INTEGER PROGRAMMING APPROACHES TO THE DOMINATING TREE PROBLEM

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

BY

SELİN AKİFOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

SEPTEMBER 2016

Approval of the thesis:

INTEGER PROGRAMMING APPROACHES TO THE DOMINATING TREE PROBLEM

submitted by **SELİN AKİFOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mustafa Murat Köksalan
Head of Department, **Industrial Engineering** _____

Assist. Prof. Dr. Mustafa Kemal Tural
Supervisor, **Industrial Engineering Department, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. İsmail Serdar Bakal
Industrial Engineering Department, METU _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering Department, METU _____

Assoc. Prof. Dr. Serhan Duran
Industrial Engineering Department, METU _____

Assist. Prof. Dr. Bahar Çavdar
Industrial Engineering Department, METU _____

Assist. Prof. Dr. Ayşegül Altın Kayhan
Industrial Engineering Department, TOBB ETU _____

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: SELİN AKİFOĞLU

Signature :

ABSTRACT

INTEGER PROGRAMMING APPROACHES TO THE DOMINATING TREE PROBLEM

Akifoğlu, Selin
M.S., Department of Industrial Engineering
Supervisor : Assist. Prof. Dr. Mustafa Kemal Tural

September 2016, 90 pages

Let $G = (V, E)$ be a simple undirected edge-weighted graph, where V and E denote the set of vertices and edges of G, respectively. The Dominating Tree Problem (DTP) searches for a minimum weighted tree in G, say DT, such that each vertex either belongs to DT or is one-hop away from DT. This problem is an $\mathcal{NP}$-hard but a practical problem. The solution of the DTP is used to construct a backbone for wireless sensor networks, which have a wide usage in many industrial and consumer applications. In this thesis, different integer programming formulations of the problem are introduced. For some instances in the literature, optimal solutions are provided for the first time and for some others best known heuristic solutions are shown to be optimal. Moreover, branch-and-cut approach is applied to the problem.

Keywords: Integer Programming, Branch-and-Cut Procedure, Dominating Tree

Problem

ÖZ

BASKIN AĞAÇ PROBLEMİ'NE TAMSAYILI PROGRAMLAMA YAKLAŞIMLARI

Akifoğlu, Selin
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Mustafa Kemal Tural

Eylül 2016 , 90 sayfa

$G = (V, E)$, V'nin düğüm kümesini, E'nin kenar kümesini temsil ettiği basit, yönlendirilmemiş ve kenarları ağırlıklandırılmış bir çizge olsun. Baskın Ağaç Problemi, DT adı verilen, G üzerindeki her düğümün ya DT'ye ait olacağı ya da DT'deki düğümlerden en az birine komşu olacağı enaz ağırlıklı bir ağaç arar. Bu problem gerçek hayat uygulamaları olan NP-zor bir problemdir. Baskın Ağaç Problemi'nin çözümü endüstri ve tüketici uygulamalarında yaygın olarak kullanılan kablosuz sensör ağları için bir omurga oluşturmak için kullanılmaktadır. Bu tezde, bahsedilen problem için farklı tamsayılı programlama formülasyonları sunulmuştur. Literatürdeki bazı problem örnekleri için en iyi sonuçlar ilk defa sağlanmış, bazıları içinse bulunmuş sezgisel en iyi sonuçların en iyi olduğu gösterilmiştir. Buna ek olarak, problem çözümünde dal-kesme yöntemi de kullanılmıştır.

Anahtar Kelimeler: Tamsayılı Programlama, Dal-Kesme Yöntemi, Baskın Ağaç Problemi

To my parents...

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my supervisor Assist. Prof. Dr. Mustafa Kemal Tural for his endless support and assistance both throughout this thesis and my new life in METU-IE Department. He is the inspiration for me to study in this department.

I also owe a word to my family for their support and trust. Although the distance we have in between, it is exclusive to feel their support right next to me. Without them, this work would not be possible.

Moreover, I acknowledge my colleagues and my friends for the patience they show to me and for cheering me up when I had difficulties throughout the work. Specially, I would like to thank my friends whom I share the stage with.

I am also grateful to my committee members Assoc. Prof. Dr. İsmail Serdar Bakal, Assoc. Prof. Dr. Serhan Duran, Assist. Prof. Dr. Bahar Çavdar, and Assist. Prof. Dr. Ayşegül Altın Kayhan for their valuable contribution to this work.

Last but not the least, I also would like to present my thanks to all my professors of METU-IE Department for letting me to put myself forward.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

BMI	Big-M improvement
CDS	Connected dominating set
CF	Cutset formulation
DS	Dominating set
DT	Dominating tree
DTP	Dominating tree problem
EE	Edge elimination
EF	Number of edges fixed
ESPP	Elementary shortest path problem
FF	Flow formulation
IP	Integer programming
IPC	Classical integer programming formulation
IPS	Strengthened integer programming formulation
MCDSP	Minimum connected dominating set problem
MCFF	Multi-commodity flow formulation
MDSP	Minimum dominating set problem
MFD	Martin's formulation for the DTP
MIP	Mixed-integer programming
MST	Minimum spanning tree
TSP	Travelling salesman problem
VeF	Number of vertices fixed
VF	Variable fixing
VNS	Variable neighborhood search
WSN	Wireless sensor network

CHAPTER 1

INTRODUCTION

Wireless Sensor Networks (WSNs) include wireless sensors that collect data from the environment they are in, and transmit the data throughout the network to a base station, which communicates with other networks, or form a backbone for other networks. As stated in [18], in the network layer of a WSN, sensors form a network and try to transmit data to the base station.

Yick, Mukherjee, and Ghosal [28] conduct a comprehensive literature review on WSNs. They point out that the wireless sensors are deployed in locations that are hard to access, and mostly their places are not even known. So, it becomes a crucial concern to keep the system working to reduce maintenance and management costs. Since these sensors include electronic gadgets, they need power to operate. Thus, as power supply, they use batteries embedded to their system, but in some cases there may also exist external power supply such as solar panels.

A WSN needs to transmit data using the available power. In some applications the data is transmitted by using the so called flooding approach, in which each sensor distributes the gathered data to all sensors that it is able to communicate with. However, in [15], Ni et al. claim that the data transmission done by flooding, causes redundancy, contention, and collision. Thus, we need a routing protocol to save from energy as stated in [25]. To serve this aim, we will build a backbone to transmit the data.

The constructed backbone provides routing information to sensors and lets them communicate with each other using this information. In other words, with the help of backbone construction, sensors do not need to flood their data to all other sensors that they are able to communicate with; but only the ones that are determined in the backbone construction.

WSNs are widely used in many application areas, such as health-care, environmental sensing, military applications, and industrial monitoring. For instance, sensors deployed to the environment can keep track of the changes of temperature, pressure, humidity, and provide high-level information to the user. Also, in hospitals advanced medical systems can keep track of a patient by collecting the raw data from the sensors, processing them using pre-defined procedures, and providing high-level information to the doctor. Therefore, many studies are held on such applications with wireless sensors as in [13], [14].

Expectedly, many problems arise from this field. One of them is constructing this backbone, so that the sensors can communicate with each other and transmit data to the base station. During the transmission, both *wires* and sensors consume energy. Please note that by *wires*, we refer to the communication opportunity, not a physical wiring system. Since sensors run on battery, the energy consumption is crucial in order to maintain the system. It is not easy to detect the sensors that are down and replace them since these sensors are widely deployed and hundreds of thousands of them may exist in the field.

Constructing a backbone which results in the least energy consumption so that the system remains longer, is one of the main purposes. While constructing this, we do not have to include all the sensors in the backbone, but the sensors that are not included in the backbone should be exactly one-hop away from a vertex included in the backbone because of the communication requirements. Actually, this application is defined in the literature as *domination* and studied with the name Dominating Set (DS) [1].

In a more formal way, a sensor u dominates sensor v, if u is in the dominating set, and u and v are able to communicate. So, all the sensors not in DS, has to be dominated by at least one sensor in DS, i.e. should be one-hop away from at

least one sensor in DS.

However, having a DS is not enough to transmit data between sensors. We need to have a set of sensors that can communicate along the given network. To overcome this issue several problems are studied in the literature. We will mention two of these here: Minimum Connected Dominating Set Problem (MCDSP) and Minimum Cost Dominating Tree Problem (DTP).

MCDSP searches for a set of sensors and *wires* which has the minimum number of sensors and forms a connected DS. While MCDSP focuses on the number of sensors included in the backbone, the DTP [21] aims to form an energy efficient backbone by using the communication cost information between the sensors and preserving connectivity across the network.

The DTP is constructed upon some assumptions. These assumptions are:

- Domination is defined only for sensors, i.e., vertices, not the communication edges.
- Sensor locations, communication opportunities between sensors are known for sure.
- Energy consumption between nodes are known for sure and they are deterministic.

The assumptions show that the given network is a perfect network having a reliable design. However, we are aware that such a design cannot exist in real world. The sensors may collapse, the communication may be blocked because of some environmental effects. We define a perfect environment without these issues causing unreliability, and motivate the application of the DTP by its measure of perfect information. In case of perfect conditions, we address the cost of the network by the DTP and analyze real world cost accordingly. Therefore, this thesis focuses on the DTP and proposes exact solution approaches for its solution.

As stated, in minimum cost DTP, one needs to minimize the communication cost across the network. So, each sensor pair included in the network, that

are able to communicate, is associated with a non-negative cost value. In the literature, we see that this cost value increases as the distance between two sensors increases. Furthermore, in the DTP literature, the cost is assumed to be proportional to the square of the distance between two sensors. [23] defines the cost, w_{uv}, between vertices u and v as:

$$w_{uv} \propto d_{uv}^2$$

where d_{uv} is the Euclidean distance between the corresponding vertices. Although this assignment gives a general idea about the power consumption, one may observe that it overlooks the fact that flow of information between some sensors may be much more than the others, thus may have high impact on the battery consumption.

In this thesis, we aim to examine the DTP, provide exact solutions by proposing integer programming formulations. Besides, we aim to decrease the time spent on finding the solution with the help of different solution approaches.

The thesis is organized as follows: In Chapter 2, we present a literature review on wireless sensor networks and related problems to the DTP. Chapter 3 defines the problem and introduces mixed-integer programming formulations for the DTP. Chapter 4 details pre-processing procedures proposed for the problem. An iterative branch-and-cut procedure and a pure cutting plane algorithm are introduced in Chapter 5. Chapter 6 presents the preliminary and detailed computational experiments. We conclude and share comments on the future research directions in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews the literature related to the DTP. First, the application area WSNs, then the related problems are examined, namely the Dominating Set Problem, the Minimum Spanning Tree Problem, and the Elementary Shortest Path Problem. The variants of the DSP are also discussed throughout the chapter.

2.1 Wireless Sensor Networks (WSNs)

WSNs have a wide application area in real-life. Therefore, many aspects of it have been studied in the literature. With the emerging applications of Internet of Things, the attention on the subject has come to a high level. As stated in [17], there are some main characteristics of WSNs: lifetime, flexibility, maintenance, and data collection. It is desired to have long-running, flexible networks to reduce the cost of maintenance and data collection. Thus, constructing such networks is crucial.

In [28], the authors state that, in the network layer of WSNs, the routing of the data is determined. There are many protocols to transmit the data that the sensors gather to the base station. However, as stated, it is desired to find one with the least cost, and this cost can be determined by means of energy, memory, and capability of computation. Moreover, some energy efficient protocols are provided in [28].

2.2 The Dominating Set Problem (DSP)

The Dominating Set Problem aims to find a dominating set on a graph $G = (V, E)$. There are several variants of this problem studied in the literature. We will briefly review some of them that are related to our study, namely, Minimum Dominating Set Problem (MDSP), Minimum Connected Dominating Set Problem (MCDSP), Minimum m-connected k-tuple Dominating Set Problem ((m,k)-CDS).

2.2.1 The Minimum Dominating Set Problem (MDSP)

The MDSP searches for a dominating set on a graph $G = (V, E)$ by minimizing the number of vertices included in the solution. Therefore, this problem is closely related to the DTP by means of selecting a dominating set on a graph.

In [10], it is stated that the MDSP is $\mathcal{NP}$-hard. Therefore, many heuristic and meta-heuristic algorithms are proposed for its solution. For instance, Nieberg and Hurink [16] propose a polynomial time approximation scheme (PTAS) for the MDSP in unit disk graphs. In [9], the authors discuss exact exponential algorithms to compute the minimum dominating set on specific type of graphs.

Moreover, Zou et al. [30] consider the MDSP with weights on vertices. They propose an approximation algorithm for the solution to the minimum weight DSP.

Now, we will introduce more specific problems to MDSP in the following section.

2.2.2 The Minimum Connected Dominating Set Problem (MCDSP)

MCDSP intends to find a minimum dominating set that is also connected via the edges of the graph. We know that MDSP is $\mathcal{NP}$-hard, and so MCDSP. The DTP also tries to find a connected dominating set, but it serves to a different objective. Therefore, some procedures, valid inequalities etc. are DTP specific. Nevertheless, the MCDSP gives a wide aspect for the solution approaches to the

DTP, by its similar problem definition.

In [22], the authors propose some valid inequalities for MCDSP and apply Benders Decomposition and branch-and-cut procedures on the problem. On the other side, Fan and Watson [8], propose many IP formulations that use Miller-Tucker-Zemlin constraints, Martin constraints, single-commodity flow constraints, and multi-commodity flow constraints.

A generalization of MCDSP, (m,k)-CDS, is studied from many aspects in the literature. Here, m refers to connectedness, and k refers to domination. If all pairs of the vertices are connected by at least m disjoint paths, and the vertices not included in the backbone are dominated by at least k vertices, then it is a solution for (m,k)-CDS Problem. In [27], the general problem is studied and two algorithms, centralized algorithm and distributed deterministic algorithm, produced better results for general k- and m-values than in the literature. The problem is reduced for specific m and k values in some works. To exemplify, in [20], they do not restrict m values, but work for $k = 1$, $k = 2$ and $k = m$. On the other hand, the k-values are not restricted but m is assumed to be 2 in the work of Li and Zhang [12]. Also, in [4], the authors study on 2-connected dominating set problem.

This problem is also related to the WSNs, since the domination and connectivity concerns are on site. It is also more realistic to have a m-connected and k-dominated backbone in order to increase reliability. However, note that, while increasing the reliability, the cost increases as well. So, one needs to examine the trade-off in between.

2.3 The Dominating Tree Problem (DTP)

Dominating Tree Problem is introduced by [29] in 2008, and proved to be $\mathcal{NP}$-hard. As it is a recently introduced problem, the literature is limited.

To the best of our knowledge, no integer programming formulations has been studied in the literature for the DTP other than the classical one which is intro-

duced in [21]. On the other hand, the studies in the literature on the DTP mainly focus on heuristic algorithms, such as variable neighborhood search (VNS) and meta-heuristic algorithms, such as genetic, artificial-bee colony, and ant-colony optimization.

For instance, in 2013, Sundar and Singh [23] propose a new heuristic algorithm and compare it with the existing ones. Besides, they work on two meta-heuristic approaches which yield satisfying results. In [3], the heuristic algorithm proposed in [23] is improved, and an additional evolutionary algorithm with guided search is studied.

The most recent work on the DTP, held by Dražić, Čangalović, and Kovačević-Vujčić [5], focuses on variable neighborhood search (VNS) based heuristic. The results achieved by VNS are compared with the results of algorithms that exist in the literature.

However, the studies held on this subject do not offer an exact approach to the problem. They utilize heuristic approaches and the results are not proved to be optimal. In this study, we aim to obtain the optimal values for those instances with IP formulations or branch-and-cut operations.

2.4 Other Related Problems

The DTP is closely related to the Minimum Spanning Tree Problem (MST) Problem, in the sense that it tries to minimize the total cost on the edges of a tree. Therefore, some applications for the MST guided our research. Moreover, the Travelling Salesman Problem (TSP), and the Elementary Shortest Path Problem (ESPP) are also related problems, since they aim to eliminate cycles in the solution.

2.4.1 The Minimum Spanning Tree Problem (MST)

The MST is a widely studied problem in the literature. It is not practical to include all here, so we will limit ourselves to the most related literature.

Trying to get rid of exponentially many sub-tour elimination constraints is one of the main concerns of the MST. Although pure MST has a polynomial time algorithm for the solution, variants of the problem still need to deal with this issue. To exemplify, Behle, Jünger, and Liers [2] work on the Degree-Constrained Minimum Spanning Tree Problem that searches for a minimum spanning tree with additional degree constraints. They present many separation procedures to eliminate sub-tours and conclude that the primal separation procedures outweighs the standard ones.

Moreover, [19] works on minimum spanning trees under conflict constraints and separate sub-tour elimination constraints with the help of Maximum Flow-Minimum Cut Algorithm.

Apart from those, in [7], the authors review many integer programming formulations for the MST. These are also related to the problem the DTP in the sense that they have similar formulations. Therefore, in this thesis we utilize the idea behind these formulations, and construct IP formulations for the DTP.

2.4.2 The Elementary Shortest Path Problem (ESPP)

The purpose of the ESPP is finding a shortest path on a graph G where edges may take negative cost values. Because of this negativity, the algorithms proposed by Bellman-Ford or Dijkstra do not apply, and the solution tends to create sub-tours. Therefore, in order to solve the problem, some sub-tour elimination procedures are needed. Since subtour elimination constraints increase exponentially in number with the number of vertices, the computational complexity increases as well. Thus, many approaches aiming to handle the sub-tour elimination constraints are studied in the literature.

For instance, in [6], Drexl discusses the separation of sub-tour elimination constraints for the ESPP. He points out the application of eliminating the sub-tours by maximum flows and by strong components. Also, Taccari [24] introduces many integer programming formulations for the ESPP and applies a Min Cut-based and a Strong Component-based separation procedures to eliminate the

sub-tours. He shows that the Strong Component-based separation procedure is faster than all other procedures.

CHAPTER 3

PROBLEM DEFINITION AND INTEGER PROGRAMMING FORMULATIONS

3.1 Problem Definition

3.1.1 Terminology

For the sake of completeness, before introducing the DTP, we first define the related terminology. Following definitions introduce the concepts from graph theory [26].

Definition 3.1. An undirected graph G consists a pair $G = (V, E)$ where V and E are the set of vertices, and the set of unordered pairs of vertices, i.e., edges, respectively.

Please note that, throughout this thesis, we will assume undirected graphs, unless otherwise stated.

Definition 3.2. Two vertices $u, v \in V$ of a graph are said to be adjacent if $(u, v) \in E$. An edge $(u, v) \in E$ is incident to the vertices $u, v \in V$. The set $\delta(u)$ consists of the vertices that are adjacent to the vertex $u \in V$, and is called as the open-neighborhood of vertex u. The set $\Gamma(u)$, called as the closed-neighborhood of a vertex is the union of the open-neighborhood of $\delta(u)$ of u and u; in other words, $\Gamma(u) = \delta(u) \cup \{u\}$.

