ABSTRACT

Title of dissertation: DUAL-BASED LOCAL SEARCH FOR DETERMINISTIC, STOCHASTIC, AND ROBUST VARIANTS OF THE CONNECTED FACILITY LOCATION PROBLEM

María Gisela Bardossy, Doctor of Philosophy, 2011

Dissertation directed by: Professor Subramanian Raghavan
Robert H. Smith School of Business

In this dissertation, we propose the study of a family of network design problems that arise in a wide range of practical settings ranging from telecommunications to data management. We investigate the use of heuristic search procedures coupled with lower bounding mechanisms to obtain high quality solutions for deterministic, stochastic and robust variants of these problems. We extend the use of well-known methods such as the sample average approximation for stochastic optimization and the Bertsimas and Sim approach for robust optimization with heuristics and lower bounding mechanisms. This is particular important for NP-complete problems where even deterministic and small instances are difficult to solve to optimality. Our extensions provide a novel way of applying these techniques while using heuristics; which from a practical perspective increases their usefulness.

We study the connected facility location (ConFL) problem, which arises in a number of applications that relate to the design of telecommunication networks as well as data distribution and management problems on networks. It combines

features of the uncapacitated facility location problem with the Steiner tree problem and is known to be NP-complete. In this setting, we wish to install a set of facilities on a communication network and assign customers to the installed facilities. In addition, the set of selected facilities needs to be connected by a Steiner tree. We propose a dual-based local search (DLS) heuristic that combines dual-ascent and local search which together yield strong lower and upper bounds on the optimal solution. Our procedure is applied to a slightly more general version of the ConFL problem that embraces a family of four different problems—the Steiner tree-star problem, the general Steiner tree-star problem, the ConFL problem, and the rent-or-buy problem—that combine facility location decisions with connectivity requirements. Consequently, our solution methodology can be successfully applied to all of them.

DUAL-BASED LOCAL SEARCH FOR DETERMINISTIC, STOCHASTIC, AND ROBUST VARIANTS OF THE CONNECTED FACILITY LOCATION PROBLEM

by

María Gisela Bardossy

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor Subramanian Raghavan, Chair/Advisor
Professor Michael Ball
Professor Zhi-Long Chen
Professor Bruce Golden
Professor Richard La

Dedication

To my nona, Selma Catalina Gariglio de Filippa.

Acknowledgments

First and foremost I'd like to thank my advisor, Professor Raghu Raghavan, for giving me the invaluable opportunity to work with him over the past five years. He has always made himself available for help and advice beyond the doctoral program and there has never been an occasion when I've knocked on his door and he hasn't given me time. Thanks are due to Professor Bruce Golden, Professor Zhi-Long Chen, Professor Michael Ball, and Professor Richard La for agreeing to serve on my dissertation committee and for sparing their invaluable time reviewing the manuscript. I have to thank Professor Kazim Ruhi for his great teaching advice and job search tips.

I am also greatly indebted to my graduate fellows: Inbal Yahav, Dina Ribbink, Dilek Gunnec, Andrew Hall and Ming Chen, for their care and love. I was lucky to meet them and share this journey with them. I also want to thank my dearest friends, Anita and Javier Bianchi, Francisca and Giorgo Sertsios, and Andrea Cavassa, who became my College Park family in these last five years. I will forever cherish the great times that we spent together and I will miss them a lot as we continue our separate paths and dreams.

I owe my deepest thanks to my husband, José, who has always stood by me so I could accomplish my dreams. Words cannot express the gratitude I owe him. I need to thank my parents-in-law for letting me take their son away from them. I would also like to thank my family, who in their very special ways always supported me. Thank you mom for your support and constant encouragement. Thank you brothers

for taking more family responsibility than you have to and always protecting me.

I also want to thank my grandparents for being my role models in family values, love, generosity and joy, and whose memory is a constant inspiration to be a better Christian.

Lastly, thank God for all your blessings!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

B&C	Branch-and-Cut
ConFL	Connected Facility Location
DA	Dual Ascent
DLS	Dual-ascent Local Search
DST	Directed Steiner Tree
EVS	Expected Value Solution
GRASP	Greedy Randomized Adaptive Search Procedure
GSTS	General Steiner Tree-Star
MVP	Mean Value Problem
RCOH	Robust Combinatorial Optimization Heuristic
RConFL	Robust Connected Facility Location
ROB	Rent-or-Buy
SAA	Sample Average Approximation
SConFL	Stochastic Connected Facility Location
SSBND	Single-Sink Buy-at-Bulk Network Design
STS	Steiner Tree-Star
UFL	Uncapacitated Facility Location
VNS	Variable Neighborhood Search

Chapter 1

Dissertation Overview

In this dissertation, we propose the study of a family of network design problems that arise in a wide range of practical settings ranging from telecommunications to data management. We investigate the use of heuristic search procedures coupled with lower bounding mechanisms to obtain high quality solutions for deterministic, stochastic and robust variants of these problems. We extend the use of well-known methods such as the sample average approximation for stochastic optimization (Shapiro and Philpott 2007) and the Bertsimas and Sim approach for robust optimization (Bertsimas and Sim 2003) with heuristics and lower bounding mechanisms. This is particular important for NP-complete problems where even deterministic and small instances are difficult to solve to optimality. Our extensions provide a novel way of applying these techniques while using heuristics; which from a practical perspective increases their usefulness.

We study the connected facility location (ConFL) problem, which arises in a number of applications that relate to the design of telecommunication networks as well as data distribution and management problems on networks. It combines features of the uncapacitated facility location problem with the Steiner tree problem and is known to be NP-complete. In this setting, we wish to install a set of facilities on a communication network and assign customers to the installed facil-

ities. In addition, the set of selected facilities needs to be connected by a Steiner tree. We propose a dual-based local search (DLS) heuristic that combines dual-ascent and local search which together yield strong lower and upper bounds to the optimal solution. Our procedure is applied to a slightly more general version of the ConFL problem that embraces a family of four different problems—the Steiner tree-star problem, the general Steiner tree-star problem, the ConFL problem, and the rent-or-buy problem—that combine facility location decisions with connectivity requirements. Consequently, our solution methodology can be successfully applied to all of them. We discuss a wide range of computational experiments, which indicate that our heuristic is a very effective procedure that finds high quality solutions very rapidly for the deterministic variant of the problem.

In the next part of the dissertation, we focus our attention mainly on the ConFL problem because as we show in Chapter 2 the remaining problems can be easily transformed into ConFL instances, and our computational experiments in Chapter 3 reveal that the ConFL problem is the hardest to solve for the DLS heuristic. Regardless, all of the subsequent analyses and solution methodologies for the stochastic and robust variants presented throughout this dissertation can be successfully applied to all of the problem within the broader family.

We consider the ConFL problem in the case where there is uncertainty on the assignment costs either due to random quantity demands, unknown customer locations, or varying edge lengths. We show that the optimal solution of the stochastic problem with random demands can be obtained by replacing the unknown values with their averages. However, when customer locations or edge lengths are random

this strategy is not optimal. Instead, we define a deterministic equivalent problem and provide a set of graph transformations that allow us to apply our DLS procedure to this equivalent problem. Our computational results show that this strategy is very effective for instances with a polynomial number of scenarios. However, as the number of scenarios increases, it becomes impractical to generate a complete deterministic equivalent problem. Under such circumstances, we propose a more effective framework that relies on the sample average approximation (SAA) approach (Shapiro and Philpott 2007) yet it utilizes a heuristic to solve the sample average problems and a lower bounding procedure to construct a confidence interval on the true optimal value. To our knowledge this is the first discussion of the SAA approach in stochastic programming with the use of heuristics to solve the underlying optimization problems.

Finally, we present a robust optimization model for the ConFL problem under customer uncertainty and we show how a DLS heuristic can be effectively used to obtain high quality solutions to this robust optimization problem. We extend Bertsimas and Sim's robust optimization solution approach (Bertsimas and Sim 2003) to situations where one has a heuristic upper bound and a lower bound on the optimal solution objective value for each nominal problem. An advantage of the DLS heuristic is that it provides high quality lower bounds in addition to the heuristic solution. We present computational results that demonstrate that the DLS heuristic rapidly obtains high-quality solutions for a large set of test instances for the robust variant of the ConFL problem.

We conclude this dissertation with a summary of contributions and directions

for future research.

Chapter 2

The Connected Facility Location Problem and Related Problems

2.1 Introduction

The recent growth of telecommunication networks coupled with digital data management has motivated a range of network design problems that combine facility location with connectivity requirements. These network design problems combine features of the uncapacitated facility location (UFL) problem with the Steiner tree problem. The connected facility location (ConFL) problem belongs to this class of network design problems and is known to be NP-complete. In this chapter we introduce a slightly more general version of the ConFL problem that subsumes a family of four closely related problems (that arise in virtual private network design and data distribution problems on networks) and propose a dual-based local search (DLS) heuristic that combines dual-ascent and local search to obtain high quality solutions rapidly.

We first describe the family of four closely related problems that are special cases of the general version of the ConFL problem that we introduce in this essay.

2.1.1 Steiner tree-star (STS) problem

The Steiner tree-star (STS) problem (Lee et al. 1993) is a virtual private network design problem that arises in the design of telecommunication networks for

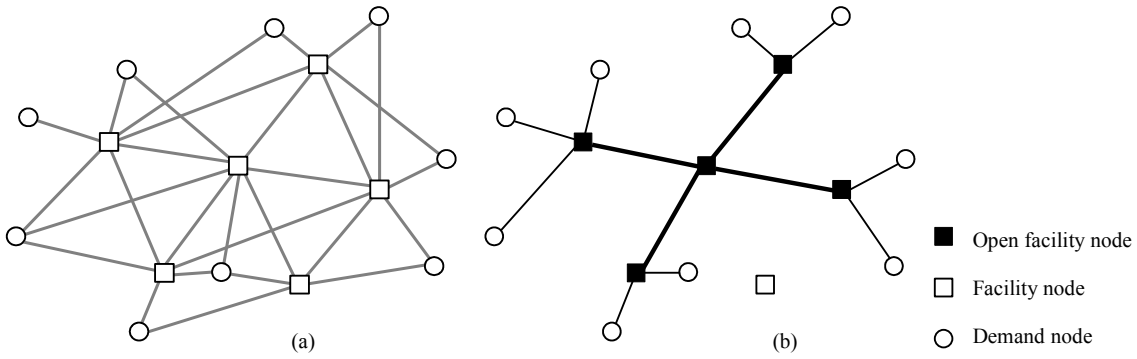

Figure 2.1: Steiner tree-star example

digital data services and can be stated as follows. Given a graph $G = (V, E)$ and a disjoint partition of the nodes in V into two sets: $D \subset V$, set of demand nodes (also referred to as target nodes in the STS literature), and $F \subset V$, set of potential facility nodes (also referred to as hubs in the STS literature), we seek to find a minimum cost tree such that every demand node is connected (or assigned) to a facility node, and the facilities serving demand nodes are connected through a node-weighted Steiner tree T constructed solely on the F nodes (i.e., on $G(F) = (F, E(F))$). Each facility has an opening (or activating) cost, $f_i \geq 0$, that is incurred if the facility is included in the final network design (regardless of whether the facility serves a demand node). The cost to connect a demand node, $j \in D$, to a facility node, $i \in F$, is given by an assignment cost a_{ij}; while the cost to connect two facilities, $i, j \in F$, comes at a significantly higher (in terms of cost per unit distance) connection cost b_{ij}. The network design cost is thus given by $\sum_{j \in D} a_{(j)j} + \sum_{i \in T} f_i + \sum_{\{i,j\} \in T} b_{ij}$, where $i(j)$ is the facility serving demand node j, and T is the Steiner tree connecting the facilities serving demand nodes. Figure 2.1 illustrates an example of the STS problem, where

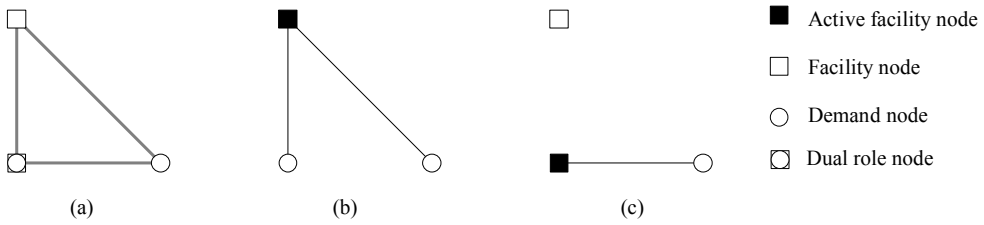

Figure 2.2: General Steiner tree-star example

(a) shows the graph G, and (b) represents a feasible solution to the STS problem.

2.1.2 General Steiner tree-star (GSTS) problem

Khuller and Zhu (2002) extended the STS problem to a more general setting where the facility and demand node sets are not disjoint and called it the *general* STS (GSTS) problem. In this case some demand nodes may host a facility in a solution to the problem. Figure 2.2(a) shows a small example of GSTS problem, where one node has dual role and can be both a facility and a demand node. Figure 2.2(b) illustrates a solution where the dual-role node takes the function of a demand node; while Figure 2.2(c) shows a solution where it takes the role of a facility node. Note that when the dual-role node takes the role of a facility node it also satisfies its demand node at no additional cost.

2.1.3 Connected Facility Location (ConFL) problem

Karger and Minkoff (2000), Krick et al. (2003), and Havet and Wennink (2004) independently introduced data distribution and management problems in a network setting that arise in information/content distribution networks (such networks are

widely prevalent at search providers for example). In the applications discussed in these three papers, there are facilities (or servers) to be located on a network that will contain (or cache) information. Demand nodes make requests for the information. When a demand node j requests a piece of information, it is served from the closest facility $i(j)$ and incurs a cost $\alpha(i(j), j)$ (if a demand node j makes multiple requests, say d_j requests, the cost is simply $d_j\alpha(i(j), j)$). Further, updates to the information on the servers are made over time. If a piece of information is updated, then it must be updated at every facility (or server) on the network. This incurs a cost $\beta(i, j)$ for every edge $\{i, j\}$ in the network on which this information is sent. Consequently, the cheapest way to update information over facilities (once a choice of which facilities to open has been made) is via a Steiner tree T on the facilities with a cost of $\sum_{\{i,j\}\in T} \beta(i, j)$ (if μ information update requests are made, then the cost is $\mu \sum_{\{i,j\}\in T} \beta(i, j)$). The goal is to determine (i) what facilities to locate (or open), (ii) which facility serves each demand node, and (iii) how to connect the open facilities; in order to minimize the total cost. In the applications discussed in Karger and Minkoff (2000) and Havet and Wennink (2004) the facility opening costs are zero, while in Krick et al. (2003) there are costs associated with opening facilities.

The problems introduced by Karger and Minkoff (2000), Krick et al. (2003) and Havet and Wennink (2004) can be modeled as ConFL problems (Gupta et al. 2001, introduced the terminology ConFL) that can be stated as follows. We are given a graph $G = (V, E)$, and three sets: $D \subseteq V$, set of demand nodes (or customers); $F \subseteq V$, set of potential facility nodes; and $S \subseteq V$, set of potential Steiner nodes,

with $D \cup F \cup S = V$ and $F \cap S = \emptyset$. We seek to find a minimum cost network where every demand node is assigned to an open facility and open facilities are connected through a Steiner tree T constructed on the subgraph of G on the nodes $F \cup S$ (i.e., $G(F \cup S) = (F \cup S, E(F \cup S))$). There are facility opening costs, $f_i \geq 0$, incurred for each facility that serves a customer; assignment costs, $a_{ij} \geq 0$, for assigning a customer $j \in D$ to a facility $i \in F$; and edge costs, $b_{ij} \geq 0$, for an edge $\{i, j\} \in E(F \cup S)$ if it is used on the Steiner tree T. The nodes in S may be viewed as pure Steiner nodes and can only be used in the tree T as Steiner nodes, while the nodes in F may be used as Steiner nodes on the tree T without incurring a facility opening cost when no customers are assigned to them. The final network cost is given by $\sum_{j \in D} a_{i(j)j} + \sum_{i \in Z} f_i + \sum_{\{i,j\} \in T} b_{ij}$, where $i(j)$ is the facility serving demand node j, $\mathcal{F}$ is the set of open facilities (or facilities serving customers), and T is a Steiner tree connecting the open facilities. The data distribution problems introduced by Karger and Minkoff (2000), Krick et al. (2003), and Havet and Wennink (2004) may be modeled as a ConFL problem by setting $a_{ij} = d_j \alpha(i, j)$, $b_{ij} = \mu \beta(i, j)$, and f_i as the facility opening cost. Figure 2.3 illustrates an example of the ConFL problem and a feasible solution.

In the definition of the ConFL problem above, it is possible that $D \cap F \neq \emptyset$ or $D \cap S \neq \emptyset$. In both these situations a demand node j can be used on the Steiner tree T. In this case, in addition to the edges adjacent to node j in the Steiner tree T, we have a connection $\{i(j), j\}$ between node j and the facility it is assigned to. Consequently, the minimum cost network is not necessarily a tree. On the other hand, if $D \cap F = \emptyset$ and $D \cap S = \emptyset$ (meaning that D, F, and S form a partition), a

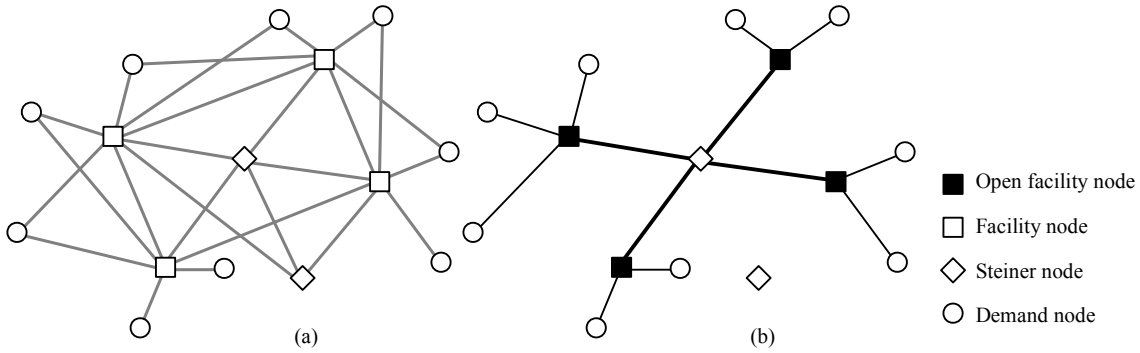

Figure 2.3: Connected facility location example

demand node cannot be used on the Steiner tree T and the minimum cost network is a tree. It can be viewed as consisting of a core tree T (where the leaf nodes must be open facility nodes), with the assignment edges dangling from open facility nodes on the core tree. Typically, the papers in the computer science literature allow for $D \cap F \neq \emptyset$ or $D \cap S \neq \emptyset$, while the papers in the operations research literature assume that D, F, and S form a partition. It is easy to transform ConFL instances where $D \cap F \neq \emptyset$ or $D \cap S \neq \emptyset$ into ones where the sets D, F, and S form a partition. We will discuss this transformation in §2.3.2.

2.1.4 Rent-or-Buy (ROB) problem

The rent-or-buy (ROB) problem, often viewed as a special case of the ConFL problem, has the feature that facilities can be opened at any node of the graph (i.e., $F = V$) at zero cost. (The term rent-or-buy comes from a related problem called the single-sink buy-at-bulk network design (SSBND) problem (Salman et al. 1997) which ROB is equivalent to when the SSBND problem has two cable types.

Here the idea is that edges on the network can either be rented, in which case the cost function a_{ij} applies, or can be purchased in which case the cost function b_{ij} applies.) In our opinion, the ROB problem can also be viewed as a special case of the GSTS problem with zero facility opening costs and $D \subseteq F = V$. In the ROB problem any demand node can act as a facility node and hence serve other customers (i.e., demand nodes). Consequently, the cost of an edge depends on the role of its adjacent nodes. If one of the end points of the edge plays the role of a demand node, meaning that the edge is connecting a demand node to an open facility, we say the edge is an assignment edge and its cost is a_{ij}. Otherwise, the edge belongs to the Steiner tree T (and we call it a tree edge) and its cost is b_{ij}. (In all four problems (STS, GSTS, ConFL, and ROB) we will use the term customer interchangeably with demand node. We will also refer to edges connecting demand nodes to facilities as assignment edges, and potential edges on the Steiner tree connecting open facilities as tree edges.) In this problem, the final network cost is given by $\sum_{j \in D} a_{i(j)j} + \sum_{\{i,j\} \in T} b_{ij}$, where $i(j)$ is the facility which serves demand node j (note that $a_{ii} = 0$ when demand node i is used as a facility), and T is the Steiner tree connecting all the open facility nodes. The data distribution and management applications introduced by Karger and Minkoff (2000) and Havet and Wennink (2004) are actually instances of the ROB problem. In a different setting, Nuggehalli et al. (2003) considered the problem of energy-conscious cache placement in wireless ad hoc networks. The objective is to find an effective strategy to cache the server information at some nodes distributed across the network while optimally considering the trade-off between energy consumption and access latency.

Interestingly, this problem is also an instance of the ROB problem (Ljubić 2007) indicating the widespread application of the ConFL and ROB problems.

All four problems—STS, GSTS, ConFL, ROB—are of significant interest from a practical perspective; both in the telecommunications/virtual private network design context as well as in the data distribution and management context. There has been a considerable amount of research devoted to these four problems from an approximation algorithms perspective, but somewhat limited study of these problems from a mathematical programming perspective.

Clearly, all four problems combine a facility location problem with a Steiner tree problem. In this chapter, we exploit the similarity between the four problems and define a slightly more general version of the ConFL problem that we call the general ConFL (or GConFL) problem. In the next chapter, we devise a high-quality dual-based local search heuristic for the GConFL problem that provides both tight lower and upper bounds. Our heuristic solution strategy consists of first formulating the GConFL problem as a directed Steiner tree problem with a unit degree constraint on the root node. We then implement a dual-ascent procedure to obtain a lower bound and an upper bound (feasible solution) to the optimal solution. We then apply a set of local improvement steps on the feasible solution obtained by the dual-ascent procedure to significantly improve the quality of the solution. We conducted an extensive set of computational experiments (reported on the next chapter) that demonstrated the efficacy and efficiency of our DLS heuristic. These results included instances on complete graphs as well as non-complete graphs. Across the four problems over the set of test instances our DLS heuristic consistently found

solutions of very high quality. Over the 2748 problems tested the DLS heuristic found solutions that were on average at most 1.07% from optimality.

2.2 Literature Review

2.2.1 Steiner tree-star (STS) and General STS problem

The STS problem was introduced by Lee et al. (1993). Later, Lee et al. (1996) described valid inequalities and facets for the STS polytope (polytope of integer feasible STS solutions) and developed a branch-and-cut procedure for the STS problem. Their procedure was able to solve Euclidean test problems with less than 200 nodes in up to 3 hours of computational time.

Xu et al. (1996a) and Xu et al. (1996b) proposed two tabu search heuristics for the STS problem which they tested in a small sample of random and grid problems with up to 600 nodes. Subsequently, Chu et al. (2000) proposed a genetic algorithm for the STS problem. Their computational experiments indicated that their genetic algorithm is of similar quality to Xu et al. (1996a). For the same set of problems, they find no difference in performance between the two heuristics; however, their genetic algorithm required less computational time. Note, however, that since neither heuristic computes a lower bound to the optimal solution, no optimality gaps were reported in neither of these papers. Khuller and Zhu (2002) introduced the GSTS problem and gave two approximation algorithms with approximation ratios of 5.16 and 5, respectively.

2.2.2 Connected Facility Location (ConFL) problem

Gupta et al. (2001) arrived at the ConFL problem when considering a virtual private network design with demand uncertainty. Here, a set of demand nodes are to be connected using a virtual private network. One is given the maximum incoming and outgoing traffic from each demand node, but one does not know the actual traffic matrix between the nodes. One wishes to construct a minimum cost tree network and provision sufficient capacity on its edges so that the tree network can support any traffic matrix where the aggregate incoming and outgoing demands respect the maximum limits for each node. Gupta et al. (2001) reduced this virtual private network design problem to the ConFL problem. They then gave a 10.66 approximation algorithm for the ConFL problem by adapting a rounding technique of Shmoys et al. (1997) on an integer programming formulation of the problem with an exponential number of constraints.

Swamy and Kumar (2004) described a primal-dual approximation algorithm for the ConFL problem. Their algorithm works in two phases and has an approximation ratio of 8.55. The first phase is a facility location phase where they decide which facilities to open, connect demands to facilities, and cluster the demands at each facility. In this phase, demands are clustered so that each open facility serves at least a certain minimum number of demand points. (In Swamy and Kumar (2004) the ratio between the costs b_{ij} and a_{ij} is constant, and $\frac{b_{ij}}{a_{ij}}$ is treated as the minimum number of demand points to cluster at a facility node.) The second phase is a Steiner phase where the open facilities are connected by a Steiner tree. Jung et al.

(2008) improved upon Swamy and Kumar's algorithm (by making changes in phase 1) and devised a 6.55 approximation algorithm for the ConFL problem.

Recently Eisenbrand et al. (2008) presented a randomized algorithm that improves the approximation ratio for the ConFL problem to 4. In their algorithm the idea is to (i) run an approximation algorithm for the UFL problem, (ii) randomly sample demand nodes, and open the facilities serving sampled customers in the approximate solution (all the demands are now assigned to these open facilities), and (iii) compute an approximate Steiner tree on the opened facilities. The authors showed that the approximation ratio degrades slightly to 4.23 when the algorithm is derandomized.

The theoretical computer science community has focused on developing approximation algorithms for the ConFL problem. In all of these papers no computational results are presented for any of the proposed approximation algorithms, and thus their effectiveness in practice is unknown.

The ConFL problem has only recently gained attention in the operations research community. Ljubić (2007) introduced a variable neighborhood search (VNS) heuristic that is combined with reactive tabu search. She also proposed a branch-and-cut (B&C) approach for solving the ConFL problem to optimality. She constructed ConFL test problems by combining Steiner tree problem instances from OR-Library [1] and UFL problem instances from UflLib [2]. She reported that the VNS heuristic found solutions that were up to 10% from the lower bound provided

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html

[2] http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/

by the branch-and-cut algorithm. It is our understanding (Ljubić 2009) that due to a computational error the values of the lower bounds described in Ljubić (2007) are incorrect and in some instances may be lower than the ones reported in the paper.

We should note that the approaches used by Swamy and Kumar (2004), Ljubić (2007), and Jung et al. (2008) assumed that one of the facilities in the optimal solution is known a priori. We find that interpretation of the ConFL somewhat restrictive. If one does not know a priori one of the facilities in the optimal solution, then their solution procedures need to be applied $|F|$ times (once for each facility node) and the best solution selected. In this chapter, we will assume that we do not have any a priori knowledge of an open facility in the optimal solution.

Tomazic and Ljubić (2008) proposed a greedy randomized adaptive search procedure (GRASP) algorithm for the ConFL problem. (Here the authors do not assume a priori knowledge of an open facility in the optimal solution.) The proposed heuristic is a multi-start iterative approach, where for each start a greedy construction defines a starting solution. Then, local improvements consisting of moves where a single facility is opened or closed are applied. Finally, a shortest path Steiner tree heuristic is applied to find a Steiner tree on the open facilities. The procedure was tested on three sets of randomly generated graphs with varying topologies and cost structures. On those instances, the GRASP procedure provided results whose average gaps were as large as 10% from the optimal solution.

2.2.3 Rent-or-Buy (ROB) problem

Since the ROB problem may be viewed as a special case of the ConFL problem, some of the heuristics proposed for the ConFL problem can be applied to the ROB problem with no deterioration in the approximation ratio, and in many cases with an improvement in the approximation ratio. Gupta et al. (2001) found that their rounding heuristic is a 9.002-approximation algorithm for the ROB problem while Swamy and Kumar (2004) showed that their primal-dual-approach is a 4.55-approximation algorithm. Nuggehalli et al. (2003) provided a 6-approximation algorithm for the ROB problem. Gupta et al. (2003) proposed a 3.55-randomized approximation algorithm for the ROB problem. The best known approximation algorithm for the ROB problem is due to Eisenbrand et al. (2008), who proposed a randomized approximation algorithm with a performance bound of 2.92 that when derandomized has an approximation ratio of 3.28.

2.3 General Connected Facility Location Problem

One of the key differences between the STS and the ConFL problems is that in the STS problem a facility node in the network design that does not serve a customer incurs a facility opening cost, f_i, while in the ConFL problem a facility incurs a facility opening cost only if it serves a customer. To create a generalization that encompasses all four problems, we slightly alter the original definition of the ConFL problem (with some additional changes) and require that a facility node in T incurs a facility opening cost regardless of whether it serves a customer or not.

Specifically, we define the GConFL problem as follows. Given a graph $G = (V, E)$ and a disjoint partition of V into three subsets of nodes: namely, D, the set of demand nodes; F, the set of potential facilities; and S, the set of potential Steiner nodes; we seek a minimum cost tree solution such that (i) a set of facility nodes is opened, (ii) every demand node is assigned to an open facility, and (iii) open facilities are connected through a Steiner tree T constructed on $G(F \cup S) = (F \cup S, E(F \cup S))$. Each facility i used in T incurs a facility opening cost, $f_i \geq 0$; the cost of assigning demand node $j \in D$ to facility $i \in F$ is given by a_{ij} (as before we refer to these edges as assignment edges and the remaining edges as tree edges); and the cost of tree edges is given by b_{ij}. The network design cost is defined by $\sum_{j \in D} a_{i(j)j} + \sum_{i \in V(T) \cap F} f_i + \sum_{\{i,j\} \in T} b_{ij}$ (where as before $i(j)$ denotes the facility serving demand node j).

Observe that the graph for the GConFL problem has some special characteristics. Since each node takes a unique role, the only edges from demand nodes (D) are to potential facility nodes (F) (i.e., there are no edges between any pair of demand nodes, and there are no edges in the graph between any demand node and any Steiner node (S)). Further, use of a facility node necessarily incurs a facility opening cost, while there are no opening costs associated with using a Steiner node.

Clearly, the STS problem is a special case of the GConFL problem with $S = \emptyset$ and sets F and D defined identically to the STS problem. Here the cost for assignment edges and tree edges remains unchanged. On the other hand, to see that the GSTS, ConFL and ROB problems are special cases of the GConFL problem, we must apply a set of graph transformations.

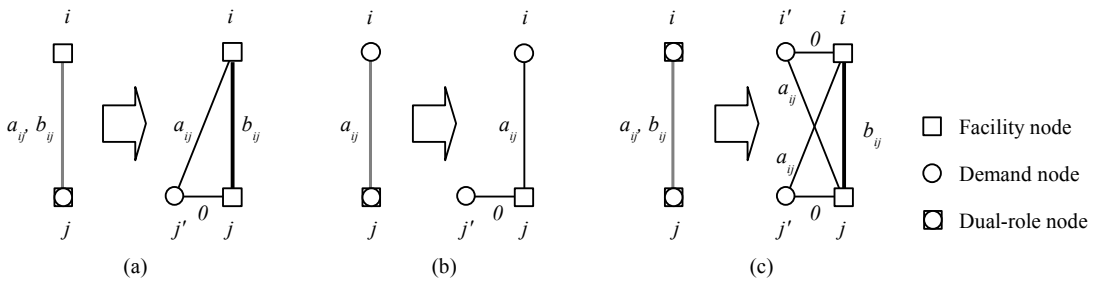

Figure 2.4: GSTS problem transformation to GConFL problem

In each of these three problems, a node can have multiple roles. For example, in the GSTS problem a node can be a demand node as well as a facility node. Similarly, in the ConFL problem a node can be a facility node (serving demand nodes and incurring a facility installation cost) or a Steiner node (not serving any demand nodes and not incurring a facility installation cost). The transformations are based on a node splitting strategy where each node has a unique role (either demand node, facility node, or Steiner node). For ease of exposition (and brevity) we illustrate each of these transformations graphically.

