REFORMULATION AND CUTTING-PLANE APPROACHES FOR SOLVING TWO-STAGE OPTIMIZATION AND NETWORK INTERDICTION PROBLEMS

By

SIQIAN SHEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2011

To my parents and grandparents; to my grandma, Zong Wu, who left us too early

ACKNOWLEDGMENTS

To my family, your love and support have been tremendously helpful and priceless. Specially to my dearest parents, Jiankang Shen and Ying Cao, who taught me confidence, kindness, and dedication to the things I love. I owe you everything.

To Cole Smith, please accept my deepest gratitude for your guidance and friendship in the past four years. Working with you is the greatest privilege of my life, and I cannot describe how much impact you have had on me, academically and personally. Your endless passion for research and your gracious care for students have inspired me to work harder to hopefully become someone like you. Another thankful note also goes to your lovely family, Cindy and the kids, for the generous care and joy they gave me.

To my dissertation committee, I would like to thank Jean-Philippe Richard for your enlightening ideas, Joseph Geunes for your intelligent humor, and Pramod Khargonekar for your remarkable vision. I thank you all for suggestions on my dissertation. I also sincerely acknowledge Shabbir Ahmed, Edwin Romeijn, and many other colleagues for their support and influence on my career path.

Moreover, my enduring appreciation goes to the Industrial and Systems Engineering program at the University of Florida, and my fantastic graduate student colleagues. Specially to Ruiwei, Chunhua, Qipeng, Zhili, Chin Hon, Vera, Zeki, Semra, Chase, John P., Kelly, Shantih, Jin-Ho, Ehsan, Soheil, and Clay & Renee, your friendship made the graduate study one of the best experiences of my life. To Jing Li, Shuang Lü, Liujiang Song, and Chuang Xuan, my good friends, the time I spent with you contributed to so many enjoyable memories of Gainesville.

The last but not the least, happy 100^{th} birthday to Tsinghua University, where I completed my bachelor degree. Specially to Lei Zhao, a great teacher and friend in Tsinghua, I owe you my happiness and the opportunity to pursue an academic career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

REFORMULATION AND CUTTING-PLANE APPROACHES FOR SOLVING TWO-STAGE OPTIMIZATION AND NETWORK INTERDICTION PROBLEMS

By

Siqian Shen

August 2011

Chair: J. Cole Smith
Major: Industrial and Systems Engineering

This dissertation investigates models and algorithms for solving a class of two-stage optimization problems arising in a variety of practical problems, including network interdiction applications. Traditional decomposition methods for solving large-scale mixed-integer programs (MIP), such as Benders decomposition, cannot be utilized for solving the problems that we consider due to the presence of discrete variables in the second-stage problems. In the problems we study, there exist a finite set of first-stage feasible solutions and general mixed-integer variables in the second stage. We employ a decomposition strategy and apply the Reformulation-Linearization Technique (RLT) to obtain the convex hull of the formulation in terms of the second-stage variables. For some of the problems, we then describe modified Benders cuts for attaining optimality. For the others, we derive valid inequalities based on the subproblem reformulations or sub-optimal polynomial-time dynamic programming (DP) solutions. Our goal is to develop novel formulation and solution methodologies for solving several two-stage (stochastic) MIPs within practical time limits.

We first consider a class of two-stage stochastic optimization problems arising in the protection of vital arcs in a critical path network. A project is completed after a series of dependent tasks are all finished. We analyze a problem in which task finishing times are uncertain, but can be insured a priori to mitigate potential delays. A decision maker must trade off costs incurred in insuring arcs with expected penalties associated

with late project completion times, where lateness penalties are assumed to be lower semi-continuous nondecreasing functions of completion time. We provide decomposition strategies to solve this problem with respect to either convex or nonconvex penalty functions. In particular, for the nonconvex penalty case, we employ RLT to make the problem amenable to solution via Benders decomposition. We also consider a chance-constrained version of this problem, in which the probability of completing a project on time is sufficiently large. We demonstrate the computational efficacy of our approach by testing a set of randomly generated instances, using the Sample Average Approximation method to guide our scenario generation.