In the light of this basic terminology, we will define connectedness, cycle, and cut-edge to enlighten the definition of tree.

Definition 3.3. A u, v-path exists between pair of vertices u and v, if a sequence of adjacent edges connecting a sequence of vertices, where vertices are distinct possibly except u and v, can be listed. Then, a graph $G = (V, E)$ is said to be connected if it has a u, v-path for all pairs of $u, v \in V$. It is said to contain a cycle if a path with the same starting and ending vertex can be found, for at least one vertex.

Definition 3.4. An edge (u, v) is a cut-edge of the graph if it belongs to no cycle. We have that there are no cycles in a graph if and only if every edge is a cut-edge.

Definition 3.5. A graph $G = (V, E)$ is a tree if at least one of the following is satisfied:

- G is connected and every edge is a cut-edge.
- G is connected and the number of edges is one less than the number of vertices.
- For all u, v-pairs, there exist exactly one path that connects them.
- G is connected, but becomes disconnected when any of the edges is removed.
- G is connected and contains no cycles.
- G is connected, and adding any edge to G creates exactly one cycle.

Now, suppose $G = (V, E)$ defines a connected graph representing a WSN. In this representation, vertices specify the wireless sensors, and edges specify the communication opportunity between two wireless sensors. Note that this graph does not need to be a tree. Next, we formally define two main problems related to the DTP.

Definition 3.6. In a graph G, a set $S \subseteq V$ is a Dominating Set (DS), if every vertex in $V \setminus S$ has at least one neighbor in S.

Definition 3.7. Minimum Connected Dominating Set Problem (MCDSP) searches for a sensor set that has minimum number of sensors, i.e., vertices, and forms a connected DS.

Definition 3.8. In a connected graph G, a subgraph DT is a dominating tree if DT is a tree and the vertices in DT dominates the ones not in DT.

These definitions lead us to the definition of the minimum cost DTP.

Definition 3.9. The Minimum Cost Dominating Tree Problem (DTP), searches for a dominating tree which has the least total cost on the edges of the tree.

Definition 3.10. The degree of a vertex is equal to the number of edges incident to that vertex.

Definition 3.11. Given a connected graph $G = (V, E)$, a spanning tree of G is a tree that connects all the vertices of the graph. In a weighted graph, a Minimum Spanning Tree (MST) minimizes the total weight of the edges used in the tree.

Definition 3.12. Let $G = (V, E)$ be an undirected graph. A connected component, CC, of G, is a subgraph of G, where there exists at least one path between each pair of vertices in CC, and no path exists between the vertices of CC and the vertices not in CC.

In other words, in a connected component of an undirected graph G, all the vertices are reachable from the other vertices of the same component, but none of the vertices of a component can be reached from the vertices that are in a different component. Consider the Figure 3.1 below.

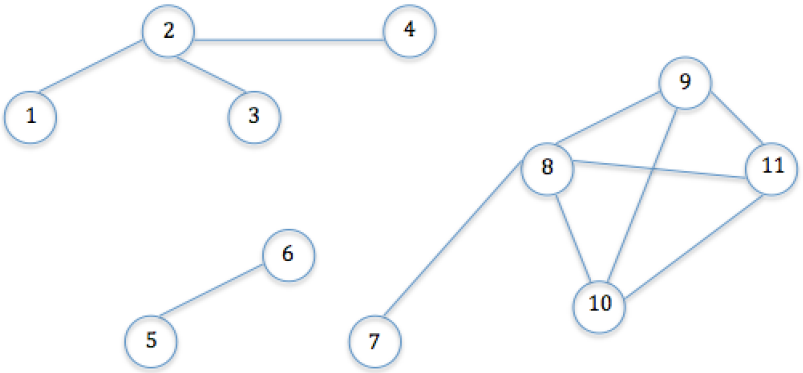


Figure 3.1: Illustration for Connected Component

This graph includes three connected components: $G_1 = \{V_1, E_1\}$, $G_2 = \{V_2, E_2\}$ and $G_3 = \{V_3, E_3\}$; where $V_1 = \{1, 2, 3, 4\}$, $V_2 = \{5, 6\}$, $V_3 = \{7, 8, 9, 10, 11\}$ and E_1, E_2 and E_3 are the corresponding edge sets. For instance, in the subgraph G_3, vertex 7 is reachable from vertex 11, by the paths *11-10-8-7*, *11-8-7*, *11-9-8-7*, *11-9-10-8-7*, *11-10-9-8-7*. Since we can find such path(s) for each pair of vertices, we say that G_3 is a connected component. Same applies for the other vertices; so this graph has three connected components.

Definition 3.13. Let $G = (V, E)$ be a graph and $S \subset V$ be a subset of vertices. The induced graph of G on set S is defined as $G' = (V', E')$, where $V' = S$ and $E' = \{(u, v) \in E | u, v \in S\}$.

3.1.2 Classical IP formulation for the DTP

We consider a network having n vertices and m edges. We represent this network as $G = (V, E)$ where V and E are the set of vertices and set of edges, respectively.

We assume that each edge (u, v) is associated with a weight w_{uv}. Here, weight values reflect the power consumption on the edges during data transmission. Moreover, the weights are assumed to be deterministic. We want to decide on which vertices and edges to appear in the dominating tree to minimize the cost of the network.

We define our decision variables as follows,

$$x_u = \begin{cases} 1 & \text{if vertex } u \text{ is selected in a dominating tree} \\ 0 & \text{otherwise} \end{cases}, \qquad u \in V$$

$$y_{uv} = \begin{cases} 1 & \text{if edge } (u, v) \text{ is selected in a dominating tree} \\ 0 & \text{otherwise} \end{cases}, \qquad (u, v) \in E$$

We want to minimize the total weight of the selected edges of the dominating tree. An integer programming formulation, presented in [21], is shown below in

(IPC). In this formulation, constraint set (3.1) defines the edge-vertex restrictions. If an edge (u, v) is selected, i.e., $y_{uv} = 1$, then both end-points should be selected as well, i.e., $x_u = x_v = 1$. Constraint set (3.2) ensures that the selected edges do not form any cycle. To see this, if the selected edges form a cycle C, then taking $S = C$ results in the violation of (3.2). Moreover, constraint (3.3) ensures that the number of vertices selected is exactly one more than the number of edges, which is the case for trees. Lastly, with the help of constraint set (3.4), the model constructs a dominating set, by selecting at least one of the vertices that are in the closed neighborhood of each vertex. The objective function minimizes the total weight of the dominating tree.

$$
\begin{aligned}
&\min \sum_{(u,v)\in E} w_{uv} y_{uv} \\
&\text{s.to} \quad x_u + x_v \geq 2y_{uv} && \forall (u,v) \in E && (3.1)\\
&\sum_{u,v\in S,(u,v)\in E} y_{uv} \leq |S| - 1 && \forall S \subset V, S \neq \emptyset, S \neq V && (3.2)\\
&\sum_{(u,v)\in E} y_{uv} = \sum_{u\in V} x_u - 1 && && (3.3)\\
&\sum_{u\in\Gamma(v)} x_u \geq 1 && \forall v \in V && (3.4)\\
&x_u \in \{0,1\} && \forall u \in V && (3.5)\\
&y_{uv} \in \{0,1\} && \forall (u,v) \in E && (3.6)
\end{aligned}
\qquad \text{(IPC)}
$$

However, this model becomes inapplicable for large size problem instances, since (3.2) includes $2^{|V|}$ many constraints. So, in order to get through this computational disadvantage posed by the formulation (IPC), we propose different integer programming based formulations for the exact solution of the DTP. Cutset, flow, Martin's [7] formulations and a multi-commodity network flow formulation [11] that were proposed for the solution of the minimum (weighted) spanning tree problem are taken as the building blocks to come up with other IP formulations for the DTP. In the following subsection, the revised formulations for the DTP are given.

3.1.3 Strengthened formulation

In this section, revisions on the (IPC) are presented.

3.1.3.1 Separating constraint set (3.1) into two components:

Firstly, constraint set (3.1) can be divided into two as follows,

$$x_u \geq y_{uv} \qquad \forall u \in V, v \in \delta(u) \qquad (3.7)$$

$$x_v \geq y_{uv} \qquad \forall u \in V, v \in \delta(u) \qquad (3.8)$$

By this way, we aim to tighten the LP-relaxation of the formulation. Note that this disaggregation is classical and widely used in the literature.

Also, this revision makes the formulation stronger since the corresponding y_{uv} value will be at least the maximum of x_u and x_v, instead of the average of the sum of those.

3.1.3.2 Revising constraint set (3.4):

The constraint set (3.4) enforces that either the vertex v or at least one of the neighbors of v should be included in any dominating set for each $v \in V$. For graphs in which there exists a vertex of degree $|V| - 1$, the solution of the dominating tree problem is trivial (select that vertex only in the DTP). Therefore, we assume that G does not have such a vertex and revise the constraint set (3.4) after this assumption as follows:

$$\sum_{v \in \delta(u)} x_u \geq 1 \qquad \forall v \in V \qquad (3.9)$$

Also, with the decrease in the left-hand side terms the formulation becomes stronger.

Moreover, in [22], one more revision on (3.4) of (IPC) is studied. It is claimed that, the inequality (3.4) can be lifted as:

$$\sum_{v\in\Gamma(u)} x_u \geq 1 + \sum_{v\in\Gamma(u)} y_{uv} \qquad \forall v \in V \qquad (3.10)$$

In the paper, the correctness of this inequality is discussed. In a similar manner, (3.9) can be strengthened by lifting it as:

$$\sum_{v\in\delta(u)} x_u \geq 1 + \sum_{v\in\delta(u)} y_{uv} \qquad \forall v \in V \qquad (3.11)$$

The inequality (3.11) can be shown to be valid for the DTP. Suppose we have a dominating tree D. If for a vertex $v \in D$, $\sum_{v\in\delta(u)} y_{uv}$=0, then $\sum_{v\in\delta(u)} x_u$ will clearly be greater than or equal to 1 by (3.9). On the other hand, if $v \in D$, $\sum_{v\in\delta(u)} y_{uv} > 0$, then the subgraph of D induced by $\delta(u)$ is a forest and we have that the number of vertices is at least 1 more than the number of edges in a forest.

3.1.3.3 Revising constraint set (3.2):

In order to further strengthen the formulation (IPC), we can revise the constraint set (3.2) as follows:

$$\sum_{\substack{u,v\in S \\ v\in\delta(u)}} y_{uv} \leq \frac{|S|-1}{|S|} \sum_{u\in S} x_u \qquad \forall S \subset V, S \neq \emptyset, S \neq V \qquad (3.12)$$

Note that, the constraint set (3.2) is a classical set of sub-tour elimination constraints. Still, we could not find its strengthened form (3.12) in the literature. We next show that (3.12) is valid for the DTP and it is stronger than (3.2).

Proof. (i) First, let us observe that the proposed constraint set (3.12) is stronger than (3.2). We know that in a dominating tree the maximum value for the number of vertices selected from a set S is $|S|$. Thus,

$$\sum_{u \in S} x_u \leq |S| \qquad \forall S \subset V, S \neq \emptyset, S \neq V \tag{3.13}$$

Then,

$$\sum_{\substack{u,v \in S \\ v \in \delta(u)}} y_{uv} \leq \frac{|S|-1}{|S|} \sum_{u \in S} x_u \leq |S| - 1 \qquad \forall S \subset V, S \neq \emptyset, S \neq V \tag{3.14}$$

Therefore, the proposed upper bound is stronger than (3.2).

(ii) Unlike the MST problem, the DTP does not include all the vertices in the optimal solution. From (3.3) we know that

$$\sum_{(u,v) \in E} y_{uv} = \sum_{u \in V} x_u - 1 \tag{3.15}$$

holds for every $u \in V$.

To prove the validity of the constraint set (3.12), we need to show that the incidence vector of any dominating tree satisfies all the constraints in (3.12). Let D be a dominating tree and $S \subset V$. The subgraph of D induced by $S \cap D$

is a forest and so $\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} \leq \sum_{u\in S} x_u - 1$. Let $k = \sum_{u\in S} x_u$. Clearly, $k-1 \leq \frac{|S|-1}{|S|}k$ holds true and using $\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} \leq k-1$ the inequality is proved to be valid.

Therefore, proposed upper bound is valid for the formulation. □

The following sub-tour elimination constraints were proposed in [22].

$$\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} \leq \sum_{u\in S\setminus\{i\}} x_u \qquad i \in S, S \subset V, S \neq \emptyset, S \neq V \tag{3.16}$$

However, (3.16) results in more constraints than (3.12), since for each subset elements we need to write one more equation, but it is still stronger.

Claim: $\{(x,\, y)|(x,\, y) \text{ satisfies (3.16)}\} \subseteq \{(x,\, y)|(x,\, y) \text{ satisfies (3.12)}\}$

Proof. Suppose $\bar{x}$ and $\bar{y}$ form a solution for (3.16), and

$$\bar{x}_{max} = max\{\bar{x}_j : j \in S\}. \tag{3.17}$$

Then,

$$\bar{x}_{max} \geq \frac{\sum_{u\in S} \bar{x}_u}{|S|} \tag{3.18}$$

since the maximum is greater than or equal to the average of all x_j values. Utilizing (3.12),

$$\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} \leq \sum_{u\in S\setminus\{i\}} x_u \tag{3.19}$$

$$\leq \sum_{u\in S} \bar{x}_u - \bar{x}_{max} \tag{3.20}$$

$$\leq \sum_{u\in S} \bar{x}_u - \frac{\sum_{u\in S}\bar{x}_u}{|S|} \tag{3.21}$$

$$\leq \frac{|S|-1}{|S|}\sum_{u\in S} \bar{x}_u \tag{3.22}$$

□

3.1.3.4 Valid inequalities

In [22] some valid inequalities are proposed regarding connectivity property. We know that sub-tour elimination constraints satisfy connectivity. So,

$$\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} \leq \sum_{u\in S} x_u - 1 \quad S\subset V, S\neq\emptyset, S\neq V, S\cap D\neq\emptyset \tag{3.23}$$

is valid, where D is a dominating tree. As stated, this holds true when the selected subset includes at least one vertex from the dominating tree. Therefore, we can write this inequality for the subsets that include a vertex of any dominating tree for sure.

For instance, the vertices that are adjacent to a degree one vertex, are certainly known to be included in the dominating tree in order to satisfy domination properties. Therefore, for subsets including such vertices this inequality is valid.

Moreover, the following valid cut inequalities (3.24) are proposed in [22].

$$\sum_{\substack{u\in S, v\in V\setminus S \\ v\in\delta(u)}} y_{uv} \geq 1 \quad S\subset V, S\neq\emptyset, S\neq V, \Gamma(S)\neq V, \Gamma(V\setminus S)\neq V \tag{3.24}$$

The inequality (3.24) holds true when the closed neighborhoods of both subsets S and $V \setminus S$ do not cover the vertex set V. If one of them was to cover the set, then we would not need to connect S and $V \setminus S$, and the inequality would not necessarily be valid.

The strengthened formulation (IPS) that we will use in the computational experiments can be summarized as follows:

$$\min \sum_{(u,v)\in E} w_{uv} y_{uv}$$

$$\text{s.to} \quad x_u \geq y_{uv} \qquad \forall (u,v)\in E \tag{3.25}$$

$$x_v \geq y_{uv} \qquad \forall (u,v)\in E \tag{3.26}$$

$$\sum_{\substack{u,v\in S \\ v\in\delta(u)}} y_{uv} \leq \frac{|S|-1}{|S|}\sum_{u\in S} x_u \qquad \forall S\subset V, S\neq\emptyset, S\neq V \tag{3.27}$$

(IPS)

$$\sum_{(u,v)\in E} y_{uv} = \sum_{u\in V} x_u - 1 \tag{3.28}$$

$$\sum_{v\in\delta(u)} x_u \geq 1 + \sum_{v\in\delta(u)} y_{uv} \qquad \forall v\in V \tag{3.29}$$

$$x_u \in \{0,1\} \qquad \forall u\in V \tag{3.30}$$

$$y_{uv} \in \{0,1\} \qquad \forall (u,v)\in E \tag{3.31}$$

3.2 Cutset Formulation

3.2.1 Formulation

Cutset formulation (CF) is another variation of the IP formulation. Instead of considering the vertices in the subsets, it considers the relation between the two subsets, that are the vertices selected in S and the vertices not selected in S, namely $V \setminus S$. In the (CF), the third constraint set (3.27), is revised as follows:

$$\sum_{\substack{u \in S \\ v \in \delta(u) \\ v \in V \setminus S}} y_{uv} \geq \frac{\sum_{u \in S} x_u}{|S|} + \frac{\sum_{u \in V \setminus S} x_u}{|V| - |S|} - 1 \quad S \subset V, S \neq \emptyset, S \neq V \tag{3.32}$$

3.2.2 Proof of correctness

As stated in (3.12), we know that the following holds:

$$\sum_{\substack{u,v \in S \\ v \in \delta(u)}} y_{uv} \leq \frac{|S| - 1}{|S|} \sum_{u \in S} x_u \qquad \forall S \subset V, S \neq \emptyset, S \neq V \tag{3.33}$$

Utilizing equation (3.15) we get the following.

$$\sum_{\substack{u,v \in S \\ u \in \delta(v)}} y_{uv} + \sum_{\substack{u \in S \\ v \in V \setminus S \\ u \in \delta(v)}} y_{uv} + \sum_{\substack{u,v \in V \setminus S \\ u \in \delta(v)}} y_{uv} = \sum_{u \in S} x_u + \sum_{u \in V \setminus S} x_u - 1 \tag{3.34}$$

We replace the first and third terms of the left-hand side of (3.34), and come up

with the following inequality:

$$\sum_{u \in S} x_u + \sum_{u \in V \setminus S} x_u - 1 - \sum_{\substack{u \in S \\ v \in \delta(u) \\ v \in V \setminus S}} y_{uv} \leq \frac{|S| - 1}{|S|} \sum_{u \in S} x_u + \frac{|V| - |S| - 1}{|V| - |S|} \sum_{u \in V \setminus S} x_u \tag{3.35}$$

Rearranging (3.35), we get (3.32).

Claim: This inequality does not cut off the incidence vector of any dominating tree.

Proof. Let D be a dominating tree and S be a subset of V, satisfying (3.25), (3.26), (3.28) and (3.29). The right-hand side of (3.32) may take a maximum value of 1, when all the vertices are included in the dominating tree. If S and $V \setminus S$ both contain vertices of D, then due to connectedness the left-hand side of (3.32) will be at least 1. On the other hand, if one of S and $V \setminus S$ does not contain any vertex of D, then (3.32) is clearly satisfied by the incidence vector of D as the left-hand side is greater than or equal to 0. □

Claim: This inequality eliminates all the sub-tours.