2.3.1 Transforming the GSTS problem to the GConFL problem

In an instance of the GSTS problem, the node sets F and D are not disjoint ($F \cap D \neq \emptyset$) and there are no potential Steiner nodes ($S = \emptyset$). To address the fact that the node sets are not disjoint, we duplicate every node $i \in (F \cap D)$ creating an additional copy i'. One copy, i, is treated as a facility node in the GConFL problem and the other copy, i', is treated as a demand node in the GConFL problem. Further, nodes i and i' are connected by a zero cost edge, and the edge $\{i, j\}$ in GSTS that has a dual-role node adjacent to it (i.e., if either $i, j \in (F \cap D)$) is replaced as

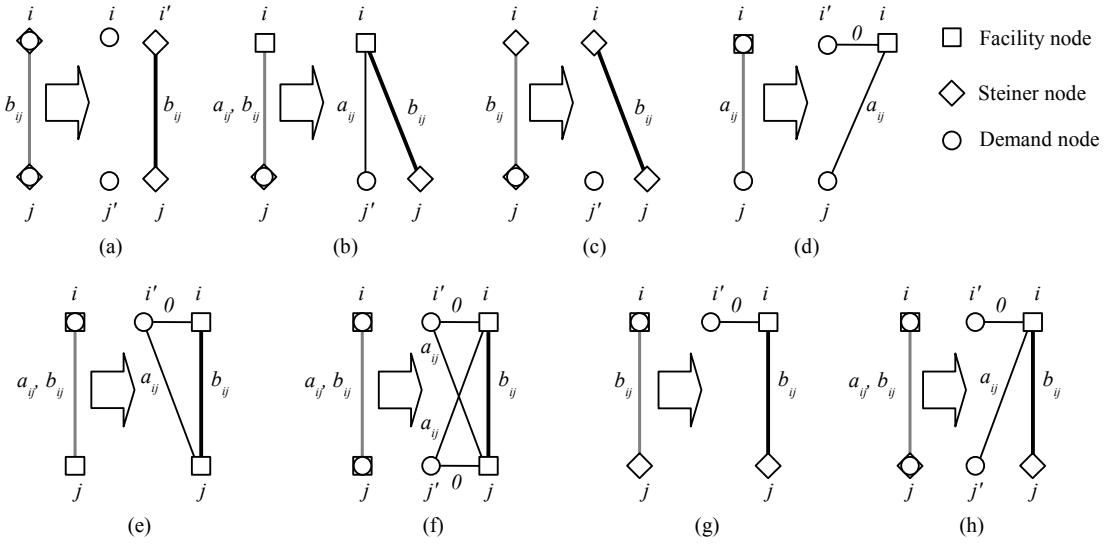

Figure 2.5: Transformations that allow us to assume D, F, and S form a partition in the ConFL problem

indicated in Figure 2.4. There are three cases: (i) either one of the end points is in $F \cap D$ and the other in F which is shown in Figure 2.4(a), or (ii) one of the end points is in $F \cap D$ and the other is in D which is shown in Figure 2.4(b), or (iii) both of the end points are in $F \cap D$ which is shown in Figure 2.4(c). Observe that while in the original GSTS problem instance some edges have different costs depending on the role of the edge (i.e., whether it is an assignment edge or a tree edge), in the transformed GConFL problem each node has a unique role and thus each edge has a unique cost.

2.3.2 Transforming the ConFL problem into the GConFL problem

We first illustrate how an instance of the ConFL problem where $D \cap F \neq \emptyset$ or $D \cap S \neq \emptyset$ can be transformed into a ConFL instance where D, F, and S form

a partition. If a demand node can also be used as a Steiner node, we simply create two copies of the node with one copy representing the node as a demand node and the other copy representing the node as a Steiner node. Similarly, if a demand node can also be used as a facility node, we create two copies of the node with one copy representing the node as a demand node and the other copy representing the node as a facility node. With this duplication, the edges between the duplicated nodes and the remaining nodes in the graph are updated as shown in Figure 2.5. (There are 8 possible cases that are illustrated in the figure). Consequently, without loss of generality, we can assume that in the ConFL problem D, F, and S form a partition.

We now show how to transform an instance of the ConFL problem into an instance of the GConFL problem. In the definition of a ConFL problem instance, the facility opening cost is only incurred when a demand node is assigned to it. To address this situation we duplicate every facility node F in the ConFL problem creating an additional copy i' for every node $i \in F$. The copy i is treated as a facility node in the GConFL problem and the other copy i' is treated as a Steiner node in the GConFL problem. Further, nodes i and i' are connected by a zero cost edge. Edges between a facility i and a node $j \in V$ in the ConFL problem are replaced as shown in Figure 2.6 to transform it into a GConFL problem. There are three cases: (i) either $j \in D$ which is shown in Figure 2.6(a), or (ii) $j \in S$ which is shown in Figure 2.6(b), or (iii) $j \in F$ which is shown in Figure 2.6(c). We should note that it is possible to use fewer edges in this transformation. For example, an alternate transformation with fewer edges is to delete edge $\{i, j\}$ in Figure 2.6(b), and delete edges $\{i, j\}$, $\{i, j'\}$, and $\{i', j\}$ in Figure 2.6(c). However, in terms of our

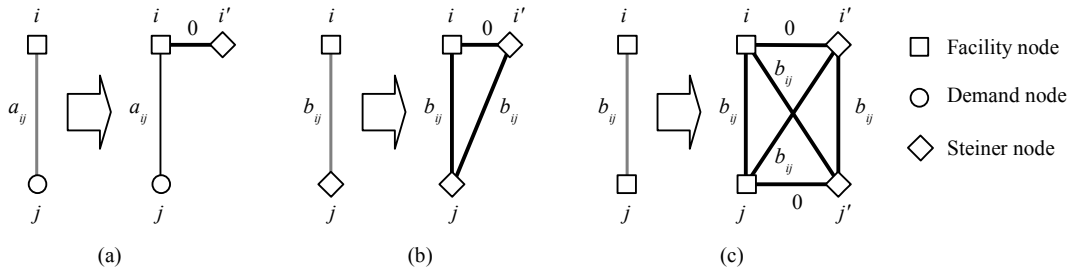

Figure 2.6: ConFL problem transformation to GConFL problem

local search heuristic (described in the next chapter) we found it convenient to use the transformations described in Figure 2.6, since they have the property that if the graph on $G(F \cup S)$ is complete for the ConFL problem, then the graph induced on $G(F \cup S)$ after transformation to the GConFL problem is also complete.

2.3.3 Transforming the ROB problem into the GConFL problem

The ROB problem is a special case of GSTS with $f_i = 0,\ \forall i \in F$. Hence, we apply the transformation described for the GSTS problem in §2.3.1 to convert the ROB problem into a GConFL problem.

2.4 Conclusions

Though the four problems arise in very different settings and so far they have received individual treatment, we have shown that they are special cases of a more general problem and can in fact be treated as one problem. The distinctive and unifying characteristics of these problems are (i) that all of them combine facility location decisions with connectivity requirements and (ii) that demand nodes are

connected through assignment edges dangling from the core tree.

Being able to define a more general problem that encompasses all of them allows us to develop heuristics with broader applicability. Furthermore, it allows us to observe that many of the approximation algorithms and heuristics proposed for some of these problems may in fact be transferable and applicable to the other problems within the family with minimum or no adjustments.

In the following chapters, we use this observation to focus our attention on the GConFL problem with the computational focus on the ConFL problem. However, the heuristics and methodologies developed throughout this dissertation are applicable to the whole family of problems discussed in this chapter.

Chapter 3

Heuristics for the General Connected Facility Connected Problem

3.1 Introduction

In this chapter we propose a powerful heuristic that combines traditional mathematical programming with local search procedures to yield high-quality solutions rapidly for the GConFL problem and consequently for the four related problems. Our heuristic has two significant advantages over the approximation algorithms and heuristics discussed in the literature and reviewed in Chapter 2. The main feature is that it yields a solution within seconds for instances with up to 200 nodes, and within minutes for larger instances with up to 500 nodes, as opposed to within hours for either case as the best-known exact methods. The second advantage of our heuristic is that with each solution it provides a quality measurement in percent, α, that specifies how far the solution is from optimality. In other words, given a solution one knows that the solution is at most $\alpha\%$ from optimality. This is a distinctive characteristic that sets it apart from any of the other heuristics in the literature. While approximation algorithms do have a constant worst-case ratio from optimality, it is often very loose (up to 4.23 times the optimal solution value with the best-known approximation algorithm for the ConFL problem) and it does not provide any concrete information to the decision-maker about the particular solution at hand. Furthermore, our computational results show that the solutions

obtained by our heuristic are consistently within less than 5% from optimality.

Our proposed heuristic, dual-based local search (DLS) heuristic, works in a sequence of stages. In the first stage we transform the problem into a pure directed network design problem with all its costs at the arc level. We specifically model the GConFL problem as a directed Steiner tree problem with a unit degree constraint on the root node. We exploit this transformation to formulate the problem as a multicommodity network flow problem and incorporate the unit degree constraint into the objective function with a sufficiently large Lagrangian multiplier. Next, we use a dual-ascent procedure, which has been shown to be successful for the directed Steiner tree problem, to obtain an initial solution to the problem and a lower bound to the optimal solution. Finally, we improve upon dual-ascent's initial solution by local search movements; closing open facilities and reconstructing the tree on the remaining open facilities, and repeat this process until we cannot find any further improvements in the solution total cost.

As an alternative we present another heuristic that combines mathematical programming and local search procedure. Some of the approximation algorithms found in the literature for the ConFL problem rely on decomposing the problem into two subproblems: uncapacitated facility location problem and Steiner tree problem. Consequently, we devise a heuristic that first solves the facility location problem to optimality disregarding connectivity requirements; secondly, constructs a Steiner tree on the open facilities using dual-ascent; and lastly, improves upon the solution by local search movements. We called this heuristic the uncapacitated facility location (UFL) heuristic. Furthermore, the UFL heuristic allows us to determine

whether the high-quality solutions yielded by the DLS heuristic could be largely attributed to the local search phase. In our computational experiments, we find that even though this seems a reasonable strategy, the solutions are consistently and significantly worse than the solutions yield by the DLS heuristic. We conclude that it is the combination of dual-ascent (on the entirety of the problem) with local-search that produces high-quality solutions.

3.2 Modeling the GConFL Problem as a Directed Steiner Tree Problem with a Unit Degree Constraint on the Root Node

We now discuss how to model the GConFL problem as a directed Steiner tree (DST) problem with a unit degree constraint on the root node. We will use this transformation to apply a dual-ascent strategy to obtain lower and upper bounds for the GConFL problem. We first construct a directed graph $H = (V, A)$ from the graph $G = (V, E)$ of the GConFL problem as follows.

1. Replace every edge $\{i, j\} \in E(F \cup S)$ by two directed arcs (i, j) and (j, i) with cost $c_{ij} = b_{ij}$, if $j \in S$ (and $c_{ji} = b_{ij}$, if $i \in S$), or cost $c_{ij} = b_{ij} + f_j$, if $j \in F$ (and $c_{ji} = b_{ij} + f_i$, if $i \in F$).

2. Replace every assignment edge $\{i, j\}$ between facility i and demand node j by an arc (i, j) with cost $c_{ij} = a_{ij}$.

3. Create an artificial root node s, and create an arc from s to every node $i \in F$ with cost $c_{si} = f_i$.

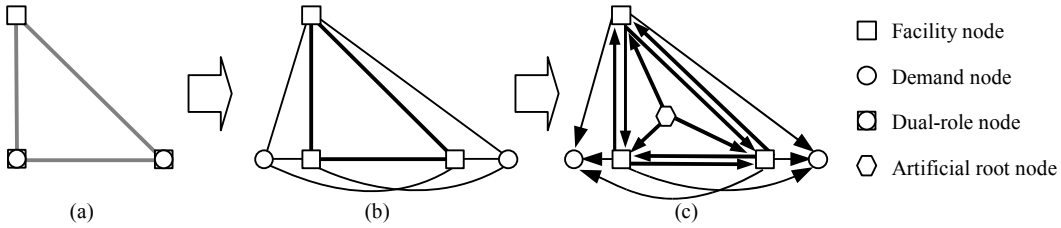

Figure 3.1: Transforming a GConFL instance, derived from the ROB problem, into a directed Steiner tree problem

Figure 3.1 illustrates the transformation for a GConFL instance, derived from a ROB problem. Figure 3.1(b) shows the transformation of the ROB problem into a GConFL problem, and Figure 3.1(c) the transformation to a directed graph. We can now view the GConFL problem as a DST problem on H. On H we would like to construct a minimum cost DST rooted at node s and connected to all demand nodes D (i.e., s has a directed path to every node in D) with the condition that the outdegree of node s is equal to one. (We note that we do not actually need to explicitly introduce an artificial root node s. Instead, we can use any of the demand nodes D as the root node. We will refer to this problem as the unit degree directed Steiner tree (UDDST) problem.) The unit outdegree of node s ensures that the graph obtained after deleting (the artificial root node) s is connected. Notice that on H no costs are associated with any of the nodes, as the facility costs are now included in the arc costs on H. It is easy to see that every feasible solution to GConFL on G can be converted into a feasible UDDST on H with the same cost (simply connect the root node s to one of the facility nodes in the solution, and direct the tree solution to the GConFL problem away from s). Likewise, every feasible solution to the UDDST problem on H corresponds to a feasible solution to the GConFL

problem on G with identical cost (simply delete the root node s, the resulting tree in an undirected sense provides a feasible solution to the GConFL problem). We now provide two formulations for the UDDST problem. Both formulations follow from well-known directed formulations for the Steiner tree problem, with an additional constraint for the unit degree constraint on the root node. The first model is a directed cut model, while the second model is a multicommodity flow model. Both models assume that arc costs are non-negative.

$$z = \text{Minimize} \quad \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{3.1}$$

$$\text{subject to:} \quad \sum_{(i,j)\in\delta^-(R)} y_{ij} \geq 1 \quad \text{for all } R \subset V,\ s \notin R,\ R \cap D \neq \emptyset \tag{3.2}$$

$$\sum_{j\in F} y_{sj} \leq 1 \tag{3.3}$$

$$y_{ij} \geq 0 \quad \text{and integer for all } (i,j) \in A. \tag{3.4}$$

In the above model y_{ij} is an integer variable denoting the number of copies of arc (i, j) in the solution. (We note that since arc costs are non-negative it is sufficient to define y_{ij} as integer instead of binary.) Constraints (3.2) are the standard directed cut constraints, and constraint (3.3) is the degree constraint on the root node. Since there must be at least one arc out of the root node, it is sufficient to define the constraint as an inequality instead of a strict equality. Alternatively, for each node $j \in D$, we create a commodity with the origin as the root node s and node j as the destination node. Let K denote the set of commodities, and $\mathcal{D}(k)$ denote the

destination node of commodity k.

$$z = \text{Minimize} \quad \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{3.5}$$

$$\text{subject to:} \quad \sum_{(j,i)\in A} f_{ji}^k - \sum_{(i,l)\in A} f_{il}^k = \begin{cases} -1 & \text{if } i = s; \\ 1 & \text{if } i = \mathcal{D}(k); \\ 0 & \text{otherwise;} \end{cases} \quad \text{for all } i \in V \;\&\; k \in K \tag{3.6}$$

$$f_{ij}^k \leq y_{ij} \qquad \text{for all } (i,j) \in A \;\&\; k \in K \tag{3.7}$$

$$\sum_{j\in F} y_{sj} \leq 1 \tag{3.8}$$

$$f_{ij}^k \geq 0 \qquad \text{for all } (i,j) \in A \;\&\; k \in K \tag{3.9}$$

$$y_{ij} \geq 0 \qquad \&\text{ integer for all } (i,j) \in A. \tag{3.10}$$

In the above directed flow formulation for the UDDST problem, constraints (3.6) are the standard flow balance constraints. Constraints (3.7) are forcing constraints that require that an arc be in the design, if there is flow on it. Finally, constraint (3.8) is the unit degree constraint on the root node.

3.3 Dual-Based Local Search Heuristic

Our heuristic can be viewed as a two-phase procedure. The first phase is a dual-ascent procedure applied to the UDDST problem that yields both a feasible solution and a lower bound on the optimal solution value. At the conclusion of this phase, we have a feasible solution to the GConFL problem consisting of a set of

open facilities $\mathcal{F} \subseteq F$, a set of Steiner nodes $\mathcal{S} \subseteq S$, and a tree solution on $\mathcal{F}$, $\mathcal{S}$, and D. The second phase is a local search phase that tries to improve the solution obtained by the dual-ascent procedure for the GConFL problem. Our local search heuristic limits itself to improvements that only include nodes in $\mathcal{F}$ and $\mathcal{S}$. In other words, it tries to obtain improvements by finding a better tree on the existing set of nodes, and by closing open facilities and reassigning demand nodes to facilities as needed.

3.3.1 Dual-Ascent Phase

If we were to relax the degree constraint on the root node to the UDDST problem, we obtain the DST problem. Dual-ascent has been a successful solution strategy for the Steiner tree problem. Our first stage is motivated by the desire to utilize this solution strategy to obtain a good lower bound for the GConFL problem as well as a high-quality initial solution for our local search phase. Suppose we relax constraint (3.8) by dualizing it. Then the resulting Lagrangian problem LR(λ) is

$$\begin{aligned} z_{LR}(\lambda) = \text{Minimize} \quad & \sum_{(i,j)\in A} c_{ij} y_{ij} + \lambda(\sum_{j\in F} y_{sj} - 1) \\ \text{subject to:} \quad & (3.6), (3.7), (3.9), (3.10). \end{aligned}$$

Ideally, we would like to obtain the best possible lower bound on z by solving the Lagrangian dual problem $z_{LD} = \max_{\lambda \geq 0} z_{LR}(\lambda)$. It is well-known that $z_{LR}(\lambda)$ is piecewise linear and concave. Further, since $\lambda(\sum_{j\in F} y_{sj} - 1) \geq 0$ for every feasible solution that satisfies constraints (3.6), (3.7), (3.9), and (3.10), $z_{LR}(\lambda)$ is a non-

decreasing function of λ. Intuitively, $z_{LR}(\lambda)$ increases with λ, until it reaches a plateau (i.e., it is flat) for sufficiently large λ. In other words, we can solve the Lagrangian dual problem by solving LR(λ) for a sufficiently large value of λ. Further, when $z_{LR}(\lambda)$ is at a plateau, the solutions to LR(λ) have unit degree (otherwise $z_{LR}(\lambda)$ would not have a slope of 0 at that point) implying that $z_{LD} = z$. Rather than working with the primal problem, we will obtain a lower bound on z by working with the dual of the linear programming relaxation of the directed flow formulation for the UDDST problem. This dual may be stated as follows.

$$t(\lambda) = \text{Maximize} \quad \sum_{k \in K} v^k_{\mathcal{D}(k)} - \lambda \tag{3.11}$$

$$\text{subject to} \quad v^k_j - v^k_i \leq w^k_{ij} \quad \text{for all } (i,j) \in A \text{ and } k \in K \tag{3.12}$$

$$\sum_{k \in K} w^k_{ij} \leq c_{ij} \quad \text{for all } (i,j) \in A, i \neq s \tag{3.13}$$

$$\sum_{k \in K} w^k_{sj} - \lambda \leq c_{ij} \quad \text{for all } (s,j) \in A \tag{3.14}$$

$$w^k_{ij} \geq 0 \quad \text{for all } (i,j) \in A \text{ and } k \in K \tag{3.15}$$

$$\lambda \geq 0. \tag{3.16}$$

Observe that for a given value of λ, $t(\lambda) \leq z_{LR}(\lambda)$. (In fact if we were to move λ to the right hand side of constraint (3.14), the above dual may be viewed as the dual to the linear relaxation of LR(λ).) Solving for $t(\lambda)$ for any choice of $\lambda \geq 0$ provides a lower bound on $z_{LR}(\lambda)$, and in particular solving for $t(\lambda)$ for a sufficiently large value of λ provides a lower bound on z. Thus, our strategy to obtain a lower bound is to choose a sufficiently large value of λ and apply the dual-ascent procedure for

the Steiner tree problem. Notice that the Lagrange multiplier λ may also be viewed as an artificial cost that is added to the cost of the arcs out of the root node to ensure that the solution has outdegree 1 at the root node. We apply the dual-ascent procedure for the Steiner tree as described in Wong (1984) and Balakrishnan et al. (1989), and refined in Raghavan (1995). If we are given values for the $\mathbf{w}$ variables and λ, the dual problem separates by commodity. The subproblem corresponding to commodity k is the dual to the directed shortest path problem between the root node and node $\mathcal{D}(k)$, with arc lengths w_{ij}^k. Dual-ascent iteratively increases the value of the $\mathbf{w}$ variables to improve the dual objective until no further increase is possible. This is achieved in the following fashion. Every arc for which either constraint (3.13) or (3.14) is satisfied at equality is called a tight arc. Any commodity k, that has a directed s–$\mathcal{D}(k)$ cut with no tight arcs across it, is a candidate for improving the dual objective. (We denote the nodes in the destination side of this cut as U_k and refer to them as an ascent set, and to the arcs across this cut as $\delta^-(U_k)$.) The dual objective can be increased by increasing all the w_{ij}^k variables in $\delta^-(U_k)$ until one of the arcs across becomes tight. This is called a basic dual-ascent step. Our dual-ascent implementation uses the commodity cycling rule of Balakrishnan et al. (1989) to choose the commodity and directed cut on which a dual-ascent iteration is performed. This iteratively grows the ascent sets for each commodity U_k, starting from $U_k = \mathcal{D}(k)$. It maintains the property that every node in U_k has a directed path to $\mathcal{D}(k)$ consisting solely of tight arcs. Raghavan (1995) shows this rule to be equivalent to considering minimal ascent sets (a set is a minimal ascent set if it does not contain an ascent set as a strict subset) in the dual-ascent iterations. At the

conclusion of the dual-ascent procedure, we have an approximate dual solution, i.e., a lower bound, and a network of tight arcs (referred to as the auxiliary network) on which there is a directed path from the root node to every demand node. A feasible solution (upper bound for the DST problem) consisting solely of tight arcs is obtained by iteratively deleting arcs in the reverse order in which they become tight, if their deletion from the auxiliary network does not result in a network where there is no directed path from s to a demand node. (A feasible solution consisting solely of tight arcs satisfies the primal complementary slackness conditions.) In our implementation, we add a sufficiently large cost λ to all of the arcs out of the root node. We then apply the dual-ascent procedure for the DST problem to obtain a lower bound and a Steiner tree. If this Steiner tree has outdegree 1 at the root node, it is also a feasible upper bound for the UDDST problem. For a sufficiently large λ, we now argue this is precisely the case. As the value of λ increases, the cost of the arcs out of the root node effectively increases. Thus for a sufficiently large λ, each of the nodes in F will belong to the ascent sets U_k for $k \in D$ before any of the arcs out of the root node s become tight in the dual-ascent procedure. In the next set of dual-ascent steps, one or more arcs out of the root node will become tight, and the dual-ascent procedure will terminate (as there will now be a path from s to all of nodes in D in the auxiliary network). Observe now in the reverse delete procedure to obtain an upper bound, we first delete arcs out of the root node as they became tight last. Consequently, all of the tight arcs out of the root node except for one will be deleted in the reverse delete step, and the upper bound produced by dual-ascent will have outdegree 1.

We make a few final notes before proceeding to the local search phase. Setting $\lambda \geq |F \cup S| \max_{(i,j)\in A} c_{ij}$ ensures that the upper bound is feasible to the UDDST problem (notice that the longest acyclic path from s to a node in D has at most $|F \cup S|$ intermediate nodes, from which the above value is computed). Also, since $z(\lambda)$ and $t(\lambda)$ have $-\lambda$ in the objective function, to obtain upper and lower bounds for the UDDST problem, we should subtract λ from the upper and lower bounds obtained by applying dual-ascent for the DST problem (once the arcs out of the root node have their cost increased by λ).

Following we illustrate this dual-ascent procedure with an example.

Dual-Ascent Example

We consider the example shown in Figure 3.2. Figures 3.2(a) shows the arcs costs, node s is the source node, and $D = \{1, 4\}$. The cost of the arcs out of the artificial source node, s, have already been increased by $\lambda = 20$. Initially, $Z = \{\emptyset\}$, $(\mathbf{v}, \mathbf{w}) = (\mathbf{0}, \mathbf{0})$, and $L = 0$.

The sets $\{1\}$ and $\{4\}$ are minimal ascent sets. The algorithm first performs a basic dual-ascent set step on $\{1\}$. As a result arc $(2, 1)$ becomes tight and is added to Z. The dual objective function increases by 2 units to $L = 2$. The algorithm then performs a basic dual-ascent step on $\{4\}$. As a result arc $(3, 4)$ becomes tight and is added to Z. The dual objective function increases by 1 unit to $L = 3$. Figure 3.2(b) shows the digraph at the conclusion of the first two iterations.

At this point, sets $\{1, 2\}$ and $\{3, 4\}$ are minimal ascent sets. The algorithm first performs a basic dual-ascent step on $\{1, 2\}$. As a result arc $(5, 1)$ and $(3, 1)$ become tight and are added to Z. The dual objective function increases by 1 unit to

$L = 4$. The algorithm then performs a basic dual-ascent step on $\{3, 4\}$. As a result arc $(2, 4)$ becomes tight and is added to Z. The dual objective function increases by 2 units to $L = 6$. Figure 3.2(c) shows the digraph at the conclusion of these two iterations.

At this stage, sets $\{1, 2, 3, 5\}$ and $\{2, 3, 4\}$ are minimal ascent sets. The algorithm performs a basic dual-ascent set step on $\{1, 2, 3, 5\}$. As a result arc $(s, 3)$ becomes tight and is added to Z. The dual objective function increases by 18 units to $L = 24$. The algorithm then tries to perform a basic dual-ascent step on $\{2, 3, 4\}$. However, the arc $(s, 3)$ is already tight.Figure 3.2(d) shows the digraph at the conclusion of these two iterations.

At this point, the network has no ascent set. Therefore, we apply the LIFO drop heuristic. The arcs of Z in the order they were added are $(2, 1)$, $(3, 4)$, $(5, 1)$, $(3, 1)$, $(2, 4)$, and $(s, 3)$. If we delete arc $(s, 3)$ the network defined by the arcs in Z does not contain a path from the source node to the demand nodes 1 and 4. Therefore, we retain arc $(s, 3)$. Following we delete arc $(2, 4)$ and observe that the network defined by the remaining arcs in Z contains a path from the source node to the demand nodes 1 and 4. We continue to proceed in this fashion, deleting arcs in LIFO order from Z and checking that the network defined by remaining arcs contains a path from the source node and the demand nodes 1 and 4. We find the solution shown in Figure 3.2(e). The total cost of this solution is 24 units. Since the lower bound is 24 units, we have obtained the optimal solution to the DST problem. Now, we verify that the out degree of the source node is 1, and consequently, we drop the artificial source node from the solution. We have found the optimal solution to

the ConFL problem with total cost equal to 4 (that is, 24 - λ).

3.3.2 Local Search Phase

In the second phase, we implement a basic version of local search to improve upon the solution yielded by dual-ascent. We search the neighborhood of the dual-ascent solution through a set of improvement steps for a solution of lower cost. If such a solution is found after the improvement steps have been completed, it replaces the current solution, and the search continues. In the search of a lower cost neighboring solution, we implement a set of steps that reconstructs the tree on the set of Steiner nodes $\mathcal{S}$, open facilities $\mathcal{F}$, and demand nodes; closes open facilities; and reassigns demand nodes as needed. Recall, at the conclusion of the dual-ascent phase, we have a feasible solution to the GConFL problem consisting of a set of open facilities $\mathcal{F} \subseteq F$, a set of Steiner nodes $\mathcal{S} \subseteq S$, and a tree solution on $\mathcal{F}$, $\mathcal{S}$, and D. Our local search phase works on the undirected graph associated with the GConFL problem. It tries to improve the solution provided by the dual-ascent procedure by using two types of improvement steps: (1) sequential improvements that try to delete Steiner nodes in $\mathcal{S}$; and (2) local improvements that at each iteration strategically close a facility in $\mathcal{F}$. In the sequential improvements, we construct a minimum spanning tree T on the set of open facility nodes, $\mathcal{F}$, and Steiner nodes, $\mathcal{S}$. Next, we iteratively remove any Steiner node from $\mathcal{S}$ that has degree 2 or less and reconstruct the minimum spanning tree, T. When the graph on $(F \cup S)$ is complete and edge costs satisfy the triangle inequality, this cannot deteriorate the

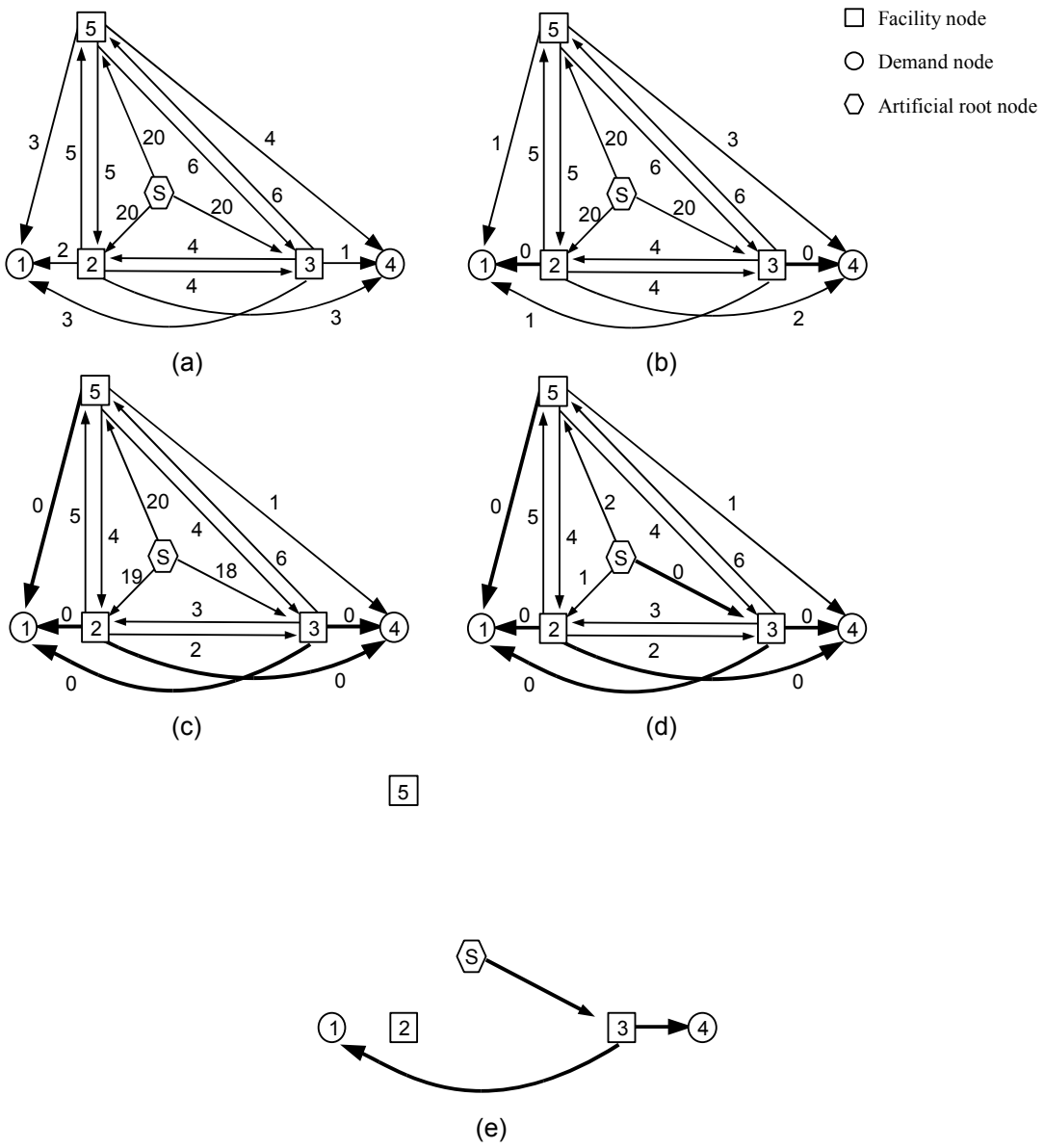

Figure 3.2: Dual-Ascent Example

objective. When the graph on $(F \cup S)$ is not complete or the edge costs do not satisfy triangle inequality, we verify that the cost of the minimum spanning tree T does not increase before removing the Steiner node. (An alternate procedure is to compute all pairs of shortest paths on $G(F \cup S)$ taking into account facility costs, and to (i) complete the graph $G(F \cup S)$, and (ii) set the cost of an edge to the cost of the shortest path between the two end points. This results in the situation where the graph on $(F \cup S)$ is complete and edge costs satisfy the triangle inequality.) At the end of the sequential improvements no Steiner nodes from $\mathcal{S}$ can be deleted without increasing the cost of T. Subsequently, in the local improvements we list open facility nodes first in order of node degree in the tree T and next by the number of demand nodes it serves (obviously, each demand node is connected to the closest open facility on the tree T). Then we move through the list at each iteration removing the next facility node from the solution, reassigning its demand nodes, recomputing the minimum spanning tree on the remaining open facilities and Steiner nodes, and computing the change in the solution cost, that we denote by Δ. If we observe a saving in the solution cost, the facility node is permanently closed and removed from $\mathcal{F}$; otherwise, the facility node is restored to the solution. We iteratively repeat this process for each facility in the list.