Then, we examine variants of the Critical Node Problem (CNP) on specially-structured graphs, which aim to identify a subset of nodes whose removal will maximally disconnect the graph. These problems lie in the intersection of network interdiction and graph theory research. The two different connectivity metrics that we consider regard the number of maximal connected components (which we attempt to maximize) and the largest component size (which we attempt to minimize). We develop polynomial-time DP algorithms for solving these problems with respect to each network-connectivity metric on tree structures and on series-parallel graphs. We also extend our discussion by expanding the core model to account for node deletion costs and weighted nodes, and also by solving the problems on generalizations of tree structures.

Furthermore, we analyze the CNP on general undirected graphs, and consider a third connectivity metric: the minimum cost required to reconnect the graph after the nodes are deleted (which we attempt to maximize). We show that these problem variants are $\mathcal{NP}$-hard in general, and formulate each problem as a mixed-integer program. Valid inequalities are studied for the first two connectivity objectives by examining intermediate DP solutions to k-hole subgraphs (a type of generalizations of tree structures). We randomly generate a set of test instances, on which we demonstrate the computational efficacy of our approaches.

Finally, we consider an optimization problem that integrates network design and broadcast domination decisions. Given an undirected graph, a feasible broadcast domination is a set of nonnegative integer powers f_i assigned to each node i, such that for any node j in the graph, there exists some node k having a positive f_k-value whose shortest distance to node j is no more than f_k. The cost of a broadcast domination solution is the sum of all node power values. The network design problem constructs edges that decrease the minimum broadcast domination cost on the graph. The overall problem we consider minimizes the sum of edge construction costs and broadcast domination costs. We show that this problem is $\mathcal{NP}$-hard in the strong sense, even on unweighted graphs. A decomposition strategy will iteratively add valid inequalities based on optimal broadcast domination solutions corresponding to the first-stage network design solutions. We demonstrate that our decomposition approach is computationally superior to the solution of a monolithic MIP formulation.

CHAPTER 1
INTRODUCTION

Decision-making processes that penetrate many aspects of our daily life often involve a set of quantitative decisions that need to be made sequentially. In the context of mathematical programming, we formulate optimization models that cast those decisions as variables, and use them to describe a set of objectives and constraints that represent facets of a specific real-world problem. For instance, consider a *facility location* problem, in which a decision maker has a certain budget to open several facilities on a subset of given locations. The facilities are operated to satisfy daily customer demands from a variety of nearby locations. The decision maker seeks a set of locations to open facilities and also a feasible assignment that will match every facility to its corresponding customer. (S)he needs to examine the tradeoff between the cost of opening a facility and long-term operational costs, while the solution needs to satisfy the budget constraint as well as the customers' daily demands.

Note that the decision of opening a facility can be characterized as a binary variable at each potential facility location. Also, consider demands that are uncertain every day over a finite time horizon. We can formulate the problem as a mathematical program that involves stochastic parameters and discrete variables. A major concern of solving this class of problems is the amount of memory and computational effort required, which normally grow exponentially in terms of the problem size (e.g., the number of discrete variables or the number of scenarios to characterize uncertainty). Indeed, the presence of discrete variables and stochastic data both significantly complicate the process of optimizing mathematical programming models. As a result, the challenge of overcoming the difficulties posed by these problems has received much attention in the literature. Benders decomposition (Benders, 1962) and its many variants are still among the most effective techniques for solving large-scale linear programming (LP) and MIP models having special decomposable structures. We refer to Saharidis

& Ierapetritou (2010) and Saharidis et al. (2010a) for recent improvements of Benders methods as demonstrated in science and engineering applications. Essentially, the Benders approach identifies a split between "here-and-now" and "recourse" variables, whereby the resolution of complicating here-and-now variables in the first stage permits the solution of second-stage subproblems involving recourse variables. At each iteration, the algorithm passes a fixed first-stage decision into the second stage, and employs optimality conditions to derive "feasibility" or "optimality" cuts that will direct the evolution of the first-stage decisions at the next iteration. The algorithm terminates when all optimality conditions are satisfied by the current first-stage solution, and no further cuts are generated. In the previous facility location problem, we decide whether or not to open a facility on the set of locations to minimize the opening cost, and an estimate for the overall operational cost, regardless of whether the solution is feasible to meet customer demands. Then in the second stage, we fix facility locations, and verify whether this solution could meet all demands, and determine the actual corresponding operational cost under each demand scenario. We then generate Benders feasibility cut(s) (if the demands are not satisfied) or optimality cut(s) (if the actual operational cost exceeds the estimate) into the first-stage problem, and iteratively repeat the procedure. We claim optimality when no cuts are generated from the subproblems.