Proof. Let D be a graph corresponding to a feasible solution of the cutset formulation having a cycle. Then, it has at least two connected components since incidence vector of D satisfies (3.28).

Suppose S includes only the vertices which belong to a connected component, thus $\sum_{u \in S} x_u = |S|$. The rest of the vertices are in $V \setminus S$, which we know to include at least one connected component, thus $\sum_{u \in V \setminus S} x_u \geq 1$.

So, the right-hand side takes a value greater than 0 and less than or equal to 1, which enforces a y_{uv} value to become positive on the left-hand side. This leads to a contradiction and proves the claim. □

3.3 Single-commodity Flow Formulation

In this section, we present the flow formulation (FF) for the DTP which is obtained by modifying the classical flow formulation proposed for the MST [7]. First, source vertex selection, then the MIP formulation are described in detail.

3.3.1 Selecting a source vertex

As in the classical flow formulation of the MST, we need a source to supply flow to the other vertices. At this point, by selecting a one-degree vertex as a source, we guarantee that the vertex adjacent to the one-degree vertex is included in the solution. However, if the graph does not contain a one-degree vertex, then we choose a vertex having the least degree, create an extra vertex, and connect it to this least-degree vertex and its neighbors. Note that when there are more than one candidate source vertex, one can break the ties arbitrarily.

Suppose that we have a graph as in Figure 3.2 having 10 vertices and 15 edges. Here, there are two vertices having degree one, 4 and 6. We arbitrarily choose 4 as a source vertex here.

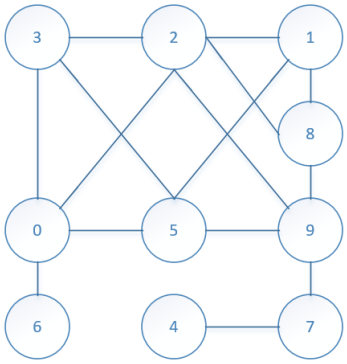


Figure 3.2: Instance with 10 vertices and 15 edges

For the graph in Figure 3.3 with 15 vertices and 30 edges, no one-degree vertex exists. Therefore, we prefer to choose a vertex having the least degree. Again, there are more than one 2-degree vertices, 1, 5, and 9. Here we select 1 arbitrarily, and connect a hypothetical source vertex to it and its neighbors. One may see the selected source vertex in Figure 3.4.

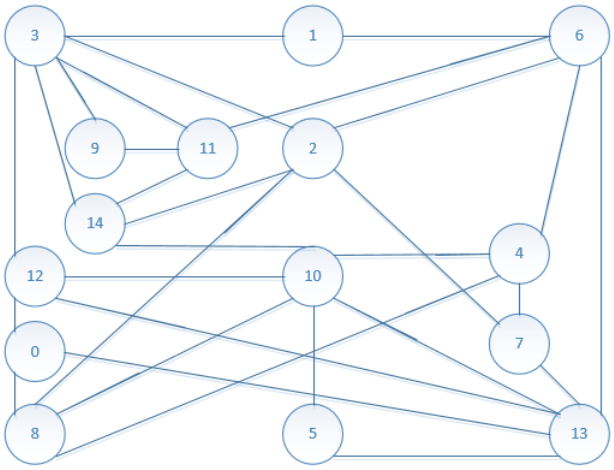


Figure 3.3: Instance with 15 vertices and 30 edges

It is good to note that we may exclude the edge connecting the source vertex and vertex 1 here, since we constraint to select at least one of the neighbors of 1. In other words, the source vertex prefers to send its flow directly to a neighbor of vertex 1 instead of following vertex 1 and a neighbor. Hereby, an extra edge is not included in the graph.

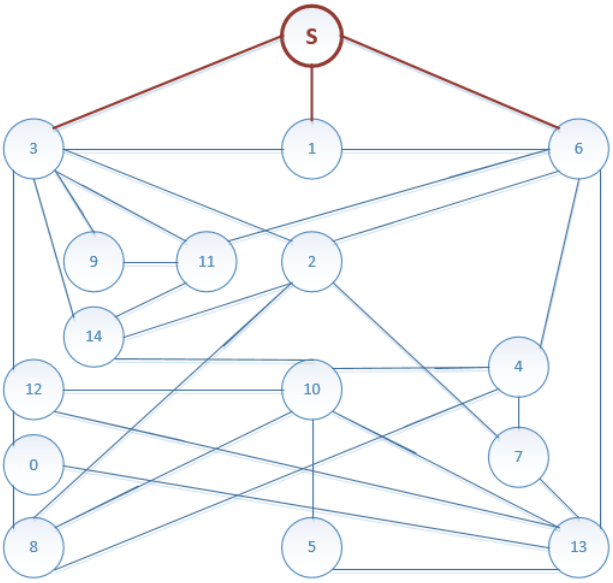


Figure 3.4: Source vertex selection for the graph in Figure 3.3

3.3.2 Formulation

In the classical flow formulation, flow is generated at a vertex, that is the source vertex, and sent to other vertices. With the flow balance restrictions, the connectedness is guaranteed. So, we may apply the main idea behind the classical single-commodity flow formulation to the DTP as well. Noting that a solution need not include all the vertices in the DTP, not all the vertices will consume flow. In other words, vertices included in the solution will consume one unit of flow, where the others will not consume any.

Below, one can find the additional decision variables used in (FF).

$\mathrm{f}_{uv} =$ The flow amount carried from vertex u to vertex v, $\quad (u, v) \in E$

$$\mathrm{a}_v = \begin{cases} 1 & \text{if the edge from source to vertex } v \text{ is selected for flow} \\ 0 & \text{otherwise} \end{cases}, v \in \delta(s)$$

As in the IPC, we have the same objective function for the problem. Next, we provide the details of the formulation (FF).

The objective function aims to minimize the total weight of the dominating tree. Inequalities (3.36) ensure the flow balance. A vertex is included in the optimal solution if and only if there is a flow passing through that vertex. However, this is valid for non-source vertices. Also, with (3.37) we enforce selected vertices to form a dominating set. Next three constraints (3.38), (3.39), and (3.40) ensure that the source is not included in the optimal solution. The first one makes sure that no flow will go into the source vertex and the edge(s) leaving the source vertex will not be included in the optimal solution by (3.39). By (3.40), we are not going to have the source vertex in the optimal solution. (3.41) and (3.42) put lower and upper bounds for the flow variables. If an edge is included in the solution, then a flow should pass through that edge, and the maximum flow that can be observed on an edge is two less than the number of vertices in the graph, since maximum $|V| - 2$ number of vertices will be included in the optimal DT. Therefore, the Big-M value is set to $|V| - 2$ throughout the model. (3.43) and (3.44) are the constraints that are already

included in the original IP formulation. (3.45) and (3.46) together make sure that exactly one edge will be used while distributing the flow from the source vertex. Last four constraints, (3.47), (3.48), (3.49), and (3.50), bind decision variables x, y, and a to binary values and f to a non-negative value, respectively.

(FF)

$$\min \sum_{(u,v)\in E} w_{uv} y_{uv}$$

$$\text{s.to} \sum_{v\in\delta(u)} f_{vu} - \sum_{t\in\delta(u)} f_{ut} = x_u \qquad \forall u \in V \setminus \{s\} \tag{3.36}$$

$$\sum_{v\in\delta(u)} x_u \geq 1 + \sum_{v\in\delta(u)} y_{uv} \quad \forall v \in V \tag{3.37}$$

$$\sum_{(u,s)\in E} f_{us} = 0 \tag{3.38}$$

$$\sum_{(u,s)\in E} y_{su} = 0 \tag{3.39}$$

$$x_s = 0 \tag{3.40}$$

$$f_{uv} \geq y_{uv} \qquad \forall v \in \delta(u)\ s.t.\ u \in V \setminus \{s\} \tag{3.41}$$

$$f_{uv} \leq M y_{uv} \qquad \forall v \in \delta(u)\ s.t.\ u \in V \setminus \{s\} \tag{3.42}$$

$$x_u \geq y_{uv} \qquad \forall u \in \delta(v)\ s.t.\ v \in V \setminus \{s\} \tag{3.43}$$

$$x_v \geq y_{uv} \qquad \forall v \in \delta(u)\ s.t.\ u \in V \setminus \{s\} \tag{3.44}$$

$$f_{sv} \leq M a_v \qquad \forall v \in \delta(v) \tag{3.45}$$

$$\sum_{v\in\delta(s)} a_v = 1 \tag{3.46}$$

$$x_u \in \{0,1\} \qquad \forall u \in V \tag{3.47}$$

$$y_{uv} \in \{0,1\} \qquad \forall u \in V,\ v \in \delta(u) \tag{3.48}$$

$$a_v \in \{0,1\} \qquad \forall v \in \delta(s) \tag{3.49}$$

$$f_{uv} \geq 0 \qquad \forall u \in V,\ v \in \delta(u) \tag{3.50}$$

In [8], the authors propose a single-commodity flow formulation (SCF) for MDSP, but there are some differences between (SCF) and (FF). For instance, as specified in Section 3.3.1, we determine the source vertex before the optimization. However, in [8], the authors decide the root vertex through the model. In other words, in (FF), the source vertex is a parameter, where in (SCF) it is a decision variable. (SCF) and (FF) are similar to each other in the sense that they both use Big-M in the formulation. However, because of the source vertex selection procedure, the formulations differ.

3.3.3 Proof of correctness

Let us consider a graph resulting from a feasible solution of (FF). Let us assume that the graph is disconnected. Since we have one source, the source will remain on side of one of the connected components. Thus, we will not be able to send flow to the connected component(s) which do not include the source vertex. Therefore, we cannot include a vertex from those connected components. This leads to a contradiction of having a disconnected graph. So, we will always have a connected graph.

Now, let us have an incidence vector of any DT. This solution is not violated by any of the constraints in (FF) since for each feasible solution we can find a path for the flow.

The domination property is satisfied with the classical domination constraints.

All in all, every feasible solution in (FF) corresponds to an incidence vector of a DT and vice versa.

3.4 Multi-commodity Flow Formulation

3.4.1 Formulation

In multi-commodity flow formulations studied in the literature, the vertices are the suppliers of flow, and each vertex supplies a unique commodity. With the

flow balance constraints, each commodity is transmitted to the sink vertex. By ensuring that the vertices included in the solution will distribute flows, and the others not, we may form a dominating tree.

In the multi-commodity flow formulation (MCFF) a sink is defined. Besides, each vertex k is defined as a source and supplies a commodity k and sends it to the sink s. Therefore, the sink is the vertex that should receive all commodities. Apart from decision variables x and y, we need one more decision variable t, and it is defined as follows:

$$t_{uv}^{k} = \begin{cases} 1 & \text{if commodity } k \text{ passes through edge } (u,v) \\ 0 & \text{otherwise} \end{cases}$$

The first three constraint sets fulfill the balance constraints. Constraint set (3.51) ensures that if vertex k, is included in the optimal solution, then the difference between outflow and inflow should be one, otherwise zero. In other words, if you supply a commodity from vertex k then you have to include that vertex in the optimal solution.

Constraint set (3.52) is written for each commodity k other than the sink s, and it makes sure that if that vertex is not the vertex k, then inflow should be equal to outflow.

Constraint set (3.53) is written for each vertex except the sink. If a vertex k supplies commodity k, then the commodity k should arrive to the sink for the sake of connectivity.

(3.54) is the constraint set for the dominating set property. (3.55) ensures that none of the edges will be included in the solution that are connected to the sink, and (3.56) makes sure that the sink is not included in the DT.

Constraints included in set (3.57) are the edge-vertex restrictions. (3.58) states that if commodity k passes through edge (u,v), then that edge has to be included in the optimal solution, and (3.59) relates the decision variables x and y. The last constraint, (3.60), uses the property of trees and imposes that the number of edges is exactly one less than the number of vertices in the DT.

The last three, (3.61), (3.62), and (3.63), enforce the decision variables to be binary for the given sets.

(MCFF)

$$\min \sum_{(u,v)\in E} w_{uv} y_{uv}$$

$$\text{s.to} \quad \sum_{v\in\delta(k)} t^k_{kv} - \sum_{k\in\delta(u)} t^k_{uk} = x_k \qquad \forall k \in V \setminus \{s\} \tag{3.51}$$

$$\sum_{v\in\delta(u)} t^k_{uv} - \sum_{m\in\delta(u)} t^k_{mu} = 0 \qquad \forall k \in V \setminus \{s\}, \forall u \in V \setminus \{s,k\} \tag{3.52}$$

$$\sum_{v\in\delta(s)} t^k_{sv} - \sum_{s\in\delta(u)} t^k_{us} = -x_k \qquad \forall k \in V \setminus \{s\} \tag{3.53}$$

$$\sum_{v\in\delta(u)} x_u \geq 1 + \sum_{v\in\delta(u)} y_{uv} \qquad \forall v \in V \tag{3.54}$$

$$\sum_{s\in\delta(u)} y_{su} + y_{us} = 0 \tag{3.55}$$

$$x_s = 0 \tag{3.56}$$

$$y_{uv} + y_{vu} \leq x_u \qquad \forall v \in \delta(u) \ s.t. \ u, v \in V \setminus \{s\} \tag{3.57}$$

$$t^k_{uv} \leq y_{uv} \qquad \forall k \in V \setminus \{s\}, v \in \delta(u), \text{ s.t. v} \in V \setminus \{s\} \tag{3.58}$$

$$\sum_{v\in\delta(u)} (y_{uv} + y_{vu}) \geq x_u \qquad \forall u \in V \tag{3.59}$$

$$\sum_{v\in\delta(u)} y_{uv} = \sum_{u\in V} x_u - 1 \tag{3.60}$$

$$x_u \in \{0,1\} \qquad \forall u \in V \tag{3.61}$$

$$y_{uv} \in \{0,1\} \qquad \forall u \in V, \ v \in \delta(u) \tag{3.62}$$

$$t^k_{uv} \in \{0,1\} \qquad \forall u \in V, \ v \in \delta(u), k \in V \tag{3.63}$$

In [8], Fan and Watson propose a multi-commodity flow formulation for MCDSP.

However, it differs from the formulation that we propose in many aspects. For instance, they use Big-M values in the formulation, while one of the main motivation for converting a single-commodity formulation to a multi-commodity formulation is getting rid of Big-M values. With this approach, we do not make use of any Big-M values in our formulation. Also, they use a decision variable for selecting the sink vertex, where in (MCFF) we do not use such a decision variable, but a sink vertex parameter.

3.4.2 Proof of correctness

The domination property is again satisfied by the set of constraints (3.54).

In (MCFF), every vertex is defined as a source. But, there is only one sink, and all the commodities generated at vertices should be transferred to the sink vertex. So, again suppose the formulation leads to a disconnected graph D as a solution. Then, at least one vertex, supplying a commodity, cannot transfer its commodity to the sink. However, constraint set (3.53) ensures that each commodity generated, must be transferred to the sink. This leads to a contradiction; therefore, (MCFF) cannot lead to disconnected graph as a solution.

Now, consider a DT being an incidence vector. We may assign each vertex in DT as a source, and send the corresponding commodities to the sink node. We are able to find a path with the help of flow balance constraints, so each commodity can reach to the sink. Therefore, DT is feasible to the model.

All in all, the proposed multi-commodity flow formulation constructs a dominating tree.

3.5 Martin's Formulation

3.5.1 Formulation

In Martin's Formulation for the DTP (MFD), the regular decisions on vertices and edges are to be made, but this time t refers to a different decision. This

formulation was previously proposed for the MST problem in [7], for the MDSP in [8], and now it is modified for the DTP.

First, let us define decision variable t. Afterwards the details of the formulation will be introduced.

$$t_{uv}^{k} = \begin{cases} 1 & \text{if edge } (u,v) \text{ and vertex } k \text{ are in the optimal solution,} \\ & \text{and } k \text{ is on the side of vertex } v \text{ when the edge } (u,v) \text{ is removed} \\ 0 & \text{otherwise} \end{cases}$$

Suppose you have a graph as in Figure 3.5. Here, a 10-vertex graph and its solution for the DTP are given. In the optimal solution, vertices 0, 2, 7 and 9, and edges (0,2), (2,9), and (9,7) are included. Since the solution is a tree, disconnecting an edge of the optimal solution results in two distinct trees. To exemplify, if we remove the edge (2,9), (0,2) with vertices 0 and 2, and (9,7) with vertices 9 and 7 refer to two different trees. Therefore, for this solution, $t_{2,9}^{0}$ takes the value of zero, since after the removal of edge (2,9), vertex 0 remains on the side of 2. On the other hand, $t_{9,2}^{0}$ takes the value of 1, since vertex 0 remains on the side of 2 when we remove the edge (2,9).

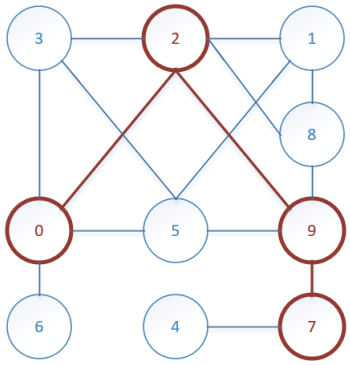


Figure 3.5: Solution to the instance with 10 vertices and 15 edges

Utilizing the decision variable t, we can prevent cycles in the solution, and form a tree. Consider the left hand side of the inequalities in (3.64) and (3.65). The first sum and y values are closely related to each other, and at most one of them should be positive. If t_{uk}^{v} is positive for some k, then it means that vertex v and the edge (u,k) exist in the optimal solution by definition, and v is on the side of k when we remove the edge (u,k). Then, in order not to have cycles, we

cannot have the edge (u, v) in the optimal solution. And if one of them, the first sum or y values, is non-zero, then we need to include the vertices u and v in the optimal solution. Also, if u and v are included in the solution, then either the edge (u, v) should be in the optimal solution, or they should be connected with the help of a vertex k. If at least one of u and v is not included in the solution, then we do not restrict y and t variables.

Moreover, constraint set (3.66) relates decision variable t to the other two decision variables x and y. It makes sure that if a decision variable t_{uv}^{k} is positive, then we need to have edge (u, v) and vertex k in the optimal solution because of the definition of the decision variable. Constraint set (3.67) puts a lower bound on t. There are 4 cases as follows:

1. Edge (u, v) and vertex k are in the optimal solution:

   Then y_{uv} and x_k take a value of one, right hand side becomes one, which means that vertex k should be reachable from either u or v.

2. Edge (u, v) is in the optimal solution, vertex k is not:

   Then y_{uv} takes a value of one, and x_k takes a value of zero, right hand side becomes zero, which means that we should not restrict t values. Note that, t is restricted to zero in this case by (3.69).

3. Edge (u, v) is not in the optimal solution, vertex k is:

   Then y_{uv} takes a value of zero, and x_k take a value of one, right hand side becomes zero, which means that we should not restrict t values.