In the local improvements, the order in which open facilities are considered for removal is critical. Ordering the nodes in increasing order of node degree in T and number of demand nodes assignments seeks to maximize the savings with each removal based on the two roles that a facility node plays in the final solution. The rationale is to aggressively remove facilities that serve the least number of demand

```
Input: set of Steiner nodes S, facility nodes F and open facility tree T
Output: updated set of Steiner nodes S, facility nodes F and open facility tree T
foreach i ∈ F do
    Calculate i's node degree, ND(i), in T;
    Calculate i's number of assigned demand nodes, D(i);
end
Create an ordered list L and list facility nodes in increasing order of ND(i) and
D(i);
foreach j ∈ L do
    Close facility node j, reassign its demand nodes, restore the tree on the
    remaining open facilities, and compute the change in the solution cost Δ;
    either savings or losses;
    if (Δ ≤ 0) then
        Update the set of open facility nodes F and open facility tree T;
    else
        Restore j into the solution;
    end
end
```

Figure 3.3: Pseudocode for local improvements

nodes and that are farther out in the tree, T. Consequently, we first remove facilities that are leaf nodes in the core tree in increasing order of their number of assigned demand nodes; and then, we move gradually into the tree attempting to close facilities with higher node degree in the tree, T. Our local improvements are described in pseudocode in Figure 3.3. For the local improvements, we tested three somewhat different implementations (and thus definitions of neighboring solutions) that yield solutions with successively greater improvements. In the first implementation, the set of Steiner nodes $\mathcal{S}$ is not updated until the completion of the local improvements. Hence, the local search phase cycles back and forth through sequential improvements followed by local improvements until there are no more improvements. In the second implementation, when Δ is less than or equal to zero, at the time of updating

the set of open facility nodes $\mathcal{F}$ and the tree T; in addition, we also eliminate any Steiner node that has degree 2 or less as specified in the sequential improvements. Hence, in this case at this step the set of Steiner nodes $\mathcal{S}$ is also updated, and the actual improvement is greater than or equal to Δ. The third implementation tries to look ahead and computes Δ by incorporating cost reductions by removing any Steiner node with degree 2 or less from the minimum spanning tree T obtained as a result of removing the facility node under consideration.

Each of these implementations provides successively better solutions (although the improvements are very slight) on average. In terms of running time, we expected each of the implementations to take successively longer times, though we did not observe any time differences on smaller graphs. Thus, we used the third implementation in our local search phase.

3.4 Computational Experiments

We now report on an extensive set of computational experiments with our DLS heuristic on the STS, GSTS, ConFL, and ROB problems. We coded our heuristics in Visual Studio 2005 (C++). We conducted all runs on an AMD Athlon^{TM} 62 X2 Dual, 2.61 GHz machine with 3GB of RAM.

3.4.1 Problem Generation and Characteristics

For the four problems, we generated a large set of Euclidean test problems with varying characteristics. We created five sets of graphs in Euclidean space with

different characteristics in terms of number of D, F, and S nodes, facility opening costs, edge costs, and network density. We generated problems by first selecting nodes randomly located on a 100 x 100 square grid. The Euclidean distances rounded up to the next integer (to preserve triangle inequality) were used as a based for the edge lengths. This problem generation method leads to the most difficult problem instances from a computational standpoint in the context of the STS problem (see Lee et al. 1996). The assignment edge costs are equal to the edge lengths between demand nodes and facility nodes, while tree edge costs are equal to the edge lengths multiplied by an M factor. The M factor illustrates the significantly higher (in terms of cost per unit distance) connection cost of edges in the tree T. The number of demand nodes and facility nodes vary between 10 and 90 in steps of 10, with the total number of demand and facility nodes equal to 100. In an instance of the problem, the facility opening costs are the same for all the facility nodes. For the STS, GSTS and ConFL problems, we generated two sets of "complete" instances with common variations. In Set 1 the facility opening cost varies between 0 and 30 in steps of 10 while the M factor is fixed at 3. In Set 2 the M factor takes values 1, 3, 5 and 7, while the facility opening cost is kept fixed at 30. In the GSTS problem instances, any demand node may host a facility. In the ConFL problem instances, an additional 20 pure potential Steiner nodes were created. Edges were created between the pure Steiner nodes and all of the facility nodes. Since demand nodes cannot be assigned to pure Steiner nodes, no edges were created between demand nodes and pure Steiner nodes. For the ROB problem we used test instances from Set 2 for the GSTS problem and set the facility opening cost to zero. We refer to

this set of instances as Set 3. Notice, in these three sets of instances, any demand node may be assigned to any facility node and the graph $G(F \cup S)$ is complete. Since our DLS heuristic performed exceedingly well on these complete instances, to test the effect of sparsity on the performance of our heuristic, we generated an additional set of test instances, Set 4, for the ConFL problem. We focused on the ConFL problem since we found, from our experiments on Sets 1-3, that they were the hardest to solve amongst the four problems. In this test set the assignment edges and the tree edges were created with a given probability. In addition, to ensure that the instances were connected we randomly constructed a cycle of tree edges through all the facility and potential Steiner nodes; and finally, we verified that there was at least one assignment edge for each demand node. We varied the sparsity of the test instances by using edge creation probabilities of 0.25, 0.50 and 0.75. As in the previous sets of ConFL instances, the number of facility nodes varies between 10 and 90 in steps of 10, and the number of demand nodes is 100 minus the number of facility nodes and the number of pure Steiner nodes is 20. The facility opening cost was 30 and the M factor was set to 7.

As we noted earlier by recomputing edge costs and completing the graph on $G(F \cup S)$, we can in general assume that the graph $G(F \cup S)$ is complete. Thus we hypothesized that in terms of sparsity, the sparsity between demand and facility nodes is more problematic than sparsity between facility nodes. To test this assertion we created a new ConFL test set, Set 5, where the bipartite graph between demand nodes and facility nodes is complete while the graph between facility nodes and potential Steiner nodes is sparse. We varied the sparsity by using edge creation

probabilities of 0.25, 0.50, and 0.75, for the tree edges. The problem instances in Set 5 are always feasible; however, to ensure that they were somewhat equivalent in characteristics to Set 4 (i.e., that all facility and Steiner nodes are potential candidates for the final solution) we randomly create a cycle of tree edges through all the facility and pure Steiner nodes. The remaining characteristics are identical to Set 4.

3.4.2 UFL Heuristic

We were interested in knowing whether the high-quality of the DLS heuristic could be largely attributed to the local search phase. In other words, we wanted to see if the dual-ascent phase provided a high quality initial solution by its selection of facilities to open and Steiner tree T connecting them, or whether any other reasonable choice of a starting solution followed by the local search phase would produce solutions of similar quality to the DLS heuristic. Consequently, we proposed the following UFL heuristic to compare against the DLS heuristic. We first ignore the requirement that the open facilities must be connected to each other by a Steiner tree, and solve the UFL problem (between demand nodes and facility nodes) optimally to obtain a set of facilities to open and the demand nodes assigned to them. We then find a Steiner tree T connecting the open facility nodes by applying dual-ascent for the DST problem to obtain a Steiner tree on $G(F \cup S)$ connecting the set of open facilities. We then use this solution as the starting solution to our local search phase. To solve the uncapacitated facility location problem to optimality

we used CPLEX 10. To obtain the Steiner tree connecting the open facilities, we transformed the graph $G = (V, E)$ to a directed graph $H = (V, A)$ as described in §3.2 except that (i) we use one of the open facilities as the root node and do not create an artificial node (thus no degree constraint is necessary), (ii) all D nodes were deleted, and (iii) the remaining open facilities are the required nodes in our directed Steiner tree.

3.4.3 Results on Complete Graphs

We are now ready to discuss our computational experiments on the different problems. We compare the upper bounds provided by only applying dual-ascent (DA), the DLS heuristic, and the UFL heuristic. We compare the quality of these results by reporting the gaps between the upper bounds provided by the three heuristics and the lower bound obtained by the dual-ascent phase. Each entry in the tables represents the average over 10 instances.

3.4.3.1 STS Problem

Tables 3.1 and 3.2 present our computational results on Set 1 and Set 2, respectively. DA yields relatively good solutions with average gaps below 8.01% for each combination of parameters. However, in some instances the gap can be quite large, and in one instance this reaches 39.72%. Our local search phase is extremely effective and reduces this gap considerably. Our DLS heuristic yields solutions with average gaps below 1.51% and for this specific instance it lowered the gap to 2.80%.

From Table 3.1 it appears that the average percentage gap decreases when the facility opening cost increases. Table 3.2 indicates that the average percentage gap decreases when the M factor increases. The average percentage gap increases at first as the proportion of demand nodes increases, before it decreases again. The location of this peak seems to increase as the M factor is increased. Overall, the performance of our heuristic is stable to a wide range of problem parameters. The average gaps are below 1.51%. The highest gap out of the 630 instances is 3.68% and the average gap computed over all 630 instances is 0.39%. The DLS heuristic is extremely fast and took less than 2 seconds in all instances. In contrast to the DLS heuristic, the UFL heuristic performed quite poorly. The average gaps of the UFL heuristic are significantly larger than the average gaps from the DLS heuristic. This indicates that the dual-ascent phase of the DLS heuristic does a significantly better job identifying which facilities to open than the UFL heuristic does.

$\|D\|$	$\|F\|$	$f_i = 0$			$f_i = 10$			$f_i = 20$			$f_i = 30$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	3.36%	0.03%	4.82%	0.00%	0.00%	3.10%	0.00%	0.00%	7.49%	0.00%	0.00%	5.69%
20	80	7.27%	0.91%	5.38%	0.69%	0.02%	5.41%	2.87%	0.43%	4.62%	0.98%	0.04%	4.12%
30	70	5.39%	1.51%	6.00%	6.66%	1.14%	5.39%	8.01%	0.93%	5.36%	4.02%	0.72%	4.26%
40	60	3.66%	0.95%	4.62%	3.51%	0.78%	4.35%	2.79%	0.51%	5.27%	3.77%	0.79%	5.44%
50	50	2.17%	0.82%	3.72%	2.23%	0.61%	4.04%	4.19%	0.48%	4.27%	5.29%	0.74%	4.12%
60	40	1.91%	0.69%	3.24%	1.91%	0.46%	3.62%	2.30%	0.47%	2.97%	2.24%	0.48%	3.56%
70	30	0.56%	0.26%	1.50%	1.22%	0.41%	1.66%	2.12%	0.50%	2.76%	1.30%	0.40%	2.52%
80	20	0.28%	0.17%	0.35%	0.69%	0.40%	0.79%	0.56%	0.31%	0.83%	1.04%	0.23%	1.30%
90	10	0.05%	0.02%	0.06%	0.25%	0.05%	0.11%	0.07%	0.03%	0.05%	0.10%	0.02%	0.20%

Table 3.1: Comparison of heuristics for the STS problem on Set 1. $M = 3$, and facility opening costs are varied

$\|D\|$	$\|F\|$	$M = 1$			$M = 3$			$M = 5$			$M = 7$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	1.45%	0.37%	5.87%	0.00%	0.00%	5.69%	0.00%	0.00%	15.34%	0.00%	0.00%	15.34%
20	80	5.78%	0.75%	2.76%	0.98%	0.04%	4.12%	0.00%	0.00%	4.69%	0.00%	0.00%	5.69%
30	70	4.43%	0.98%	3.22%	4.02%	0.72%	4.26%	1.59%	0.01%	3.26%	0.00%	0.00%	2.82%
40	60	3.43%	0.43%	1.36%	3.77%	0.79%	5.44%	4.11%	0.40%	5.25%	3.68%	0.13%	3.36%
50	50	2.97%	0.47%	1.08%	5.29%	0.74%	4.12%	5.26%	0.55%	3.71%	6.66%	0.37%	5.00%
60	40	2.97%	0.44%	1.13%	2.24%	0.48%	3.56%	3.56%	0.55%	4.59%	4.10%	0.59%	3.15%
70	30	1.17%	0.25%	0.50%	1.30%	0.40%	2.52%	2.81%	0.43%	3.10%	2.72%	0.24%	4.07%
80	20	0.16%	0.09%	0.19%	1.04%	0.23%	1.30%	1.56%	0.19%	2.59%	3.13%	0.28%	2.38%
90	10	0.00%	0.00%	0.03%	0.10%	0.02%	0.20%	0.17%	0.17%	0.23%	1.39%	0.45%	0.23%

Table 3.2: Comparison of heuristics for the STS problem on Set 2. $f_i = 30$, and M factor is varied

3.4.3.2 GSTS Problem

Our results for the GSTS problem are as promising as the results for the STS problem. Tables 3.3 and 3.4 present our computational results on Set 1 and Set 2. Recall, in these instances any demand node may host a facility. Compared to the STS problem the gaps for all three upper bound heuristics increase slightly on the GSTS problem. For DA, the highest average gap is 12.82% and the worst gap over 630 instances is 31.93%. Once again the local search phase achieves a significant improvement and the average gaps fell below 1.47%. The DLS continues to show a very consistent performance. Out of 630 instances, the worst gap for the DLS heuristic is 4.00% and the average gap over all instances is 0.74%. As the proportion of demand nodes and facility nodes is varied, it appears that instances with a low proportion of demand nodes (10%) are easier, while the remaining instances show similar gaps. Again the DLS heuristic is extremely fast taking at most 3 seconds to obtain a solution for an instance.

$\|D\|$	$\|F\|$	$f_i = 0$			$f_i = 10$			$f_i = 20$			$f_i = 30$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	0.00%	0.00%	5.50%	0.00%	0.00%	5.27%	0.00%	0.00%	5.08%	0.00%	0.00%	13.01%
20	80	2.82%	0.72%	3.23%	5.77%	0.29%	3.80%	3.05%	0.16%	4.77%	2.55%	0.20%	3.84%
30	70	3.20%	0.81%	2.92%	9.33%	1.47%	3.22%	8.77%	1.01%	6.09%	7.32%	0.70%	4.78%
40	60	3.40%	0.89%	2.93%	8.48%	1.16%	5.17%	8.01%	0.78%	4.85%	8.30%	1.05%	5.62%
50	50	4.13%	0.93%	2.79%	9.28%	1.28%	4.42%	12.82%	1.46%	4.31%	10.62%	0.99%	4.72%
60	40	2.07%	0.59%	3.02%	8.38%	0.98%	4.53%	9.35%	1.12%	4.22%	8.55%	1.25%	3.63%
70	30	2.85%	0.99%	2.83%	7.59%	1.34%	5.33%	9.93%	1.06%	4.20%	9.01%	1.22%	4.15%
80	20	2.39%	0.77%	3.50%	9.61%	1.33%	4.85%	10.67%	1.22%	5.00%	10.32%	1.31%	5.00%
90	10	2.22%	0.83%	3.45%	7.82%	1.06%	4.33%	10.88%	1.14%	4.15%	8.77%	0.74%	5.14%

Table 3.3: Comparison of heuristics for the GSTS problem on Set 1. $M = 3$, and facility opening costs are varied

$\|D\|$	$\|F\|$	$M = 1$			$M = 3$			$M = 5$			$M = 7$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	2.06%	0.42%	5.10%	0.00%	0.00%	13.01%	0.00%	0.00%	20.66%	0.00%	0.00%	20.66%
20	80	12.63%	1.42%	3.77%	2.55%	0.20%	3.84%	0.00%	0.00%	3.95%	0.00%	0.00%	6.00%
30	70	5.81%	0.63%	2.88%	7.32%	0.70%	4.78%	2.57%	0.11%	4.90%	0.00%	0.00%	4.03%
40	60	7.91%	0.60%	2.11%	8.30%	1.05%	5.62%	6.10%	0.69%	3.68%	2.26%	0.16%	2.22%
50	50	6.97%	0.59%	2.27%	10.62%	0.99%	4.72%	4.22%	0.55%	5.28%	4.85%	0.58%	4.71%
60	40	7.22%	0.42%	1.48%	8.55%	1.25%	3.63%	9.78%	1.21%	4.68%	6.12%	1.18%	5.53%
70	30	5.18%	0.48%	1.79%	9.01%	1.22%	4.15%	11.09%	1.11%	5.01%	3.90%	0.44%	5.57%
80	20	5.17%	0.40%	1.86%	10.32%	1.31%	5.00%	5.62%	0.89%	5.81%	7.65%	0.94%	5.20%
90	10	5.86%	0.58%	1.55%	8.77%	0.74%	5.14%	8.82%	0.96%	5.57%	12.23%	1.13%	5.01%

Table 3.4: Comparison of heuristics for the GSTS problem on Set 2. $f_i = 30$, and M factor is varied

The solutions obtained by the UFL heuristic on the GSTS problem are quite poor. Again, we can state that the dual-ascent phase does an excellent job finding a subset of facility nodes to open (and Steiner nodes to use). The difference between the DLS and UFL heuristics is more marked when the number of facility nodes in an instance is large. Here the problem of identifying the set of facility nodes to open is combinatorially more challenging, and in these instances the UFL heuristic does very badly.

3.4.3.3 ROB Problem

We now focus our attention on the ROB problem. Table 3.5 summarizes our results. The ROB problem with $M = 1$ is essentially a Steiner tree problem. Consequently, for $M = 1$ DA yields high-quality solutions for all combinations of demand nodes and facility nodes, which our local search phase improves even further. As the M factor increases, on average the average gap for DA and the DLS heuristic increase. The average gap for DA is less than 8% for all combinations of parameters. With the addition of the local search phase the results are even better. Our DLS heuristic has average gaps below 0.5% for $M = 1$, and below 2.02% for all other combinations of parameters. Over the entire set of 360 instances, the average gap is below 0.70% for the DLS heuristic. At the instance level, the highest gap for the DLS heuristic remains below 4%, while the worst gap for DA over the 360 instances is 22.29%. Once again, the consistency of the results obtained by the DLS heuristic is quite compelling. Again the DLS heuristic is extremely fast taking at most 3

seconds to obtain a solution for an instance.

Recall, in the case of the ROB problem facilities can be opened at any node in the graph including demand nodes and the facility opening cost is zero. Consequently, for the UFL heuristic, the optimal solution to the uncapacitated facility location problem is to open a facility at each demand node. Thus, the starting solution for the UFL heuristic is simply the set of facilities at the demand nodes, which later is enhanced by dual-ascent to include Steiner nodes and by the local search to improve the total solution cost. Again, the average gaps obtained by UFL are consistently worse than the ones achieved by the DLS heuristic. Specifically, at the instance level in 351 out of 360 instances the solution from the DLS heuristic was superior to the UFL heuristic.

$\|D\|$	$\|F\|$	$M=1$			$M=3$			$M=5$			$M=7$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	2.30%	0.35%	1.93%	0.00%	0.00%	5.50%	0.00%	0.00%	6.17%	0.00%	0.00%	6.17%
20	80	0.90%	0.15%	0.66%	2.82%	0.72%	3.23%	2.51%	0.48%	3.08%	0.00%	0.00%	3.95%
30	70	0.49%	0.00%	0.58%	3.20%	0.81%	3.04%	3.94%	1.02%	3.65%	1.26%	0.07%	3.46%
40	60	0.38%	0.04%	0.43%	3.40%	0.89%	2.93%	2.69%	1.39%	4.21%	7.94%	1.14%	4.79%
50	50	0.33%	0.08%	0.22%	4.13%	0.93%	2.79%	5.52%	1.19%	4.21%	5.13%	1.19%	3.73%
60	40	0.09%	0.02%	0.16%	2.07%	0.59%	3.02%	4.35%	1.37%	5.79%	5.74%	1.62%	4.93%
70	30	0.10%	0.02%	0.18%	2.85%	0.99%	2.83%	4.55%	1.33%	4.19%	3.55%	1.37%	4.51%
80	20	0.02%	0.00%	0.11%	2.39%	0.77%	3.50%	4.40%	1.21%	5.04%	5.95%	1.74%	5.40%
90	10	0.00%	0.00%	0.06%	2.22%	0.83%	3.45%	2.95%	1.04%	4.62%	5.43%	2.02%	6.00%

Table 3.5: Comparison of heuristics for the ROB problem on Set 3. M factor is varied

3.4.3.4 ConFL Problem

Tables 3.6 and 3.7 summarize our computational results for Set 1 and Set 2, respectively. The gaps for DA exhibit a concave behavior as the fraction of demand nodes increases. For a low proportion of demand nodes the gaps are low. They rapidly jump as the proportion of demand nodes increases before decreasing again as the proportion of demand nodes gets higher. For DA the average percentage gaps get as high as 16%. On the other hand, the average percentage gaps for the DLS heuristic are always below 4.27%. In all cases the local search phase significantly improved the upper bounds obtained from the dual-ascent phase. Over 630 instances the worst gap for the DLS heuristic was 7.72%, in contrast to a worst gap of 29.67% for DA. Averaged over the 630 instances the average gap for the DLS heuristic was 1.74%. We observed that as the facility opening costs increase the average gaps for the DLS heuristic increase. In contrast, as the M factor increases the average gap for the DLS heuristic first increases but later decreases reaching the maximum at $M = 3$. In summary, our results continue to indicate that the DLS heuristic's performance is stable over a wide range of parameters. We note that the DLS heuristic took at most 4.74 seconds to solve an instance of the ConFL problem. The performance of the UFL heuristic on the ConFL instances was significantly worse than the DLS heuristic. As before, it appears the dual-ascent phase is quite consistent in identifying a good starting solution for the local search phase.

$\|D\|$	$\|F\|$	$f_i = 0$			$f_i = 10$			$f_i = 20$			$f_i = 30$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	0.54%	0.08%	1.76%	2.30%	0.48%	4.49%	2.18%	0.30%	8.43%	0.00%	0.00%	10.59%
20	80	6.79%	0.91%	5.52%	12.41%	3.16%	4.94%	11.38%	3.05%	5.17%	15.62%	3.95%	7.90%
30	70	4.86%	1.57%	5.32%	9.95%	2.80%	5.33%	14.24%	4.27%	5.71%	14.13%	3.70%	5.35%
40	60	5.31%	1.80%	4.83%	9.80%	2.63%	6.19%	8.97%	3.02%	6.69%	14.13%	3.97%	6.17%
50	50	3.83%	1.43%	3.37%	7.90%	2.45%	3.94%	8.73%	2.94%	5.55%	10.08%	3.27%	4.85%
60	40	3.50%	0.96%	2.63%	5.51%	1.80%	3.77%	6.86%	2.29%	4.31%	7.54%	3.04%	4.93%
70	30	1.28%	0.81%	1.69%	2.70%	1.13%	1.93%	4.88%	1.57%	3.46%	5.44%	1.89%	3.23%
80	20	0.53%	0.28%	14.98%	1.02%	0.57%	1.18%	1.84%	0.79%	1.24%	3.24%	1.08%	1.28%
90	10	0.23%	0.17%	0.17%	0.23%	0.02%	0.13%	0.43%	0.05%	0.30%	0.09%	0.04%	0.28%

Table 3.6: Comparison of heuristics for the ConFL problem on Set 1. $M = 3$, and facility opening costs are varied

$\|D\|$	$\|F\|$	$M = 1$			$M = 3$			$M = 5$			$M = 7$		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	10.30%	3.25%	5.83%	0.00%	0.00%	10.59%	0.00%	0.00%	10.59%	0.00%	0.00%	10.59%
20	80	12.21%	3.15%	4.22%	15.62%	3.95%	7.90%	8.29%	1.11%	5.76%	0.94%	0.01%	6.08%
30	70	7.90%	2.47%	2.93%	14.13%	3.70%	5.35%	7.74%	2.97%	5.72%	3.17%	1.36%	6.67%
40	60	7.18%	1.92%	3.24%	14.13%	3.97%	6.17%	12.23%	3.76%	5.37%	8.22%	2.73%	6.43%
50	50	5.32%	1.63%	1.53%	10.08%	3.27%	4.85%	7.83%	3.03%	5.93%	7.69%	3.01%	7.56%
60	40	4.78%	1.40%	2.12%	7.54%	3.04%	4.93%	9.05%	3.54%	5.56%	8.39%	2.41%	6.57%
70	30	2.59%	0.84%	0.75%	5.44%	1.89%	3.23%	4.83%	2.08%	4.66%	6.17%	2.24%	6.14%
80	20	1.56%	0.33%	0.32%	3.24%	1.08%	1.28%	3.06%	1.49%	2.27%	4.43%	1.64%	3.19%
90	10	0.00%	0.00%	0.19%	0.09%	0.04%	0.28%	0.30%	0.28%	0.47%	1.14%	0.56%	0.51%

Table 3.7: Comparison of heuristics for the ConFL problem on Set 2. $f_i = 30$, and M factor is varied

3.4.4 Results on Sparse Instances

Table 3.8 summarizes our computational results on Set 4. Recall that Set 4 was created to understand the performance of the DLS heuristic for sparse instances, and that we focussed our attention on the ConFL problem as it was the hardest (in terms of gaps) of the four problems to solve on complete graphs. Table 3.8 indicates that as the graph gets sparser the average gaps increase considerably for all three upper bound heuristics. Average gaps for DA go up to 18.12% and for the DLS heuristic increase to 14.36%. For individual instances, the worst case gap for DA is 37.16% while the worst case gap for the DLS heuristic is 22.38%. Notice, however, that both heuristics yield significantly better solutions than the UFL heuristic which is disastrous. Its worst case gap for individual instances is 151.04% and the average gap gets as high as 76.74%.

In interpreting the results in Table 3.8 we wanted to understand whether the large gaps for the DLS heuristic on sparse instances were due to the quality of the upper bounds or the lower bounds produced by the dual-ascent phase. To address this issue we were very fortunate to access Dr. Ljubić's state of the art B&C code (Ljubić 2007) as an alternate method of generating lower bounds. We limited the running time of the B&C code to 1 hour for each instance, and use the better lower bound from both the B&C code and the dual-ascent procedure to recalculate the average gaps for our DLS heuristic. These recomputed average gaps are shown in Table 3.9. The column DLS indicates the average gaps obtained using the dual-ascent phase's lower bound, while the column DLS' indicates the recomputed average

$\|D\|$	$\|F\|$	Probability of Edge Creation								
		0.25			0.50			0.75		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	10.63%	9.72%	74.71%	5.48%	5.17%	76.74%	3.71%	2.57%	43.29%
20	80	18.12%	14.36%	50.45%	7.15%	5.44%	38.55%	4.51%	2.90%	34.48%
30	70	12.36%	10.44%	38.68%	12.64%	7.92%	25.71%	10.50%	4.99%	25.04%
40	60	16.15%	12.28%	32.09%	11.18%	7.31%	24.40%	10.52%	4.54%	15.66%
50	50	12.87%	8.86%	24.41%	8.75%	5.33%	16.87%	10.85%	4.86%	12.03%
60	40	9.28%	7.06%	20.31%	9.47%	5.45%	14.78%	9.53%	4.77%	10.59%
70	30	7.69%	4.81%	14.56%	7.25%	4.26%	11.34%	8.00%	3.49%	8.42%
80	20	3.98%	3.06%	5.06%	4.50%	2.73%	5.57%	4.69%	2.20%	4.88%
90	10	0.30%	0.11%	0.16%	0.29%	0.22%	1.46%	1.34%	0.56%	1.09%

Table 3.8: Comparison of heuristics for the ConFL problem on Set 4

$\|D\|$	$\|F\|$	Probability of Edge Creation								
		0.25			0.50			0.75		
		DLS'	DLS	Imp	DLS'	DLS	Imp	DLS'	DLS	Imp
10	90	1.81%	9.72%	9	1.26%	5.17%	9	1.42%	2.57%	4
20	80	4.78%	14.36%	10	0.38%	5.44%	10	0.20%	2.90%	9
30	70	3.03%	10.44%	10	3.09%	7.92%	10	2.48%	4.99%	7
40	60	4.50%	12.28%	10	2.92%	7.31%	9	2.82%	4.54%	5
50	50	1.48%	8.86%	10	0.74%	5.33%	10	2.59%	4.86%	7
60	40	1.52%	7.06%	10	1.26%	5.45%	10	2.01%	4.77%	8
70	30	0.29%	4.81%	10	1.04%	4.26%	10	0.66%	3.49%	10
80	20	0.50%	3.06%	10	0.45%	2.73%	10	0.31%	2.20%	10
90	10	0.11%	0.11%	0	0.05%	0.22%	4	0.14%	0.56%	8

Table 3.9: Performance of the DLS heuristic computed using the best lower bound on Set 4

gaps. The column Imp indicates the number of times over the 10 instances B&C was able to improve the lower bound. In most cases average gaps decrease to one third of their previous value. B&C is able to obtain tighter lower bounds in practically all of the very sparse (probability 0.25) instances except for those with 90 demand nodes. However, as the network becomes more dense it becomes a little harder for B&C to improve the lower bound and even when the lower bound is improved the relative improvement is much smaller. We note that the running times of the B&C code increase as the density of the graph increases.

The analysis in Table 3.9 indicates that DLS heuristic performs quite well and is consistent. As the density of the graph increases its performance improves. The average gaps are always lower than 4.78%. Over the 270 instances, the average gap of the DLS heuristic is 1.55%. The worst gap over the 270 instances for the DLS heuristic drops to 16.43%. Incidentally, the instance with the 16.43% gap is one where the lower bound from B&C is worse than the dual-ascent lower bound (meaning that even after 1 hour of running time it is unable to improve the dual-ascent lower bound). In summary, the quality of the solutions obtained by the DLS heuristic are consistently of high quality, even for sparse problems. We now report on our computational experience with Set 5. Recall instances in Set 5 are sparse problems where the bipartite graph between demand nodes and facility nodes is complete. Table 3.10 summarizes the results of the three heuristics with the gaps computed using the dual-ascent lower bound. These results are quite similar to those for the three heuristics on Set 1 and Set 2. In other words, the quality of the solutions provided by the DLS heuristic are quite high. In particular, the average gap of the DLS heuristic is below 2.46% (or 2.05% when computed using the best lower bound). Over the 270 instances the worst gap for the DLS heuristic is 5.09% (or 4.72% with the best lower bound) in contrast to the worst gap for DA which is 34.58%. The average gaps for the DLS heuristic increase as the instances become denser. Averaged over the 270 instances the DLS heuristic has a gap of 1.12% (or 0.46% with the best lower bound).

Table 3.11 shows the gaps for the DLS heuristic gaps recomputed using the best of the lower bounds from the B&C code and dual-ascent. The results in the

$\|D\|$	$\|F\|$	Probability of Edge Creation								
		0.25			0.50			0.75		
		DA	DLS	UFL	DA	DLS	UFL	DA	DLS	UFL
10	90	0.00%	0.00%	14.39%	0.00%	0.00%	11.91%	0.00%	0.00%	10.59%
20	80	0.00%	0.00%	12.48%	0.00%	0.00%	9.15%	0.00%	0.00%	5.59%
30	70	2.54%	0.80%	5.99%	3.43%	0.72%	5.50%	3.38%	1.18%	4.43%
40	60	7.23%	0.96%	8.74%	5.59%	1.47%	7.36%	9.91%	2.46%	7.30%
50	50	4.31%	1.47%	7.02%	7.52%	1.94%	6.10%	6.98%	1.78%	6.18%
60	40	2.56%	1.30%	11.57%	6.01%	2.19%	8.98%	8.62%	2.43%	6.34%
70	30	4.07%	1.42%	6.41%	4.94%	1.74%	6.10%	5.27%	2.32%	5.35%
80	20	2.71%	1.24%	4.57%	6.50%	1.55%	4.34%	3.72%	1.90%	3.77%
90	10	1.00%	0.48%	3.08%	0.64%	0.29%	2.21%	1.28%	0.72%	1.02%

Table 3.10: Comparison of heuristics for the ConFL problem on Set 5

$\|D\|$	$\|F\|$	Probability of Edge Creation								
		0.25			0.50			0.75		
		DLS'	DLS	Imp	DLS'	DLS	Imp	DLS'	DLS	Imp
10	90	0.00%	0.00%	0	0.00%	0.00%	0	0.00%	0.00%	0
20	80	0.00%	0.00%	0	0.00%	0.00%	0	0.00%	0.00%	0
30	70	0.29%	0.80%	7	0.24%	0.72%	5	0.71%	1.18%	5
40	60	0.34%	0.96%	10	0.64%	1.47%	6	2.05%	2.46%	4
50	50	0.44%	1.47%	10	0.86%	1.94%	10	0.96%	1.78%	4
60	40	0.20%	1.30%	10	1.04%	2.19%	10	1.08%	2.43%	8
70	30	0.35%	1.42%	10	0.60%	1.74%	10	0.69%	2.32%	9
80	20	0.43%	1.24%	9	0.35%	1.55%	10	0.75%	1.90%	9
90	10	0.14%	0.48%	7	0.10%	0.29%	7	0.23%	0.72%	10

Table 3.11: Performance of the DLS heuristic computed using the best lower bound on Set 5

table indicate that B&C yielded a slightly smaller number of improvements, and at the same time the improvements in the lower bound were less pronounced. In summary, we can conclude that sparsity between demand nodes and facility nodes causes a weaker dual-ascent lower bound and hence a wider gap when it is used to compute the average gaps. In other words, ConFL problems with sparsity between demand nodes and facility nodes are hardest for our DLS heuristic.