The advantages of the above decomposition approach are the following. In the first-stage problem, we solve a relaxed mater problem that has much fewer variables and constraints, and thus is more computationally tractable. Moreover, stochastic parameters are only in present in the subproblems, each of which corresponds to a realized scenario. By fixing the values of the first-stage variables, we solve the subproblems in parallel, and independently generate cuts into the master problem from each subproblem. Compared with directly solving a monolithic mathematical program, this strategy solves a series of smaller problems, allows large-scale parallel computing, and provides an attractive alternative for solving stochastic mixed-integer

programs (MIP) within reasonable computational limits. In the case where there are a finite number of feasible solutions in the first stage, this cutting-plane algorithm will guarantee convergence to an optimal solution in finite number of steps. Furthermore, the two-stage MIP model and its solution methodologies share many attributes with a *network interdiction* model. The latter is usually formulated as a *Stackelberg game*, in which one player "moves" first, followed by the other player's recourse reaction. The first player seeks a set of moves that would minimize the maximum gain, or would maximize the minimum loss of the second-player. Optimality conditions are also employed to combine the two players' problems into a single optimization program, and to compute an optimal solution.

However, assuming the presence of discrete variables in the second stage, we can neither directly apply traditional Benders-like decomposition approaches to the two-stage MIPs, nor combine the two player's formulations in a network-interdiction setting. This is due to the absence of dual information that we can glean from the nonconvex subproblem formulations. In particular, this situation is often encountered when modeling two-stage stochastic programs, whose many variants have been used to cast a variety of real-world problems. The topic of two-stage stochastic modeling and its solution methodologies are briefly reviewed as follows.

Optimization problems formulated as stochastic programs with recourse commonly arise in areas of military, health care industry, financial business, emergency response (Haneveld & Van der Vlerk, 1999; Schultz, 2003). Birge & Louveaux (1997) and Kall & Wallace (1994) discuss general decomposition strategies for solving stochastic programs, in which a set of decisions are made *a priori*, when uncertain information has not yet been completely available. Then, given first-stage decisions and the revealed random second-stage data following some distribution, values of subsequent recourse are decided based on a relationship between the first- and second-stage variables.

Two-stage stochastic linear programs were first introduced by Dantzig (1955), and then investigated by Walkup & Wets (1967). Here we study general two-stage stochastic mixed-integer programming models (SMIP). The easiest case deals with SMIPs having only continuous variables in the second stage, which yields convex subproblem formulations. An effective decomposition approach, such as Benders decomposition (Benders, 1962), will categorize the decisions as "here-and-now" and "wait-and-see," such that the former type of decisions are independent of uncertain data realizations and need to be made in the first stage. The latter type of decisions are then considered as recourse, whose values are decided in the second-stage subproblems. A traditional Benders cutting-plane algorithm has been shown to finitely converge if the subproblems are linear, or are nonlinear convex programs that satisfy certain feasibility and boundedness assumptions (Flippo & Rinnooy Kan, 1993; Geoffrion, 1972; Van Slyke & Wets, 1969; Wollmer, 1980).