4. Edge (u, v) and vertex k are not in the optimal solution:

   Then y_{uv} takes a value of zero, and x_k take a value of zero, right hand side becomes -1, and again t values should not be restricted.

Since we define t for the optimal solution, both the edge (u, v) and the vertex k should be in the optimal solution, and these are satisfied by (3.68) and (3.69), respectively. (3.70) relates edges and vertices, and also ensures that we cannot include both (u,v) and (v,u) at the same time since this is an undirected network

and x cannot take a value greater than zero. (3.71) defines the dominating set, and (3.72) ensures the tree property. (3.73) makes sure that if a vertex u is included in the optimal solution, then at least one of its neighbor edges should be included in the optimal solution as well.

Lastly, (3.74) makes sure that a vertex cannot be connected to itself through other vertices, where (3.75) satisfies that a vertex is not reachable from another vertex through more than one distinct paths. (3.76), (3.77), and (3.78) present the binary restrictions.

(MFD)

$$\min \sum_{(u,v)\in E} w_{uv} y_{uv}$$

$$\text{s.to} \sum_{\substack{k\in V\setminus\{u,v\} \\ k\in\delta(u)}} t^v_{uk} + y_{uv} + y_{vu} \leq \frac{x_u + x_v}{2} \qquad \forall u \in V, v \in \delta(u) \tag{3.64}$$

$$\sum_{\substack{k\in V\setminus\{u,v\} \\ k\in\delta(u)}} t^v_{uk} + y_{uv} + y_{vu} \geq x_u + x_v - 1 \qquad \forall u \in V, v \in \delta(u) \tag{3.65}$$

$$x_k + y_{uv} + y_{vu} \geq 2t^k_{uv} \qquad \forall u \in V, v \in \delta(u), k \in V \tag{3.66}$$

$$x_k + y_{uv} + y_{vu} - 1 \leq t^k_{uv} + t^k_{vu} \qquad \forall u \in V, v \in \delta(u), k \in V \tag{3.67}$$

$$y_{uv} + y_{vu} \geq t^k_{uv} + t^k_{vu} \qquad \forall u \in V, v \in \delta(u), k \in V \tag{3.68}$$

$$x_k \geq t^k_{uv} + t^k_{vu} \qquad \forall u \in V, v \in \delta(u), k \in V \tag{3.69}$$

$$y_{uv} + y_{vu} \leq x_u \qquad \forall u \in V, v \in \delta(u) \tag{3.70}$$

$$\sum_{v\in\delta(u)} x_u \geq 1 + \sum_{v\in\delta(u)} y_{uv} \qquad \forall v \in V \tag{3.71}$$

$$\sum_{v \in \delta(u)} y_{uv} = \sum_{u \in V} x_u - 1 \tag{3.72}$$

$$\sum_{v \in \delta(u)} y_{uv} + y_{vu} \geq x_u \qquad \forall u \in V \tag{3.73}$$

$$t^u_{uv} = 0 \qquad \forall v \in V, u \in \delta(v) \tag{3.74}$$

(MFD-cont'd)

$$t^k_{uv} + t^k_{um} \leq 1 \qquad \begin{array}{l} \forall m \in V, v \in V \setminus \{m\}, \\ \text{k} \in V \setminus \{v, m\}, \\ \text{s.t.u} \in \delta(v), u \in \delta(m) \end{array} \tag{3.75}$$

$$x_u \in \{0, 1\} \qquad \forall u \in V \tag{3.76}$$

$$y_{uv} \in \{0, 1\} \qquad \forall u \in V,\ v \in \delta(u) \tag{3.77}$$

$$t^k_{uv} \in \{0, 1\} \qquad \forall u \in V,\ v \in \delta(u), k \in V \tag{3.78}$$

Although the feasible area of the MDSP and the DTP are the same, and the formulation used in [8] can be used, we have updated some of the constraints, and came up with a different formulation. In [8], they use Big-M values in the formulation. On the other hand, in our formulation, Big-M values do not appear.

3.5.2 Proof of correctness

Suppose a disconnected graph D results as a feasible solution. By (3.72) D contains a cycle. Then, at least for one (u, v) pair, we have the left-hand side of (3.64), 2. However, the right-hand side is 1 as u and v are in the solution. Thus, (3.64) eliminates the cycle, and leads to a contradiction.

Also, (MFD) does not exclude any dominating tree solutions while preventing the cycles. We know that the set of constraints (3.71) satisfy the domination property. So, assume a vertex set $D \subseteq S$ satisfying (3.66)-(3.75). Then, for the vertices in D, at least one of the incident edges is included in the dominating tree. (3.65) puts a lower bound only if both u and v are in D. In this case, either u and v are connected by an edge which means that y_{uv} is 1, or u and v are

connected via some vertex k, which means that t_{uv}^{k} is one for some $k \in D$. So the left-hand side of (3.65) becomes at least one, the constraint set is satisfied for all dominating trees.

For the constraint set (3.64), the right-hand side is one when $u, v \in D$. Then, the left-hand side is one for the same reason explained above for the constraint set (3.65). It puts an upper bound of $\frac{1}{2}$ when only one, 0 when none of u and v are in D. Since the left-hand side variables are binary, both cases restrict the left-hand side to zero. In those cases, where at least one of u, v is not in D, corresponding t and y values should be zero because of (3.69)-(3.70). Thus, it does not eliminate any dominating tree as well.

All in all, the proposed Martin's formulation for the DTP constructs a dominating tree.

CHAPTER 4

PRE-PROCESSING PROCEDURES

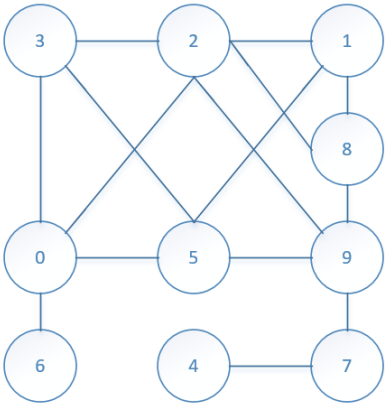
In this chapter, two pre-processing procedures, variable fixing (VF) and edge elimination (EE), applicable in general, and one pre-processing procedure, Big-M improvement (BMI), applicable only in (FF) are presented.

4.1 Variable Fixing

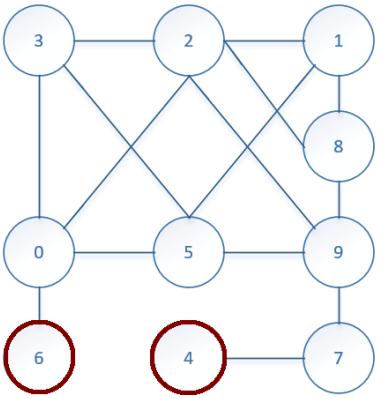
Given an edge-weighted connected graph G, we would like to solve the DTP. There may exist vertices of degree one. In G, if a vertex is of degree one, then it is pointless to include it in the optimal solution since including that vertex results in including the only edge incident to it, and therefore the only neighbor adjacent to it. However, it is sufficient to have the neighbor of v in the optimal solution without having v and satisfy domination with less cost.

Employing this property, we can fix x variables corresponding to vertices of degree one to zero, x variables associated with the vertices adjacent to one-degree vertices to one, and y variables for the edges that are incident to vertices of degree one to zero.

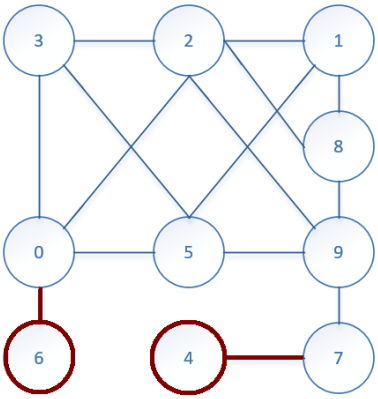
After that, we may apply one more procedure on the resulting graph. If the vertices whose x values are fixed to one, i.e., formerly adjacent to a one-degree vertex, have a degree of one after the removal of the one-degree vertices, then we need to include the only incident edge and the only adjacent vertex to the vertex in the optimal solution so that we can maintain connectivity and domination properties. We apply this fixing procedure until no candidates are left.



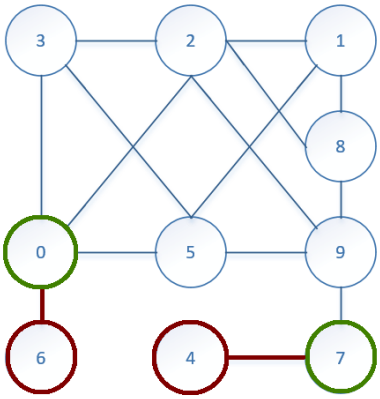
(a) Initial Graph



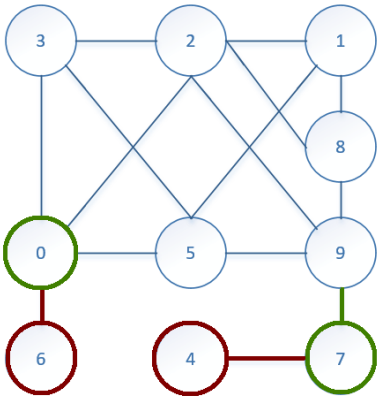
(b) Iteration 1



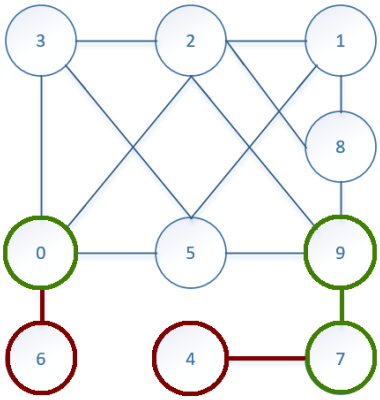
(c) Iteration 2



(d) Iteration 3



(e) Iteration 4



(f) Iteration 5

Figure 4.1: Variable Fixing Procedure

To illustrate, consider the example given in Figure 4.1. The variable fixing procedure applies as follows.

- Iteration 1: We look if there are any degree one vertices in the graph, and see that 4 and 6 are degree one vertices. We fix these vertices to 0.
- Iteration 2: The edges incident to vertices 4 and 6, that are $(0, 6)$ and $(4, 7)$, are fixed to 0 as well.
- Iteration 3: Since 4 and 6 have to be dominated, we fix the adjacent vertices, 0 and 7 to 1.
- Iteration 4: We again look for degree one vertices. Since 4 and 6 are removed from the graph, now 7 is a vertex of degree one. Since 7 is a degree one vertex, it should be connected to the only adjacent vertex, in order to form a connected graph. Therefore, the edge $(7, 9)$ is also fixed to 1.
- Iteration 5: Fixing $(7, 9)$ brings along fixing the vertex 9 to 1.
- Iteration 6: We again, look for degree one vertices, since there does not exist any, the algorithm stops.

Eventually, for this specific example, we fix vertices 4 and 6 to zero, 0, 7 and 9 to one; edges $(0, 6)$ and $(4, 7)$ to zero, $(7, 9)$ to one. By this way, we aim to improve the linear programming relaxation values.

Notice that, this procedure can also be applied to MCDSP, since the algorithm does not depend on the edge-weights.

4.2 Edge Elimination

In the DTP, the model minimizes the total weight on the edges. Therefore, it may prefer many edges if the cost associated to those set of edges is less than fewer edges with high cost. To be more specific, let u and v be two adjacent vertices, i.e., $(u, v) \in E$. If there is any other path in the graph connecting u

and v other than the edge (u, v) with less total cost, then the model prefers that path connecting u and v over the edge (u, v). There are two reasons for this decision:

1. The model tries to dominate the vertices that are not in the solution. Therefore, the more vertices included in the solution, the more vertices are tend to be dominated. By selecting the shortest path between two vertices, it includes more vertices than the case of selecting the direct edge.

2. It minimizes the cost over the network by connecting two vertices with less cost.

With the help of edge elimination (EE), we are able to decrease the number of edges and reduce the number of variables and constraints throughout the mathematical models constructed.

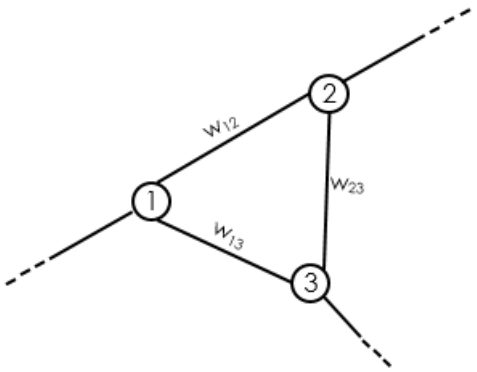


Figure 4.2: Edge Elimination Procedure

To illustrate the edge elimination procedure, consider the illustration in Figure 4.2. Suppose

$$w_{12} > w_{13} + w_{23}$$

There are two cases for the edge $(1, 2)$,

- Case 1: Both u and v are in the solution, therefore they should be connected. In other words, (u, v) edge may be used in the solution.

- Case 2: At least one of them is not in the solution, that is the edge (u, v) will not be used for sure.

So, eliminating the edge in case 2, does not affect the solution; but we need to analyze the first case. We assumed that w_{12} is strictly greater than the sum of w_{13} and w_{23}. To reduce the cost over the network, the model tends to include edges $(1, 3)$ and $(2, 3)$ in the solution, which makes the edge $(1, 2)$ redundant. Besides, by $(1, 2)$ and $(1, 3)$, we include 3 vertices in the solution, so that we have the chance to cover more vertices with less cost.

All in all, an edge becomes redundant and can be removed, if there exists a path with less total cost than the direct edge cost.

It is crucial to observe that this procedure is problem specific and cannot be used in MCDSP. This is due to the fact that MCDSP tries to minimize the number of vertices included in the solution, and by EE we actually increase the number of vertices which contradicts with the idea behind MCDSP.

4.3 Big-M Improvement for (FF)

The Big-M value proposed in (FF), was $|V| - 2$. However, further improvement is possible on that value. The procedure uses the distance information from source vertex to the corresponding edge. The details of the Big-M Improvement (BMI) procedure are as follows.

In order to include a vertex in the solution, the flow supplied from the source vertex has to pass through a number of edges and reach to the vertex. This number of edges is at least the number of edges on the shortest path to edge (u, v) from the source.

To obtain the shortest path to the edge (u, v), we need to calculate shortest paths to u and v from the source vertex. Then the minimum of these values gives us the shortest path to that edge. In other words, for each vertex u, shortest path,

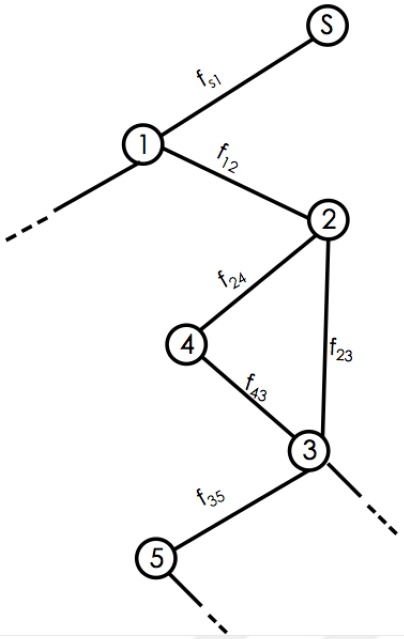


Figure 4.3: Big-M Improvement Procedure for (FF)

s_u, from the source vertex is calculated. For each edge (u, v),

$$A_{uv} = min\{s_u, s_v\}$$

is determined. This value represents that at least A_{uv} amount of flow will be dropped off until this edge in the case that it is included in the graph. So, we can decrease the Big-M value by A_{uv} in the constraint set (3.42), since the flow on this edge can be at most $|V| - 2 - A_{uv}$.

For instance, consider the graph in Figure 4.3. Here, the vertex S defines the source, and the remaining vertices are a subset of the graph. Assume that there are no other paths connecting pair of vertices included in this graph. That is to say $(1, 2)$ and $(3, 5)$ are cut-edges, and all the adjacent vertices to vertices 2 and 4 are shown in the illustration.

Let us consider the edge $(2, 3)$. The shortest path from the source vertex to vertex 2 follows $S - 1 - 2$, and the shortest path from the source vertex to vertex 3 follows $S - 1 - 2 - 3$. Then,

$$s_2 = 2 \quad s_3 = 3$$

which leads

$$A_{23} = min\{s_2, s_3\}$$
$$A_{23} = min\{2, 3\}$$
$$A_{23} = 2$$

So M_{23} value becomes $|V| - 4$, where $|V|$ is the number of vertices in the graph. To interpret, the maximum possible flow value that can pass through the edge $(2, 3)$ is $|V| - 4$, since it needs to distribute at least two flows to vertices 1 and 2.

With the help of BMI, we aim to improve the LP-relaxation of (FF). In the LP-relaxation upper bounds for flow values, f_{uv}, will be tighter. Thus, the solution space of the LP-relaxation will be closer to that of the MIP and the solution time is expected to decrease.

CHAPTER 5

OTHER SOLUTION APPROACHES

5.1 Iterative Branch-and-Cut Approach

Sub-tour elimination constraints included in (IPC) and (IPS) formulations increase the number of constraints exponentially. Therefore, it becomes impossible to enumerate all the constraints and solve the problem in an exact manner within a reasonable time limit. To overcome this difficulty, we propose an iterative branch-and-cut approach.

Basically, we remove the exponentially many sub-tour elimination constraints from the formulation, and solve the resulting problem (IPWS). According to the solution, we apply a separation procedure in order to determine the most violated sub-tour elimination constraint, which is then added to the problem. The procedure continues until no more violated sub-tour elimination constraint is found. The main idea of the algorithm for both formulations is the same and as follows:

Algorithm 1 Iterative Branch-and-Cut Approach

```
1: procedure IterativeBranch–and–CutApproach
2:     PR ⇐ IPWS
3:     while 1 do
4:         if there exists a violated sub-tour elimination constraint C then
5:             PR ⇐ PR ∪ C
6:         else
7:             break
```

In the following subsections, we give the details of the separation procedures applied to both for (IPC) and (IPS).

5.1.1 Classical IP formulation

Let us recall the sub-tour elimination constraints of (IPC),

$$\sum_{u,v\in S,(u,v)\in E} y_{uv} \leq |S| - 1 \qquad \forall S \subset V, S \neq \emptyset, S \neq V \qquad (5.1)$$

To find the most violated inequality we need to maximize $\sum_{u\in V,v\in\delta(u)} y_{uv} - |S|$. We can easily observe that if we can find a connected subgraph that is not a tree, then the subset corresponding to that component violates the corresponding sub-tour elimination constraint.

Claim: If we have a connected subgraph including a cycle, adding one more vertex connected to this connected subgraph, and enlarging it does not make the violation worse.