3.4.5 Large-Scale Instances and Comparison to Ljubić's VNS heuristic

We tested the DLS heuristic on a set of large-scale instances introduced by Ljubić (2007) for the ConFL problem. She constructed ConFL test problems by combining Steiner tree problem instances from OR-Library and UFL problem instances from UflLib (see Ljubić (2007) for more details about these test instances). These instances are representative for each type of problem. However, their combination departs from the typical convention that we have seen in the literature on the four problems in that tree edges are typically more expensive per unit length than assignment edges. In the instances constructed in Ljubić (2007) the cost of tree edges is given by the Steiner tree instances while the cost of the assignment edges is determined by the UFL instances with no attempt to scale these costs (and thus the per unit length cost of tree edges is cheaper than the per unit length cost of assignment edges). However, testing the DLS heuristic on these instances gives us the opportunity to evaluate its performance on larger instances (up to 1300 nodes) that are on non-Euclidean graphs, and compare their performance against a variable neighborhood search (VNS) heuristic that was used in Ljubić (2007). In these instances knowledge of an open facility in the solution was assumed a priori. Consequently, we modified our DLS heuristic to incorporate this knowledge. Hence, in the dual-ascent phase we simply use the open facility as the root node, and do not create an artificial source node s or impose a unit degree constraint. In addition, in the local search phase this open facility is never removed even if its removal would

result in a lower cost solution.

Table 3.12 shows the results obtained by DA, the DLS heuristic, and Ljubić's VNS procedure. (To compute the gaps we use dual-ascent's lower bounds since it is our understanding that there is an error (Ljubić 2009) in the lower bounds reported on Ljubić (2007).) Over the 48 test instances reported in Table 3.12, the worst gap of DA is 17.23%, VNS's worst gap is 21.16%, while the DLS heuristic's worst gap is 8.63%. Averaged over the 48 instances, the average gap of the DLS heuristic is 3.83%, while VNS's average gap is 5.75%. The DLS heuristic finds a better solution than the VNS procedure in 36 out of the 48 instances. The average computational time required by VNS (these are as reported in Ljubić 2007) and our DLS heuristic appear to be quite similar and around 500 seconds. The running times reported in Ljubić (2007) are the average over ten runs of the VNS procedure, and the solution reported is the best over the ten instances. So a more accurate assessment of the running time of the VNS procedure would be a tenfold increase in the running times reported. However, our heuristic shows a lot more variability with running times as high as 5422.2 seconds. On the other hand the gaps of the DLS heuristic are lower, have much less variability, and are quite stable to a wide range of parameters. We should note that the excessive running times for the DLS heuristic seem to occur in the last group of 16 instances. If we use the second implementation of the local improvements in the local search phase instead, the running time goes down considerably and the performance actually does not deteriorate. For example, the instance that takes 5422.2 seconds takes 1640.5 seconds with the second implementation, and its gap increases from 0.45% to 0.46%. Consequently, when computational time is

important and the instance has a large number of nodes, we could either impose a time limit to the DLS heuristic, or use an alternate implementation (i.e., the second implementation).

3.5 Conclusions

In this chapter we considered a family of four important network design problems that combine facility location with connectivity requirements. We provided a common framework and methodology to address these four problems. In particular, we devise a DLS heuristic that works in two phases. In the first phase it applies dual-ascent to obtain both a lower bound and an initial solution to the problem. In the second phase it applies local search, limiting its attention to the set of facilities and Steiner nodes selected in the first phase. From a broad algorithmic perspective our work is closely related to the primal-dual algorithm by Swamy and Kumar (2004), with approximation ratio 8.55. However, while Swamy and Kumar's focus is to develop approximation algorithms with provable worst case bounds, our goal is to find tight formulations that combined with local search perform effectively in practice. In that sense our approach is more comprehensive. Although the first phase of Swamy and Kumar's heuristics attempts to incorporate some of the connectivity requirement costs in the selection of open facilities (by insisting that each facility serve a minimum number of demand points), the final cost of the core tree network is only considered in the second phase when facilities have already been selected. Our approach selects the open facilities and constructs a Steiner tree among open

facilities simultaneously ensuring a global treatment of the problem. Our extensive computational experiments show that across the family of problems, our DLS heuristic obtains high quality solutions rapidly. Further, the results are quite consistent in the sense that the variance of the performance gap is quite low, and smaller than the other heuristics considered for the problem. Among the four problems, the ConFL problem seems to be the hardest to solve for the DLS heuristic. In particular ConFL instances with higher sparsity between demand nodes and facility nodes are harder for our DLS

Our heuristic can be viewed as one that successfully partners mathematical programming approaches (i.e., dual-ascent) with local search. As our experiments show both phases of the heuristic strategy contribute to its success. When we replaced the DA solution by an initial solution from the UFL heuristic, the local search phase found solutions that were significantly worse. The local search phase also significantly improves the solution obtained from the dual-ascent phase. We should note that on the large scale ConFL problem instances our DLS procedure significantly outperformed a state-of-the-art VNS procedure. In contrast to other heuristic methodologies, another significant advantage of our DLS heuristic is that it provides a high-quality lower bound along with each solution; thus providing a guarantee on the quality of the solution.

Instances		Gap			Time (sec)		
UFLP	STP	DA	DLS	VNS_{best}	DA	DLS	$VNS_{average}$
mp1	c5	5.88%	4.29%	13.14%	68.03	69.34	98.60
mp2	c5	9.84%	5.34%	9.95%	87.44	89.61	363.10
mp1	c10	5.76%	4.59%	9.58%	61.55	63.00	389.00
mp2	c10	5.41%	5.13%	5.85%	78.34	79.05	187.20
mp1	c15	5.45%	4.94%	7.75%	77.91	79.58	406.40
mp2	c15	9.34%	6.59%	9.18%	84.41	85.00	186.10
mp1	c20	5.35%	4.66%	6.05%	128.30	128.95	300.30
mp2	c20	9.28%	4.21%	5.72%	95.38	95.66	305.20
mq1	c5	11.08%	6.78%	21.16%	355.73	357.52	177.70
mq2	c5	12.74%	5.32%	16.20%	413.98	416.00	353.00
mq1	c10	10.79%	6.62%	10.67%	332.91	334.61	365.00
mq2	c10	17.23%	4.79%	10.29%	387.47	392.13	340.10
mq1	c15	10.63%	7.06%	12.21%	346.64	347.89	528.90
mq2	c15	11.96%	4.50%	6.01%	367.23	367.81	250.50
mq1	c20	10.48%	7.15%	10.83%	376.11	376.75	401.20
mq2	c20	16.65%	4.73%	7.40%	667.44	668.59	375.50
mp1	d5	6.00%	4.06%	5.52%	62.36	65.88	402.90
mp2	d5	15.70%	5.64%	5.32%	84.08	87.61	482.30
mp1	d10	5.73%	4.25%	6.19%	78.80	82.39	366.80
mp2	d10	9.70%	5.08%	3.01%	107.34	111.03	365.00
mp1	d15	5.44%	4.97%	6.62%	65.52	68.25	328.50
mp2	d15	14.28%	6.37%	5.02%	86.92	90.89	379.00
mp1	d20	5.31%	4.22%	4.83%	118.86	120.86	453.40
mp2	d20	9.27%	6.66%	2.86%	359.92	361.61	321.90
mq1	d5	10.88%	7.11%	9.12%	339.34	346.48	508.10
mq2	d5	8.81%	5.24%	11.52%	391.75	395.56	460.70
mq1	d10	10.87%	5.88%	8.78%	360.88	364.67	511.10
mq2	d10	12.30%	4.31%	6.43%	422.13	430.11	593.80
mq1	d15	10.68%	6.42%	6.77%	354.23	358.02	652.80
mq2	d15	12.15%	4.64%	7.45%	482.49	488.17	627.00
mq1	d20	10.59%	7.32%	10.47%	474.31	477.78	490.40
mq2	d20	16.57%	8.62%	6.47%	535.13	536.89	495.50
gs250a_1	c5	0.73%	0.52%	0.36%	10.41	292.81	523.50
gs250a_2	c5	0.59%	0.37%	0.30%	5.58	364.83	458.30
gs250a_1	c10	0.57%	0.37%	0.39%	6.09	272.66	668.50
gs250a_2	c10	0.69%	0.54%	0.38%	6.61	406.28	341.70
gs250a_1	c15	0.66%	0.45%	0.34%	15.13	218.06	548.60
gs250a_2	c15	0.55%	0.38%	0.27%	16.75	103.16	598.30
gs250a_1	c20	0.67%	0.41%	0.36%	140.13	160.53	598.00
gs250a_2	c20	0.67%	0.41%	0.28%	121.28	144.88	697.40
gs500a_1	c5	0.58%	0.41%	0.68%	36.42	2312.39	838.10
gs500a_2	c5	0.66%	0.45%	0.72%	40.73	5422.21	845.90
gs500a_1	c10	0.51%	0.41%	0.63%	27.55	1595.16	881.10
gs500a_2	c10	0.52%	0.37%	0.66%	29.28	1373.59	939.50
gs500a_1	c15	0.60%	0.39%	0.61%	84.63	1081.00	928.00
gs500a_2	c15	0.51%	0.36%	0.60%	138.77	837.94	871.20
gs500a_1	c20	0.58%	0.38%	0.55%	568.05	717.47	943.80
gs500a_2	c20	0.57%	0.36%	0.51%	111.27	216.88	906.00

Table 3.12: Comparison of the DLS heuristic with the VNS procedure on large-scale instances

Chapter 4

The Stochastic Connected Facility Location Problem

4.1 Introduction

Most of the applications presented in Chapter 2 have associated uncertainty which is often simplified in the abstraction of the problem. For example, in the caching problem described by Krick et al. (2003), the exact number of read and write requests is unknown at the time that the network is designed. Similarly, in the problem described by Nuggehalli et al. (2003) the assignment cost might not be revealed until the last moment when the assignment edges are rented. In both problems the literature assumes average values as an approximation; however, under some circumstances we might wish to obtain the solution of the stochastic problem, perhaps to assess the validity of the approximation.

In this chapter we seek to explore the value of explicitly modeling uncertainty into the ConFL problem. Here we introduce a variant of the Connected Facility Location (ConFL) problem that addresses these uncertainties and we call it the Stochastic ConFL (SConFL) problem. When uncertainties exist, the objective is to minimize the expected cost of the network.

In an instance of the SConFL problem, there are certain and uncertain costs. The facility opening costs and the connection costs between them is assumed to be known beforehand while the assignment costs are unknown and dependent on a

random scenario that causes them to be uncertain. This type of problem is known as a two-stage stochastic problem with fixed recourse as originated by Dantzig (1955) and described by Birge and Louveaux (1997). The two stages of the problem are as follows. In the first stage, a set of facilities must be opened and a Steiner tree that connects them constructed. In the second stage, uncertainty on the assignment costs is unveiled (i.e., one scenario is realized) and customers must be assigned to open facilities. The objective is to minimize the network design cost including the core network—opening of facilities and their connection—and the expected assignment cost. This sequence of events clearly describes the problem introduced by Krick et al. (2003) in the data management setting where for operational reasons the construction of the core network must be performed ahead of time while the actual requests to read and/or write data from the servers come later in time.

In the SConFL problem we can identify two types of uncertainty: (i) each customer's demand quantity and (ii) each customer's location (or travel time to potential facilities). Both types of uncertainty affect the assignment costs; however, they have a very different impact that requires independent treatment. When demand quantities are unknown, we show that the SConFL problem can be optimally solved by replacing all random variables by their expected values. The two-stage stochastic problem can be nicely reduced into a one stage problem without recourse. The reason is that demand quantities affect all assignment costs in the same way. Hence, once facilities have been opened, demand nodes are assigned to the closest (per unit of demand) facility, regardless of the actual demand quantity. Consequently, there is no recourse in the second stage and the problem can be solved in

one stage. The value of the stochastic solution is null (i.e., using average demand values as in Krick et al. (2003) actually solves the problem).

Another source of uncertainty in assignment costs is the customer locations. In this category we also include other sources of uncertainty in assignment costs that have a similar effect and do not affect all assignment costs in the same direction (i.e., some costs can increase and some can decrease). For example, when a network provider offers discounts or special rates on certain links in the network to balance out the overall flow, the customer location may be known beforehand but the closest or cheapest facility (per unit of demand) may change overtime tied to the traffic flow on the network. For the ConFL problem one can specify three different types of probability distributions on the scenarios: (i) the polynomial-scenario model, where one assumes that there is only a polynomial number of scenarios that occur with positive probability, and these are explicitly enumerated, (ii) the independent-activation model, where each assignment edge has an independent probability distribution, and (iii) the black-box model, where nothing is assumed about the probability distribution.

In this chapter our analysis focuses on the polynomial-scenario model and the independent-activation model. While in theory one can use the same methodology proposed here for the polynomial-scenario model for the independent-activation model; in practice our solution approach devised for the polynomial-scenario model becomes impracticable because the number of scenarios grows rapidly when assignment edges vary independently.

The method proposed for the polynomial-scenario model is based on construct-

ing a deterministic equivalent of the stochastic problem. We propose a set of transformations to obtain a deterministic ConFL problem that coupled with our DLS heuristic yields high-quality solutions. Furthermore, we use this strategy to obtain high-quality solutions for the independent-activation model within a Monte-Carlo simulation framework such as Sample Average Approximation (SAA) (see Kleywegt et al. (2002)). We report computational results on a comprehensive set of randomly generated instances for both the polynomial-scenario model and the independent-activation model using the SAA framework. The novelty of our implementation of the SAA framework is that we use a heuristic to solve the sample average problems. The SAA framework relies on the fact that sample average problems are solved to optimality. However, here we show how to implement SAA using a lower bounding procedure jointly with a heuristic and yet obtain tight confidence bounds on the optimal solution.

4.2 Literature Review

The introduction of uncertainty in linear programming dates back to 1955 when Dantzig (1955) introduced a computation procedure for two-stage linear programming models and a set of convexity theorems on the objective function of multiple stage models. Real world applications are flooded with uncertainty, which stochastic optimization allows us to model into the decision process as probability distributions to better represent the problem under consideration. As a result the stochastic models provide significant value to the decision process because they

better capture the nature of problem. In many situations it has been shown that just replacing uncertain input values for their expected values is not necessarily a good strategy. For a good introduction to the field of stochastic optimization see Ruszczynski and Shapiro (2003), Birge and Louveaux (1997) and Kall and Wallace (1994).

There have been several papers in the literature that deal with facility location or network design with uncertain demands or link lengths in various contexts. However, to our knowledge there is no prior work on the stochastic connected facility location problem. Mirchandani (1975) and Mirchandani and Odoni (1979) extend the concept of p-median location to networks whose edge costs are random variables. Their main motivation is the deployment of a service vehicle in a city when the travel times vary randomly and throughout the day due to traffic congestion. The objective of the problem is to minimize the expected travel time to any destination node in the network. Weaver and Church (1983) address the same problem and develop a computational procedure. There is no recourse in this problem. Berman (1978) and Berman and Odoni (1982) add the option of relocating the service vehicle once travel times are revealed. Berman (1978)'s heuristic is generalized to multiple facilities by Berman and Odoni (1982).

In another set of facility location problems, the uncertainty element relies instead on the customer demands. Snyder (2006) provides a comprehensive review on stochastic and robust facility location models. Laporte et al. (1994) analyze the capacitated facility location problem with uncertain demand. They state the problem as a two-stage program with recourse where the first stage decisions define

the location of the facilities and their capacity and the second-stage decisions determine the quantities delivered to each demand node. The paper by Louveaux and Peeters (1992), which deals with a more general version of the problem that models uncertain demands and edge costs, is very relevant to our study in this chapter in terms of methodology. They propose a dual-based heuristic for a two-stage stochastic program with recourse when there is uncertainty on demands, selling prices, and production and transportation costs. In the first stage, decisions regarding location and capacities of the plants are taken. And in the second stage, after demands, prices and costs are revealed, the allocation of demands is determined. The optimal capacity of the facilities arises from the trade-off between the cost of increasing the capacity and the net profit at the various random demand levels. They extend the dual-based procedure of Erlenkotter (1978) for the uncapacitated facility location problem and once again prove the effectiveness of dual-ascent schemes for facility location problems. Although the problem setting and algorithmic strategy is different; we also exploit the known virtues of a dual-ascent scheme to generate a dual-solution (and lower bound) and a primal solution that later improves with local search.

The SConFL problem is different from these two types for facility location problems in that we assume no capacity limits on the edges. Capacity limits call for a trade-off between the first stage and second stage decisions as some demand might be lost due to capacity decisions taken at an earlier stage. In the SConFL problem demands are always met; however, the facility node that serves the demand may change according to the realized edge costs. The trade-off lies between the cost of the core network and the realized assignment cost. Installing more facilities and

locating them closer to the demand nodes would increase the cost of the core network while potentially decreasing the assignment cost of the second stage decisions.

Another vein of research that relates to our problem are network design problem in telecommunications. Sen et al. (1994) study the problem of private-line services with random demands. They define a two-stage problem where the first-stage decision variables correspond to the installation of capacity on the edges of a network and the second stage decision variables deal with routing demand between origins and destinations. The objective in that problem is to minimize unmet demand. Riis and Andersen (2003) discuss the same problem and develop a procedure based on an L-Shaped algorithm (Van Slyke and Wets 1969).

A different path of research has been the development of approximation heuristics for these problems. Gupta et al. (2004) find a constant factor approximation heuristic for the stochastic Steiner tree problem and single sink network design problem. Ravi and Sinha (2006) consider two-stage finite scenario stochastic versions of various combinatorial optimization problems among them the facility location problem. In the first-stage facilities are opened while in the second-stage open facilities can be modified at a higher cost. They find an 8-approximation heuristic for this problem, which Swamy (2004) improve with a $4.127 + \epsilon$-approximation algorithm. In this chapter, we do not seek to devise approximation algorithms. Instead our objective is to develop a heuristic that performs efficiently in practice.

All the previously discussed problems have some aspect of the SConFL problem; yet none of the methods proposed for these problems easily extends to the SConFL problem. There has been considerable research on two-stage stochastic

problem with recourse (see, Birge and Louveaux (1997)); however, most of these methods assume linearity on the decisions of the first and second stage decision variables. Integer (and binary) decisions on both stages makes the stochastic problem even harder to solve. We present in this chapter a heuristic that exploits the characteristics of the SConFL problem and obtains high-quality solutions to a two-stage stochastic integer programming problem.

4.3 A Note on Two-Stage Linear Programs with Fixed Recourse

Stochastic linear programs are linear programs in which some problem data may be considered uncertain. Recourse programs are those in which some decisions or recourse actions can be taken after the uncertainty is revealed. That means that some of the problem parameters can be represented as random variables. We represent by ω the random event such as market conditions that determines some of our problem parameters such as demand quantities $d_i(\omega)$ or assignment costs $a_{ij}(\omega)$, and we denote by ξ the set of problem parameters that are tied to the random event ω. We represent by $\xi = \xi(\omega)$ the relationship between the uncertain event and the parameters of our problem. Although this relationship is not a functional relation between the random event and the problem inputs; we assume that each scenario $\omega \in \Omega$ fully determines the problem parameters in ξ. The realization of the random event divides the set of decisions into two groups:

- Decisions that have to be taken before the realization of the random event are called *first-stage decisions* and take place during the first stage.

- Decisions that take place after uncertainty is unveiled are *second-stage decisions* or *recourse decisions*. These decisions take place during the second-stage.

Within the stochastic optimization literature, it is customary to represent first-stage decisions by x and second-stage decisions by $y(x, \omega)$. Then, the classical two-stage stochastic linear program with fixed recourse (introduced by Dantzig (1955) and Beale (1955)) is the problem of finding

$$\text{Minimize } z = c^T x + \mathbb{E}_\omega[\min\ q(\omega)^T y(x, \omega)] \tag{4.1a}$$

subject to

$$Ax = b \tag{4.1b}$$

$$T(\omega)x + W(\omega)y(x, \omega) = h(\omega) \tag{4.1c}$$

$$x \geq 0, \quad y(x, \omega) \geq 0. \tag{4.1d}$$

Each component q, T, W, and h is a possible random variable determined by a realization of ω. Then ξ is the set of these random components, $\xi(\omega) = \{q(\omega)^T$, $h(\omega)^T$, $T(\omega), W(\omega)\}$. We assume that Ξ is the support of ξ. In addition, for the SConFL we assume that Ξ is finite.

The objective function (4.1a) contains a deterministic term, $c^T x$, and the expectation of the second-stage objective, $q(\omega)^T y(x, \omega)$. In many stochastic programming problems, this second-stage term is usually hard to compute because for each

ω the value of $y(x, \omega)$ is the solution of an optimization problem. In other words, for each ω and first-stage decisions x, we must solve the following optimization program

$$Q(x, \xi(\omega)) = \text{Minimize } q(\omega)^T y \tag{4.2a}$$

$$\text{subject to}$$

$$W(\omega)y = h(\omega) - T(\omega)x \tag{4.2b}$$

$$y \geq 0. \tag{4.2c}$$

$Q(x, \xi(\omega))$ is referred as the second-stage value function. Suppose we denote the expected second-stage value function as $\mathcal{Q}(x) = \mathbb{E}_\omega Q(x, \xi(\omega))$, then the *deterministic equivalent program* (DEP) of the stochastic programming problem is:

$$\text{Minimize } z = c^T x + \mathcal{Q}(x) \tag{4.3a}$$

$$\text{subject to}$$

$$Ax = b \tag{4.3b}$$

$$x \geq 0 \tag{4.3c}$$

This representation of a stochastic program shows that the main difference from a deterministic formulation is in the expected second-stage value function. If we can find a closed form representation for the second-stage value function,

which could be nonlinear, the stochastic program becomes an ordinary nonlinear program. For the SConFL problem we find this second-stage value function when Ξ has polynomial size, and furthermore, show that it is linear. This allow us to apply our DLS heuristic on the deterministic equivalent program to find high-quality solutions.

In the SConFL the first-stage decisions are the set of open facilities, $\mathbf{z}$, and the Steiner tree that connects them, $\mathbf{y}$, and the second-stage decisions involve the allocation of customers to open facilities, $\mathbf{x}$. We represent second-stage decisions by $\mathbf{x}(\mathbf{z}, \omega)$ to emphasize their dependence on the first-stage decisions, $\mathbf{z}$, and the realized scenario, ω.

4.4 Problem Formulation

In this section we explore the formulation of the Stochastic ConFL and show its transformation into a deterministic ConFL problem (i.e. its deterministic equivalent program) such that our dual-ascent local search heuristic, introduced in Chapter 3, can be used to obtain high-quality solutions as well as assisting lower bounds.

We first define a cutset formulation for the deterministic ConFL, i.e. a ConFL problem with known demand quantities and assignment costs. The objective function (4.4a) has three terms: the opening facility cost, the core tree cost and the assignment cost. Constraints (4.4b) and (4.4c) impose the condition that the open facilities are connected by a Steiner tree, while constraints (4.4d) and (4.4e) ensure that each demand node is assigned to an open facility.

Cutset formulation for the deterministic ConFL problem:

$$\text{Minimize} \sum_{i \in F} f_i z_i + \sum_{(i,j) \in (S \cup F)} c_{ij} y_{ij} + \sum_{i \in D, j \in F} a_{ij} x_{ij} \tag{4.4a}$$

subject to

$$\sum_{(i,j) \in (S \cup F)} y_{ij} = \sum_{l \in (S \cup F)} z_l - 1 \tag{4.4b}$$

$$\sum_{(i,j) \in R} y_{ij} \leq \sum_{l \in R} z_l - 1, \ \forall R \subset (S \cup F) \tag{4.4c}$$

$$\sum_{j \in F} x_{ij} \geq 1, \ \forall i \in D \tag{4.4d}$$

$$x_{ij} \leq z_j, \ \forall i \in D, \forall j \in F \tag{4.4e}$$

$$x_{ij}, y_{ij}, z_i \in \{0, 1\} \tag{4.4f}$$

In the stochastic version of the ConFL problem, assignment costs, a_{ij}, are uncertain and depend on the realization of a random variable ω or scenario. In general terms, ξ pieces together the stochastic components of the problem. We assume that the random variable ξ has discrete and finite support in Ξ. In other words, there is a finite set of known possible scenarios and $P(\Xi) = 1$. Then, $a_{ij}(\omega)$ represents the assignment cost under scenario ω and p_ω the probability of occurrence of scenario ω. The following formulation shows the analogous cutset formulation of this problem as a two-stage stochastic program with fixed recourse.

Cutset formulation for the Stochastic ConFL problem:

$$\text{Minimize} \sum_{i \in F} f_i z_i + \sum_{i,j \in (S \cup F)} c_{ij} y_{ij} + \mathbb{E}_{\omega}(Q(\mathbf{z}, \xi(\omega))) \tag{4.5a}$$

subject to

$$\sum_{(i,j) \in (S \cup F)} y_{ij} = \sum_{l \in (S \cup F)} z_l - 1 \tag{4.5b}$$

$$\sum_{(i,j) \in R} y_{ij} \leq \sum_{l \in R} z_l - 1, \ \forall R \subset (S \cup F) \tag{4.5c}$$

$$y_{ij}, z_i \in \{0, 1\} \tag{4.5d}$$

In the stochastic version of the ConFL problem, the assignment cost is unknown in the first stage and hence we replace the third term in the objective function (4.4a) by its expected value, (i.e. the expected value of the second stage decision problem), yielding the objective function (4.5a). In other words, the assignment decision for each demand node is delayed until the second stage when the recourse minimization problem (4.6) is solved.

$$Q(\mathbf{z}, \xi(\omega)) = \text{Minimize } q(\mathbf{z}, \xi(\omega))(\mathbf{x}) = \sum_{i \in D, j \in F} a_{ij}(\omega) x_{ij} \tag{4.6a}$$

subject to

$$\sum_{i \in F} x_{ij} \geq 1, \ \forall j \in D \tag{4.6b}$$

$$x_{ij} \leq z_j, \ \forall i \in D, \forall j \in F \tag{4.6c}$$

$$x_{ij} \in \{0, 1\} \tag{4.6d}$$

Clearly, once open facilities are defined in the first stage, the recourse problem reduces to an assignment problem where demand nodes are assigned to the closest open facility. To determine the closest open facility for each demand node we must wait until assignment costs are realized; consequently, the solution to the assignment problem may vary for each scenario.

4.4.1 SConFL with Uncertain Demands

In the case where uncertainty on assignment costs is due to unknown demand quantities, we assume that the per unit assignment cost (denoted by b_{ij}) is fixed and known before hand. Here when scenario ω is unveiled, we mean that the demand quantity $d_i(\omega)$ is discovered for each demand node i, and hence the assignment cost $a_{ij}(\omega) = d_i(\omega) b_{ij}$ is revealed. In this setting, the assignment cost matrix, A, is the only random input parameter in the problem; then, $\xi(\omega) = (A(\omega))$. For this specific

realization of events, we show that the value of the stochastic solution is null. In other words, the optimal solution for this stochastic ConFL is the optimal solution of a deterministic ConFL problem when average demands are assumed.

Theorem 4.4.1. *The optimal solution of the stochastic ConFL with uncertain demands is equal to the optimal solution of the deterministic ConFL obtained by replacing all random variables by their expected values.*

Before we can prove Theorem 4.4.1, we need to show the following two lemmas.

Lemma 4.4.2. *Given a first-stage decision,* $\mathbf{z}$*, the optimal allocation solution,* $\mathbf{x}^*$*, to the recourse problem,* $Q(\mathbf{z}, \xi(\omega))$*, for the SConFL with uncertain demands is invariant to demand realizations.*

Proof of Lemma 4.4.2

We must show that if $\mathbf{x}^*$ is an optimal solution for $Q(\mathbf{z}, \xi(\tilde{\omega}))$ for some $\tilde{\omega} \in \Omega$, then $\mathbf{x}^*$ is an optimal solution for $Q(\mathbf{z}, \xi(\omega))$ for all $\omega \in \Omega$.

Let $X^{\mathbf{z}}$ be the feasible region defined by $\mathbf{z}$, and $\mathbf{x}^* \in X^{\mathbf{z}}$ be an optimal solution for $Q(\mathbf{z}, \xi(\tilde{\omega}))$ for some $\tilde{\omega} \in \Omega$. Note that the feasible region $X^{\mathbf{z}}$ is not dependent on ω. Then,

$$\begin{aligned} q(\mathbf{z}, \xi(\tilde{\omega}))(\mathbf{x}^*) &= \textstyle\sum_{i \in D}(\sum_{j \in F} a_{ij}(\tilde{\omega}) x^*_{ij}) \\ &= \textstyle\sum_{i \in D}(\sum_{j \in F} d_i(\tilde{\omega}) b_{ij} x^*_{ij}) \\ &= \textstyle\sum_{i \in D} d_i(\tilde{\omega})(\sum_{j \in F} b_{ij} x^*_{ij}) \quad \leq \quad \sum_{i \in D} d_i(\tilde{\omega})(\sum_{j \in F} b_{ij} x_{ij}), \forall \mathbf{x} \in X^z. \end{aligned}$$

In vector notation,

$$\sum_{i \in D} d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*) \leq \sum_{i \in D} d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i), \forall \mathbf{x} \in X^z. \tag{4.7}$$

From inequality (4.7) we can show[1] that not only the inequality holds for the summation, but also for each individual term in it. That is,

$$d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*) \leq d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i), \forall i \in D, \forall \mathbf{x} \in X^z; \tag{4.8}$$

which implies,

$$\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^* \leq \mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i, \forall i \in D, \forall \mathbf{x} \in X^z. \tag{4.9}$$

We know that $\mathbf{x}^* \in X^{\mathbf{z}}$ and hence $\mathbf{x}^*$ is a feasible solution for $Q(\mathbf{z}, \xi(\omega)), \forall \omega \in \Omega$. Now, assume that $\mathbf{x}^*$ is not an optimal solution to $Q(\mathbf{z}, \xi(\omega))$ for some $\omega \in \Omega$. Then, there exists an $\mathbf{x}' \neq \mathbf{x}^* \in X^{\mathbf{z}}$ such that $q(\mathbf{z}, \xi(\omega))(\mathbf{x}') < q(\mathbf{z}, \xi(\omega))(\mathbf{x}^*)$.

$$\begin{array}{rcll} q(\mathbf{z}, \xi(\omega))(\mathbf{x}') & = & \sum_{i \in D} d_i(\omega)(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i') & < \quad \sum_{i \in D} d_i(\omega)(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*) \\ & \Rightarrow & d_i(\omega)(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i') & < \quad d_i(\omega)(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*), \exists i \in D \\ & \Rightarrow & \mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i' & < \quad \mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*, \exists i \in D \Rightarrow\Leftarrow \end{array}$$

This contradicts equation (4.9) and proves by contradiction our Lemma 4.4.2 $\square$.

[1] If there exists a demand node $i \in D$, such that $d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i^*) > d_i(\tilde{\omega})(\mathbf{b}_i^{\mathrm{T}} \mathbf{x}_i')$ for some $\mathbf{x}_i'$. Then we could replace i's assignment in $\mathbf{x}^*$ and obtain a lower objective function. This would contradict our assumption that $\mathbf{x}^*$ is an optimal solution.