Next, we describe methods that solve two-stage SMIPs having integer recourse variables in the subproblems, which are nonconvex in general. First consider the case in which the relaxed master problem has *pure* binary variables. Laporte & Louveaux (1993) provide a cutting-plane algorithm for general two-stage MIPs in which the first-stage problem has only binary variables, and the second-stage problem may have a combination of continuous and discrete variables. The so-called Laporte-Louveaux (LL) inequalities that their technique employs have been used to solve stochastic location and routing problems within reasonable computational limits (Laporte et al., 1992, 1994). Carøe & Tind (1997) provide an approach for solving two-stage stochastic programming problems having integer second-stage variables by employing general duality theory, in which the master problem has nonlinear constraints. This method establishes important stochastic integer programming foundations, but is not practically effective unless the reformulated nonlinear master program can be efficiently solved. Sen & Higle (2005) introduce a disjunctive-decompostion (D^2) algorithm, using

disjunctive programming to convexify subproblems and derive valid cutting planes. However, computational implementations and results have been shown to be slow for problems that have large-scale stochastic data and discrete decision variables (Ntaimo & Sen, 2008). Sherali & Fraticelli (2002) generate modified Benders cuts by using the Reformulation-Linearization Technique (RLT) (Sherali & Adams, 1990, 1994), yielding integral subproblems given first-stage solutions. Rather than explicitly implement RLT, they sequentially generate a suitable partial description of the convex hull representation as needed in the process of deriving valid Benders cuts, which are reusable and globally valid. Sen & Sherali (2006) propose a decomposition-related branch-and-cut (BAC) algorithm. They enhance the computational efficacy by putting the D^2 algorithm into a BAC framework, which allows the SMIPs to be solved by dividing a large problem into smaller subproblems, which can be solved in parallel. Yuan & Sen (2009) study computational enhancements to speed up cut generation process associated with D^2-BAC.

However, when mixed-integer variables are present in both stages, by allowing fractional first-stage solution values, one may generate infinite number of cuts from the subproblems by considering infinite choices of first-stage solutions. Ahmed et al. (2004) explore a class of two-stage SMIPs having general first-stage variables and pure integer second-stage recourse. They develop a finite branch-and-bound approach by implementing a partitioning process in a transformed space. Sherali & Zhu (2007) also use a branch-and-bound approach, and a modified Benders decomposition method in Sherali & Fraticelli (2002) to define a lower bound value after each branching procedure. They show that the algorithm will converge to a global optimal solution, but may take infinite iterations.

In this dissertation, we analyze several practical problems having general discrete subproblems, and try to derive dual information of the subproblems by examining the convex hull of the original problem. We then use the dual formulation to generate

modified Benders cuts or valid inequalities for solving the corresponding two-stage MIPs, or to derive a single-level model for solving network interdiction models. We demonstrate the computational efficacy of our approaches on a set of randomly generated instances. We introduce the details of our models and algorithms in each chapter as follows, in which combinatorial optimization methods, graph theory, and RLT are employed to obtain the convex hull representations.

Chapter 2 considers a class of two-stage stochastic optimization problems arising in the protection of vital arcs in a critical path network. A project is completed after a series of dependent tasks are all finished. We analyze a problem in which task finishing times are uncertain, but can be insured a priori to mitigate potential delays. A decision maker must trade off costs incurred in insuring arcs with expected penalties associated with late project completion times, where lateness penalties are assumed to be lower semi-continuous nondecreasing functions of completion time. We provide decomposition strategies to solve this problem with respect to either convex or nonconvex penalty functions. In particular, for the nonconvex penalty case, we employ RLT to make the problem amenable to solution via Benders decomposition. We also consider a chance-constrained version of this problem, in which the probability of completing a project on time is sufficiently large. The Sample Average Approximation method is employed to guide our scenario generation.

Chapter 3 examines variants of the Critical Node Problem on specially-structured graphs, which aim to identify a subset of nodes whose removal will maximally disconnect the graph. These problems lie in the intersection of network interdiction and graph theory, and are relevant to several practical optimization problems. The two connectivity metrics that we consider regard the number of maximal connected components (which we attempt to maximize) and the largest component size (which we attempt to minimize). We develop polynomial-time dynamic programming algorithms for solving these problems with respect to each network-connectivity metric on tree structures

and on series-parallel graphs. We also extend our discussion by considering node deletion costs, node weights, and generalizations of tree structures. The solution methodology we developed in this chapter prepares for the extension of maximizing the disconnectivity of *general* graphs in Chapter 4. Aside from the two problem variants in Chapter 3, we consider a third one based on maximizing a metric as the minimum cost required to reconnect the graph after the nodes are deleted. We formulate each problem as a MIP, and then study valid inequalities for the first two connectivity objectives by examining intermediate dynamic programming solutions to k-hole subgraphs.