Proof: Let S_1 be a connected subgraph and v be a vertex in $V \in S_1$ such that v is adjacent to at least one vertex in S_1. Letting $S_2 = S_1 \cup \{v\}$, we have that

$$\sum_{u,v\in S_2,v\in\delta(u)} y_{uv} - |S_2| - \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} - |S_1| \geq 0$$

as $S_2 = S_1 + 1$ and $\sum_{u,v\in S_2,v\in\delta(u)} y_{uv} \geq \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} + 1$.

Thus, S_2 violates the constraint at least as much as S_1. Therefore, we may search for connected components while searching for the most violated inequalities.

Claim: If we have two independent connected components including cycles, combining them and enlarging the subset does not make the violation worse.

Proof: Suppose a solution has two independent connected components, S_1 and S_2. So, if

$$\sum_{u,v\in S_1\cup S_2, v\in\delta(u)} y_{uv} - (|S_1| + |S_2|) \geq \sum_{u,v\in S_1, v\in\delta(u)} y_{uv} - |S_1|$$

we may add constraints corresponding to $S_1 \cup S_2$ vertices.

$$\sum_{u,v\in S_1\cup S_2, v\in\delta(u)} y_{uv} - (|S_1| + |S_2|)$$
$$= \sum_{u,v\in S_1, v\in\delta(u)} y_{uv} - |S_1| + \sum_{u,v\in S_2, v\in\delta(u)} y_{uv} - |S_2|$$

holds true since the two connected components are independent.

Since $\sum_{u,v\in S_1, v\in\delta(u)} y_{uv} - |S_1| \geq 0$ and $\sum_{u,v\in S_2, v\in\delta(u)} y_{uv} - |S_2| \geq 0$, the inequality written for $S_1 \cup S_2$ does not worsen the violation.

Since the connected components include cycles, this inequality holds. Thus, combining two or more connected components including cycles does not worsen the violation.

5.1.2 Strengthened IP formulation

Let us recall the sub-tour elimination constraints of (IPS),

$$\sum_{u,v\in S,(u,v)\in E} y_{uv} \leq \frac{|S|-1}{|S|}\sum_{u\in S} x_u \quad \forall S \subset V, S \neq \emptyset, S \neq V \qquad (5.2)$$

To find the most violated inequality we need to maximize

$$\sum_{u,v\in S, v\in\delta(u)} y_{uv} - \frac{|S|-1}{|S|}\sum_{u\in S} x_u.$$

Claim: If we have a connected subgraph including a cycle, adding one more vertex connected to this connected subgraph, and enlarging it does not make the violation worse.

Proof: Suppose we have $S_1, S_2 \subset V$,corresponding to two connected subgraphs where $S_2 \supset S_1$. So, if

$$\sum_{u,v\in S_2,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_2|})\sum_{u\in S_2} x_u \geq \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_1|})\sum_{u\in S_1} x_u$$

then, we may add constraints corresponding to S_2 vertices.

$$\begin{aligned}
&\sum_{u,v\in S_2,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_2|})\sum_{u\in S_2} x_u \\
&= \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} + 1 + a - (1-\frac{1}{|S_1|+1})(\sum_{u\in S_1} x_u + 1)
\end{aligned}$$

where a is a non-negative value, as $S_2 \supset S_1$. In other words, since the vertex to be added to S_1 is connected to at least one of the vertices in S_1, there will be additional $a+1$ edges to $\sum_{u,v\in S_1,v\in\delta(u)} y_{uv}$.

$$\begin{aligned}
&= \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} + 1 + a - \Big(\sum_{u\in S_1} x_u + 1 - (\frac{1}{|S_1|+1})\sum_{u\in S_1} x_u - (\frac{1}{|S_1|+1})\Big) \\
&= \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} + 1 + a - (1-\frac{1}{|S_1|+1})\sum_{u\in S_1} x_u - 1 + (\frac{1}{|S_1|+1}) \\
&= \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_1|+1})\sum_{u\in S_1} x_u + a + (\frac{1}{|S_1|+1})
\end{aligned}$$

Then,

$$\sum_{u,v\in S_1,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_1|+1})\sum_{u\in S_1} x_u + a + (\frac{1}{|S_1|+1})$$

$$\stackrel{?}{\geq} \sum_{u,v\in S_1,v\in\delta(u)} y_{uv} - (1-\frac{1}{|S_1|})\sum_{u\in S_1} x_u$$

$$(\frac{|S_1|-1}{|S_1|} - \frac{|S_1|}{|S_1|+1})\sum_{u\in S_1} x_u + a + \frac{1}{|S_1|+1} \stackrel{?}{\geq} 0$$

$$a + \frac{1}{|S_1|+1} \stackrel{?}{\geq} (\frac{1}{|S_1|(|S_1|+1)})\sum_{u\in S_1} x_u$$

Since

$$\sum_{u \in S_1} x_u \leq |S_1|$$

the inequality holds true. Thus, S_2 violates a constraint at least as much as S_1. Therefore, we may search for connected components while searching for the most violated inequalities for the strengthened formulation as well.

Claim: If we have two independent connected components including cycles, combining them and enlarging does not make the violation worse.

Proof: Suppose that the sets S_1 and S_2 are the vertex sets of two connected components. So, if

$$\sum_{u,v \in S_1 \cup S_2, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1| + |S_2|}) \sum_{u \in S_1 \cup S_2} x_u \geq \sum_{u,v \in S_1, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1|}) \sum_{u \in S_1} x_u$$

then, we may add constraints corresponding to $S_1 \cup S_2$ vertices.

$$\sum_{u,v \in S_1, v \in \delta(u)} y_{uv} + \sum_{u,v \in S_2, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1| + |S_2|})(\sum_{u \in S_1} x_u + \sum_{u \in S_2} x_u) \overset{?}{\geq} \sum_{u,v \in S_1, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1|}) \sum_{u \in S_1} x_u$$

$$\sum_{u,v \in S_1 \cup S_2, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1| + |S_2|}) \sum_{u \in S_1 \cup S_2} x_u \overset{?}{\geq} \sum_{u,v \in S_1, v \in \delta(u)} y_{uv} - (1 - \frac{1}{|S_1|}) \sum_{u \in S_1} x_u$$

holds true since the two connected components are independent. With basic mathematical operations we get,

$$\sum_{u,v \in S_2, v \in \delta(u)} y_{uv} \geq |S_2|$$

Since S_2 includes cycles, this inequality holds. Thus, combining two connected components including cycles does not worsen the violation in the strengthened formulation as well.

5.2 Dynamic Branch-and-Cut Approach

In the literature branch-and-cut applications are studied for many problems. In this section, we will introduce, the branch-and-cut procedure applied to (FF), and detail the separation procedure applied to determine the cuts.

We satisfy connectivity with the help of the flow variables in the implementation of (FF). Thus, we do not have exponentially many constraints as in (IPC) or (IPS). However, having a mixed integer program results in fractional solutions in the branch-and-bound tree. At this step, one may add cuts to the nodes wisely so that some fractional solutions are not considered anymore. Thus, a separation problem arises at this point. With the application of this separation procedure, we are able to decrease the depth of the branch-and-bound tree and get the result faster.

In the branch-and-cut procedure, we are only allowed to separate fractional solutions, and the feasible area of the original problem should not change with the additional cuts. When we have fractional variables, we may have disconnected solution and cycles. Therefore, we may define the separation procedure as preventing cycles in the solution. To serve this aim, we may add the sub-tour elimination constraints stated in (IPC).

At this point it is crucial to find the most violated inequality so that we may cut more fractional solutions from the feasible region of the LP-relaxation. Let us recall the sub-tour elimination constraints introduced in (IPS),

$$\sum_{\substack{u,v \in S \\ v \in \delta(u)}} y_{uv} \leq \frac{|S|-1}{|S|} \sum_{u \in S} x_u \qquad\qquad S \subset V, S \neq \emptyset, S \neq V$$

In order to find the most violated inequality, we need to maximize the following

$$\sum_{\substack{u,v\in S\\ v\in\delta(u)}} y_{uv} - \sum_{u\in S} x_u - \frac{1}{|S|}\sum_{u\in S} x_u$$

and determine the set S resulting in the maximum violation. We define two sets of decision variables as follows:

$$t_u = \begin{cases} 1 & \text{if vertex } u \text{ is selected in the set S} \\ 0 & \text{otherwise} \end{cases} \qquad u \in V$$

$$z_{uv} = \begin{cases} 1 & \text{if edge } (u,v) \text{ is selected in the set S} \\ 0 & \text{otherwise} \end{cases} \qquad (u,v) \in E$$

One may notice that the definitions of newly introduced decision variables, t and z, have a similar meaning with the original x and y, respectively. Utilizing the current LP-relaxation solution to the problem, we can decide on which variables to include in the set S with the following mathematical model as a heuristic.

(SEP1)

$$\max \sum_{u\in S, v\in\delta(u)} z_{uv}y_{uv} - \sum_{u\in V} t_u x_u$$

$$\text{s.to} \quad t_u \geq z_{uv} \qquad \forall u \in S, v \in \delta(u) \qquad (5.3)$$

$$t_v \geq z_{uv} \qquad \forall u \in S, v \in \delta(u) \qquad (5.4)$$

$$0 \leq t_u \leq 1 \qquad \forall u \in V \qquad (5.5)$$

$$0 \leq z_{uv} \leq 1 \qquad \forall u \in S, v \in \delta(u) \qquad (5.6)$$

(SEP1) model aims to select a set S such that $\sum_{u,v\in S v\in\delta(u)} y_{uv} - \sum_{u\in S} x_u$ is mostly violated. Therefore, for the vertices that have the t value 1, we can construct the corresponding sub-tour elimination constraint (SEC). Determined SEC is added as a cut to the problem, and the branch-and-bound procedure continues with the added cut. Note that this separation procedure ignores the part $\frac{1}{|S|}\sum_{u\in S} x_u$.

Overview of the algorithm can be seen below in Algorithm 2:

Algorithm 2 Dynamic Branch-and-Cut Approach

procedure DynamicBranch–and–CutApproach
  Solve relaxation of (FF)
  while 1 **do**
    Determine x and y values for the current node
    Solve (SEP1) with given x and y
    if $\max \sum_{u \in S, v \in \delta(u)} z_{uv} y_{uv} - \sum_{u \in V} t_u x_u > -\frac{1}{\sum_{u \in V} t_u} \sum_{u \in S} x_u$ **then**
      add $\sum_{\substack{u,v \in S \\ v \in \delta(u)}} y_{uv} \leq \frac{|S|-1}{|S|} \sum_{u \in S} x_u$
    else
      break

It also worths to note that we obtain integral solution when we solve the LP-relaxation of (SEP1). Suppose we have a solution $\bar{t}$ and $\bar{z}$, including some fractional values for some u and v. Selecting an ϵ small enough, we may add and subtract it from that fractional value $\bar{t}_u$ and $\bar{z}_{uv}$, and still remain in the feasible area P. Since, we can write these solutions as convex combinations of other solutions, these do not exist as extreme points. So, we are ensured to get binary values for t and z variables.

CHAPTER 6

COMPUTATIONAL EXPERIMENTS

In this chapter, we will perform computational experiments on the instances provided in the literature. First, we will conduct preliminary computational experiments to decide on the pre-processing procedures to be used. Then, with the determined sets of pre-processing procedures we will provide the detailed computational experiments. With the help of these experiments, we aim to comment on the strengths of the integer programming formulations and other solution approaches.

6.1 Preliminary Computational Experiments

In the literature, there exist two data sets, created by Sundar and Singh [23] and Dražić, Čangalović, and Kovačević-Vujčić [5] The former one includes instances with up to 500 vertices while the latter one with up to 300 vertices. In the small size instances of [5], the number of vertices vary from 10 to 20, while the large size instances vary from 100 to 300. On the other hand, instances in [23] include 50 to 500 vertices. Moreover, the test instances generated in [5] reflect sparser networks than those of [23].

In [5], the authors generate the instances in a random manner. The number of vertices and edges are predetermined, but the communication between those vertices are determined randomly. Moreover, these edges are created such that the graph will not be disconnected, i.e., it includes only one connected component which is the graph itself. If the randomly generated graph is detected to

be disconnected, then it is ignored and a new generation process starts. Besides, the weight values are also randomly generated. A real value between [1,10] is selected, and assigned to the edge. The related data can be found on `http://poincare.matf.bg.ac.rs/zdrazic/dtp`.

In the generation of the instances in [23], the authors use a more realistic approach while determining the edge weights. Predetermined number of vertices are randomly distributed in a 500m×500m area, and the transmission range of the vertices is assumed to be 100m. That is, the sensors can communicate if the distance between them is less than or equal to 100m. In [3], they also consider 125m and 150m for the transmission range parameter. The data sets can be found on `http://dcis.uohyd.ernet.in/~alokcs/dtp.zip`. Note that as of September 3^{rd}, 2016 the given link does not work.

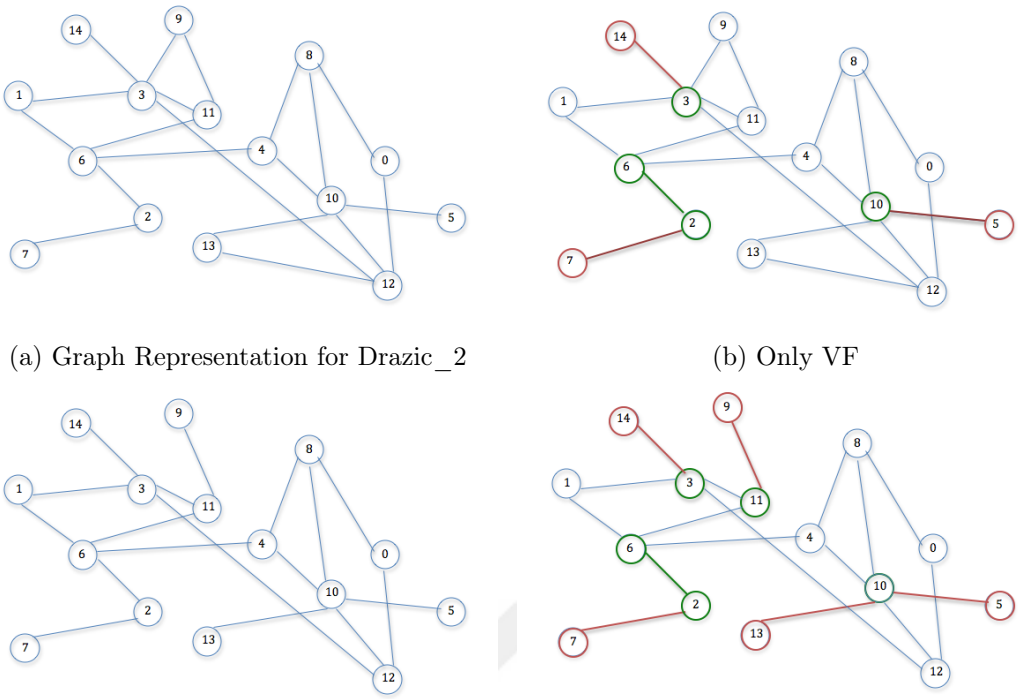
We perform the computational experiments regarding the mathematical programming formulations in AMPL environment, which is a modeling tool, and we use CPLEX 12.6 as the optimizer. Runs are held on an Intel(R) Core(TM) i7-4790 CPU @3.60GHz 8.00 GB DDR3 computer. Branch-and-cut applications are implemented using NetBeans IDE 8.1 in Java programming language. Again, CPLEX 12.6 is used as the optimizer in Java applications.

We proposed three pre-processing procedures, VF, EE and BMI in Chapter 4. In this chapter, we will investigate the impacts of these procedures in preliminary computational experiments.

Two instances from [5], *dtp*_10_15_0 and *dtp*_15_20_0 and 2 instances from [3], 50_2 from range 100 and 100_2 from range 150, are selected arbitrarily. These instances reflect sparsity and density. From now on, we will refer *dtp*_10_15_0 as Drazic_1, *dtp*_15_20_0 as Drazic_2, 50_2 from range 100 as Alok_1 and 100_2 from range 150 as Alok_2.

For (IPC), (IPS), (CF), (MCFF), and (MFD), only variable fixing and edge elimination procedures are applicable. For those, we will examine the effect of these procedures.

One may observe that VF tends to fix more variables when the vertices have



(a) Graph Representation for Drazic_2

(b) Only VF

(c) Only EE

(d) VF applied after EE

Figure 6.1: Edge Elimination and Variable Fixing Procedures

small degree. In other words, when there are fewer edges, i.e., when the graph is sparser. Therefore, we expect that applying VF after EE results in better solutions.

At this point, we need to analyze whether using only VF or only EE is better than using the combination of them. To illustrate the situation, consider the graph given in Figure 6.1a. It also corresponds to the data in Drazic_2. Here, the red edges and vertices refer to edges and vertices fixed to zero; green edges and vertices refer to edges and vertices fixed to one.

When only VF procedure is used, Figure 6.1b, we still have 15 vertices and 20 edges, where 7 of the vertices and 4 of the edges are fixed to a value with the procedure.

For only EE procedure, Figure 6.1c, we see that the number of edges decreases from 20 to 18, which leads to less number of constraints. Therefore, the model becomes simpler.

In Figure 6.1d, we illustrate the combination of these procedures. First EE is applied on the graph, then with VF we determine the vertices that can be fixed to a value. With this combination we end up in with a 15-vertex 18-edge graph having 10 vertices and 6 edges fixed to a binary value.

We can infer that the model gives better LP-relaxation values for the combination of those. To support this, we provide the LP-relaxation values for (MFD) corresponding to those in Table 6.1. For Alok_2 we refer to the best solution found in the literature as the optimal value.

As seen, when we apply EE and VF procedures together, the model gets tighter and solution space to the relaxed problem approximates the IP solution space better. Therefore, for further experiments we will use both pre-processing procedures to get better results.

We deduced that using EE-VF together results in tighter solution space. For (FF), one more pre-processing procedure, BMI, can be applied. Thus, we need to decide on whether we will use only EE-VF, only BMI or the combination of these three procedures.

In (FF), smaller Big-M values are desired to tighten the solution space. The Big-M value determined by BMI is related to the distance of the edge from the source vertex. As the distance increases between an edge and the source vertex, Big-M value decreases. In other words, Big-M value and the distance of the shortest path between source vertex and the related edge are inversely proportional.

Therefore, we need to have distant edges, no short-cuts from the source vertex to the related edge to decrease the Big-M value. This is possible when the graph is sparse. Therefore, we expect to have better results after EE procedure. Since VF does not eliminate any vertices or edges, the order of EE-VF-BMI does not differ from EE-BMI-VF. Optimal values and LP-relaxation values for only EE-VF, only BMI, and EE-VF-BMI procedures can be seen in Table 6.2.

Looking to the results, we see that application of only EE-VF or only BMI does not dominate each other. For instance, in Drazic_2, LP-relaxation value

regarding BMI is smaller than that regarding EE-VF. On the other hand, for Alok_2, the case is vice versa. However, it is sure that the combination of these three procedures constitute a tighter feasible area, and results in better LP-relaxation values.