Lemma 4.4.3. *Given a first-stage decision,* $\mathbf{z}$*, the expected value of the recourse program,* $Q(\mathbf{z}, \xi(\omega))$*, for the SConFL with uncertain demands equals the objective function value of the recourse program with expected demands.*

Proof of Lemma 4.4.3

We must show that $\mathbb{E}_\omega(Q(\mathbf{z}, \xi(\omega))) = \text{Minimize}_{\mathbf{x}\in X^{\mathbf{z}}} \sum_{i\in D} \sum_{j\in F} \mathbb{E}_\omega(d_i(\omega))b_{ij}x_{ij}$.

$$\mathbb{E}_\omega(Q(\mathbf{z}, \xi(\omega))) = \sum_{\omega\in\Omega} p_\omega Q(\mathbf{z}, \xi(\omega)) \tag{4.10}$$

$$= \sum_{\omega\in\Omega} p_\omega \ \text{Min}_{\mathbf{x}\in X^{\mathbf{z}}} \sum_{i\in D, j\in F} d_i(\omega)b_{ij}x_{ij} \tag{4.11}$$

By Lemma 4.4.2, we can take the minimization outside the first summation. Moreover, we can rearrange the order of summations. Then,

$$\mathbb{E}_\omega(Q(\mathbf{z}, \xi(\omega))) = \text{Min}_{\mathbf{x}\in X^{\mathbf{z}}} \sum_{\omega\in\Omega} p_\omega \sum_{i\in D, j\in F} d_i(\omega)b_{ij}x_{ij} \tag{4.12}$$

$$= \text{Min}_{\mathbf{x}\in X^{\mathbf{z}}} \sum_{i\in D, j\in F} (\sum_{\omega\in\Omega} p_\omega d_i(\omega))b_{ij}x_{ij} \tag{4.13}$$

$$= \text{Min}_{\mathbf{x}\in X^{\mathbf{z}}} \sum_{i\in D, j\in F} \mathbb{E}_\omega(d_i)b_{ij}x_{ij} \tag{4.14}$$

This proves Lemma 4.4.3□.

Proof of Theorem 4.4.1

Finally, the mean value problem of the recourse problem solves the recourse problem, and consequently Theorem 4.4.1 directly follows from Lemma 4.4.3 □.

4.4.2 SConFL with Uncertain Locations

When variability on assignment costs is due to uncertainty on customers' location or other factors that do not affect assignment costs proportionally, simplifying the problem by replacing the random variables by their expected value does not necessary lead to good solutions. This is because the location of the closest open facility depends on the realized scenario. In this case we show that even though we cannot replace random variables by their expected values, we can transform the SConFL problem into a deterministic ConFL problem with multiple copies of demand nodes that our dual-ascent local search heuristic can successfully solve. We assume the polynomial-scenario model such that there exists a limited number of scenarios, $\omega \in \Omega$, that determine the whole set of assignment costs, $a_{ij}(\omega)$ $\forall i \in D$ and $j \in F$.

Theorem 4.4.4. *The SConFL problem is equivalent to a deterministic ConFL problem with $|\Omega|$ copies of each demand node—one copy for each scenario and with assignment cost equal to $p_\omega a_{ij}(\omega)$.*

Proof of Theorem 4.4.4

Recall the second stage recourse problem,

$$Q(\mathbf{z}, \omega) = \text{Min}_{\,\mathbf{x} \in X^{\mathbf{z}}} \sum_{i \in D, j \in F} a_{ij}(\omega) x_{ij}(\omega). \tag{4.15a}$$

Hence, the expected value of the recourse problem is the weighted sum of each scenario given the decisions of the first stage problem, $\mathbf{z}$. Here, we have to specify

that the allocation solution x_{ij} is a function of ω.

$$\mathbb{E}_{\omega}(Q(z,\xi(\omega))) = \sum_{\omega\in\Omega} p_{\omega}\ \mathrm{Min}_{x\in X} \sum_{i\in D, j\in F} a_{ij}(\omega)x_{ij}(\omega) \tag{4.16a}$$

$$= \mathrm{Min}_{x\in X} \sum_{\omega\in\Omega} p_{\omega} \sum_{i\in D, j\in F} a_{ij}(\omega)x_{ij}(\omega) \tag{4.16b}$$

$$= \mathrm{Min}_{x\in X} \sum_{\omega\in\Omega} \sum_{i\in D, j\in F} p_{\omega}a_{ij}(\omega)x_{ij}(\omega) \tag{4.16c}$$

In other words, given equation (4.16c) we can explicitly introduce the expected value into our two-stage linear problem with recourse formulation. The resulting formulation (4.17), hence, is equivalent to our original cutset formulation for the ConFL problem when assignment costs are known, see formulation (4.4). There are as many copies of the demand nodes as scenarios there exist, and the assignment costs are given by the product of the probability of the scenario and the original assignment cost.

This finalizes our proof of Theorem 4.4.4 $\square$.

Cutset formulation for the Stochastic ConFL problem:

$$\text{Minimize } Z = \sum_{i \in F} f_i z_i + \sum_{i,j \in (S \cup F)} c_{ij} y_{ij} + \sum_{\omega \in \Omega} \sum_{i \in D, j \in F} p_\omega a_{ij}(\omega) x_{ij}(\omega) \tag{4.17a}$$

subject to

$$\sum_{i \in F} x_{ij}(\omega) \geq 1, \ \forall j \in D, \forall \omega \in \Omega \tag{4.17b}$$

$$x_{ij}(\omega) \leq z_j, \ \forall i \in D, \forall j \in F, \forall \omega \in \Omega \tag{4.17c}$$

$$\sum_{(i,j) \in (S \cup F)} y_{ij} = \sum_{l \in (S \cup F)} z_l - 1 \tag{4.17d}$$

$$\sum_{(i,j) \in R} y_{ij} \leq \sum_{l \in R} z_l - 1, \ \forall R \subset (S \cup F) \tag{4.17e}$$

$$x_{ij}(\omega), y_{ij}, z_i \in \{0, 1\} \tag{4.17f}$$

While our transformation is also applicable to the independent-activation model, it suffers from what is often referred to as the curse of dimensionality. For example, a problem with only two facilities, $|F| = 2$, and three demand nodes, $|D| = 3$, with two assignment cost levels each, $|L| = 2$, where any demand node can be assigned to any facility node would have a total of 64 scenarios to consider, $|L|^{|D|^{|F|}}$. When the location of demand nodes are independent from each other, the number of scenarios increases rapidly and to simply solve the deterministic equivalent problem with multiple demand node copies is impractical and computationally infeasible. However, taking advantage of the deterministic equivalent problem, we can obtain high-quality solutions using Monte Carlo simulation for the independent-

activation model. Furthermore, the lower bounding procedure allows as to assess the quality of the heuristic solution and construct tight confidence bounds for the optimal solution value.

4.5 Sample Average Approximation Method

The sample average approximation (SAA) method is an approach for solving stochastic optimization problems by using Monte Carlo simulation. Kleywegt et al. (2002), Verweij et al. (2003), and Shapiro and Philpott (2007) provide good introductions to this approach. In this technique the expected objective function of the stochastic problem is approximated by a sample average estimate derived from a random sample. The resulting sample average approximation problem, a deterministic variant of the problem, is then solved by optimization techniques. The process is repeated with different samples to obtain candidate solutions along with statistical estimates of their optimality gaps.

Unlike the approach followed in the literature in the SAA method, here we solve the sample problems with a heuristic coupled with a lower bounding method. We show that it is possible to construct tight confidence intervals on the optimal value function even if the sample problems are not solved to optimality. This result is particularly important for problems such as the ConFL problem that are costly to solve to optimality (see Ljubić (2007)). Certainly, the quality of the solution yielded will depend on the quality of the solution obtained by the heuristic for the sample problems; and the width of the confidence interval will also depend on the sample

problem size, the quality of the lower bounds, and the variability of the solution values.

In the SAA method the expected value function $\mathbb{E}[Q(x, \xi(\omega))]$ is approximated by the sample average function $\sum_{n=1}^{N} Q(x, \xi(\omega^n))/N$, where a sample $\{\omega^1, \omega^2, ..., \omega^N\}$ of N sample scenarios is generated from Ω according to probability distribution P. The SAA problem

$$z^N = \min_{x \in X} c^T x + \frac{1}{N} \sum_{n=1}^{N} Q(x, \xi(\omega^n)), \tag{4.18}$$

corresponding to the original two-stage stochastic problem is then solved using a deterministic optimization algorithm. The optimal value z^N and an optimal solution $\hat{x}$ to the SAA problem provide estimates of their true counterparts in the stochastic program. By generating R independent samples, each of size N, and solving the associated SAA problems, objective values $z^{N1}, z^{N2}, ..., z^{NR}$ and candidate solutions $\hat{x}^1, \hat{x}^2, ..., \hat{x}^R$ are obtained. Let

$$\bar{z}^N = \frac{1}{R} \sum_{m=1}^{R} z^{Nm} \tag{4.19}$$

denote the average of the R optimal values of the SAA problems.

This procedure produces up to R different candidate solutions. Out of these R different candidate solutions, we have to select one as the approximation to the optimal solution of the original stochastic program. One generally accepted strategy is to generate a sample problem with a significantly large number of scenarios,

$N' >> N$. Then, it is natural to take $\hat{x}^*$ as one of the optimal solutions $\hat{x}^1, \hat{x}^2, ..., \hat{x}^R$ of the R SAA problems that has the smallest estimated objective value, that is,

$$\hat{x}^* \in \arg\min\{\hat{z}^{N'}(\hat{x}) | \hat{x} \in \{\hat{x}^1, \hat{x}^2, ..., \hat{x}^R\}\} \tag{4.20}$$

where $\{\omega^1, \omega^2, ..., \omega^{N'}\}$ is the sample of scenarios chosen to evaluate the candidate solutions.

Using the DLS heuristic, we generate R heuristic candidate solutions x_H^1, x_H^2,..., x_H^R with their corresponding objective values, $z_H^{N1}, z_H^{N2}, ..., z_H^{NR}$ and lower bounds, z_{LB}^{N1}, z_{LB}^{N2},..., z_{LB}^{NR}. Similarly, we take as the heuristic solution to the stochastic program the heuristic solution x_H^i that has the smallest estimated objective value in the sample problem with N' scenarios.

4.5.1 Quality of the Solution

Kleywegt et al. (2002) provides performance bounds on the quality of the solution yielded by the SAA method to the stochastic program. Following and extending their argument, in this section we provide performance bounds on the quality of the solution yielded by the SAA method using the DLS heuristic.

Given a feasible solution $x \in X$, we have to evaluate the quality of this point viewed as a candidate for solving the true problem. Since the point x is feasible, we clearly have that $g(x) \geq v^*$, where $v^* = \min_{x \in X} g(x)$ is the optimal value of the stochastic problem, and g is the true stochastic objective function. The quality of

x can be measured by the optimality gap

$$gap(x) := g(x) - v^*. \tag{4.21}$$

The true value of $g(x)$ can be estimated by Monte Carlo sampling. That is, an iid random sample ω^j, $j = 1, ..., N'$, of ω is generated and $g(x)$ is estimated by the corresponding sample average $\bar{g}_{N'}(x) = c^T x + \bar{q}_{N'}(x)$. At the same time the sample variance

$$\sigma^2_{N'}(x) := \frac{1}{N'(N'-1)} \sum_{j=1}^{N'} [Q(x, \omega^j) - \bar{q}_{N'}(x)]^2 \tag{4.22}$$

of $\bar{q}_{N'}(x)$ is calculated. Then we can calculate an approximate $100(1-\alpha)\%$ confidence upper bound for $g(x)$ by

$$U_{N'}(x) := \bar{g}_{N'}(x) + z_\alpha \sigma_{N'}(x). \tag{4.23}$$

This bound is justified by the Central Limit Theorem with the critical value $z_\alpha = \Phi^{-1}(1-\alpha)$, where $\Phi(z)$ is the cumulative distribution function (cdf) of the standard normal distribution.

In order to calculate a lower bound for v^* we proceed as follows. Denote by z^N_{LB} the lower bound yielded by the DLS heuristic for the SAA problem based on a sample of size N. Note that z^N_{LB} is a function of the (random) sample and hence is random. To obtain a lower bound for v^* observe that $\mathbb{E}[\bar{g}_N(x)] = g(x)$, i.e., the sample average $\bar{g}_N$ is an unbiased estimator of the expectation $g(x)$. We also have that for any $x \in X$ the inequality $\bar{g}_N(x) \geq \inf_{x' \in X} \bar{g}_N(x') \geq z^N_{LB}$ holds, so for any

$x \in X$, we have

$$g(x) = \mathbb{E}[\bar{g}_N(x)] \geq \mathbb{E}[\inf_{x' \in X} \bar{g}_N(x')] \geq \mathbb{E}[z^N_{LB}]. \tag{4.24}$$

By taking the minimum over $x \in X$ of the left hand side of the above inequality we obtain $v^* \geq \mathbb{E}[z^N_{LB}]$.

We can estimate $\mathbb{E}[z^N_{LB}]$ by solving the SAA problems several times and averaging the lower bounds calculated by dual ascent. That is, the SAA problems based on independently generated samples, each of size N, are solved to obtain a dual-ascent bound R times. Let $z^{N1}_{LB}, z^{N2}_{LB}, ..., z^{NR}_{LB}$ be the computed lower bound values for these SAA problems. Then,

$$\bar{z}^{NR}_{LB} := \frac{1}{R} \sum_{j=1}^{R} z^{Nj}_{LB} \tag{4.25}$$

is an unbiased estimator of $\mathbb{E}[z^N_{LB}]$, the dual-ascent lower bound yield for SAA problems with N scenarios. Since the samples, and hence $z^{N1}_{LB}, z^{N2}_{LB}, ..., z^{NR}_{LB}$, are independent, we can estimate the variance of $\bar{z}^{NR}_{LB}$ by

$$\sigma^2_{NR} := \frac{1}{R(R-1)} \sum_{j=1}^{R} (z^{Nj}_{LB} - \bar{z}^{NR}_{LB})^2. \tag{4.26}$$

A confidence $100(1-\alpha)\%$ lower bound for $\mathbb{E}[z^N_{LB}]$ is then given by

$$L_{NR} := \bar{z}^{NR}_{LB} + t_{\alpha,\nu} \sigma_{NR}, \tag{4.27}$$

where $\nu = R-1$ and $t_{\alpha,\nu}$ is the α-critical value of the t-distribution with ν degrees of

freedom. Since $v^* \geq \mathbb{E}[z_{LB}^N]$, we have that L_{NR} gives a valid lower statistical bound for v^* as well. Consequently,

$$\hat{gap}(x) := U_{N'}(x) - L_{NR} \tag{4.28}$$

gives a statistically valid (with confidence at least 1 - 2α) bound on the true $gap(x)$. Alternatively, we can express this gap as a percentage by

$$\hat{gap}(x)[\%] := \frac{U_{N'}(x) - L_{NR}}{L_{NR}} \times 100[\%], \tag{4.29}$$

with the following interpretation: the heuristic solution is within $x\%$ from the true optimal solution with confidence at least 1 - 2α. It can be noted that the lower bound L_{NR} is somewhat conservative and depends on the quality of the lower bounding mechanism.

4.6 Proposed Heuristic

Our proposed heuristic relies on the fact that the SConFL problem with a polynomial number of scenarios can be formulated as a deterministic ConFL problem with multiple copies of the demand nodes and assignment costs equal to the original assignment cost multiplied by its probability of occurrence. Once the problem has been transformed into a deterministic ConFL problem, we apply our DLS heuristic to obtain a high-quality solution and lower bound as described in Chapter 3.

4.7 Computational Experiments

In this section we solve a set of SConFL problems and explore the benefits of solving the stochastic problem as an integer linear two-stage recourse problem. We report on a set of computational experiments with our DLS heuristic on the SConFL problem.

We report on results for both: (i) the polynomial-scenario model and (ii) the independent-activation model. We use the insights from our polynomial-scenario computational experiments to generate the sample average problems for the independent-activation computational experiments. We note that the sample average problems are simply SConFL problems with a preset number of scenarios. For the polynomial-scenario problem, we also solve the associated mean value problem and report the duality gap of the expected value solution. We coded our heuristics in Visual Studio 2005 (C++). We conducted all runs on an AMD Athlon^{TM} 62 X2 Dual, 2.61 GHz machine with 3GB of RAM.

4.7.1 Expected Value Solutions

In order to calculate the Expected Value Solution (EVS) we need to first find the solution that solves the associated mean value problem (MVP). That is, the deterministic ConFL problem whose random assignment costs, $a_{ij}(\omega)$, have been replaced by their expected value, $\bar{a}_{ij}$. We denote this solution as $\mathbf{x}(\bar{\omega})$ and we use

it to calculate the EVS as,

$$EVS = \sum_{\omega \in \Omega} p_\omega Z(\mathbf{x}(\bar{\omega}), \omega) \tag{4.30}$$

where $Z(\mathbf{x}(\bar{\omega}), \omega)$ is the objective function of the SConFL model for the value of the decision variables $\mathbf{x}(\bar{\omega})$ and a realization of the random variable ω. Therefore, the EVS is the expected value yielded by implementing the solution obtained by assuming expected values on the random inputs. In many settings this solution is not even guaranteed to be feasible; however, for the ConFL problem the solution obtained by solving the average problem is always feasible.

The Value of the Stochastic Solution (VSS) is then given by,

$$VSS = EVS - Z \tag{4.31}$$

where Z is the objective of the stochastic solution.

In our computational experiments we use a heuristic to solve the MVP and the stochastic problem; consequently, we cannot precisely compute the value of the stochastic solution. However, we can compare the quality of both solutions and compute the lower bound gap of the MVP solution using the lower bound yielded by DA for the SConFL problem.

4.7.2 Problem Generation and Characteristics

In this section we describe how we generated instances for both the polynomial-scenario model and sample average approximation problems. We generated instances by first selecting nodes randomly located on a 100 x 100 square grid. The location, i.e. x- and y-coordinates, of each facility, Steiner node and demand node is randomly generated on the grid. Furthermore, to represent the uncertainty in the assignment costs we assume that the exact location of each demand node is uncertain and generate as many copies as scenarios of each demand node varying its location. As a first step we generated a base location for each demand node; secondly, we disturbed that location by an error term, e, in the x- and y-coordinates drawn from a discrete uniform distribution according to a given variability, v. In our first set of instances, v ranges from 5 to 30 in steps of 5; i.e. if $v = 5$ then $e \sim U[-5, 5]$.

The Euclidean distances rounded up to the next integer (to preserve triangle inequality) were used as a basis for the edge lengths. The assignment edge costs are equal to the edge lengths between demand nodes and facility nodes, while tree edge costs are equal to the edge lengths multiplied by an M factor. The M factor illustrates the significantly higher (in terms of cost per unit distance) connection cost of edges in the tree T. We set $M = 7$ for the polynomial-scenario instances and $M = 3$ for the sample average problems.

The number of demand nodes and facility nodes vary between 10 and 90 in steps of 10, with the total number of demand and facility nodes equal to 100. The

number of Steiner nodes is 20 for all the instances. In an instance of the problem, the facility opening costs are equal to 30 and the same for all the facility nodes. Finally, the number of scenarios considered for the polynomial-scenario instances is 5.

For the polynomial-scenario instances to compare the performance of our DLS heuristic on the SConFL with respect to its performance on the average representation of the problem, we generate an "average instance" where the assignment costs for each demand node equal its average assignment cost. In these instances assignment costs are no longer integer.

For the sample approximation method, we set the number of scenarios N for the sample average problems to 20, and the number of replications R to 10. Lastly, we set the number of scenarios N' to 2000 to compute the sample variance of the solution yielded by the heuristic.

4.7.3 Polynomial-Scenario Model Results

Tables 4.1 and 4.2 present our computational results of the dual-ascent heuristic and the DLS heuristic on the MVP and the stochastic formulation. The MVP results were calculated using the original assignment costs; that is, after the DLS heuristic yielded a solution using the average assignment costs, we use that solution (set of open facilities) to calculate the real assignment costs with recourse.

DA yields relatively good solutions with average gaps below 11.01% for every range of variability on the location of the demand nodes and proportion of demand

and facility nodes. However, in some instances the gap can be quite large, and in one instance this reaches 23.74%. The local search is quite effective to reduce the gaps on either formulations and yields in all cases solutions with gaps below 9.25% on the MVP and 7.70% on the stochastic formulation. The DLS heuristic yields solutions with average gaps below 5.22% and 4.07% on the MVP and stochastic formulation, respectively, for all the combinations of parameters.

There are no considerable differences between the solutions yielded by DLS on the MVP and on the stochastic formulation for low values of variability. This result was expected. When there is low variability on the assignment costs, to obtain a solution using the average assignment costs seems quite reasonable given that the DLS takes a fraction of the time on this formulation with respect to the time it requires on the complete formulation of the problem. However, as the variability on the assignment costs increases, DLS finds better solutions when run on the stochastic formulation of the problem. Nevertheless, we must note that neither formulation yields the best solution for all combinations of parameters or variability. Even when the variability on the assignment costs is high, in a few instances the DLS heuristic found a better solution using the MVP formulation.

4.7.4 SAA Results (Independent-Activation Model)

Tables 4.3, 4.4 and 4.5 show our computational results for the SConFL problem using the SAA method and the DLS heuristic. Each entry in Table 4.3 shows the average gap over ten instances, while Tables 4.4 and 4.5 show the minimum

$\|D\|$	$\|F\|$	$v = 5$			$v = 10$			$v = 15$		
		DA	MVP	DLS	DA	MVP	DLS	DA	MVP	DLS
10	90	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%
20	80	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%
30	70	3.05%	1.37%	1.39%	4.88%	1.21%	1.11%	4.27%	1.64%	1.25%
40	60	7.63%	2.38%	2.64%	8.59%	3.02%	3.03%	8.47%	3.38%	2.69%
50	50	7.20%	3.50%	3.70%	8.40%	3.72%	3.52%	10.28%	4.49%	3.68%
60	40	8.55%	3.27%	3.53%	9.58%	3.34%	3.84%	9.84%	4.30%	4.07%
70	30	6.22%	2.71%	2.77%	6.79%	3.06%	2.89%	7.12%	3.38%	2.66%
80	20	2.87%	1.30%	1.53%	3.16%	1.68%	1.47%	3.59%	1.87%	1.46%
90	10	1.66%	0.38%	0.38%	1.38%	0.47%	0.48%	1.53%	0.61%	0.57%

Table 4.1: Comparison of heuristics for the Stochastic ConFL. $f_i = 30$, $M = 7$, and v factor is varied

$\|D\|$	$\|F\|$	$v = 20$			$v = 25$			$v = 30$		
		DA	MVP	DLS	DA	MVP	DLS	DA	MVP	DLS
10	90	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%	0.04%
20	80	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%	0.02%
30	70	6.15%	1.28%	1.47%	4.56%	1.91%	1.73%	5.98%	1.67%	1.63%
40	60	7.01%	3.24%	2.93%	8.91%	3.16%	2.55%	8.40%	4.08%	2.79%
50	50	7.67%	3.59%	3.32%	9.67%	4.11%	3.74%	11.01%	4.85%	3.67%
60	40	10.61%	3.33%	3.98%	9.11%	5.22%	3.82%	8.43%	4.29%	3.21%
70	30	6.77%	3.57%	3.02%	8.74%	3.83%	2.81%	9.35%	4.54%	3.42%
80	20	4.10%	2.53%	1.83%	5.40%	2.31%	1.62%	5.32%	2.98%	2.33%
90	10	1.15%	0.66%	0.41%	1.82%	1.23%	0.32%	1.72%	1.50%	0.51%

Table 4.2: Comparison of heuristics for the Stochastic ConFL. $f_i = 30$, $M = 7$, and v factor is varied

and maximum gap within those ten instances, respectively. The values reported are 98% confidence gaps. That is, with 98% confidence the optimal value of the true stochastic problem is x% from the lower bound. Overall, these gaps follow the behaviour observed for the deterministic instances and the polynomial-scenario instances. Lower gaps are observed for either high proportions of demand nodes or facility nodes. On the contrary, higher gaps are observed for balanced instances with similar numbers of demand nodes and facility nodes. Furthermore, these confidence

$\|D\|$	$\|F\|$	v					
		5	10	15	20	25	30
10	90	0.59%	1.03%	1.96%	2.47%	2.64%	2.67%
20	80	2.58%	2.93%	3.55%	3.60%	4.25%	4.75%
30	70	2.69%	2.95%	3.51%	4.59%	5.01%	5.17%
40	60	2.59%	3.08%	4.14%	4.48%	5.00%	5.42%
50	50	3.08%	3.39%	3.76%	4.51%	4.75%	5.25%
60	40	2.91%	3.31%	3.42%	3.90%	4.17%	4.78%
70	30	1.76%	2.12%	2.53%	2.83%	3.23%	3.35%
80	20	0.82%	1.20%	1.75%	1.91%	2.26%	2.52%
90	10	0.36%	0.80%	0.95%	1.09%	1.36%	1.19%

Table 4.3: Average 98% confidence gaps for the SConFL, $f_i = 30$, $M = 3$, and v factor is varied

$\|D\|$	$\|F\|$	v					
		5	10	15	20	25	30
10	90	0.33%	0.43%	1.08%	1.48%	1.40%	0.74%
20	80	1.27%	1.54%	2.17%	2.06%	1.71%	3.04%
30	70	1.01%	1.69%	1.57%	2.66%	3.29%	3.18%
40	60	0.83%	1.22%	3.13%	3.09%	3.55%	4.17%
50	50	1.75%	2.23%	2.64%	3.81%	3.39%	4.25%
60	40	1.40%	1.76%	2.71%	3.08%	3.14%	2.82%
70	30	0.80%	1.19%	1.63%	1.77%	2.01%	1.89%
80	20	0.30%	0.59%	1.28%	1.39%	1.65%	1.53%
90	10	0.20%	0.32%	0.48%	0.69%	1.05%	0.44%

Table 4.4: Minimum 98% confidence gaps for the SConFL, $f_i = 30$, $M = 3$, and v factor is varied

gaps increase for higher levels of uncertainty. We can outline two explanation for this behaviour. First, as the uncertainty level increases, the duality gaps obtained by DLS increase. We observe this behaviour for the polynomial-scenario instances. Secondly, as the uncertainty level increases, the sample variance increases as well, and the width of the confidence interval increases.

Tables 4.4 and 4.5 show that there are no large disparities in the gaps for a particular set of parameters. The smallest gap corresponds to an instance with 90

$\|D\|$	$\|F\|$	v					
		5	10	15	20	25	30
10	90	1.14%	1.60%	3.34%	3.85%	3.99%	4.53%
20	80	3.77%	6.20%	6.11%	5.48%	6.81%	7.82%
30	70	4.49%	4.21%	5.15%	7.85%	8.28%	8.84%
40	60	4.12%	4.19%	5.38%	5.15%	6.53%	6.45%
50	50	4.91%	6.03%	5.63%	5.76%	5.80%	6.44%
60	40	4.29%	4.08%	4.40%	4.65%	5.19%	6.02%
70	30	3.66%	3.61%	4.42%	3.94%	4.42%	4.25%
80	20	1.55%	1.87%	2.40%	2.48%	3.65%	3.59%
90	10	0.80%	1.45%	1.57%	1.58%	1.73%	2.00%

Table 4.5: Maximum 98% confidence gaps for the SConFL, $f_i = 30$, $M = 3$, and v factor is varied

customer nodes, 10 facility nodes and the lowest variability ($v = 5$). On the other hand, the highest gap is observed for an instance with 30 demand nodes, 70 facility nodes and the highest variability ($v = 30$).

In terms of computational time, to solve one sample average problem with 20 scenarios takes approximately 90 seconds. Consequently, to obtain a confidence interval on the optimal function value of the true stochastic problem with 10 replications takes approximately 900 seconds plus approximately 100 seconds for post processing. One could attempt to find better solutions for the true stochastic problem and tighter confidence intervals increasing the number of scenarios in each sample average problem, the number of replications for each sample average problem or the number of scenarios to evaluate each solution. However, any of these alternatives would indisputably require a longer computational time. There is a natural trade-off between the performance of the SAA and the computational effort to produce tighter performance bounds.

4.7.5 Sample sizes N and N' and number of replications R

In this section we explore the trade-off between sample size N (i.e., number of scenarios per sample average problem) and the number of replications R (i.e., number of sample average problems). In addition, we describe how we determine the number of scenarios N' selected to test each of the solutions generated by the sample average problems for our computational experiments.

Earlier we mentioned that we can use a large number of scenarios to assess the quality of a solution, x, and calculate an approximate $100(1-\alpha)\%$ confidence upper bound for $g(x)$ by equation (4.28). Clearly, the sample variance is one of the key factors that determines the width of such bound. Figure 4.1 shows how the sample variance of $\bar{q}_{N'}(x)$ changes as the number of scenarios, N', increases. This figure corresponds to one instance with 50 demand nodes, 50 facility nodes, $M = 3$, facility opening cost equal to 30 and variability up to ± 10 in the demand nodes coordinates. The sample variance decreases abruptly at the beginning but later it level offs reaching a plateau at around 2000 scenarios. This behavior was representative for the whole set of problems. Based on this observation, we determined that $N' = 2000$ was an appropriate number of scenarios for our computational experiments. Evaluating more scenarios would increase our computational time with very little gain in terms of the quality of the bound.

Figure 4.1 gives further insights regarding the solutions yielded by each replication. In theory each distinct sample average problem (or replication) would yield a distinct solution. However, in this case we can observe that the 10 replications

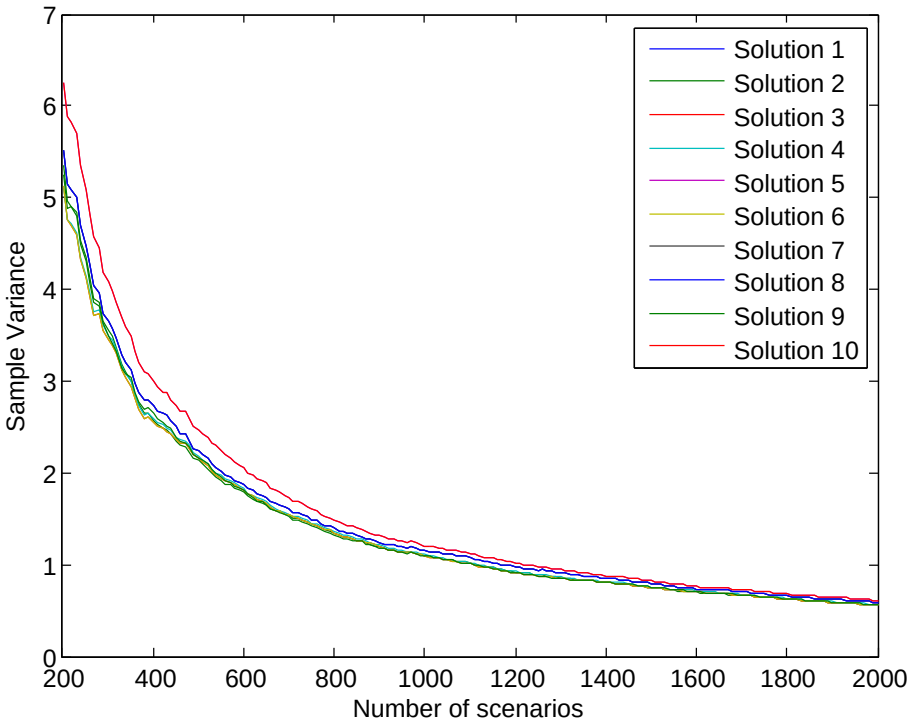

Figure 4.1: Sample variance of $\bar{q}_{N'}(x)$ for increasing number of scenarios, N'

provide only 7 distinct solutions. This is an important observation at the time of deciding the number of replications. Increasing the number of replications does not necessarily produce more candidate solutions.

Another reason to increase the number of replications, R, is to produce a tighter lower bound. In order to improve the lower bound defined by equation (4.27), one must decrease the sample variance of $\mathbb{E}[z_{LB}^N]$ either increasing the number of scenarios, N, for each sample average problem, or the number of replications, R. The results reported in the previous section were calculated using $N = 20$ and $R = 10$. If we had twice the computational time available, we could increase either N from 20 to 30, or R from 10 to 20. Table 4.6 shows the sample variance and total computational time for either 20 or 30 scenarios and 10 or 20 replications. Each

Instance	$N = 20$				$N = 30$	
	$R = 10$		$R = 20$		$R = 10$	
	variance	time (sec)	variance	time (sec)	variance	time (sec)
1	3.4922	805.76	1.8242	1589.31	2.7715	1512.27
2	2.7314	837.45	1.2617	1656.42	2.7534	1586.78
3	8.8446	801.28	3.7902	1582.86	5.7064	1510.22
4	6.1019	769.58	3.5685	1521.37	6.2817	1464.41
5	2.7945	804.11	2.6196	1587.08	3.0258	1501.25
6	6.1907	826.36	2.3549	1639.23	3.1859	1581.80
7	2.7922	817.08	2.4721	1614.63	3.3145	1561.22
8	3.4815	792.14	2.3656	1568.42	5.9125	1523.25
9	3.1519	869.80	1.7980	1721.59	0.7566	1670.31
10	1.7649	811.98	1.3088	1600.97	2.5983	1519.61

Table 4.6: Sample variance of $\mathbb{E}[z_{LB}^N]$ and computational time for various N and R values.

row represents the results for an instance with 50 demand nodes, 50 facility nodes, $M = 3$, facility opening cost equal to 30 and variability up to ± 10 in demand node coordinates. Interestingly, while we observed earlier that increasing the number of replications does not improve the quality of the upper bound, it does decrease the lower bound sample variance and consequently improve the lower bound. Such improvement is greater than the one obtained by increasing the number of scenarios per sample average problem. These results indicate that increasing the number of replications might be the preferred strategy to produce tighter confidence interval around the true optimal cost and improve the quality of the lower bound. Once again these results were representative for the whole set of instances.

4.8 Conclusions

In this chapter we have extended the ConFL problem to address two types of uncertainties: demand quantities and assignment edge costs. Furthermore, we have considered two types of models: polynomial-scenario and independent-activation.

In either of these models, for demand quantities we show that the problem can be solved optimally by the mean expected value problem. That is, the probability distribution of demands can be ignored as long as the risk of the solution is not considered. As the linear program proposed is optimizing expected values, we do not make any assumptions on the variability of the total cost of the solution. However, we could introduce a constraint that limits the maximum cost of the recourse problem to control for very unfavorable scenarios which would invariably change the problem at hand.

In the more general case, where variability occurs on the assignment edges, we show how to transform the problem into a deterministic equivalent formulation on a larger graph and apply our dual-ascent local search heuristic. We tested successfully this approach on a set of randomly generated instances with varying network structures and uncertainty levels. We find this strategy especially valuable for the polynomial-scenario model and for the independent-activation model within the sample average approximation method. We find that the stochastic solution has significant value. The value of the stochastic solution obtained by the DLS heuristic, as expected, depends on the variability level on the assignment costs. As the uncertainty is higher, the value of the stochastic solution is higher.

Lastly, we show how to use sample average approximation with a heuristic and a lower bounding procedure. Furthermore, we find that the DLS heuristic within the SAA method yields tight confidence intervals for the SConFL problem under a wide range of parameters. We explore the trade-offs between the number of scenarios and number of replications; and conclude that in general increasing the number of replications yields tighter performance confidence bounds for the true stochastic optimal value.

Chapter 5

Robust Optimization for the Connected Facility Connected Problem

5.1 Introduction

In this chapter we attempt to address the uncertainty in the ConFL problem by robust optimization, and search for solutions that perform well under varying customer realizations. One example that motivates this approach is the fiber-to-the-nodes (FTTN) technology, which the ConFL problem models (Gollowitzer and Ljubic 2011). There is growing interest to ensure broadband accessibility to every household (see Economist 2009a, Frenzel 2010); and consequently, to upgrade and extend fiber optic networks. However, there is limited information regarding which customers will subscribe to the service and their demand. On the other hand, demand ranges and coverage areas are easily identifiable.

From a broad perspective the FTTN layout is as follows. Serving offices distribute the signal to neighborhoods or homes through fiber optic connections. Fiber optic cables run from the serving offices to a cabinet serving a neighborhood, where end users connect using their existing copper (or even fiber optic, Economist 2009b) connections. Switching devices stored in these cabinets and fiber optic cables are expensive. Consequently, the problem is to determine how many cabinets (and hence switches) to deploy, where to place them, which customers to connect to them, and how to reconnect cabinets among each other and to the backbone.

A considerable amount of time elapses between the moment that cabinets (or facilities) are installed and interconnected and customers are assigned to each facility. Consequently, decision makers have to determine the "optimal" location of these cabinets with partial or limited information based on best estimates and forecasts of the location of future customers and their demand. Furthermore, to explicitly incorporate information on the location of each individual customer might not be possible or even practical. Evidence of the difficulty to estimate the location of customers is the fact that while AT&T can provide broadband access to 22 million of household, today only 2.5 millions are current subscribers of the U-verse service. Under this premise we propose a robust optimization approach that would explicitly incorporate the uncertainty of customer locations and provide a solution that mitigates the adverse effects of specific realizations. As we highlighted in the previous chapter, the ConFL problem models other similar problems that combine facility location decisions with connectivity requirements that face similar uncertainty.

We follow the optimization approach introduced by Bertsimas and Sim (2003) to search for robust solutions to the original problem. Under Bertsimas and Sim's approach, a robust solution is an optimal solution that satisfies an uncertainty budget constraint determined by the decision-maker. For robust optimization problems with uncertainty limited to the objective function coefficients, the budget constraint determines the number of such coefficients that can take their highest value at any solution and the objective function becomes a minimax function. Bertsimas and Sim proposed an algorithmic procedure to solve this minimax optimization problem that consists of solving a family of deterministic, called *nominal*, optimization prob-

lems. These nominal problems are deterministic versions of the original problem yet their cost matrices do not necessarily have the same structure and properties as the original problem. The main algorithm focuses on finding the optimal solution to the nominal problems and therefore an optimal solution to the robust optimization problem. For the case where the nominal problems are not solved to optimality, Bertsimas and Sim (2003) also show that an α-approximation algorithm for the deterministic problem can be used to obtain an α-approximation algorithm for the robust problem. We extend their approach to heuristics that do not necessarily have a known worst case approximation ratio but where it is possible to calculate lower bounds on the optimal solutions of the nominal problem.

For the ConFL problem, we look for network designs that are insensitive to different customer realizations yet exhibit low cost under different environments. We define customer location within an area. The final location of a customer may fall anywhere within this area. This strategy is specially appropriate for the broadband network problem, where neighborhoods can be described by geographic regions. Furthermore, this approach can be easily extended to assign range demand values for each customer or group of customers in a given region. We examine how such uncertainty affects the optimal network design and propose an efficient strategy to obtain high-quality solutions to the robust ConFL problem.

In summary, the contributions of this chapter are threefold. First, we introduce the ConFL problem under customer uncertainty and formalize its robust counterpart. Secondly, we extend Bertsimas and Sim's robust optimization solution approach to situations where one has a heuristic upper bound and a lower bound on

the optimal solution objective value for each nominal problem. In other words as long as we have a heuristic and a lower bounding mechanism for the deterministic problem, we can use them to find a heuristic solution for the robust problem together with a lower bound on the optimal objective value of the robust optimization problem. And finally, we propose an algorithm based on a DLS heuristic that yields high-quality solutions to the ConFL problem's robust counterpart.

In the next section we explore the black-box model for uncertainty and analyze how the formulation in (4.4) is affected by incertitude in the location and/or demand quantity of customers. The robust ConFL problem belongs to the family of robust problems whose uncertainty takes place only in the objective function (as opposed to the feasibility set). Consequently, we focus on a model that exploits this characteristic and specifically applies to problems with box uncertainty in the objective function.

5.2 Robust Optimization and The ConFL Problem

When the location of demand nodes or their demand is unknown and there is limited information regarding the probability distribution of this uncertainty, it is not possible to implement an expected value approach that minimizes the expected value of the objective function. Furthermore, optimizing over the expected value might lead to solutions that are very expensive under certain realizations, although they have the minimum expected value. Since this may not be desirable for the decision maker, the robust optimization approach provides a more appropriate

solution strategy in these situations.

Under the robust optimization framework one optimizes against the worst instances that might arise by using a min-max objective. In the ConFL problem, uncertain customer locations and/or demand quantities translate to uncertain assignment costs. Hence, one robust approach is to optimize against the worst-case scenario (i.e., worst-case analysis) and assume the highest assignment cost for each pair of demand node and facility node. However, indisputably this strategy yields a very conservative and most likely a very expensive solution. Alternatively, Bertsimas and Sim (2003) propose a robust optimization method that allows the decision maker to adjust the conservatism level of the solution sought. For the case where uncertainty exists only on the coefficients of the objective function, as it is the case in the ConFL problem under *customer uncertainty*[1], each solution is penalized by a weighted deviation term that incorporates the uncertainty cost of the solution.

In the ConFL problem under customer uncertainty, we consider two sources of uncertainty: customer location and demand quantities. To model customer location uncertainty we assume that demand node locations are unknown on a plane and consider two types of uncertainty regions: circular and rectangular. For the circular uncertainty region the location of each demand node is described by a circular disk defined by two parameters: center coordinates and radius (or diameter). For the rectangular uncertainty region the location is defined by a set of parameters: center coordinates and distinct deviation ranges for each axis direction. Then, the

[1]We use the expression *customer uncertainty* to refer jointly to location and/or demand quantity uncertainty

assignment costs range between the closest Euclidean distance between the facility node and the demand node uncertainty region, a_{ij}, and the longest distance between them, $a_{ij}+d_{ij}$; that is, the uncertain assignment cost, $\tilde{a}_{ij} \in [a_{ij}, a_{ij}+d_{ij}]$. When the facility node falls within the uncertainty region, $a_{ij} = 0$. Our analysis here easily extends to other shapes for the uncertain region. However, within this chapter we focus our attention to only these two shapes, circular and rectangular, as they model the uncertainty generally seen in practice. It turns out that the circle is simple enough to derive theoretical results, and the rectangle is sufficiently irregular to convey general characteristics pertaining to other shapes.

On the other hand, when demand quantities are uncertain we assume that they also range within a predetermined range. Let $\tilde{q}_i \in [q_i, q_i + \delta_i]$ be the random demand quantity for customer i, and α_{ij} be the per unit demand assignment cost for customer i to facility j. Then, the random assignment cost is given by $\tilde{a}_{ij} \in [\alpha_{ij}q_i, \alpha_{ij}q_i + \alpha_{ij}\delta_i] = [a_{ij}, a_{ij} + d_{ij}]$, where $a_{ij} = \alpha_{ij}q_i$ and $d_{ij} = \alpha_{ij}\delta_i$.

In both cases uncertainty translates into interval uncertainty as unknown assignment costs vary between a minimum and a maximum value without reference to a probability distribution. Note that if there is uncertainty on both location and demand, it is easy to see that the result is indeed uncertainty in the assignment costs.

5.2.1 Bertsimas and Sim's Robust Optimization Model

In general terms, given the following nominal combinatorial optimization problem

$$\begin{aligned} &\text{Minimize} && \tilde{c}^T \mathbf{v} \\ &\text{subject to} && \mathbf{v} \in X \end{aligned} \tag{5.1}$$

where $X \subseteq \{0,1\}^n$, Bertsimas and Sim (2003) define the robust counterpart, where each entry $\tilde{c}_j, j \in N = \{1,2,...,n\}$ takes values in $[c_j, c_j + d_j]$, $d_j \geq 0$, and X is a discrete set, as follows

$$\begin{aligned} \mathcal{Z} = \text{Minimize} \quad & c^T \mathbf{v} + \max_{\{R|R\subseteq N, |R|\leq \Gamma\}} \sum_{j\in R} d_j v_j \\ \text{subject to} \quad & \mathbf{v} \in X. \end{aligned} \tag{5.2}$$

The interpretation of this formulation is that at most Γ of the uncertain values in any solution will take their highest value. Consequently, the decision maker wants to minimize the maximum cost of a solution with at most Γ coefficients at their highest (or wort case) values. In other words, Γ represents a budget constraint that allows the decision maker to adjust the conservatism level of the solution sought.

The deviation term (or penalty term), $\max_{\{R|R\subseteq N, |R|\leq \Gamma\}} \sum_{j\in R} d_j v_j$, represents the sum of the maximum deviation of a specified number, Γ, of uncertain coefficients in the solution. The decision maker can select the conservatism parameter Γ between zero and the maximum number of uncertain coefficients in the problem, n.

A parameter of $\Gamma = 0$ corresponds to the optimistic case solution, which completely disregards the deviation terms in the cost coefficients, and assumes the best-case scenario with the minimum coefficient cost for each decision variable. A value of $\Gamma = n$ yields the worst-case scenario solution and each cost coefficient includes the deviation term.

Bertsimas and Sim (2003) propose an algorithm to find a solution to problem (5.2). They show that one can find the optimal solution to problem (5.2) by solving at most $n + 1$ nominal (deterministic) problems. To apply this method one must first identify the values of the deviation coefficients, d_j, and label them in decreasing order such that $d_1 \geq d_2 \geq ... \geq d_n \geq d_{n+1} = 0$. Then for each deviation coefficient, d_l, one defines a nominal problem, G_l, given by

$$\begin{aligned} G_l = \Gamma d_l + \text{ Minimize} \quad & c^T \mathbf{v} + \sum_{j=1}^{l} (d_j - d_l) v_j \\ \text{subject to} \quad & \mathbf{v} \in X. \end{aligned} \tag{5.3}$$

Note that for equal d_l values, one must solve only one G_l nominal problem. Consequently, the number of nominal problems is at most $n + 1$.

The problem $G_l - \Gamma d_l$ is a deterministic instance of the original problem (5.1) with cost coefficients equal to $c_j + \max(d_j - d_l, 0)$. Then, $G_1 - \Gamma d_1$ represents the best-case scenario problem and $G_{n+1} - \Gamma d_{n+1}$ yields the worst-case scenario problem.

Theorem 5.2.1. *[Bertsimas and Sim (2003)] The optimal function value, $\mathcal{Z}$, to problem (5.2) is given by $\mathcal{Z} = \min_{l=1,...,n+1} G_l$ and the optimal solution, $\boldsymbol{v}^* =$*

$\arg\min_{l=1,\ldots,n+1} G_l$.

Below we restate a sketch of the proof of Theorem 5.2.1 in Bertsimas and Sim (2003). This will be helpful in our subsequent analysis of using upper and lower bounds instead of solving the nominal problems to optimality.

Proof of Theorem 5.2.1

For a given value of $\mathbf{v}$, we can alternatively express the inner maximization problem in (5.2) as the following:

$$\begin{aligned} \zeta(\mathbf{v}) = \text{Maximize} \quad & \sum_{j \in N} (d_j v_j) u_j \\ \text{subject to} \quad & u_j \leq 1, \ \forall j \in N \\ & \sum_{j \in N} u_j \leq \Gamma \\ & u_j \geq 0, \ \forall j \in N. \end{aligned} \tag{5.4}$$

Note that $\zeta(\mathbf{v})$ has an integral solution for u_j (i.e., the polyhedron has integer extreme points). The values u_j represent whether a certain value d_j in the solution (i.e., with $v_j = 1$) belongs to the set of the Γ highest deviations in the solution $\mathbf{v}$. The dual problem to $\zeta(\mathbf{v})$ is given by the following minimization problem:

$$\begin{aligned} \omega(\mathbf{v}) = \text{Minimize} \quad & \Gamma\theta + \sum_{j \in N} \pi_j \\ \text{subject to} \quad & \pi_j + \theta \geq d_j v_j, \ \forall j \in N \\ & \theta, \pi_j \quad \geq 0 \end{aligned} \tag{5.5}$$

where π_j are the dual variables to the first set of constraints and θ to the conservatism

budget constraint. By strong duality $\zeta(\mathbf{v}) = \omega(\mathbf{v})$ and consequently we can replace $\zeta(\mathbf{v})$ by $\omega(\mathbf{v})$ in problem (5.2), which leads to the following minimization problem

$$\begin{aligned} \mathcal{Z} = \text{Minimize} \quad & \Gamma\theta + \sum_{j \in N} c_j v_j + \sum_{j \in N} \pi_j \\ \text{subject to} \quad & \mathbf{v} \in X \\ & \pi_j + \theta \geq d_j v_j, \ \forall j \in N \\ & \theta, \pi_j \geq 0. \end{aligned} \tag{5.6}$$

We can observe that constraint $\pi_j + \theta \geq d_j v_j$ is binding at the optimal solution. If for a given j this constraint were non-binding in the solution, we can improve its total cost by decreasing the value of π_j. Consequently, π_j satisfies $\pi_j = \max(d_j v_j - \theta, 0) = \max(d_j - \theta, 0) v_j$. Note that we can only take v_j outside the maximization function because $v_j \in \{0, 1\}$. Then we can substitute π_j in the formulation, rearrange the terms and rewrite the problem as

$$\begin{aligned} \mathcal{Z} = \text{Minimize} \quad & \Gamma\theta + \sum_{j \in N} (c_j + \max(d_j - \theta, 0)) v_j \\ \text{subject to} \quad & \mathbf{v} \in X, \ \theta \geq 0. \end{aligned} \tag{5.7}$$

We define as $\mathcal{Z}(\theta)$ problem (5.7) for a given value of θ. Then to find the optimal solution to the robust counterpart, we have to find the optimal solution $\theta^* \in \Re^+$ that minimizes $\mathcal{Z}(\theta)$. In other words, $\mathcal{Z} = \min_{\theta \in \Re^+} \mathcal{Z}(\theta)$.

$\mathcal{Z}(\theta)$ is neither a convex or concave function; however, $\mathcal{Z}(\theta)$ is linear over

Input: Problem instance and Γ.
Output: Solution, $\mathbf{v}^B$, and Solution value, Z^B
foreach d_l **do**
  Find an α-approximate solution v_H^l using Algorithm H for the nominal problem: $G_l - \Gamma d_l = \min_{\mathbf{v}\in X} c'\mathbf{v} + \sum_{j=1}^{l}(d_j - d_l)v_j$.
  Let $Z_l^H = c'\mathbf{v}_l^H + \max_{\{R|R\subseteq N,|R|\leq\Gamma\}} \sum_{j\in R} d_j(\mathbf{v}_l^H)_j$.
end
Let $l^* = \arg\min_{l=1,\ldots,n+1} Z_l^H$.
$Z^B = Z_{l^*}^H$; $\mathbf{v}^B = \mathbf{v}_{l*}^H$.

Figure 5.1: Bertsimas and Sim's Algorithm B

$\theta \in [d_{l+1}, d_l]$. The optimal solution to $\mathcal{Z}(\theta)$ for $\theta \in [d_l, d_{l+1}]$ must lie on one of the end points of the interval $[d_l, d_{l+1}]$. In other words, $\min_{\theta\in[d_l,d_{l+1}]} \mathcal{Z}(\theta) = \min(\mathcal{Z}(d_l), \mathcal{Z}(d_{l+1}))$. In a similar way, if we extend the interval such that $\theta \in [d_l, d_{l+2}]$, then $\min_{\theta\in[d_l,d_{l+2}]} \mathcal{Z}(\theta) = \min(\mathcal{Z}(d_l), \mathcal{Z}(d_{l+1}), \mathcal{Z}(d_{l+2}))$. Lastly, for $\theta \in \Re^+$, $\mathcal{Z}^*(\theta) = \min_{l=1,2,\ldots,n+1} \mathcal{Z}(d_l)$ (and $\theta^* = \arg\min_{l=1,2,\ldots,n+1} \mathcal{Z}(d_l)$). Furthermore, $\mathcal{Z}(d_l) = G_l$. This concludes the proof of Theorem 5.2.1.□

5.2.2 Extending Bertsimas and Sim's Robust Approximation Algorithm

We first describe Bertsimas and Sim's robust approximation algorithm. Figure 5.1 depicts the algorithm, which is referred to as Algorithm B in their paper. First, one finds an α-approximate solution, $\mathbf{v}_l^H$, to the nominal problem $G_l - \Gamma d_l$, for each deviation term, d_l. Following that one computes the total cost of the α-approximate solution, $\mathbf{v}_l^H$, in the robust counterpart problem (5.2). Lastly, the solution that yields the lowest total cost is the algorithmic solution to the robust counterpart problem.

Bertsimas and Sim show that given a polynomial time α-approximation algorithm capable of solving the nominal problems, Algorithm B yields a polynomial time α-approximate solution to its robust counterpart. The approach is predicated on having an approximation algorithm available for the combinatorial optimization problem. We show that this approach can be extended to any heuristic procedure if we are able to generate a lower bound for each nominal problem. In cases where the nominal problem is hard to solve, the approach we outline can be used to obtain high quality heuristic solutions to the robust combinatorial optimization problem. We point out that for certain problems even though there might exist a high quality approximation algorithm to solve the deterministic version of the problem, the same algorithm might not be suitable to solve the nominal problems. That is the case with the 3/2-approximation algorithm for the traveling salesman problem (Christofides 1976), which requires that the cost matrix satisfies the triangle inequality.

We modify Algorithm B to use any heuristic (as opposed to an approximation algorithm) together with a lower bounding procedure for the nominal problems, $G_l - \Gamma d_l$. Figure 5.2 depicts this modified Algorithm B, which we will call Robust Combinatorial Optimization Heuristic (RCOH), and finds a lower bound for each nominal problem in conjunction with a heuristic solution.

For each deviation coefficient, d_l, we first find a heuristic solution, $\mathbf{v}_l^H$, and a lower bound, Ω_l^{LB}, using Heuristic H and a lower bounding procedure for the nominal problem $\Omega_l = G_l - \Gamma d_l = \min_{\mathbf{v}\in X} c'\mathbf{v} + \sum_{j=1}^{l}(d_j - d_l)v_j$. Let Ω_l^H be the objective value of the heuristic solution, $\mathbf{v}_l^H$. Let α_l be the lower bound gap of solution $\mathbf{v}_l^H$; i.e.,

Input: Problem instance and Γ.
Output: Solution, $\mathbf{v}^B$, solution value, Z^B, and a solution quality assessment, β
foreach d_l **do**
  Find a heuristic solution $\mathbf{v}_l^H$ and lower bound Ω_l^{LB} using Heuristic H and a lower bounding procedure for the nominal problem:
  $G_l - \Gamma d_l = \min_{\mathbf{v}\in X} c'\mathbf{v} + \sum_{j=1}^{l}(d_j - d_l)v_j$.
  Let $Z_l^H = c'\mathbf{v}_l^H + \max_{\{R|R\subseteq N,|R|\leq\Gamma\}} \sum_{j\in S} d_j(v_l^H)_j$.
end
Let $l^* = \arg\min_{l=1,\ldots,n+1} Z_l^H$.
Let $Z^{LB} = \min_{l=1,\ldots,n+1} \Gamma d_l + \Omega_l^{LB}$.
$Z^B = Z_{l^*}^H$; $\mathbf{v}^B = \mathbf{v}_{l^*}^H$.
$\beta = \frac{Z^B - Z^{LB}}{Z^{LB}}$

Figure 5.2: Robust Combinatorial Optimization Heuristic (RCOH)

$\alpha_l = \frac{\Omega_l^H - \Omega_l^{LB}}{\Omega_l^{LB}}$ and let $\alpha = \max_{l=1,\ldots,n+1} \alpha_l$. Then we show that the solution yielded by RCOH using Heuristic H for the robust counterpart problem, Z^B, has a lower bound gap β, less than or equal to α. In other words, $Z^{LB} \leq \mathcal{Z} \leq Z^B \leq (1+\beta)Z^{LB}$ and $\beta \leq \alpha$.

Theorem 5.2.2. *If $\alpha = \max_{l=1,\ldots,n+1} \alpha_l$, where α_l is the a bound gap for nominal problem Ω_l, then the solution yielded by RCOH for the robust counterpart has a lower bound gap β, which is less than or equal to α.*

In order to prove Theorem 5.2.2 we need the following four lemmas.

Lemma 5.2.3. *Let $G_l^{LB} = \Gamma d_l + \Omega_l^{LB}$ be a lower bound to nominal problem, G_l, then $\mathcal{Z}^{LB} = \min_{l=1,\ldots,n+1} G_l^{LB}$ is a lower bound to $\mathcal{Z}$.*

Proof.

Let $\mathcal{Z} = G_{\bar{l}}$, where $\bar{l}$ is the nominal problem that solves (5.2) to optimality, then $G_{\bar{l}}^{LB}$ is a lower bound to $\mathcal{Z}$. Furthermore, $Z^{LB} = \min_{l=1,\ldots,n+1} G_l^{LB} \leq G_{\bar{l}}^{LB}$; and consequently, Z^{LB} is a lower bound to $\mathcal{Z}$.□

Lemma 5.2.4. *For all* $\boldsymbol{v} \in X$ *and* $\theta \in \Re^{+}$,

$$\max_{\{R|R\subseteq N,|R|\leq\Gamma\}} \sum_{j\in R} d_j v_j \leq \sum_{j\in N} \max(d_j - \theta, 0)v_j + \Gamma\theta. \tag{5.8}$$

Before we present a formal proof for Lemma (5.2.4), we discuss a simple example to illustrate and understand the meaning of this lemma. Figure 5.3(a) shows ten deviation values in decreasing order. Each bar represents one d_l value with unit width and height equal to d_l. In this example, θ is equal to 52 and Γ equals 6. Then, the light blue area in Figure 5.3(b) highlights the region that corresponds to the left-hand side of equation (5.8). On the other hand, the light blue area in Figure 5.3(c) depicts the area that corresponds to the right-hand side of equation (5.8). By comparing both light blue areas, we can conclude that inequality (5.8) strictly holds. Furthermore, we can conclude that inequality (5.8) holds strictly for $\theta > 43.18$ and $\theta < 40.62$, and holds at equality for $\theta \in [40.62, 43.18]$.

Proof of Lemma 5.2.4.

Given a solution $\mathbf{v} \in X$, we denote by $\mathcal{D}_{\mathbf{v}}$ the subset of deviations, d_j, whose corresponding v_j element is non-zero in the solution. Furthermore, we label the elements in $\mathcal{D}_{\mathbf{v}}$ in decreasing order such that $d_1 \geq d_2 \geq ... \geq d_{|\mathcal{D}_{\mathbf{v}}|}$. Then, we can calculate the left-hand side maximization term by adding up the highest Γ deviations in $\mathcal{D}_{\mathbf{v}}$. That is,

$$\max_{\{R|R\subseteq\{1,...,|\mathcal{D}_{\mathbf{v}}|\},|R|\leq\Gamma\}} \sum_{j\in R} d_j = \sum_{j=1}^{\Gamma} d_j. \tag{5.9}$$

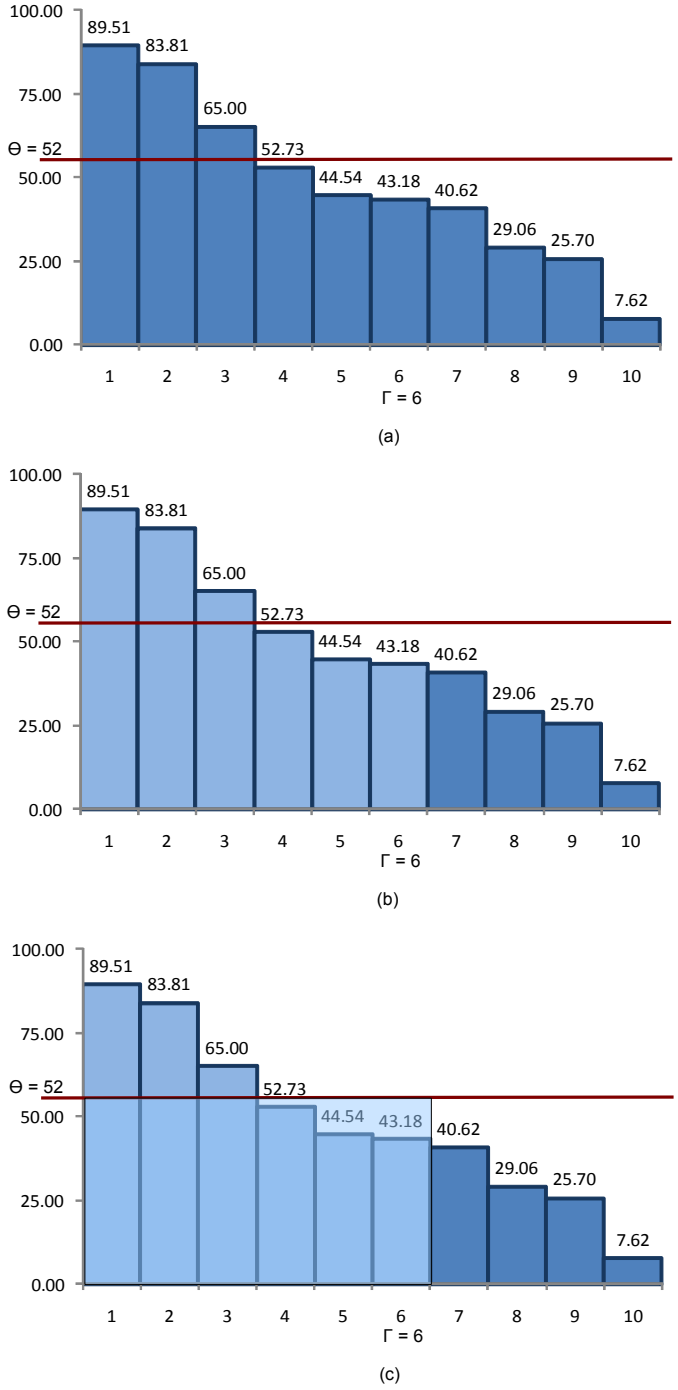

Figure 5.3: Example for Lemma 5.2.4

Without loss of generality, we assume that $\Gamma \leq |\mathcal{D}_{\mathbf{v}}|$. Otherwise, the summation on the right-hand side of (5.9) goes up to $\min\{\Gamma, |\mathcal{D}_{\mathbf{v}}|\}$.

We have to consider three situations depending on the value of θ:

Case (A): $\theta > d_1$

In this case $\sum_{j \in |\mathcal{D}_{\mathbf{v}}|} \max(d_j - \theta, 0) = 0$, and clearly, $\sum_{j=1}^{\Gamma} d_j \leq \Gamma\theta$.

Case (B): $\theta \leq d_{|\mathcal{D}_{\mathbf{v}}|}$

In this case,

$$\begin{aligned}\sum_{j=1}^{|\mathcal{D}_{\mathbf{v}}|} \max(d_j - \theta, 0) &= \sum_{j=1}^{|\mathcal{D}_{\mathbf{v}}|} (d_j - \theta) \\ &\geq \sum_{j=1}^{\Gamma} (d_j - \theta) \\ &= \sum_{j=1}^{\Gamma} d_j - \Gamma\theta.\end{aligned}$$

In other words, by reordering the terms, we observe that

$$\sum_{j=1}^{\Gamma} d_j \leq \sum_{j=1}^{|\mathcal{D}_{\mathbf{v}}|} \max(d_j - \theta, 0) + \Gamma\theta.$$

Case (C): $\theta \in (d_{|\mathcal{D}_{\mathbf{v}}|}, d_1]$

Without loss of generality, assume that $\theta \in (d_{l+1}, d_l]$. Then, we have to consider two cases: (i) $\Gamma \geq l$ (and $d_\Gamma \leq d_l$) and (ii) $\Gamma < l$ (and $d_\Gamma \geq d_l$). Note that all of the deviations, d_j's, do not need to be distinct. Given $\theta \in (d_{|\mathcal{D}_{\mathbf{v}}|}, d_1]$, we can always find an l such that the interval $(d_{l+1}, d_l]$ contains θ.

Case (i). $\Gamma \geq l$, then

$$\begin{aligned}\max_{\{R|R\subseteq\{1,\ldots,|\mathcal{D}_{\mathbf{v}}|\},|R|\leq\Gamma\}} \sum_{j\in R} d_j &= \sum_{j=1}^{l} d_j + \sum_{j=l+1}^{\Gamma} d_j + l\theta - l\theta \\ &= \sum_{j=1}^{l}(d_j-\theta) + \sum_{j=l+1}^{\Gamma} d_j + l\theta\end{aligned}$$

By assumption $\forall j \geq l+1,\ \theta > d_j,$ thus

$$\begin{aligned}&\leq \sum_{j=1}^{l}(d_j-\theta) + \sum_{j=l+1}^{\Gamma} \theta + l\theta \\ &= \sum_{j=1}^{l}(d_j-\theta) + (\Gamma-l)\theta + l\theta \\ &= \sum_{j=1}^{l}(d_j-\theta) + \Gamma\theta\end{aligned}$$

Note that $\max(d_j-\theta,0) = d_j-\theta$ for $j \leq l,$ and

$$\max(d_j-\theta,0) = 0 \text{ for } j \geq l+1, \text{ thus}$$

$$\max_{\{R|R\subseteq\{1,\ldots,|\mathcal{D}_{\mathbf{v}}|\},|R|\leq\Gamma\}} \sum_{j\in R} d_j \leq \sum_{j=1}^{|\mathcal{D}_{\mathbf{v}}|} \max(d_j-\theta,0) + \Gamma\theta. \tag{5.10}$$

Case (ii). $\Gamma < l$, then

$$\begin{aligned}
\max_{\{R|R\subseteq\{1,\ldots,|\mathcal{D}_{\mathbf{v}}|\},|R|\leq\Gamma\}} \sum_{j\in R} d_j &= \sum_{j=1}^{l} d_j - \sum_{j=\Gamma+1}^{l} d_j + l\theta - l\theta \\
&= \sum_{j=1}^{l} (d_j - \theta) + l\theta - \sum_{j=\Gamma+1}^{l} d_j \\
&= \sum_{j=1}^{l} (d_j - \theta) + \Gamma\theta + (l-\Gamma)\theta - \sum_{j=\Gamma+1}^{l} d_j \\
&= \sum_{j=1}^{l} (d_j - \theta) + \Gamma\theta + \sum_{j=\Gamma+1}^{l} \theta - \sum_{j=\Gamma+1}^{l} d_j \\
&= \sum_{j=1}^{l} (d_j - \theta) + \Gamma\theta + \sum_{j=\Gamma+1}^{l} (\theta - d_j)
\end{aligned}$$

$$\text{By assumption } \theta \leq d_j \ \forall j \leq l, \text{ then } \sum_{j=\Gamma+1}^{l} (\theta - d_j) \leq 0, \text{ and}$$

$$\begin{aligned}
&\leq \sum_{j=1}^{l} (d_j - \theta) + \Gamma\theta \\
\max_{\{R|R\subseteq\{1,\ldots,|\mathcal{D}_{\mathbf{v}}|\},|R|\leq\Gamma\}} \sum_{j\in R} d_j &\leq \sum_{j=1}^{|\mathcal{D}_{\mathbf{v}}|} \max(d_j - \theta, 0) + \Gamma\theta
\end{aligned}$$

Now we can reintroduce the decision variables v_j with its zero terms and conclude that

$$\max_{\{R|R\subseteq N,|R|\leq\Gamma\}} \sum_{j\in R} d_j v_j \leq \sum_{j\in N} \max(d_j - \theta, 0) v_j + \Gamma\theta \text{ for all } \mathbf{v} \in X. \tag{5.11}$$

This finalizes our proof of Lemma 5.2.4.□

We now use Lemma 5.2.4 to show our next lemma. Lemma 5.2.5 states that the total cost of any solution in the robust optimization problem (5.2) is less than

its total cost in the nominal problem defined by any θ value in $\Re^+$ and $\Gamma \in \mathbb{Z}^+$.