On the other hand, pre-processing times of the instances increases with the size of instance. For Drazic_1 and Drazic_2, the total pre-processing time is less than a second; where for Alok_1 and Alok_2 it increases to 5 and 7 seconds. However, since these procedures are applied one-time for an instance, we do not consider them in the running time.

Regarding to the results, for the full computational experiments, we will use EE-VF sequence for (IPC), (IPS), (CF), (MCFF), and (MFD); and EE-VF-BMI sequence for (FF).

Table 6.1: LP-relaxation values to (MFD) for the EE, VF and EE-VF

Instance ID	Opt. value	None	Only EE	Only VF	EE-VF
Drazic_1	5.89	5.89	5.89	5.89	5.89
Drazic_2	18.87	18.46	18.46	18.46	18.87
Alok_1	1340.44	624.94	627.49	627.49	634.94
Alok_2	657.35*	68.04	97.33	68.04	97.75

*The best solution value found in the literature.

Table 6.2: LP-relaxation values to (FF) for the EE-VF, BMI and EE-VF-BMI

Instance ID	Opt. value	None	Only EE-VF	Only BMI	EE-VF-BMI
Drazic_1	5.89	4.47	4.53	4.53	4.53
Drazic_2	18.87	9.42	14.71	11.61	14.98
Alok_1	1340.44	446.04	460.63	458.14	462.57
Alok_2	657.35*	47.90	47.90	48.07	52.78

*The best solution value found in the literature.

6.2 Detailed Computational Experiments

6.2.1 IP formulations

In Chapter 6.1, we introduced the data sets and determined the procedure sequence that will be used in the detailed computational experiments. So,

- For (IPC), (IPS), (CF), (MCFF), and (MFD); first EE, then VF is applied.
- For (FF); EE, VF and BMI are applied sequentially.

The proposed mathematical models differ both by logic behind them, and the number of decision variables and constraints. Below, we identify the complexity of the models by means of the number of decision variables and constraints.

In (IPC), we have $|V|+|E|$ number of binary decision variables. Constraint sets (3.1), (3.3) and (3.4) put in $|E|$, 1 and $|V|-2$ number of constraints, respectively. Moreover, (3.2) contributes to the number of constraints by $\sim 2^{|V|}$, which leads a total number of constraints $|E|+|V|+2^{|V|}$.

For (IPS), the number of binary decision variables does not differ from (IPC), but separating constraint set (3.1) into two, the number of constraints increases by $|E|$, and becomes $2|E|+|V|+2^{|V|}$. Since (CF) is only a variation of (IPS), and includes as many constraints for the sub-tour elimination as (IPS), the number of decision variables and constraints are same with (IPS).

In (FF), we confront additional $|E|$ number of continuous decision variables corresponding to flow values. Constraint sets (3.36)-(3.37) contribute $2(|V|-1)$ many constraints where (3.41)-(3.44) contribute $4(|E|-1)$. Moreover, with (3.38)-(3.39), (3.40), (3.45)-(3.46) additional $4+m$ constraints are added to the model, where m is the number of vertices connected to the source vertex. Consequently, (FF) includes $4|E|+2|V|+m-2$ constraints, $|V|+|E|$ number of binary and $|E|$ number of continuous decision variables.

In (MCFF), (3.52) contributes with $(|V|-1)(|V|-2)$, (3.57) with $|E|-m$ and (3.58) with $(|V|-1)|E|$ number of constraints. We have $4|V|$ number of

constraints resulting from the rest of the constraints. In total, we have $|V|(|V|+|E|+1)+m+2$ number of constraints, where m is the number of adjacent vertices incident to the sink vertex. Moreover, we have $|V|+|E|+|V||E|$ number of binary decision variables.

Lastly, (MFD) presents a more complicated model with $|V|+|E|+|V||E|$ number of binary decision variables as in (MCFF). With (3.64), (3.65), (3.70) and (3.75), we have $4|E|$; with (3.72)-(3.74), we have $2|V|+1$; with (3.66)-(3.69) and (3.76), we have $5|V||E|$ number of constraints. In total we get $5|V||E|+4|E|+2|V|+1$ number of constraints.

One may find the summary of the number of constraints and decision variables in Table 6.3.

Table 6.3: Number of DVs and constraints for proposed formulations

	# Binary DVs	# Continuous DVs	∼# Constraints
IPC	|E|+|V|	-	|E|+|V|+$2^{\|V\|}$
IPS	|E|+|V|	-	2|E|+|V|+$2^{\|V\|}$
CF	|E|+|V|	-	2|E|+|V|+$2^{\|V\|}$
FF	|E|+|V|	|E|	4|E|+2|V|
MCFF	|E|+|V|+|V||E|	-	|V|+|V||E|+|V|^2
MFD	|E|+|V|+|V||E|	-	4|E|+2|V|+5|V||E|

*m is the number of adjacent vertices to the source/sink vertex.

6.2.1.1 Results on the dataset by Dražić, Čangalović, and Kovačević-Vujčić [5]

The data set used in [5] includes 15 small size, and 18 large size instances. In their work, they are able to solve the small size instances using variable neighborhood search (VNS) at most within 0.20 seconds using an Intel Core i7-4702MQ 2.2 GHz with 4 GB RAM under Windows XP operating system (Table A.2).

However, the algorithm becomes rather inefficient when the size of the instance increases. They run the algorithm for 600 seconds, make 20 replications for

each instance, and record the resulting solution values. For larger instances, the resulting solution values for each instance differs more from replication to replication, i.e., the standard deviation of the solution values becomes greater.

In the following figure (Figure 6.2) one can see the relative running times in seconds for the related formulations. Here, (IPC), (IPS), and (CF) need to run significantly more than the VNS algorithm, (FF), (MCFF), and (MFD) formulations to reach the optimal solution (Figure 6.2a). So, let us compare VNS algorithm, (FF), (MCFF), and (MFD) running times in detail to address the differences between them (Figure 6.2b).

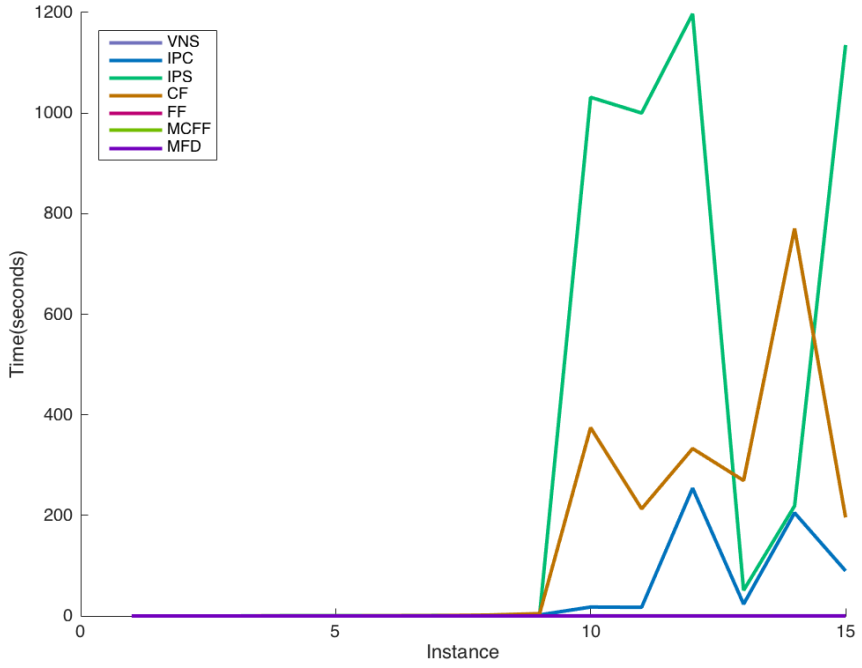
We see that for small size instances (FF) and (MFD) perform better than VNS algorithm and (MCFF). Furthermore, VNS is never better than (FF) and (MFD).

Herein, we may look for the LP-relaxation values of small size instances to understand the strength relation between the formulations. The exact values of the relaxations for small size instances can be found in Appendix A, in Table A.3. Since (IPC), (IPS), and (CF) are not able to solve instances with more than 50 vertices, their LP-relaxation values for large size instances are not available; but, one may find the values for (FF), (MCFF), and (MFD) in Appendix A, in Tables A.4, A.5, and A.7, respectively.

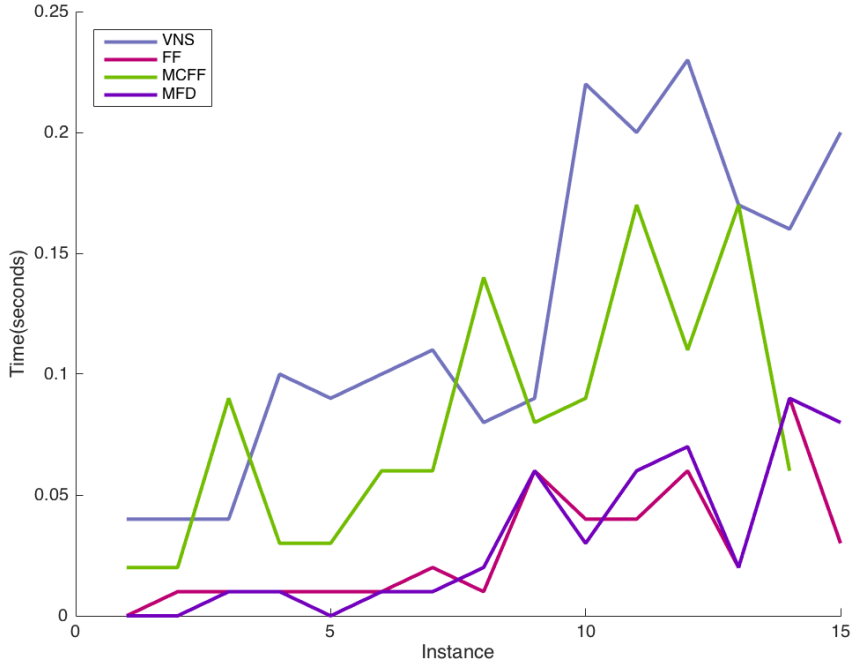
Let us examine the LP-relaxation values for small size instances (Figure 6.3).

Here, we can see that, (MCFF) is the one that converges to the optimal value the most. In 7 of the 15 instances, it achieves to the optimal value with the LP-relaxation. Afterwards, the (MFD) comes with 5 instances achieving the optimal solution. (IPS) achieves the optimal solution for only 1 instance, but it can have relaxed values better than (MFD) for some instances. Therefore, we cannot say anything about the strength between these two formulations.

Comparing the strengths, (FF) comes after (MCFF), (MFD), and (IPS). However, it is not able to catch the optimal values for any of the instances. But, still it stays stronger than (IPC) and (CF) with lower run-time values.



(a) VNS algorithm, IPC, IPS, CF, FF, MCFF, MFD



(b) VNS algorithm, FF, MCFF, MFD

Figure 6.2: Time comparison for small size instances provided in [5]

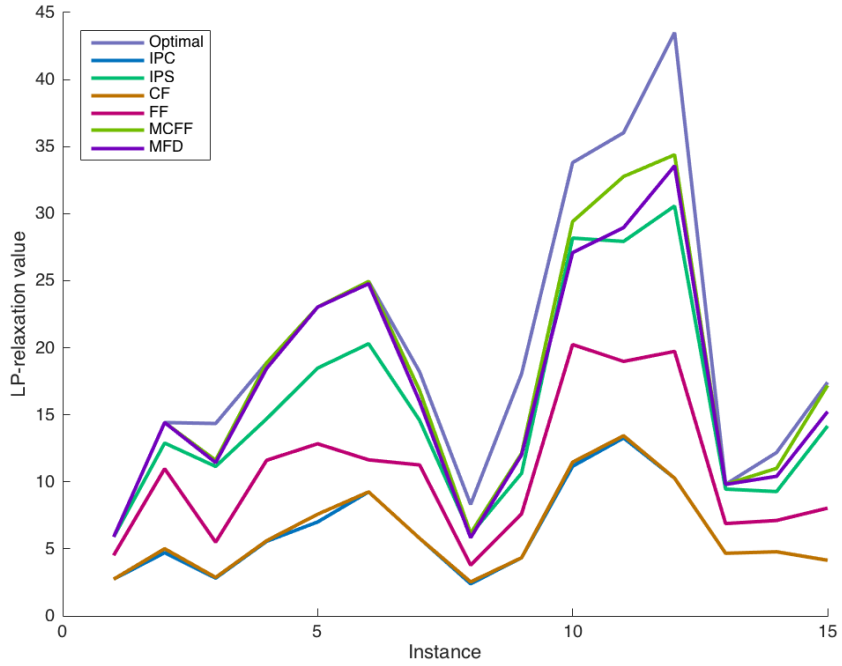


Figure 6.3: LP-relaxation values for small size instances provided in [5]

The picture changes when we get to the large size instances. (IPC), (IPS) and (CF) become unable to solve the system because of exponentially many constraints. Therefore, the larger size instances are solved with (MCFF), (MFD), and (FF).

In the work [5], the large size instances are run with VNS algorithm for 600 seconds and the obtained solution is reported as the best solution. Therefore, the algorithm is not able to achieve optimal solutions for some instances. However, with our integer programming formulations, we are able to reach the optimal solution values of most of the instances in 600 seconds.

In our computational experiment, we run our models for 10800 seconds, i.e., 3 hours.

Consider the (FF), where the detailed run-time results are given in Table A.4. We can observe that excluding 5 instances, the model reaches the optimal solution in 600 seconds. Moreover, (MCFF) behaves similar to (FF) and it finds the optimal values in 600 seconds except 4 instances. On the other hand, (MFD)

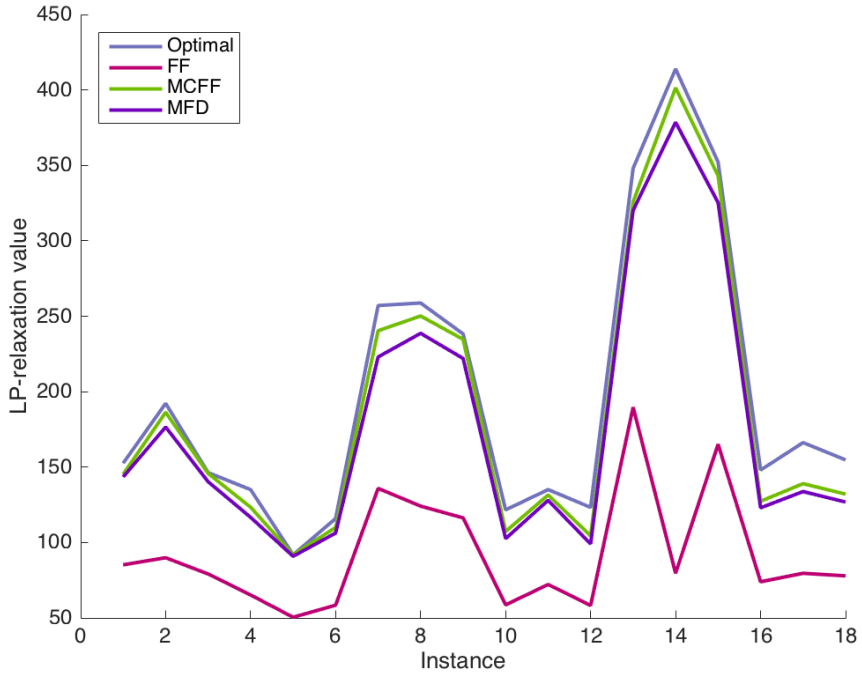


Figure 6.4: LP-relaxation values for large size instances provided in [5]

does not perform well with large size instances. Other than the first 6 instances, which corresponds to instances with 100 vertices, the model reaches the time limit, with a nonzero optimality gap.

It is also worth to examine LP-relaxation values for large size instances, since they provide promising results. In Figure 6.4, one may see the relation between the LP-relaxation values of three mathematical models and the optimal value. (FF) always stays behind (MCFF) and (MFD), as in small size instances. Moreover, (MCFF) is slightly better than (MFD). Nevertheless, (MCFF) spends more time for even the LP-relaxation solution than (MFD). Therefore, this trade-off can be analyzed for the application areas. If solution time is much more important, then (MFD) can be utilized to save time.

Actually, this result is not surprising since the flow formulation uses $|V| + |E|$ number of binary and $|E|$ number of continuous decision variables, where (MCFF) and (MFD) uses $|V|+|E|+|V||E|$ number of binary decision variables. Therefore, (FF) is expected to be weaker but faster rather than (MCFF) and (MFD).

All in all, we observe that (FF) provides rather reasonable running times both for IP and LP-relaxation, where (MCFF) and (MFD) dominates the others by means of strength. Moreover, although (MFD) performs well in small size instances, as the instance gets larger, the number of decision variables lead to increase in run-time.

In [5], the authors are able to solve the small size instances using CPLEX, and prove that the found values by VNS algorithm are optimal values. However, they lack to prove the large size instances because of memory limitation of CPLEX. With the introduced formulations, we are able to prove that the first 6 large size instances are indeed the optimal solutions to the related instances. Moreover, we improve the results found by VNS algorithm for the remaining large size instances. In other words, we provide exact results for most of the large size instances. For the ones, that result with optimality gap non-zero, we are still able to improve the solutions provided in [5]. For a summary of those improvements one may refer to Table A.4 in Appendix A.

6.2.1.2 Dataset of Chaurasia and Singh [3]

In the test instances used in [3], the number of vertices and edges are as in Table 6.4. Here the instance ID's are arranged as the first number referring to the number of vertices, and the second number to the ordinal number of the instance data. The number of edges differs from range to range because of the definition. Moreover, one may see the number of reduced edges, number of fixed vertices, and number of fixed edges in Table 6.5.

The creation of the data in [3] is as follows: The predetermined number of vertices are deployed to a 500m×500m area randomly. Then, the vertices are connected to each other if Euclidean distance between them does not exceed the considered range value. For instance, for the data set 50_1 Range 125, they deploy 50 vertices to the area randomly, and connect the vertices if the corresponding distance between them is less than or equal to 125m. Moreover, when an instance is created, i.e., vertices deployed to the area, three instances for three range values are created on the same orientation. Therefore, as the range

value increases we see an increase in the number of edges since more vertices are likely to be connected when the range is large.

Table 6.4: Number of edges for the instances in the data set provided in [3]

Instance	Range 100	Range 125	Range 150
50_1	121	194	276
50_2	118	192	262
50_3	126	188	269
100_1	535	798	1041
100_2	526	781	1078
100_3	481	726	1013
200_1	2188	3244	4345
200_2	2147	3221	4446
200_3	2069	3090	4246
300_1	4983	7406	10039
300_2	4737	7008	9693
300_3	4577	6841	9418
400_1	8738	12958	17629
400_2	8314	12419	17058
400_3	8109	11942	16340
500_1	13716	20377	27720
500_2	13069	19374	26676
500_3	12681	18791	25814

The formulations (IPC), (IPS), and (CF) lack of generating the constraints regarding to sub-tour elimination because of memory limitation of CPLEX. Besides, except for the first 6 instances, which corresponds to the instances with 50 and 100 vertices, we either get out of memory error, or we exceed the time limit, which is 3 hours, with no integer solution, or we stay in more than 90% optimality gap after 3 hours. Thus, in this section, we will examine the behavior of (FF), (MCFF) and (MFD), for the first 6 instances only.