Lemma 5.2.5. *Given* $\boldsymbol{v} \in X$, $Z(\boldsymbol{v}) \leq \Gamma\theta + \Omega(\boldsymbol{v}, \theta)$ *for all* $\theta \in \Re^+$ *and* $\Gamma \in \mathbb{Z}^+$.

Proof.

By definition,

$$Z(\mathbf{v}) = c'\mathbf{v} + \max_{\{R|R\subseteq N, |R|\leq\Gamma\}} \sum_{j\in R} d_j(\mathbf{v})_j$$

By Lemma 5.2.4,

$$\leq c'\mathbf{v} + \sum_{j\in N} \max(d_j - \theta, 0)(\mathbf{v})_j + \Gamma\theta$$

By definition of $\Omega(\mathbf{v}, \theta) = c'\mathbf{v} + \sum_{j\in N} \max(d_j - \theta, 0)(\mathbf{v})_j$,

$$= \Omega(\mathbf{v}, \theta) + \Gamma\theta$$

Consequently,

$$Z(\mathbf{v}) \leq \Gamma\theta + \Omega(\mathbf{v}, \theta).\square$$

Lastly, Lemma 5.2.6 states that the heuristic solution obtained for the robust problem (5.2) by RCOH has a total cost less than the total cost of any heuristic solution in the nominal problems, G_l, for all $l \in N$.

Lemma 5.2.6. $Z^B \leq \Gamma d_l + \Omega_l^H$ *for all* $l \in N$.

Proof.

Since $Z^B \leq Z_l^H$ for all $l \in N$, by Lemma 5.2.5 $Z^B \leq \Gamma d_l + \Omega_l^H$. $\square$

Finally, we can prove Theorem 5.2.2 that says that the lower bound gap of the robust solution yielded by RCOH is less than the worst lower bound gap for any

individual nominal problem, $G_l - \Gamma d_l$.

Proof of Theorem 5.2.2.

Let β be the lower bound gap of the robust solution yielded by RCOH. That is,

$$\beta = \frac{Z^B - Z^{LB}}{Z^{LB}}. \tag{5.12}$$

Let $Z^{LB} = \Omega_j^{LB} + \Gamma d_j$ for some $j \in N$. Then, by Lemma 5.2.6

$$\begin{aligned}
\beta &\leq \frac{\Omega_j^H + \Gamma d_j - (\Omega_j^{LB} + \Gamma d_j)}{\Omega_j^{LB} + \Gamma d_j} \\
&= \frac{\Omega_j^H - \Omega_j^{LB}}{\Omega_j^{LB} + \Gamma d_j} \\
&= \frac{\alpha_j \Omega_j^{LB}}{\Omega_j^{LB} + \Gamma d_j} \\
&\leq \alpha \left[\frac{\Omega_j^{LB}}{\Omega_j^{LB} + \Gamma d_j} \right] \\
&\text{since } \Gamma d_j \geq 0,\ 0 \leq \frac{\Omega_j^{LB}}{\Gamma d_l + \Omega_j^{LB}} \leq 1, \text{ thus} \qquad (5.13) \\
\beta &\leq \alpha. \square
\end{aligned}$$

Based on this result one would be tempted to conclude that β decreases for higher values of Γ. While that is the behavior that we observe in the majority of our computational experiments, it is not necessarily true in all cases. The value of d_j varies for different Γ values. Consequently, β does decrease for higher values of Γ as long as the decrease in d_j does not outperform the change in Γ and Γd_j increases.

5.2.3 The ConFL problem robust counterpart

The ConFL problem is an NP-complete problem costly to solve to optimality, Ljubić (2007). Consequently, a series of approximation algorithms and heuristics have been proposed to address the ConFL problem (see Swamy and Kumar 2004, Tomazic and Ljubić 2008, Eisenbrand et al. 2008, Gollowitzer and Ljubic 2011, Jung et al. 2008, Ljubić 2007). The question remains whether a heuristic (as opposed to an approximation algorithm with a performance guarantee) can be implemented to solve the nominal problems within Theorem 5.2.1 framework and yet find high-quality solutions to its robust counterpart.

In Bardossy and Raghavan (2010) we propose a high quality heuristic based on dual-ascent and local search (DLS) and here we show that the same DLS heuristic can be implemented within a variant of Algorithm B to find high-quality solutions for the robust counterpart and lower bound gaps for the solution.

The robust counterpart of the ConFL problem concerns the problem where each assignment cost $\tilde{a}_{ij}$, $\{i,j\} \in E(D)$, takes values in $[a_{ij}, a_{ij}+d_{ij}]$, $d_{ij} \geq 0, \{i,j\} \in E(D)$, but the set of feasible solutions $(x,y,z) \in \mathcal{X}$ that satisfy constraints (4.4b)-(4.4f) does not change for different realizations.

Using Bertsimas and Sim (2003) definition of the robust counterpart, we would like to find a solution $(x,y,z) \in \mathcal{X}$ that minimizes the maximum cost $\sum_{i\in F} f_i z_i + \sum_{\{i,j\}\in E(S\cup F)} b_{ij} y_{ij} + \sum_{i\in F, j\in D} \tilde{a}_{ij} x_{ij}$ such that at most Γ of the coefficients $\tilde{a}_j$ are allowed to change. In other words, we want to minimize a deterministic ConFL problem objective function, $\mathcal{W}(x,y,z) = \sum_{i\in F} f_i z_i + \sum_{\{i,j\}\in E(S\cup F)} b_{ij} y_{ij} + \sum_{i\in F, j\in D} a_{ij} x_{ij}$,

plus the maximum deviation in Γ assignment edges. The robust ConFL is given by the following formulation.

$$\begin{aligned} \mathcal{Z} = \text{Minimize} \quad & \mathcal{W}(x,y,z) + \max_{\{R|R\subseteq E(D),|R|\leq\Gamma\}} \sum_{\{i,j\}\in R} d_{ij}x_{ij} \\ \text{subject to:} \quad & (x,y,z) \in \mathcal{X}. \end{aligned} \tag{5.14}$$

In any solution to a ConFL instance there is a deviation term for each demand node; consequently, the penalty term incorporates the uncertainty in the location or demand quantity of each demand node. In other words, the optimal solution of the robust problem is determined by the uncertainty level of the Γ most uncertain demand nodes.

The added term in the objective function weighs in the uncertainty of the Γ highest possible deviations in the assignment costs for each feasible solution. Alternatively, we can interpret this term as a buffer or protection term in the total cost solution, $\mathcal{Z}$. Let (x^*, y^*, z^*) be the optimal solution to problem (5.14). The realized total cost will always be higher than or equal to $\mathcal{W}(x^*, y^*, z^*)$ and with some probability below $\mathcal{Z}^*$. The probability that the realized cost is less than $\mathcal{Z}^*$ is directly proportional to Γ.

Once we have selected solution (x^*, y^*, z^*), that is, opened facilities z^*, connected them using tree y^*, and assigned customers to open facilities according to x^*; when the assignment costs are known, we may observe with some probability a total cost greater than $\mathcal{Z}$. However, depending on the value of Γ the probability

Input: ConFL instance: $G = (V, E)$, deviation matrix, $\mathcal{D}$, and Γ.
Output: Total cost, Z^B, (x^B, y^B, z^B) solution to robust ConFL problem, lower bound, Z^{LB}, and lower bound gap, β.
Initialize $Z^B = \infty+$, and $Z^{LB} = \infty+$;
foreach $d_l \in \mathcal{D}$ **do**
  Use the DLS heuristic to obtain a solution, (x_l, y_l, z_l), and lower bound, Ω_l^{LB}, to the ConFL problem, Ω_l;
  Calculate Z_l^H;
  if $Z_l^H < Z^B$ **then**
    Update robust solution: $Z^B = Z_l^H$ and $(x^B, y^B, z^B) = (x_l, y_l, z_l)$;
  end
  if $\Gamma d_l + \Omega_l^{LB} < Z^{LB}$ **then**
    Update robust lower bound: $Z^{LB} = \Gamma d_l + \Omega_l^{LB}$;
  end
end
Calculate solution lower bound gap: $\beta = \frac{Z^B - Z^{LB}}{Z^{LB}}$;

Figure 5.4: RCOH for the Robust ConFL problem

that the total cost realized will be less or greater than $\mathcal{Z}$ will vary. The higher Γ yields a higher $\mathcal{Z}$ and also ensures that the probability of experiencing a cost higher than $\mathcal{Z}$ decreases. Higher Γ values put higher weight to costly realizations and cap the total cost under more unfavorable scenarios.

Figure 5.4 depicts how we implement RCOH using our DLS heuristic for the robust ConFL problem. This algorithm uses the DLS heuristic to find a solution to each nominal problem, computes the total cost of each solution in the objective function of problem (5.14), and stores the minimum cost solution. Lastly, the algorithm provides the lower bound gap, β of the robust solution.

One

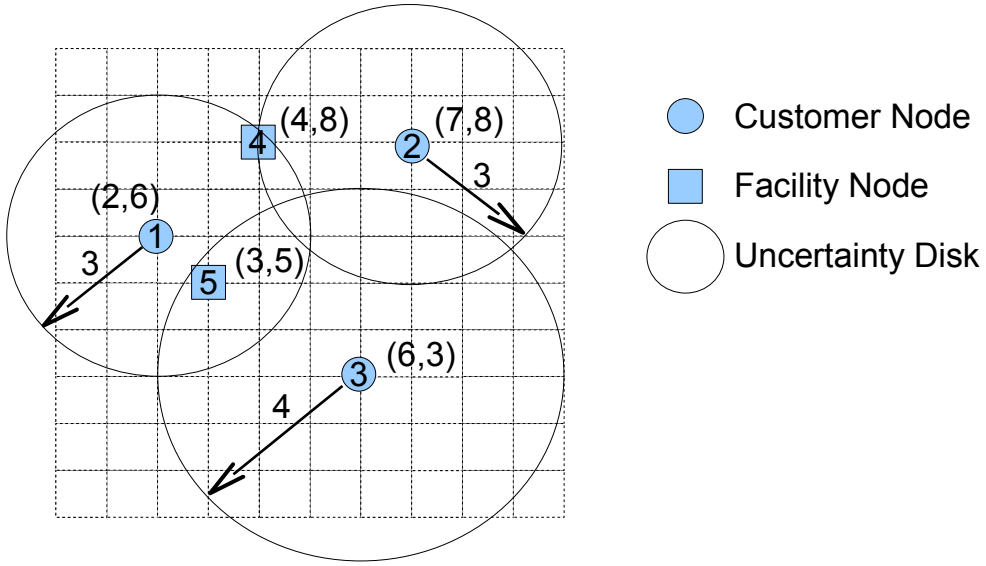

Figure 5.5: Example of robust ConFL problem

5.3 An Example of RConFL problem using RCOH

In this section we provide an example to show how we apply RCOH using a small RConFL instance. Figure 5.5 depicts an instance with two facility nodes and three customer nodes where each customer has an uncertainty location within a disk. For this example we set the facility opening cost equal to 1 and the M factor equal to 2. Table 5.1 shows the coordinates for each nodes' center and the radius of the demand nodes' uncertainty disk.

Next we calculate assignment costs, that is, minimum assignment cost and maximum possible deviation, for each pair of facility and customer nodes, and tree edge costs. Table 5.2 shows these costs.

The minimum assignment cost is given by $a_{ij} = \max\{\sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2} - r_i, 0\}$ and the maximum deviation by $d_{ij} = \min\{\sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2} + r_i, 2r_i\}$,

Customer Nodes			
Node	x-coord	y-coord	radius
1	2	6	3
2	7	8	3
3	6	3	4
Facility Nodes			
Node	x-coord	y-coord	
4	4	8	
5	3	5	

Table 5.1: Coordinates for nodes in the example

where (i_1, i_2) are coordinates of node i.

We observe that there are five distinct deviation values, d_l; consequently, following Bertsimas and Sim (2003)'s algorithm we need to solve six distinct nominal problems, one for each deviation value, plus an additional problem for $d_l = 0$, $\mathcal{D} = \{8.00, 7.61, 6.00, 5.83, 4.41, 0\}$.

Table 5.3 depicts the assignment costs for each nominal problem l, $G_l - \Gamma d_l$. We use our dual-ascent local search (DLS) heuristic to obtain a solution, $(x, y, z)_l$, for each nominal problem. Table 5.4 shows the lower bounds obtained for each nominal problem using dual-ascent (DA), the best solution value yielded by DLS (note that for this small example the solutions by DA and DLS are equal), and lastly the total cost yielded by the solution $(x, y, z)_l$ in the robust formulation for $\Gamma = 2$. Based on these results, we observe that $Z^{LB} = 16.385$ and $Z^B = 16.385$ and can conclude that the best and optimal solution is to open facility 4. In this case, DA's lower bound proves that the solution obtained by the heuristic is the optimal solution.

Assignment Costs			
Customer	Facility	Minimum Cost	Deviation
1	4	0.00	5.83
1	5	0.00	4.41
2	4	0.00	6.00
2	5	2.00	6.00
3	4	1.39	8.00
3	5	0.00	7.61

Tree Edge Cost		
Facility	Facility	Cost
4	5	6.32

Table 5.2: Assignment and tree edge costs in the example

Assignment Costs							
		Deviation d_l					
Customer	Facility	8.00	7.61	6.00	5.83	4.41	0.00
1	4	0.00	0.00	0.00	0.00	1.41	5.83
1	5	0.00	0.00	0.00	0.00	0.00	4.41
2	4	0.00	0.00	0.00	0.17	1.59	6.00
2	5	2.00	2.00	2.00	2.17	3.59	8.00
3	4	1.39	1.78	3.39	3.56	4.97	9.39
3	5	0.00	0.00	1.61	1.78	3.19	7.61

Table 5.3: Assignment costs for each nominal problem in the example

			$\Gamma = 2$		
d_l	Ω_l^{LB}	G_l^{LB}	G_l^H	Z_l^H	Solution
4.41	7.777	16.605	16.605	16.605	Open Facility 5
5.83	4.729	16.386	16.386	16.385	Open Facility 4
6.00	4.385	**16.385**	16.385	**16.385**	Open Facility 4
7.61	2.780	17.991	17.991	16.385	Open Facility 4
8.00	2.385	18.385	18.385	16.385	Open Facility 4
0.00	21.020	21.020	21.020	16.605	Open Facility 5

Table 5.4: Preliminary results for the example

	Γ			
Solution	0	1	2	3
Open Facility 4	**2.385**	**10.385**	**16.385**	22.214
Open Facility 5	3.000	10.606	16.606	**21.020**
Z^{LB}	2.385	10.385	16.385	21.020

Table 5.5: Example results for $0 \leq \Gamma \leq 3$

We initially set Γ equal to 2. However, we could vary Γ between 0 and 3 and observe how the optimal solution changes. A nice feature of this robust optimization approach is that one can easily vary Γ and observe whether the optimal solution changes. Once we obtain a lower bound and feasible solution for each nominal problem using DLS, we can quickly recalculate the lower bound, Z^{LB}, for a different Γ value by adding the term Γd_l to each Ω_l^{LB}, and reassessing each solution cost in Problem (5.2).

When $\Gamma = 0$ we obtain the best-case scenario solution, while when $\Gamma = 3$ we obtain the worst-case scenario solution. (Any $\Gamma > 3$ yields the same solution and total cost than $\Gamma = 3$.) We observe that the six nominal problems, $G_l - \Gamma d_l$, produced only two distinct solutions; that is, to either open facility 4 or facility 5. Table 5.5 shows the lower bound for each problem and the value function for each solution in the robust optimization formulation. To open facility 4 is optimal for every Γ value except 3. In other words, to open facility 4 is a robust solution under a wide range of realizations except when the worst-case scenario takes place.

5.4 Special Case

The ConFL problem is related to the Weber problem under location uncertainty introduced in Cooper (1978), where the location of demand nodes is uncertain and within an uncertainty disk. The *centered problem* refers to the Weber problem that assumes that demands are located in the center of the uncertainty disk. Cooper (1978) found (and later Juel (1981) correctly noted) that if the optimal solution to the centered problem on the Euclidean norm lies outside of all of the uncertainty circular disks, then the optimal solution to the worst-case problem is the solution to the centered problem and its optimal solution value is equal to the optimal solution value for the centered problem plus a constant term.

We can use a similar argument to note that if none of the potential facilities falls within an uncertainty circular disk, one can find the optimal solution for the whole family of Γ robust problems by solving only the best-case scenario problem. Furthermore, the optimal solution is the same for all Γ parameters.

This result does not hold when the optimal solution to the best-case scenario includes at least one facility node within a customer uncertainty disk. That was the case in our earlier example, where facility node 4 falls within the uncertainty disk for demand node 1.

The only difference between the robust ConFL formulation and the ConFL formulation lies in the objective function—specifically, in the penalty term, that is, the maximization term in the objective function. When this term is constant and equal to $\max_{\{\mathcal{D}|\mathcal{D}\subset D,|\mathcal{D}|\leq\Gamma\}} \sum_{i\in\mathcal{D}} d_i$, it does not depend on the assignment decision

x_{ij} and the optimal solution to the best-case scenario problem is an optimal solution to the robust problem. We can show that this result holds for the disk uncertainty.

Theorem 5.4.1. *Let $(x^*, y^*, z^*)_{BC}$ be an optimal solution to the best-case problem, and $\mathcal{F}$ be the set of open facilities in the solution, that is, $\mathcal{F} = \{j | y_j^* = 1\}$. Now, if every facility node $j \in \mathcal{F}$, falls outside the uncertainty disk for every demand node $i \in D$, then $(x^*, y^*, z^*)_{BC}$ is also an optimal solution to the robust problem (5.14) for any value Γ.*

Proof.

By assumption $j \in \mathcal{F}$ lies outside the uncertainty disk with diameter d_i for all $i \in D$, and assignment cost $\tilde{a}_{ji} \in [a_{ji}, a_{ji} + d_{ji}]$. Since any $j \in F$ lies outside i's uncertainty disk, $d_{ji} = d_i, \forall j \in F$. Hence, we can rearrange the order of summation in equation (4.4a) as follows: $\sum_{i \in \mathcal{D}} \sum_{j \in F} d_{ij} x_{ij} = \sum_{i \in \mathcal{D}} d_i \sum_{j \in F} x_{ij}$. However, at optimality $\sum_{j \in F} x_{ij} = 1$. (While constraint (4.4d) enforces that this sum is greater than or equal to zero, the non-negative coefficients, a_{ij}, ensure that the constraint is satisfied with equality in the optimal solution.) Then, $\sum_{i \in \mathcal{D}} \sum_{j \in F} d_{ij} x_{ij} = \sum_{i \in \mathcal{D}} d_i$. Clearly, the penalty term $\max_{\{\mathcal{D} | \mathcal{D} \subset D, |\mathcal{D}| \leq \Gamma\}} \sum_{i \in \mathcal{D}} d_i$ is a constant dependent on Γ but unaffected by the problem solution. Thus, the optimal solution to problem (5.14) is obtained by solving the best-case problem.□

Corollary 5.4.2. *If every $j \in F$ falls outside the uncertainty disk for any demand node $i \in D$, then the optimal solution to the best-case problem is the optimal solution to the robust problem (5.14).*

Proof.

Clearly, $\mathcal{F} \subseteq F$ and the rest follows from Theorem 5.4.1. $\square$

5.5 Computational Experiments

We now report on a set of computational experiments with our DLS heuristic on the robust ConFL problem. The purpose of these experiments is to assess the effectiveness of DLS, in terms of solution quality and computational time, to solve the RConFL problem under various uncertainty levels and uncertainty regions. Furthermore, these experiments allow us to observe whether the proposed solution changes under different conservatism parameters Γ. We coded our heuristic in Visual Studio 2005 (C++). We conducted all runs on an AMD Athlon$^{\text{TM}}$ 62 X2 Dual, 2.61 GHz machine with 3GB of RAM.

5.5.1 Problem Generation and Characteristics

We generated problems by first selecting nodes randomly located on a 100 x 100 square grid. The Euclidean distances were used as a basis for the edge lengths. The assignment edge costs are equal to the edge lengths between demand nodes and facility nodes, while tree edge costs are equal to the edge lengths multiplied by an M factor. The M factor illustrates the significantly higher (in terms of cost per unit distance) connection cost of edges in the tree T. For our test instances, we set the M factor equal to 3 and 7, respectively. Bardossy and Raghavan (2010) found in their extensive computational experiments that as the M factor increases the average gap for the DLS heuristic first increases but later decreases, reaching the

maximum at $M = 3$. In addition, we expect that the uncertainty in the assignment costs to have a higher impact on the overall cost of the solution for smaller values of M, and consequently, we anticipate to observe smaller optimality gaps for the instances with $M = 7$ than for $M = 3$.

Each of the instances has 50 demand nodes, 50 facility nodes and 20 pure potential Steiner nodes. Bardossy and Raghavan (2010) found that instances with equal number of demand and facility nodes are usually the most difficult one. Lastly, the facility opening cost was set to 30, which leads to the most difficult instances. In summary, these problem characteristics lead to a set of hard ConFL instances in terms of the difficulty for the DLS heuristic of Bardossy and Raghavan (2010).

In order to evaluate how the shape of the uncertainty region affects the DLS heuristic performance, in particular the computational times, we generated three sets of instances with various uncertainty regions: circular, square and rectangular. In Set 1 the location of demand nodes is represented by an uncertainty disk whose radius is randomly generated. To evaluate the effect of uncertainty on the heuristic and the solutions obtained, we considered various ranges for the disk radius and generated them between 0 and 2, 5, 10 or 20 (i.e., radii were randomly generated in the ranges: [0,2], [0,5], [0,10] and [0,20]) on each subset of instances. Consequently, for each demand node, i, we defined a center location (i_1, i_2) and an uncertainty radius r_i. Then the minimum assignment cost for demand node i from facility j, a_{ij}, is the maximum of (1) the Euclidean distance between the two nodes minus the radius and (2) zero (since the minimum assignment cost is bounded by zero when they are collocated). On the other hand, the maximum de-

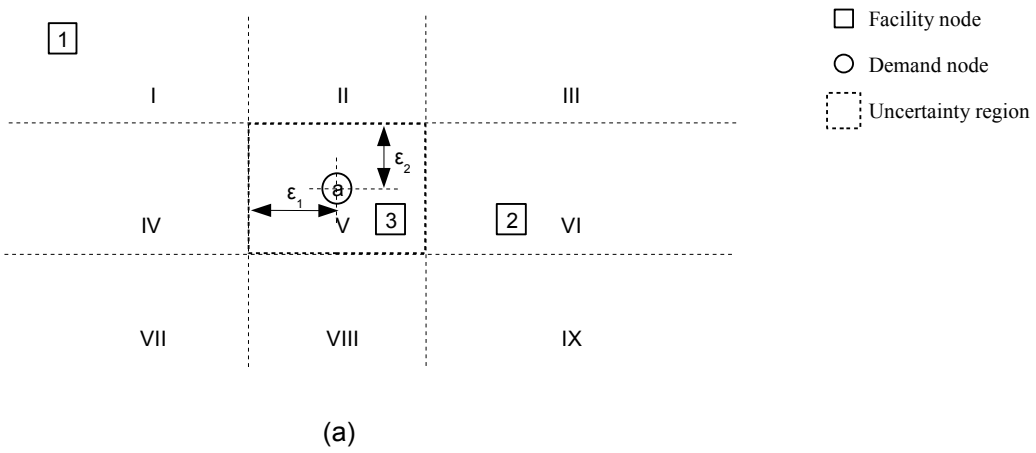

Figure 5.6: Example of rectangular uncertainty region

viation in the assignment cost, d_{ij}, is the minimum of (1) the Euclidean distance between the two nodes plus the radius, and (2) the disk diameter (i.e., twice the radius). In mathematical notation, $a_{ij} = \max\{\sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2} - r_i, 0\}$ and $d_{ij} = \min\{\sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2} + r_i, 2r_i\}$.

In Set 2 the location of demand nodes is represented by a square uncertainty region. Also to evaluate the effect of uncertainty, we created instances with various ϵ_{i1} and ϵ_{i2} deviations for each coordinate axis, which were uniformly generated between 0 and 2, 5, 10 and 20. For each demand node we defined a center location (i_1, i_2) and an uncertainty deviation ϵ_i (i.e., for square uncertainty region $\epsilon_{i1} = \epsilon_{i2}$.) Similarly to Set 1, we then calculate the minimum assignment cost and maximum deviation for each pair of demand nodes and facility node. To define the minimum assignment cost, we had to determine the location of each facility node with respect to the nine regions around the demand node location as shown in Figure 5.6.

Table 5.6 provides the coordinates of the closest possible location of demand

Region	Closest
I	$(a_1 - \epsilon_1, a_2 + \epsilon_2)$
II	$(j_1, a_2 + \epsilon_2)$
III	$(a_1 + \epsilon_1, a_2 + \epsilon_2)$
IV	$(a_1 - \epsilon_1, j_2)$
V	(j_1, j_2)
VI	$(a_1 + \epsilon_1, j_2)$
VII	$(a_1 - \epsilon_1, a_2 - \epsilon_2)$
VIII	$(j_1, a_2 - \epsilon_2)$
IX	$(a_1 + \epsilon_1, a_2 - \epsilon_2)$

Table 5.6: Facility node location and closest possible demand node location

Region	Farthest
A	$(a_1 + \epsilon_1, a_2 - \epsilon_2)$
B	$(a_1 - \epsilon_1, a_2 - \epsilon_2)$
C	$(a_1 + \epsilon_1, a_2 + \epsilon_2)$
D	$(a_1 + \epsilon_1, a_2 - \epsilon_2)$

Table 5.7: Facility node location and farthest possible demand node location

node a, when a facility node j falls within a given region with respect to demand node a as highlight in Figure 5.6.

Then, to determine the maximum deviation in assignment cost, d_{ij}, we calculate the maximum possible assignment cost and take the difference from the minimum. To calculate the maximum assignment cost we find the farthest point within the uncertainty region from the facility node. Figure 5.7 shows the four quadrants used to determine the farthest location for the demand node and Table 5.7 the coordinates of the farthest demand location given the location of the facility node.

In Set 3 the location of demand nodes is represented by a rectangular uncertainty region with different variations on each coordinate axis, (i.e., $\epsilon_{i1} \neq \epsilon_{i2}$.)

For the example shown in Figure 5.6 and 5.7 with three facility nodes, the

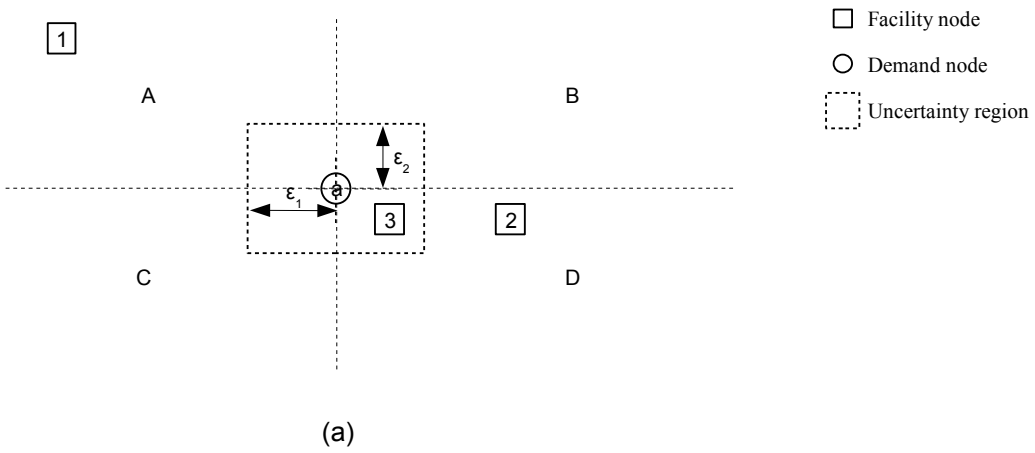

Figure 5.7: Example of rectangular uncertainty region

Facility Node	a_{ij}	d_{ij}
1	$\sqrt{(1_1-(a_1-\epsilon_1))^2+(1_2-(a_2+\epsilon_2))^2}$	$\sqrt{(1_1-(a_1+\epsilon_1))^2+(1_2-(a_2-\epsilon_2))^2}-a_{1a}$
2	$\sqrt{(2_1-(a_1+\epsilon_1))^2}$	$\sqrt{(2_1-(a_1-\epsilon_1))^2+(2_2-(a_2+\epsilon_2))^2}-a_{2a}$
3	0	$\sqrt{(2_1-(a_1-\epsilon_1))^2+(2_2-(a_2+\epsilon_2))^2}$

Table 5.8: Assignment costs for the example shown in Figure 5.6 and 5.7

assignment costs are as shown in Table 5.8.

We used various levels of conservatism or Γ to assess its effect on the solution and the performance of the heuristic. As we noted earlier, one advantage of this heuristic is that Γ does not enter into the solution procedure until the last step. Once the nominal problems have been solved, considering various levels of Γ requires simple computations that do not involve resolving the nominal problems. We considered Γ values between 0 and 50 in steps of 10. We generated 10 instances for each combination of problem characteristics.

Note that the final location of a demand node may fall outside the 100 x 100

	Γ					
Radius	0	10	20	30	40	50
2	2.04%	1.98%	1.94%	1.92%	1.90%	1.90%
5	2.47%	2.29%	2.18%	2.10%	2.04%	2.03%
10	2.73%	2.35%	2.02%	1.88%	1.79%	1.77%
20	3.34%	2.61%	1.75%	1.63%	1.48%	1.43%

Table 5.9: Average Gaps for Set 1 ($M = 3$)

grid. While the demand node center location will always fall within the 100 x 100 grid by construction, the uncertainty area may extend outside the predefined grid, and consequently, the final location of a demand node may fall outside the grid.

5.5.2 Results on the RConFL Problem

Here we report on our computational experiments on the three sets of problems. Each entry in the tables represents the average over ten instances. The percentage gaps represent the percentage difference of the RCOH solution over the lower bound obtained by DA. That is, $[\%] = \frac{UB-LB}{LB}$. For each instance we must solve a significant number of nominal problems (up to $|F||D| = (50)(50) = 2500$). DA provides a lower bound for each nominal problem and the minimum of the nominal problems' lower bounds is the lower bound for an instance.