The instances in [3] are solved using heuristic and meta-heuristic methods. For

Table 6.5: Number of edges eliminated, number of vertices fixed, and number of edges fixed for the instances in the data set provided in [3]

Instance	Range 100			Range 125			Range 150		
ID	#EE	#VeF	#EF	#EE	#VeF	#EF	#EE	#VeF	#EF
50_1	0	2	1	0	0	0	0	0	0
50_2	1	6	3	1	0	0	1	0	0
50_3	0	7	4	0	0	0	0	0	0
100_1	0	2	1	5	0	0	8	0	0
100_2	2	0	0	5	0	0	6	0	0
100_3	0	0	0	1	0	0	2	0	0
200_1	11	0	0	33	0	0	44	0	0
200_2	15	0	0	32	0	0	47	0	0
200_3	12	0	0	16	0	0	29	0	0
300_1	36	0	0	95	0	0	155	0	0
300_2	36	0	0	89	0	0	140	0	0
300_3	35	0	0	69	0	0	115	0	0
400_1	98	0	0	234	0	0	355	0	0
400_2	87	0	0	190	0	0	315	0	0
400_3	82	0	0	162	0	0	277	0	0
500_1	214	0	0	448	0	0	683	0	0
500_2	178	0	0	389	0	0	626	0	0
500_3	158	0	0	335	0	0	565	0	0

meta-heuristic approaches they get 20 replications for each instance. In some of the instances, they report the standard deviation in 20 replications as zero.

However, as in [5], they are not able to prove that the obtained results provide the optimal solution for sure. With the help of formulations introduced in this study, we prove that some of the best solutions provided as indeed optimal. One may see the related numeric results for the solutions in Appendix B, in Tables B.2, B.3, and B.4.

Moreover, in Figure 6.5, we see that (MCFF) provides better LP-relaxation

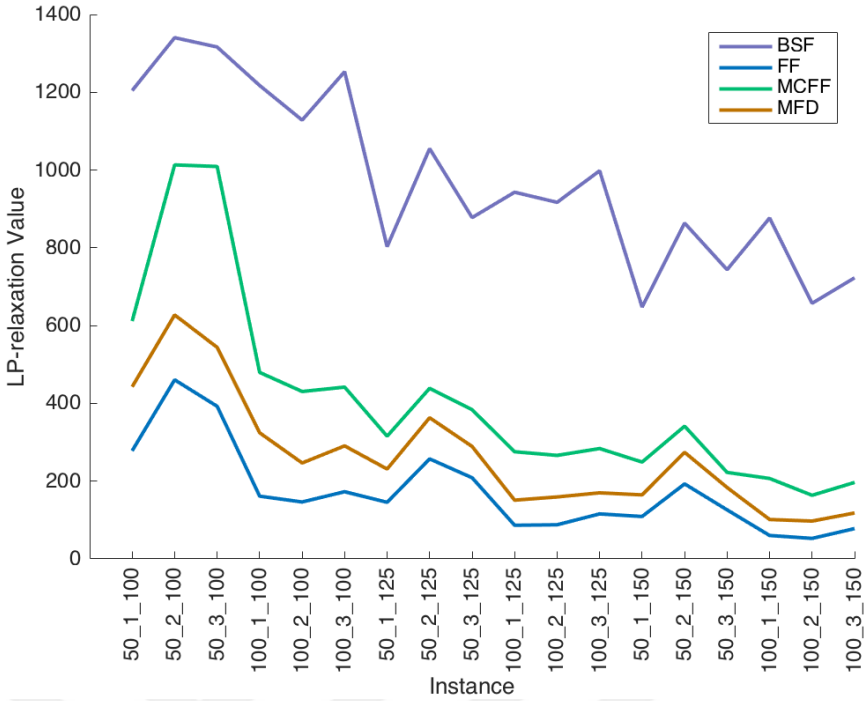


Figure 6.5: LP-relaxation values for the instances provided in [3]

values for the problem, where (FF) stays the worst. Here, BSF provides the best solution values found in the literature. Recalling the results in Section 6.2.1.1, we observe that the behavior of the formulations are consistent.

6.2.2 Iterative branch-and-cut

Although the IP formulations provide better LP-relaxation values regarding to IPS, we see that the computational times differ much, especially when the number of vertices is greater than 100. It may not be practical and desirable to wait for 3 hours and obtain solution values with optimality gap. Therefore, proposed iterative branch-and-cut approach will be discussed in this section.

Also, please note that because of the memory limitations, the instances more than 100 vertices could not be examined. The algorithm related to this approach is given in Section 5.1. Hereby, we provide the corresponding results.

Below in Table 6.6, one may see the number of iterations needed to achieve the optimal solution. As shown in Section 5.1, when we find more than one

connected component in the current solution, adding the sub-tour elimination constraint corresponding to all of those, results in at least as good as adding only one of them. For instance, consider the instance *dtp*_20_30_2. When the reduced model is solved, and the most violated inequality corresponding to a single connected component is added to the model, it took 9 iterations to solve the problem. However, when we add the combination of those connected components, we reach to optimality in one iteration.

Table 6.6: Results of iterative branch-and-cut for small size instances provided in [5]

	Add the most violated		Add all	
Instance ID	# iterations	Time (sec)	# iterations	Time (sec)
dtp_10_15_0	1	<0.01	1	<0.01
dtp_10_15_1	1	<0.01	1	<0.01
dtp_10_15_2	3	<0.01	1	<0.01
dtp_15_20_0	1	<0.01	1	<0.01
dtp_15_20_1	1	<0.01	1	<0.01
dtp_15_20_2	1	<0.01	1	<0.01
dtp_15_30_0	3	<0.01	3	<0.01
dtp_15_30_1	1	<0.01	1	<0.01
dtp_15_30_2	5	<0.01	5	<0.01
dtp_20_30_0	1	<0.01	1	<0.01
dtp_20_30_1	5	<0.01	1	<0.01
dtp_20_30_2	9	<0.01	1	<0.01
dtp_20_50_0	1	<0.01	1	<0.01
dtp_20_50_1	1	<0.01	1	<0.01
dtp_20_50_2	4	<0.01	4	<0.01

For larger size instances of [5], we are able to solve the first 6 instances because of the memory limitations. In these 6 instances, we achieve the optimal solution in 3 hours for 5 of the instances. For *dtp*_100_200_0, we cannot obtain the optimal solution in 3 hours. But, for the others we attain the optimal solutions. The exact time and solution values are provided in Table 6.8.

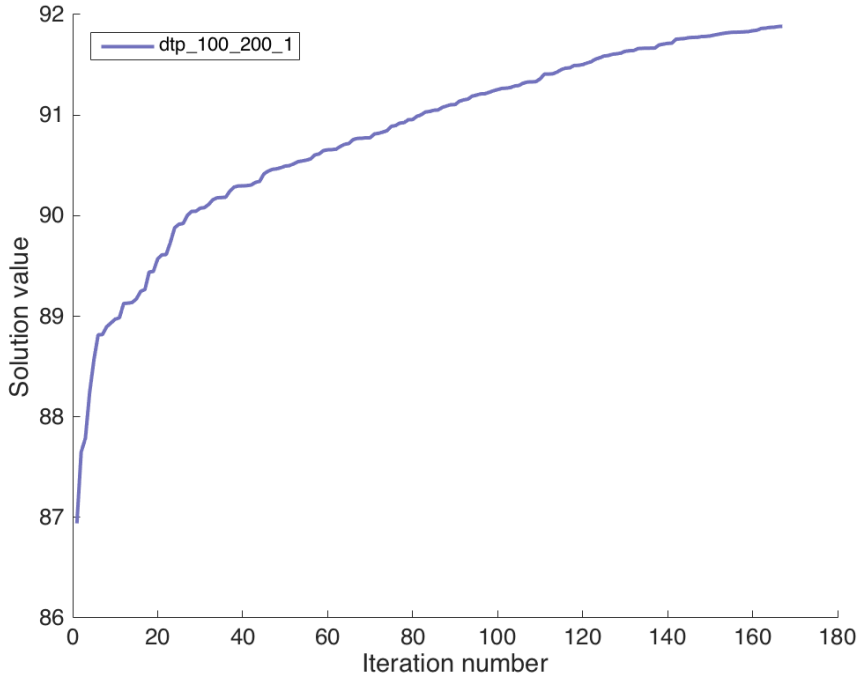


Figure 6.6: Progress of the objective function value when iterative branch-and-cut is applied on the instance dtp_100_200_1

To better investigate the approximation to the optimal value, one may refer to Figure 6.6. As seen, after some value, the optimal value starts to improve less.

However, it also worths to note that the starting value is so close to the optimal value. In other instances, we confront with the same situation as well. So, by using a stronger formulation to start with, the iterations may improve the solution value faster.

In Table 6.7 one may see the results regarding the instances provided in [3]. Here the values regarding $Time$ is in seconds. Since this set of instances reflects denser networks than the instances in [5], the optimal solutions are not available within provided time. Therefore, in Table 6.7, we report the initial solution value, that is the solution to the model without sub-tour elimination constraints, and the last solution value for the related number of iterations. As in the large size instances provided in [5], the objective function improves slower as the iteration number increases. This behavior is due to the size and density of the instances.

Table 6.7: Results of iterative branch-and-cut for instances provided in [3]

Instance ID	Initial solution	Last recorded solution	# iterations
Rng100 50_1	590.49	965.77	152
Rng100 50_2	726.99	969.78	223
Rng100 50_3	702.57	995.23	186
Rng100 100_1	529.92	673.61	23
Rng100 100_2	403.63	576.21	26
Rng100 100_3	420.50	668.22	18
Rng125 50_1	341.45	629.53	280
Rng125 50_2	639.18	819.32	168
Rng125 50_3	412.64	720.89	189
Rng125 100_1	306.07	419.58	46
Rng125 100_2	274.44	418.87	37
Rng125 100_3	256.32	462.2	19
Rng150 50_1	281.90	506.97	112
Rng150 50_2	476.5	658.54	107
Rng150 50_3	320.12	547.34	93
Rng150 100_1	243.66	334.58	28
Rng150 100_2	195.29	278.46	30
Rng150 100_3	167.24	313.28	18

Table 6.8: Results of iterative branch-and-cut for large size instances provided in [5]

Instance ID	Solution value	# iterations	Time (sec)
dtp_100_150_0	152.57	1017	236
dtp_100_150_1	192.21	1175	6595
dtp_100_150_2	146.34	485	236
dtp_100_200_0	130.58*	1138	10800
dtp_100_200_1	91.88	167	49
dtp_100_200_2	115.93	1253	8388

*The solution value with the specified number of iterations.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

Wireless Sensor Networks are getting more popular due to their use of area. In health-care, military, environmental applications they are widely used to gather data. However, since there are many of them deployed to the application area, it is hard to detect the places of them on a break-down and maintain the system. Therefore, they are required to transmit the data they gather in an efficient way. Thus, we aim to construct a backbone for their communication protocols, and make the system work in an efficient way.

This problem can be solved using the so called Dominating Tree Problem. This problem aims to find an energy-efficient backbone to the WSN by considering the communication difficulty between the sensors. Moreover, the system should be able to communicate with each other and the system should not be disconnected. Lastly, the sensors not in the backbone should be able to communicate with at least one of the sensors lying in the backbone.

In this work, the DTP is analyzed in detail. The problem makes some assumptions on the network which are:

- The cost of the communication is known for sure.
- The location of the sensors is known for sure.
- The system is 100% reliable.

However, it is known for sure that a system cannot be 100% reliable. So, when

a backbone is constructed, even though it is the most cost-efficient one, it may break-down for some reason, such as lack of battery power, or environmental effects. This brings us to the motivation behind the problem we study. With the solutions to the DTP, we aim to analyze the trade-off between the reliability of a system and the cost. So, under perfect conditions, solution to the DTP is actually the cost of the network. In other words, we see that the cost cannot be less than the solution of the DTP even in a 100% reliable network.

Throughout this thesis, we proposed integer programming formulations to the DTP. The classical IP formulation provided by [21] is strengthened and a cutset formulation is provided. Apart from these, a single-commodity and a multi-commodity integer programming formulations are presented. Lastly, Martin's formulation is modified for the DTP inspiring from [7].

Moreover, three pre-processing procedures are addressed, vertex fixing, edge elimination, and Big-M improvement. These procedures aim to decrease the number of decision variables, and the corresponding values are provided in Chapter 4.

During the computational studies, the instances provided in [5] and [3] are used. These five integer programming formulations are analyzed according to LP-relaxation values and running times. According to results, (MCFF) is seen to be the strongest among all.

Furthermore, two branch-and-cut procedures are studied on the problem. Iterative branch-and-cut algorithm is seen to be inefficient on the large-size problems, and the application of dynamic branch-and-cut algorithm is left for further studies.

Because of the wide application area of WSNs, we are motivated with a number of future research directions as follows.

- Exact separation on the DTP can be examined.
- Because of the nature of the problem, Benders' decomposition can be used to solve it.

- The generalization of the problem m-connected k-dominated Dominating Tree Problem may be introduced and studied in order to reflect the real-world better.
- Apart from the edge-weights, vertex-weights can also be considered.
- The application of dynamic branch-and-cut approach can be studied.
- Dynamic branch-and-cut approach can be modified with the (MCFF) as a base model. By this way, we may achieve the optimal solution faster since its LP-relaxation values are better than the other formulations.
- The relation between the DTP and wired telecommunication networks can be examined.

REFERENCES

[1] R. B. Allan and R. Laskar. On domination and independent domination numbers of a graph. *Discrete Mathematics*, 23(2):73 – 76, 1978.

[2] M. Behle, M. Jünger, and F. Liers. A primal branch-and-cut algorithm for the degree-constrained minimum spanning tree problem. In C. Demetrescu, editor, *Experimental Algorithms, 6th International Workshop, WEA 2007, Rome, Italy, June 6-8, 2007, Proceedings*, volume 4525 of *Lecture Notes in Computer Science*, pages 379–392. Springer, 2007.

[3] S. N. Chaurasia and A. Singh. A hybrid heuristic for dominating tree problem. *Soft Computing*, 20(1):377–397, 2014.

[4] V. L. do Forte, A. Lucenaa, and N. Maculana. Formulations for the minimum 2-connected dominating set problem. In *International Network Optimization Conference*, 2013.

[5] Z. Dražić, M. Čangalović, and V. Kovačević-Vujčić. A metaheuristic approach to the dominating tree problem. *Optimization Letters*, pages 1–13, 2016.

[6] M. Drexl. A note on the separation of subtour elimination constraints in elementary shortest path problems. *European Journal of Operational Research*, 229(3):595 – 598, 2013.

[7] N. Fan and M. Golari. *Combinatorial Optimization and Applications: 8th International Conference, COCOA 2014, Wailea, Maui, HI, USA, December 19-21, 2014, Proceedings*, chapter Integer Programming Formulations for Minimum Spanning Forests and Connected Components in Sparse Graphs, pages 613–622. Springer International Publishing, Cham, 2014.

[8] N. Fan and J.-P. Watson. *Solving the Connected Dominating Set Problem and Power Dominating Set Problem by Integer Programming*, pages 371–383. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[9] F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In J. Hromkovic, M. Nagl, and B. Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, volume 3353 of *Lecture Notes in Computer Science*, pages 245–256. Springer, 2004.

[10] F. Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4(2):209 – 214, 2006.

[11] S. T. Henn. *Weight-constrained minimum spanning tree problem*. PhD thesis, Technische Universität Kaiserslautern, 2007.

[12] X. Li and Z. Zhang. Two algorithms for minimum 2-connected r-hop dominating set. *Information Processing Letters*, 110(22):986 – 991, 2010.

[13] J. P. Lynch and K. J. Loh. A summary review of wireless sensors and sensor networks for structural health monitoring. *The Shock and Vibration Digest*, 38(2):91–128, 2006.

[14] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM, 2002.

[15] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and S. Jang-Ping. The broadcast storm problem in a mobile ad hoc network. *Mobicom*, pages 151–162, 1999.

[16] T. Nieberg and J. Hurink. *A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs*, pages 296–306. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[17] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and Systems Magazine*, 5(3):19–31, 2005.

[18] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors. *Wireless Sensor Networks*. Springer US, 1 edition, 2004.

[19] P. Samer and S. Urrutia. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, 9(1):41–55, 2015.

[20] W. Shang, P. Wan, F. Yao, and X. Hu. Algorithms for minimum m-connected k-tuple dominating set problem. *Theoretical Computer Science*, 381(1):241 – 247, 2007.

[21] I. Shin, Y. Shen, and M. T. Thai. On approximation of dominating tree in wireless sensor networks. *Optimization Letters*, 4(3):393–403, 2010.

[22] L. Simonetti, A. Salles da Cunha, and A. Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. *Springer-Verlag*, pages 162–169, 2011.

[23] S. Sundar and A. Singh. New heuristic approaches for the dominating tree problem. *Applied Soft Computing*, 13(12):4695 – 4703, 2013.

[24] L. Taccari. Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, 252:122–130, 2016.

[25] J. A. Torkestani. Backbone formation in wireless sensor networks. *Sensors and Actuators A: Physical*, 185:117 – 126, 2012.

[26] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, 2001.

[27] Y. Wu, F. Wang, M. T. Thai, and Y. Li. Constructing k-connected m-dominating sets in wireless sensor networks. In *MILCOM 2007-IEEE Military Communications Conference*, pages 1–7. IEEE, 2007.

[28] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[29] N. Zhang, I. Shin, B. Li, C. Boyaci, R. Tiwari, and M. T. Thai. New approximation for minimum-weight routing backbone in wireless sensor network. In *Proceedings of the Third International Conference on Wireless Algorithms, Systems, and Applications*, WASA '08, pages 96–108, Berlin, Heidelberg, 2008. Springer-Verlag.

[30] F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan, and W. Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198–208, 2011.