5.5.2.1 Results for Set 1 - Disk Uncertainty Area

The DLS heuristic yields high-quality solutions rapidly for the problems in Set 1 ($M = 3$ and $M = 7$). We report average lower bound gaps for different Γ levels and uncertainty radii in Table 5.9 and 5.10. The average gaps are under

	Γ					
Radius	0	10	20	30	40	50
2	2.83%	2.77%	2.72%	2.69%	2.67%	2.67%
5	1.79%	1.70%	1.63%	1.59%	1.57%	1.56%
10	2.64%	2.39%	2.12%	2.00%	1.94%	1.92%
20	3.23%	2.58%	2.23%	2.03%	1.91%	1.88%

Table 5.10: Average Gaps for Set 1 ($M = 7$)

Radius	Nominal Problems	DA	Local Search	Post-Processing	Total
2	52	19.599	0.529	0.009	20.137
5	59	22.253	0.709	0.009	22.971
10	77	29.700	0.855	0.013	30.568
20	141	55.884	1.551	0.022	57.457

Table 5.11: Average Computational Times in Seconds for Set 1 ($M = 3$)

3.34% and the highest gap for all instances is below 8.76%. Furthermore, we observe that gaps and average gaps decrease for higher values of Γ (as hinted by Theorem 5.2.1.) On the other hand, for higher levels of uncertainty the average gaps remain stable. Actually, contrary to our earlier conjecture that gaps would increase for higher uncertainty levels and lower M factors, the highest gap is observed for an instance with uncertainty radius equal to 2 and M factor equal to 7. There are no significant differences across M factors for the average gaps.

Radius	Nominal Problems	DA	Local Search	Post-Processing	Total
2	52	28.606	0.184	0.009	28.799
5	57	31.425	0.171	0.009	31.606
10	74	39.558	0.274	0.013	39.845
20	128	69.241	0.423	0.022	69.686

Table 5.12: Average Computational Times in Seconds for Set 1 ($M = 7$)

Tables 5.11 and 5.12 show the average number of nominal problems solved for each instance, dual-ascent (DA)'s computational time, local search's computational time and post-processing time. The post-processing time accounts for the time required to calculate the cost of the solution in problem (5.14) and select the best solution out of all the solutions obtained for the nominal problems. The post-processing time observed is infinitesimally different (less than 0.001 second) for each Γ level. (In theory the post-processing time should be equal for all Γ values.) The average number of nominal problems increases with uncertainty because as the radius of the uncertainty disk increases, facility nodes are more likely to fall within uncertainty disks. Hence, the average heuristic time increases as the number of nominal problems increases. Regardless, the average time is below 1 minute when the radii vary up to 10 or less and slightly above 1 minute when the radii go up to 20. Moreover, the maximum time required to solve one instance was below 2 minutes. Dual-ascent takes the bulk of the heuristic time.

Another advantage of this method is that we can easily observe how the solution changes as the decision-maker increases her conservatism level (i.e., Γ value) and calculate the cost difference between solutions. Figures 5.8 and 5.9 show the heuristic solution for each instance for different values of Γ. Each plot corresponds to ten instances with the same uncertainty level. The x-axis represents the Γ value and the y-axis the solution that yielded the lowest total cost for the instance. In other words, each line represents the solution path for an instance as Γ varies. The plots in these figures show limited changes as a result of changes of Γ. There are more changes for the lower M factor and for higher levels of uncertainty. In the first

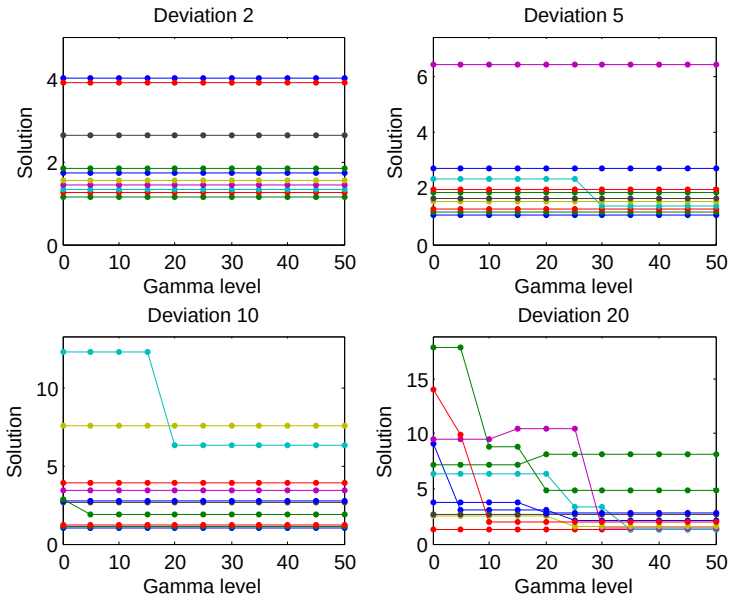

Figure 5.8: Solutions for different Γ values for Set 1 ($M = 3$)

plot for deviation up to 2, for both Figures 5.8 and 5.9, we observe that the solution does not change as the Γ level increases for any of the instances.

Theorem 5.4.1 implies that under certain circumstances one might be able to solve the robust problem by solving the best-case problem. We test the effectiveness of this strategy even when those special conditions do not hold. We use the DLS heuristic to obtain a solution, $(x, y, z)_{BC}$, to the best-case problem; we then calculate its cost for the different Γ levels in problem (5.14); and lastly, we calculate the percentage cost difference over the robust solution yielded by RCOH. Tables 5.13 and 5.13 show the average percentage difference in cost between the best-case scenario solution by DLS and the robust solution by RCOH. For every level of uncertainty the average loss difference is below 1.66%. Out of 480 instances, the highest loss was 4.37%. In addition, we can note that even though a $\Gamma = 0$ corresponds to the best-case problem in some instances the DLS solution is slightly worse than the

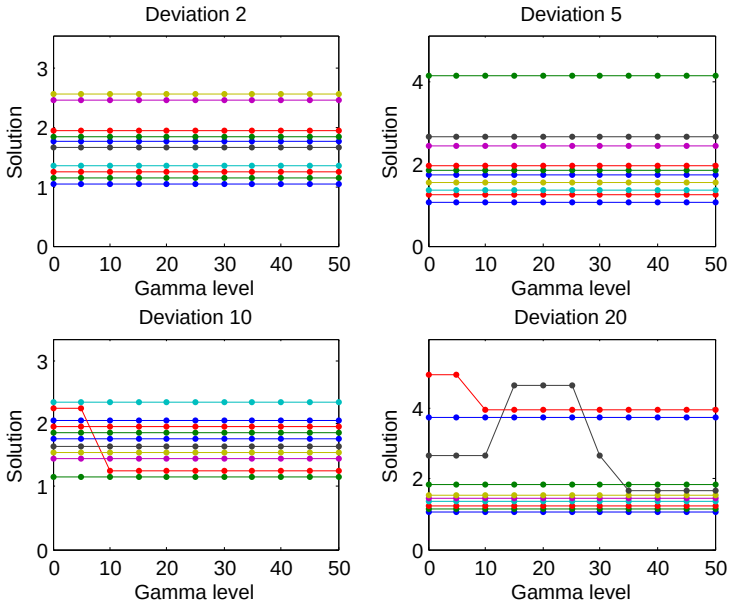

Figure 5.9: Solutions for different Γ values for Set 1 ($M = 7$)

	Γ					
Radius	0	10	20	30	40	50
2	0.44%	0.43%	0.42%	0.41%	0.41%	0.41%
5	0.03%	0.02%	0.02%	0.02%	0.02%	0.02%
10	0.47%	0.45%	0.47%	0.44%	0.43%	0.40%
20	0.92%	0.88%	1.30%	1.61%	1.66%	1.64%

Table 5.13: Average best-case solution loss for Set 1 ($M = 3$)

solution yielded by the RCOH for the same problem (see first column in Table 5.13 and 5.14.) The best-case problem, jointly with many other nominal problems, is solved as part of the RCOH; consequently, the RCOH solution is at least as good or better than the solution obtained by DLS. These results show that as the uncertainty level increases there is a higher value in implementing RCOH to solve the best-case problem, as RCOH might yield a better solution at a slightly higher computational cost.

	Γ					
Radius	0	10	20	30	40	50
2	0.33%	0.32%	0.32%	0.32%	0.31%	0.31%
5	0.32%	0.29%	0.28%	0.28%	0.27%	0.27%
10	0.30%	0.32%	0.33%	0.30%	0.29%	0.29%
20	0.28%	0.29%	0.32%	0.32%	0.32%	0.33%

Table 5.14: Average best-case solution loss for Set 1 ($M = 7$)

	Γ					
Deviation	0	10	20	30	40	50
2	1.79%	1.71%	1.63%	1.59%	1.58%	1.57%
5	1.98%	1.73%	1.56%	1.51%	1.48%	1.46%
10	2.45%	1.85%	1.69%	1.71%	1.61%	1.58%
20	3.40%	2.03%	1.32%	1.27%	1.18%	1.17%

Table 5.15: Average Gaps for Set 2 ($M = 3$)

5.5.2.2 Results for Set 2 - Square Uncertainty Area

The RCOH also provides high-quality solutions for Set 2. The average gaps are under 3.40% and the maximum gap is below 6.55%. The average gaps also decrease as Γ increases and there is no significant difference in the average gaps for higher levels of uncertainty. For low values of Γ, the average gap is slightly higher

	Γ					
Deviation	0	10	20	30	40	50
2	1.95%	1.90%	1.87%	1.87%	1.84%	1.84%
5	2.25%	2.10%	1.94%	1.88%	1.81%	1.80%
10	2.27%	1.82%	1.73%	1.64%	1.57%	1.56%
20	3.38%	2.44%	1.92%	1.71%	1.74%	1.72%

Table 5.16: Average Gaps for Set 2 ($M = 7$)

Deviation	Nominal Problems	DA	Local Search	Processing	Total
2	2485	958.792	30.116	0.437	989.345
5	2484	960.863	28.764	0.436	990.063
10	2478	968.769	29.190	0.531	998.491
20	2480	1005.351	27.770	0.539	1033.660

Table 5.17: Average Computational Times in Seconds for Set 2 ($M = 3$)

Deviation	Nominal Problems	DA	Local Search	Processing	Total
2	2474	1328.071	9.275	0.437	1337.783
5	2473	1288.235	8.263	0.436	1296.934
10	2477	1287.162	9.185	0.531	1296.879
20	2473	1343.423	8.657	0.539	1352.619

Table 5.18: Average Computational Times in Seconds for Set 2 ($M = 7$)

for higher uncertainty levels. However, this tendency fades away as Γ increases. There are no significant differences in the average gaps for $M = 3$ and $M = 7$.

In regards to computational times, the instances in Set 2 take a significantly higher amount of time than the instances in Set 1. The main reason is that to solve the robust problem, we have to solve a significant number of nominal problems. The average number of nominal problems for each instance in Set 2 is above 2400. Recall that the maximum number of possible nominal problems in these instances is 2500. Moreover, the number of nominal problems is independent of the uncertainty level. Consequently, the total computational times are much higher and they average around 22 minutes. Table 5.17 and 5.18 show the average number of nominal problems and computational times for instances in Set 2 with M = 3 and M = 7, respectively. Even though the computational times are higher, they are substantially shorter than the ones required by other methods. Based on the results reported in

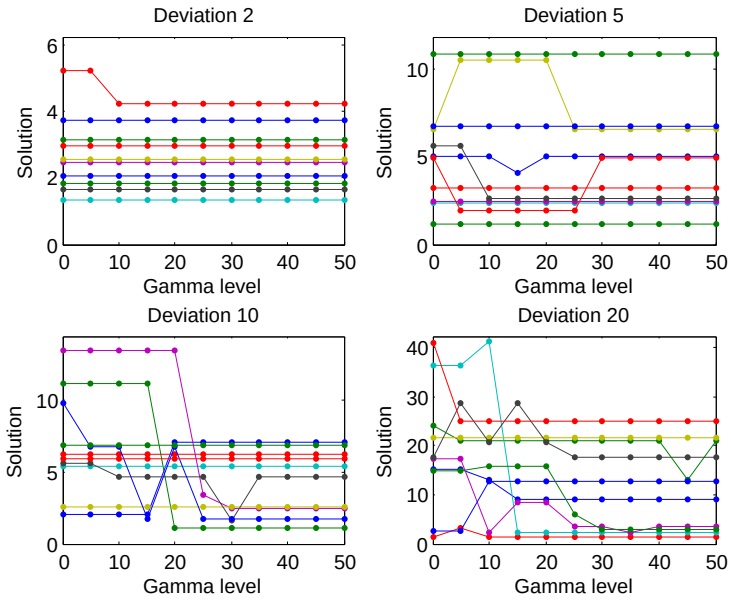

Figure 5.10: Solutions for different Γ values for Set 2 ($M = 3$)

Ljubić (2007), one could expect the average computational time for each nominal problem to be around one hour that would translate in an average of 2400 hours for each instance! That means that RCOH using DLS takes only 0.015% of the needed time by the exact method (i.e., Algorithm A of Bertsimas and Sim (2003)).

Figures 5.10 and 5.11 show how the heuristic solution changes for each instance for each uncertainty level. We observe that there are more changes in the solution for $M = 3$ than for $M = 7$ for all level of uncertainty. One explanation might be that assignment costs have a higher impact on the overall total cost of the solution when M is lower; and consequently, the Γ value that determines the number of assignment cost at their highest in the solution have also a higher effect on determining the best solution. Similar to what we observed in Set 1, changes in Γ have higher impact on the solution for higher levels of uncertainty.

Lastly, Tables 5.19 and 5.20 show the average difference between the cost of the

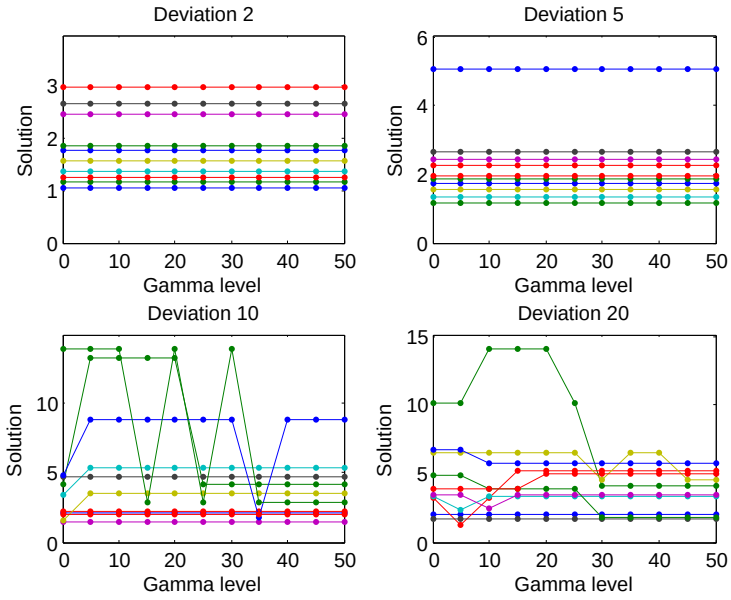

Figure 5.11: Solutions for different Γ values for Set 2 ($M = 7$)

	Γ					
Radius	0	10	20	30	40	50
2	0.54%	0.53%	0.53%	0.54%	0.53%	0.52%
5	0.35%	0.32%	0.42%	0.42%	0.40%	0.39%
10	0.37%	0.75%	0.71%	0.70%	0.75%	0.73%
20	1.17%	1.75%	2.74%	2.83%	2.82%	2.74%

Table 5.19: Average best-case solution loss for Set 2 ($M = 3$)

	Γ					
Radius	0	10	20	30	40	50
2	0.20%	0.20%	0.22%	0.22%	0.22%	0.22%
5	0.58%	0.59%	0.69%	0.67%	0.68%	0.68%
10	0.40%	0.49%	0.46%	0.45%	0.42%	0.42%
20	1.35%	1.23%	1.42%	1.39%	1.37%	1.34%

Table 5.20: Average best-case solution loss for Set 2 ($M = 7$)

	Γ					
Deviation	0	10	20	30	40	50
2	1.72%	1.68%	1.64%	1.61%	1.60%	1.59%
5	1.83%	1.63%	1.62%	1.49%	1.37%	1.42%
10	2.27%	1.87%	1.74%	1.62%	1.64%	1.65%
20	2.62%	1.66%	1.29%	1.30%	1.48%	1.77%

Table 5.21: Average Gaps for Set 3 ($M = 3$)

	Γ					
Deviation	0	10	20	30	40	50
2	2.22%	2.17%	2.13%	2.10%	2.05%	2.03%
5	1.91%	1.75%	1.69%	1.63%	1.61%	1.62%
10	2.48%	2.18%	1.99%	2.08%	2.07%	1.91%
20	1.39%	1.26%	1.47%	1.56%	1.73%	1.78%

Table 5.22: Average Gaps for Set 3 ($M = 7$)

best-case scenario solution and the robust solution. The robust solution is slightly better than the best-case scenario solution for low levels of uncertainty. However, for the highest level of uncertainty in our computational results, the best-case scenario solution is up to 5.39 % more costly than the robust solution. The average loss reaches up to 2.89%.

5.5.2.3 Results for Set 3 - Rectangular Uncertainty Area

Tables 5.21 and 5.22 show the average gaps for Set 3. RCOH also provides high-quality solutions for Set 3. The average gaps are under 2.62%. Similarly, the highest gap in Set 3 is below 5.54%. The average gaps show an overall tendency to decline as Γ increases with one exception. For high uncertainty levels, they

Deviation	Nominal Problems	DA	Local Search	Processing	Total
2	2494	962.571	28.969	1.638	993.177
5	2494	962.457	29.430	1.569	993.456
10	2489	971.110	28.427	1.039	1000.577
20	2491	1011.519	26.699	1.021	1039.238

Table 5.23: Computational Times for Set 3 ($M = 3$)

Deviation	Nominal Problems	DA	Local Search	Processing	Total
2	2483	1278.859	7.727	1.638	1288.223
5	2489	1272.502	9.599	1.569	1283.670
10	2483	1345.038	9.830	1.039	1355.907
20	2485	1357.619	7.875	1.021	1366.514

Table 5.24: Computational Times for Set 3 ($M = 7$)

first slightly decrease and later increase when Γ increases. There are no significant differences in average gaps across M factors.

The average number of nominal problems is slightly higher in Set 3; and consequently, the computational times are proportionally higher. The average total time per instance is approximately 22 minutes but less than 25 minutes for any instance.

Figures 5.12 and 5.13 show how the heuristic solution for each instance changes for different values of Γ. Again there is slightly more activity for $M = 3$ than for $M = 7$ and for higher values of uncertainty.

Tables 5.25 and 5.26 show the average cost increase of the best-case scenario solution over the robust solution. The average loss reaches up to 3.21% and the highest loss for one instance goes up to 5.08%. Once again in the first column we observe that even for the best-case problem RCOH yields better solutions than DLS. In one instance that difference reaches 3.34%. Interestingly, average losses are

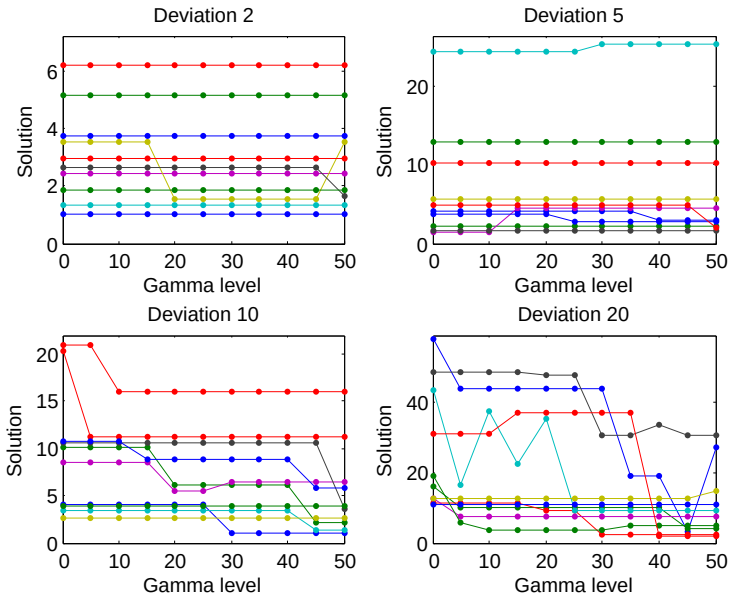

Figure 5.12: Solutions for different Γ values for Set 3 ($M = 3$)

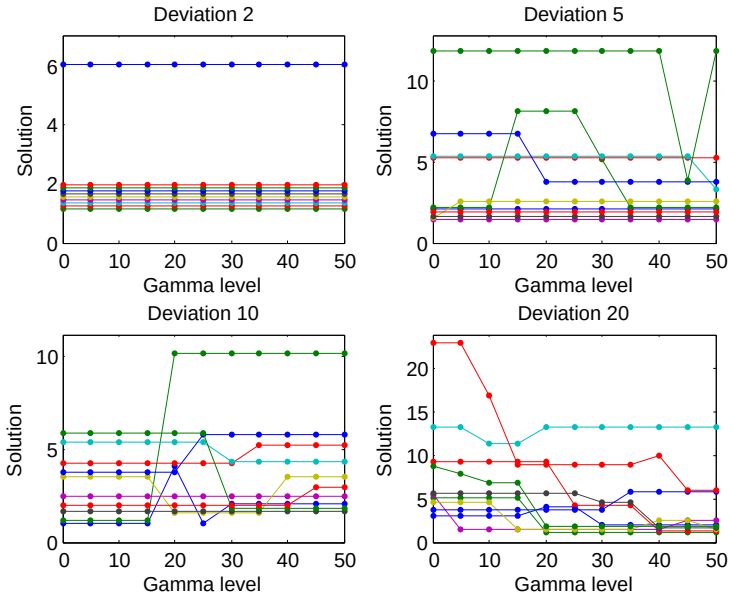

Figure 5.13: Solutions for different Γ values for Set 3 ($M = 7$)

	Γ					
Radius	0	10	20	30	40	50
2	0.53%	0.52%	0.53%	0.54%	0.53%	0.52%
5	0.54%	0.52%	0.56%	0.62%	0.65%	0.65%
10	0.64%	0.60%	0.75%	0.97%	1.02%	0.94%
20	1.00%	1.83%	2.63%	2.97%	3.21%	3.17%

Table 5.25: Average best-case solution loss for Set 3 ($M = 3$)

	Γ					
Radius	0	10	20	30	40	50
2	0.27%	0.28%	0.26%	0.25%	0.25%	0.24%
5	0.31%	0.33%	0.31%	0.38%	0.40%	0.40%
10	0.88%	0.82%	0.84%	0.95%	1.09%	1.25%
20	0.81%	0.99%	1.16%	1.52%	1.56%	1.58%

Table 5.26: Average best-case solution loss for Set 3 ($M = 7$)

slightly higher for Set 2 than Set 3.

5.6 Conclusions

In this chapter we address the uncertainty in the ConFL problem by robust optimization. Specifically, we extend Bertsimas and Sim's approach to situations where one uses a lower bounding mechanism jointly with a heuristic to obtain a solution to the nominal problems (as opposed to solving the nominal problems to optimality). We tested this strategy on the ConFL problem using the DLS heuristic and found high-quality solutions for various levels of uncertainty and budget of conservatism, which highlights the potential and applicability of the RCOH to other robust combinatorial optimization problems.

In our computational experiments we observe that the computational time of RCOH is directly proportional to the the number of deviation terms. There is one nominal problem for each deviation term. Consequently, we leave as future research to explore and devise strategies to cut down the number of nominal problems necessary to solve to find a high-quality heuristic solution for the robust problem. Notice that if the set X in problem (5.1) is a convex set then $\mathcal{Z}(\theta)$ is a convex function of θ. Suppose we consider the linear relaxation in problem (5.7) and denote the objective as $\mathcal{Z}^{LP}(\theta)$, since this function is piecewise linear and convex to determine the minimum value of $\mathcal{Z}^{LP}(\theta)$ for $\theta \geq 0$ we need only to consider the nominal problems. Our dual ascent lower bound for a nominal problem is $\mathcal{Z}^{LB}(\theta) \leq \mathcal{Z}^{LP}(\theta)$. We can use this fact together with the convexity of $\mathcal{Z}^{LB}(\theta)$ to prioritize which nominal problem to solve first or at all. Specifically, if the lower bound in a region is higher than the best solution (upper bound) in another region, we can disregard any θ value beyond the value that produced the lower bound. This could significantly reduce the total computational time of the RCOH.

The following observations relate to the Bertsimas and Sim's robust optimization approach itself and concern a deeper understanding of the method and the implications derived by its solutions for the ConFL problem. The following are some interesting questions that might be worth exploring in the future:

1. What does the difference in the total solution cost for different Γ values indicate?

2. What does that difference in cost indicate if the solution changes for adjacent

Γ values?

3. What does that difference in cost mean if the solution does not change? Recall that the Γ value is subjective and determined by the decision-maker. In real world applications the decision-maker does not have control over how many of the deviations will in fact take their highest value.

4. How should the decision-maker make a choice for the Γ value for the problem at hand? Should the uncertainty in the demand nodes be considered to determine Γ? Should there be a relationship between the number of demand nodes with the highest uncertainty level and the Γ considered?

Another note worth discussing regarding robust optimization for the ConFL problem is that in order to calculate the assignment costs and the maximum deviation in assignment costs, we consider one facility node and demand node at the time. However, one might think of situations where the actual assignment of the demand nodes does not take place until the location of the demand node is revealed. Consequently, the assignment may change depending on the final location of the demand node as in a two-stage problem. In such situations, the earlier calculated maximum deviations in costs might never occur because the demand node may get routed to a different facility node before reaching the maximum deviation. In other words, the maximum assignment cost depends on the final set of opened facilities.

These questions and observations may reveal some weaknesses in the method, specially for the ConFL problem. However, we think that they are actually opportunities worth understanding and exploring in the future and we look forward to

devising stronger robust optimization formulations that would better address the ConFL problem.

Bibliography

Balakrishnan, A., T. L. Magnanti, R. T. Wong. 1989. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research* **37**(5) 716–740.

Bardossy, M.G., S. Raghavan. 2010. Dual-based local search for the connected facility location and related problems. *INFORMS Journal on Computing* **22**(4) 584–602.

Beale, E.M.L. 1955. On Minimizing A Convex Function Subject to Linear Inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)* 173–184.

Berman, O. 1978. Dynamic positioning of mobile servers on networks. Ph.D. thesis, Massachusetts Institute of Technology.

Berman, O., A.R. Odoni. 1982. Locating Mobile Servers on a Network With Markovian Properties. *Networks* **12**(1) 73–86.

Bertsimas, D., M. Sim. 2003. Robust discrete optimization and network flows. *Mathematical Programming* **98**(1) 49–71.

Birge, J.R., F. Louveaux. 1997. *Introduction to stochastic programming*. Springer Berlin.

Christofides, N. 1976. Worst case analysis of a new heuristic for the traveling salesman problem. Tech. Rep. 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

Chu, C. H., G. Premkumar, H. Chou. 2000. Digital data networks design using genetic algorithms. *European Journal of Operational Research* **127**(1) 140–158.

Cooper, L. 1978. Bounds on the Weber problem solution under conditions of uncertainty. *Journal of Regional Science* **18**(1) 87–92.

Dantzig, G.B. 1955. Linear programming under uncertainty. *Management Science* **1**(3) 197–206.

Economist. 2009a. And access for all. *The Economist* **392**(8636) 60–01.

Economist. 2009b. Who pays for the pipes? *The Economist* **393**(8661) 6–8.

Eisenbrand, F., F. Grandoni, T. Rothvoß, G. Schäfer. 2008. Approximating connected facility location problems via random facility sampling and core detouring. *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms* 1174–1183.

Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research* **26**(6) 992–1009.

Frenzel, L. 2010. Broadband for everyone. Tech. Rep. 7, Electronic Design.

Gollowitzer, S., I. Ljubic. 2011. MIP models for connected facility location: A theoretical and computational study. *Computers & Operations Research* **38**(2) 435–449.

Gupta, A., J. Kleinberg, A. Kumar, R. Rastogi, B. Yener. 2001. Provisioning a virtual private network: a network design problem for multicommodity flow. *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing* 389–398.

Gupta, A., A. Kumar, T. Roughgarden. 2003. Simpler and better approximation algorithms for network design. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing* 365–372.

Gupta, A., R. Ravi, A. Sinha. 2004. An edge in time saves nine: LP rounding approximation algorithms for stochastic network design. *Annual Symposium on Foundations of Computer Science*, vol. 45. IEEE Computer Society Press, 218–227.

Havet, F., M. Wennink. 2004. The push tree problem. *Networks* **44**(4) 281–291.

Juel, H. 1981. Bounds in the generalized Weber problem under locational uncertainty. *Operations Research* **29**(6) 1219–1227.

Jung, H., M. K. Hasan, K. Y. Chwa. 2008. Improved primal-dual approximation algorithm for the connected facility location problem. *Lecture Notes in Computer Science* **5165** 265–277.

Kall, P., S.W. Wallace. 1994. *Stochastic programming*. John Wiley & Sons Inc.

Karger, D. R., M. Minkoff. 2000. Building Steiner trees with incomplete global knowledge. *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science* 613–623.

Khuller, S., A. Zhu. 2002. The general Steiner tree-star problem. *Information Processing Letters* **84**(4) 215–220.

Kleywegt, A.J., A. Shapiro, T. Homem-de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12**(2) 479–502.

Krick, C., H. Räcke, M. Westermann. 2003. Approximation algorithms for data management in networks. *Theory of Computing Systems* **36**(5) 497–519.

Laporte, G., FV Louveaux, L. Van Hamme. 1994. Exact solution to a location problem with stochastic demands. *Transportation Science* **28**(2) 95–103.

Lee, Y., S. Y. Chiu, J. Ryan. 1996. A branch and cut algorithm for a Steiner tree-star problem. *INFORMS Journal on Computing* **8**(3) 194–201.

Lee, Y., L. Lu, Y. Qiu, F. Glover. 1993. Strong formulations and cutting planes for designing digital data service networks. *Telecommunication Systems* **2**(1) 261–274.

Ljubić, I. 2007. A hybrid VNS for connected facility location. *Lecture Notes in Computer Science* **4771** 157–169.

Ljubić, I. 2009. Private communication.

Louveaux, F.V., D. Peeters. 1992. A dual-based procedure for stochastic facility location. *Operations Research* **40**(3) 564.

Mirchandani, P. B., A. R. Odoni. 1979. Locations of medians on stochastic networks. *Transportation Science* **13** 85–97.

Mirchandani, P.B. 1975. Analysis of stochastic networks in emergency service systems. Ph.D. thesis, Massachusetts Institute of Technology.

Nuggehalli, P., V. Srinivasan, C. F. Chiasserini. 2003. Energy-efficient caching strategies in ad hoc wireless networks. *Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking & Computing* 25–34.

Raghavan, S. 1995. Formulations and algorithms for network design problems with connectivity requirements. Ph.D. thesis, Massachusetts Institute of Technology.

Ravi, R., A. Sinha. 2006. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming* **108**(1) 97–114.

Riis, M., K.A. Andersen. 2003. Capacitated network design with uncertain demand. *INFORMS Journal on Computing* **14**(3) 247–260.

Ruszczynski, A., A. Shapiro. 2003. *Stochastic Programming. Handbook in Operations Research and Management Science*, vol. 40. Elsevier.

Salman, FS, J. Cheriyan, R. Ravi, S. Subramanian. 1997. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *Proceedings of the 8th Annual*

ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 619–628.

Sen, S., R.D. Doverspike, S. Cosares. 1994. Network planning with random demand. *Telecommunication Systems* **3**(1) 11–30.

Shapiro, A., A. Philpott. 2007. A tutorial on stochastic programming.

Shmoys, D. B., É. Tardos, K. Aardal. 1997. Approximation algorithms for facility location problems. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* 265–274.

Snyder, L.V. 2006. Facility location under uncertainty: A review. *IIE Transactions* **38**(7) 547–564.

Swamy, C. 2004. Approximation algorithms for clustering problems. Ph.D. thesis, Cornell University.

Swamy, C., A. Kumar. 2004. Primal–dual algorithms for connected facility location problems. *Algorithmica* **40**(4) 245–269.

Tomazic, A., I. Ljubić. 2008. A GRASP algorithm for the connected facility location problem. *2008 International Symposium on Applications and the Internet* 257–260.

Van Slyke, R.M., R. Wets. 1969. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* **17**(4) 638–663.

Verweij, Bram, Shabbir Ahmed, Anton J. Kleywegt, George Nemhauser, Alexander Shapiro. 2003. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications* **24** 289–333.

Weaver, JR, RL Church. 1983. Computational procedure for location problems on stochastic networks. *Transportation Science* **17**(2) 168–180.

Wong, R. T. 1984. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming* **28**(3) 271–287.

Xu, J., S. Chiu, F. Glover. 1996a. Using tabu search to solve Steiner tree-star problem in telecommunications. *Telecommunications Systems* 117–125.

Xu, J., S.Y. Chiu, F. Glover. 1996b. Probabilistic tabu search for telecommunications network design. *Journal of Combinatorial Optimization* **1** 69–94.