APPENDIX A

COMPUTATIONAL RESULTS FOR THE INSTANCES PROVIDED IN DČK (2016)

Table A.1: The abbreviations used throughout Appendix A

Abbreviation	Meaning
VNS	Variable Neighborhood Search
sol	Solution to VNS
t_{tot}	Total time spent for VNS
opt	Optimal solution to the instance
LPR	LP-relaxation value
BBN	Number of branch-and-bound nodes
RG	Relative gap
AG	Absolute gap
Time	Corresponding runtime in seconds

Table A.2: Results for small size instances provided in [5]

Instance	VNS			Time					
	sol	t_{tot}	opt	IPC	IPS	CF	FF	MCFF	MFD
dtp_10_15_0	5.89	0.04	5.89	<0.01	<0.01	<0.01	<0.01	0.02	<0.01
dtp_10_15_1	14.42	0.04	14.42	0.01	<0.01	<0.01	0.01	0.02	<0.01
dtp_10_15_2	14.35	0.04	14.35	0.04	0.02	0.03	0.01	0.09	0.01
dtp_15_20_0	18.87	0.10	18.87	0.03	0.60	0.22	0.01	0.03	0.01
dtp_15_20_1	23.03	0.09	23.03	0.02	0.60	0.17	0.01	0.03	<0.01
dtp_15_20_2	24.95	0.10	24.95	0.10	0.65	0.48	0.01	0.03	0.01
dtp_15_30_0	18.20	0.11	18.20	0.24	1.03	0.75	0.02	0.06	0.01
dtp_15_30_1	8.32	0.08	8.32	0.31	1.25	1.79	0.01	0.06	0.02
dtp_15_30_2	18.07	0.09	18.07	2.16	2.00	4.65	0.06	0.14	0.06
dtp_20_30_0	33.81	0.22	33.81	17.70	1031.41	374.44	0.04	0.08	0.03
dtp_20_30_1	36.03	0.20	36.03	17.20	1000.08	213.02	0.04	0.09	0.06
dtp_20_30_2	43.50	0.23	43.50	254.04	1197.38	332.87	0.06	0.17	0.07
dtp_20_50_0	9.81	0.17	9.81	23.57	50.88	269.27	0.02	0.11	0.02
dtp_20_50_1	12.19	0.16	12.19	205.02	218.83	770.42	0.09	0.17	0.09
dtp_20_50_2	17.42	0.20	17.42	89.56	1135.42	196.02	0.03	0.06	0.08

Table A.3: LP-relaxation values for small size instances provided in [5]

Instance		IPC		IPS		CF		FF		MCFF		MFD	
	opt	LPR	Time	LPR	Time	LPR	Time	LPR	Time	LPR	Time	LPR	Time
dtp_10_15_0	5.89	2.74	0.39	5.89	<0.01	2.74	<0.01	4.53	0.02	5.89	0.02	5.89	<0.01
dtp_10_15_1	14.42	4.72	0.67	12.89	<0.01	5.01	<0.01	10.98	0.02	14.42	0.02	14.42	<0.01
dtp_10_15_2	14.35	2.82	0.40	11.15	<0.01	2.88	<0.01	5.50	0.02	11.61	0.02	11.44	<0.01
dtp_15_20_0	18.87	5.56	0.79	14.70	0.01	5.61	0.03	14.98	0.02	18.87	0.02	18.87	<0.01
dtp_15_20_1	23.03	7.01	1.00	18.48	0.02	7.58	0.04	12.84	0.02	23.03	0.02	23.03	<0.01
dtp_15_20_2	24.95	9.25	1.32	20.30	0.02	9.25	0.05	11.64	0.02	24.95	0.02	24.78	<0.01
dtp_15_30_0	18.20	5.79	0.83	14.59	0.04	5.79	0.04	11.26	0.02	16.81	0.02	15.99	<0.01
dtp_15_30_1	8.32	2.40	0.34	6.08	0.04	2.54	0.04	3.79	0.02	6.16	0.02	5.84	<0.01
dtp_15_30_2	18.07	4.34	0.62	10.64	0.05	4.34	0.04	7.62	0.02	12.12	0.02	12.02	<0.01
dtp_20_30_0	33.81	11.16	1.59	28.18	5.60	11.47	4.13	20.24	0.02	29.41	0.02	27.09	0.01
dtp_20_30_1	36.03	13.27	1.90	27.93	2.72	13.44	3.93	28.98	0.02	32.96	0.02	28.96	<0.01
dtp_20_30_2	43.50	10.27	1.47	30.57	6.83	10.27	4.66	19.73	0.02	34.39	0.02	33.57	0.01
dtp_20_50_0	9.81	4.67	0.67	9.46	4.53	4.67	5.49	6.89	0.02	9.81	0.03	9.81	0.01
dtp_20_50_1	12.19	4.79	0.68	9.28	9.67	4.79	5.40	7.12	<0.01	11.01	0.05	10.41	0.01
dtp_20_50_2	17.42	4.15	0.59	14.17	9.50	4.15	4.43	8.04	<0.01	17.42	0.02	15.24	<0.01

Table A.4: Results and LP-relaxation values of FF for large size instances of [5]

Instance	VNS		FF						
	sol	t_{tot}	Solution	Time	BBN	RG	AG	LPR	Time
dtp_100_150_0	152.57	600.00	152.57	<0.01	270	<0.01	<0.01	85.22	0.03
dtp_100_150_1	192.21	600.00	192.21	<0.01	232	<0.01	0.01	89.92	0.02
dtp_100_150_2	146.34	600.00	146.34	<0.01	0	<0.01	<0.01	79.14	0.02
dtp_100_200_0	135.04	600.00	135.04	<0.01	1439	<0.01	<0.01	65.23	0.03
dtp_100_200_1	91.88	600.00	91.88	0.53	0	<0.01	<0.01	50.47	0.03
dtp_100_200_2	115.93	600.00	115.93	3.19	919	<0.01	0.01	58.47	0.05
dtp_200_400_0	306.06	600.00	257.09	112.13	11126	<0.01	0.02	135.87	0.08
dtp_200_400_1	303.53	600.00	258.77	57.50	7202	<0.01	0.03	124.13	0.11
dtp_200_400_2	274.37	600.00	238.27	21.27	4871	<0.01	0.02	116.32	0.08
dtp_200_600_0	132.49	600.00	121.62	538.19	20101	<0.01	0.01	58.67	0.25
dtp_200_600_1	162.92	600.00	135.08	33.02	1164	<0.01	0.01	72.17	0.20
dtp_200_600_2	139.08	600.00	123.31	2182.59	75520	<0.01	0.01	58.28	0.22
dtp_300_600_0	471.69	600.00	348.03	216.80	6654	<0.01	0.03	189.66	0.25
dtp_300_600_1	494.91	600.00	413.93	73.19	3599	<0.01	0.02	79.60	0.64
dtp_300_600_2	500.72	600.00	352.15	25.38	576	<0.01	0.03	165.12	0.19
dtp_300_1000_0	257.72	600.00	148.04	9941.75	85099	<0.01	0.01	73.98	0.67
dtp_300_1000_1	242.79	600.00	165.27	10800.00	46828	0.06	10.38	79.60	0.67
dtp_300_1000_2	223.18	600.00	155.58	10800.00	55180	0.04	5.67	77.90	0.63

Table A.5: Results and LP-relaxation values of MCFF for large size random instances provided in [5]

Instance	VNS		MCFF						
	sol	t_{tot}	Solution	Time	BBN	RG	AG	LPR	Time
dtp_100_150_0	152.57	600.00	152.57	<0.01	270	<0.01	<0.01	145.23	2.09
dtp_100_150_1	192.21	600.00	192.21	<0.01	232	<0.01	0.01	186.31	3.91
dtp_100_150_2	146.34	600.00	146.34	<0.01	0	<0.01	<0.01	145.76	1.06
dtp_100_200_0	135.04	600.00	135.04	<0.01	1439	<0.01	<0.01	123.38	7.61
dtp_100_200_1	91.88	600.00	91.88	0.53	0	<0.01	<0.01	91.88	1.14
dtp_100_200_2	115.93	600.00	115.93	3.19	919	<0.01	0.01	109.99	4.89
dtp_200_400_0	306.06	600.00	271.52	10800.00	0	0.10	26.43	240.41	100.58
dtp_200_400_1	303.53	600.00	268.12	10800.00	0	0.05	14.27	250.15	106.39
dtp_200_400_2	274.37	600.00	238.27	2176.61	0	<0.01	<0.01	234.68	116.78
dtp_200_600_0	132.49	600.00	314.65	10800.00	0	0.66	207.33	107.32	290.00
dtp_200_600_1	162.92	600.00	139.91	10800.00	0	0.05	6.56	131.51	233.52
dtp_200_600_2	139.08	600.00	133.90	10800.00	0	0.20	27.41	104.48	275.88
dtp_300_600_0	471.69	600.00	Time limit with no integer solution					325.82	700.20
dtp_300_600_1	494.91	600.00	Time limit with no integer solution					401.42	579.24
dtp_300_600_2	500.72	600.00	Time limit with no integer solution					343.07	634.33
dtp_300_1000_0	257.72	600.00	500.45	10800.00	0	1.00	500.45	127.36	2149.42
dtp_300_1000_1	242.79	600.00	446.05	10800.00	0	1.00	446.05	139.01	2261.38
dtp_300_1000_2	223.18	600.00	436.90	10800.00	0	1.00	436.90	132.07	2129.86

Table A.6: Results and LP-relaxation values of MFD for large size random instances provided in [5]

Instance	VNS		MFD						
	sol	t_tot	Solution	Time	BBN	RG	AG	LPR	Time
dtp_100_150_0	152.57	600.00	152.57	224.35	2229	<0.01	0.01	143.46	3.94
dtp_100_150_1	192.21	600.00	192.21	28.61	173	<0.01	<0.01	176.56	4.58
dtp_100_150_2	146.34	600.00	146.34	6.10	22	<0.01	<0.01	140.27	4.52
dtp_100_200_0	135.04	600.00	135.04	347.50	1265	<0.01	0.01	116.65	6.17
dtp_100_200_1	91.88	600.00	91.88	4.08	0	<0.01	<0.01	90.85	6.20
dtp_100_200_2	115.93	600.00	115.93	28.22	89	<0.01	<0.01	106.30	6.98
dtp_200_400_0	306.06	600.00	276.06	10800.00	970	0.13	35.72	223.02	61.67
dtp_200_400_1	303.53	600.00	259.30	10800.00	947	0.01	2.62	238.67	68.73
dtp_200_400_2	274.37	600.00	238.27	10800.00	779	<0.01	<0.01	222.00	52.75
dtp_200_600_0	132.49	600.00	342.90	10800.00	527	0.69	236.78	102.61	122.28
dtp_200_600_1	162.92	600.00	137.14	10800.00	901	0.03	4.19	127.97	125.13
dtp_200_600_2	139.08	600.00	132.83	10800.00	2240	0.19	25.60	99.06	73.11
dtp_300_600_0	471.69	600.00	775.93	10800.00	4	0.58	449.54	320.09	235.56
dtp_300_600_1	494.91	600.00	497.52	10800.00	314	0.21	103.07	378.55	275.38
dtp_300_600_2	500.72	600.00	490.75	10800.00	455	0.32	154.94	325.29	213.92
dtp_300_1000_0	257.72	600.00	1587.10	10800.00	2	0.92	1463.59	123.01	555.30
dtp_300_1000_1	242.79	600.00	1629.61	10800.00	0	0.92	1493.90	133.79	559.50
dtp_300_1000_2	223.18	600.00	1533.29	10800.00	9	0.92	1404.48	126.83	488.92

Table A.7: Results and LP-relaxation values of MFD for large size random instances provided in [5]

Instance	VNS solution	Improved solution
dtp_200_400_0	306.06	257.09
dtp_200_400_1	303.53	258.77
dtp_200_400_2	274.37	238.27
dtp_200_600_0	132.49	121.62
dtp_200_600_1	162.92	135.08
dtp_200_600_2	139.08	123.31
dtp_300_600_0	471.69	348.03
dtp_300_600_1	494.91	413.93
dtp_300_600_2	500.72	352.15
dtp_300_1000_0	257.72	148.04
dtp_300_1000_1	242.79	165.27
dtp_300_1000_2	223.18	155.58

APPENDIX B

COMPUTATIONAL RESULTS FOR THE INSTANCES PROVIDED IN CS (2014)

Table B.1: The abbreviations used throughout Appendix B

Abbreviation	Meaning
NV	Number of vertices
ID	Ordinal number of the instance
BSF	Best solution found in the literature
Algorithm	The algorithm corresponding to the BSF
LPR	LP-relaxation value
BBN	Number of branch-and-bound nodes
RG	Relative gap
AG	Absolute gap
Time	Corresponding runtime in seconds

Table B.2: Results of FF for the instances provided in [3]

Range	NV	ID	BSF	Algorithm	Time	FF Solution	FF Time	BBN	RG	AG	LPR	Time
100	50	1	1204.41	ACO_DT	2.41	1204.41	7.70	4289	<0.01	<0.01	278.06	<0.01
		2	1340.44	ACO_DT	4.18	1340.44	9.27	7479	<0.01	<0.01	460.63	<0.01
		3	1316.39	ACO_DT	2.50	1316.39	5.63	6234	<0.01	<0.01	392.45	<0.01
	100	1	1217.47	EA/G-MP	11.06	1217.47	10800.00	532570	0.09	111.66	161.39	<0.01
		2	1128.40	EA/G-MP	9.93	1128.40	10800.00	423933	0.12	136.18	146.41	<0.01
		3	1252.99	ABC_DT	28.39	1253.49	10800.00	457012	0.31	391.53	172.84	<0.01
125	50	1	802.95	ACO_DT	2.16	802.95	28.56	9763	<0.01	0.05	145.58	0.03
		2	1055.10	ACO_DT	2.01	1055.10	94.05	67578	<0.01	0.10	257.28	0.02
		3	877.77	ACO_DT	1.40	877.77	42.28	23108	<0.01	0.05	208.71	0.03
	100	1	943.01	ACO_DT	7.29	947.50	10800.00	363920	0.31	290.94	86.59	0.23
		2	917.00	ABC_DT	19.34	935.28	10800.00	233355	0.35	323.85	87.83	0.23
		3	998.18	ACO_DT	7.61	1005.54	10800.00	288330	0.41	414.03	115.79	0.19
150	50	1	647.75	ACO_DT	1.02	647.75	75.69	31153	<0.01	0.06	109.09	0.05
		2	863.69	ACO_DT	1.70	863.69	727.66	347541	<0.01	0.09	192.92	0.03
		3	743.74	ABC_DT	7.38	743.94	267.08	105870	<0.01	0.07	126.14	2.83
	100	1	876.69	EA/G-MP	6.61	892.78	10800.00	270390	0.39	344.52	60.34	0.36
		2	657.35	ACO_DT	4.23	657.35	10800.00	210940	0.38	247.57	52.78	313.00
		3	722.87	ACO_DT	4.80	760.38	10800.00	243624	0.34	256.24	78.00	0.34

Table B.3: Results of MCFF for the instances provided in [3]

Range	NV	ID	BSF	Algorithm	Time	MCFF Solution	Time	BBN	RG	AG	LPR	Time
100	50	1	1204.41	ACO_DT	2.41	1204.41	82.52	921	<0.01	<0.01	611.64	0.10
		2	1340.44	ACO_DT	4.18	1340.44	3.70	0	<0.01	<0.01	1013.26	0.10
		3	1316.39	ACO_DT	2.50	1316.39	6.91	0	<0.01	<0.01	1009.17	0.10
	100	1	1217.47	EA/G-MP	11.06	1352.20	10800.00	97	0.48	653.12	479.59	70.66
		2	1128.40	EA/G-MP	9.93	1299.17	10800.00	201	0.46	603.11	430.54	71.22
		3	1252.99	ABC_DT	28.39	1281.94	10800.00	13	0.58	746.73	441.81	58.59
125	50	1	802.95	ACO_DT	2.16	Time limit with no integer solution					315.56	0.01
		2	1055.10	ACO_DT	2.01	Time limit with no integer solution					438.76	0.01
		3	877.77	ACO_DT	1.40	Time limit with no integer solution					383.58	0.01
	100	1	943.01	ACO_DT	7.29	1078.31	10800.00	7	0.67	726.60	275.63	668.78
		2	917.00	ABC_DT	19.34	921.16	10800.00	168	0.62	573.72	266.24	282.24
		3	998.18	ACO_DT	7.61	Time limit with no integer solution					283.87	109.88
150	50	1	647.75	ACO_DT	1.02	Time limit with no integer solution					249.09	8.75
		2	863.69	ACO_DT	1.70	Time limit with no integer solution					341.40	3.05
		3	743.74	ABC_DT	7.38	Time limit with no integer solution					222.47	3.73
	100	1	876.69	EA/G-MP	6.61	961.15	10800.00	3	0.74	709.42	206.86	1322.48
		2	657.35	ACO_DT	4.23	Time limit with no integer solution					163.65	1170.59
		3	722.87	ACO_DT	4.80	Time limit with no integer solution					196.74	1155.84

Table B.4: Results of MFD for the instances provided in [3]

Range	NV	ID	BSF	Algorithm	Time	MFD Solution	MFD Time	BBN	RG	AG	LPR	Time
100	50	1	1204.41	ACO_DT	2.41	1204.41	8637.14	131327	<0.01	0.09	442.65	0.52
		2	1340.44	ACO_DT	4.18	1340.44	2237.01	82401	<0.01	0.13	627.49	0.27
		3	1316.39	ACO_DT	2.50	1316.39	1727.07	34111	<0.01	0.11	544.05	0.23
	100	1	1217.47	EA/G-MP	11.06	1492.26	10800.00	19196	0.61	914.54	324.36	4.63
		2	1128.40	EA/G-MP	9.93	1478.78	10800.00	31153	0.69	1018.34	246.73	5.83
		3	1252.99	ABC_DT	28.39	1609.98	10800.00	27989	0.70	1123.41	290.69	5.47
125	50	1	802.95	ACO_DT	2.16	817.76	10800.00	139712	0.18	147.91	231.18	0.50
		2	1055.10	ACO_DT	2.01	1089.52	10800.00	159761	0.23	253.26	363.12	0.47
		3	877.77	ACO_DT	1.40	882.69	10800.00	219476	0.11	97.23	289.18	0.45
	100	1	943.01	ACO_DT	7.29	1444.78	10800.00	8084	0.80	1156.31	151.10	11.05
		2	917.00	ABC_DT	19.34	1179.82	10800.00	8756	0.75	885.63	159.39	9.67
		3	998.18	ACO_DT	7.61	1246.55	10800.00	2598	0.83	1038.75	169.97	7.48
150	50	1	647.75	ACO_DT	1.02	649.07	10800.00	111343	0.11	71.19	164.69	0.91
		2	863.69	ACO_DT	1.70	946.26	10800.00	85335	0.34	319.49	274.10	0.88
		3	743.74	ABC_DT	7.38	857.84	10800.00	82647	0.42	356.18	183.70	1.13
	100	1	876.69	EA/G-MP	6.61	1243.25	10800.00	2929	0.83	1037.55	101.35	17.75
		2	657.35	ACO_DT	4.23	1004.52	10800.00	2847	0.83	834.41	97.33	37.25
		3	722.87	ACO_DT	4.80	1246.55	10800.00	2598	0.83	1038.75	118.18	26.59