In Chapter 5, we consider an optimization problem that integrates network design and broadcast domination decisions. Given an undirected graph, a feasible broadcast domination is a set of nonnegative integer powers f_i assigned to each node i, such that for any node j in the graph, there exists some node k having a positive f_k-value whose shortest distance to node j is no more than f_k. The cost of a broadcast domination solution is the sum of all node power values. The network design problem constructs edges that decrease the minimum broadcast domination cost on the graph. The overall problem we consider minimizes the sum of edge construction costs and broadcast domination costs. We show that this problem is $\mathcal{NP}$-hard in the strong sense, even on unweighted graphs. We then propose a decomposition strategy, which iteratively adds valid inequalities based on optimal broadcast domination solutions corresponding to the first-stage network design solutions. The computational results show that our approach is computationally far superior to the solution of a single large-scale MIP formulation.

Lastly, Chapter 6 concludes the paper and states the future research tasks corresponding to the problems in each chapter. The idea of seeking convex-hull representations of general large-scale discrete optimization problems is shown to be helpful for generating effective cutting planes and for eliminate the presence of bilevel models in network-interdiction settings. Thus, we want to derive standard procedures for such MIP models having specially-structured constraints. It is also interesting to

test the efficacy of the approaches that we develop in this dissertation by tuning the computational parameters. Our methodology can be extended or applied to solve many other two-stage or bilevel optimization problems arising in different practical problems.

CHAPTER 2
CRITICAL PATH INSURANCE PROBLEM

2.1 Problem Description and Literature Survey

In this chapter, we analyze a class of Critical Path Management (CPM) problems associated with the scheduling of complex projects that consist of several dependent activities. These problems are often modeled by a directed network whose arcs represent dependent tasks, and whose arc weights consist of associated task durations. Graph vertices represent milestones, and enforce precedence constraints by requiring that all tasks associated with arcs entering a node must be completed before any tasks associated with arcs exiting the node may begin. Also, there exists a node representing the start of the project, and a node representing its termination. If the duration for each activity is not known with certainty, the Program Evaluation and Review Technique (PERT) can be used to estimate the probability that a project will be completed by a given deadline. We refer to Kelley (1961, 1963); Moehring (1984) for basic CPM/PERT literature. Given the CPM network associated with a project, the total project completion time is given by the length of the network's longest path, referred to as its critical path. Critical path lengths are carefully monitored in most business applications since financial penalties often accrue as a monotonic function of project completion time.

Some recent models and algorithms for resource-constrained project scheduling problems include works by Brucker et al. (1999), Chtourou & Haouari (2008), Ozdamar & Ulusoy (1995), and Patterson (1984). In particular, Elmaghraby et al. (2000), Hagstrom (1990), and Iida (2000) compute lower and upper bounds for the distribution function of PERT project durations in order to schedule tasks. From a multi-stage dynamic perspective, Bowman & Muckstadt (1993), Hindelang & Muth (1979), and Kulkarni & Adlakha (1986) employ dynamic programming and finite-stage, continuous-time Markov chains to solve these problems.

Several methodologies are used to handle parameter uncertainty in CPM, including heuristic- and Monte-Carlo-simulation-based techniques (Burt & Garman (1971); Bowman (1995); Mitchell & Klastorin (2007)). Critical Chain Project Management (CCPM) (Goldratt, 1997) is a method based on Theory of Constraints that emphasizes keeping resources flexible at start times and quickly switching resources between tasks as necessary. Herroelen & Leus (2001) highlight the merits and disadvantages of CCPM based on literature and experimental studies on commercial CCPM software. By contrast, in this chapter we consider situations in which resources cannot be dynamically switched among tasks during the execution of the project. These situations occur for instance when resources must be allocated well in advance (e.g., in capital budgeting plans), or when communication/transportation logistics make resource transfer impossible.

Herroelen & Leus (2005) review fundamental approaches for project management under uncertainty, including reactive scheduling, stochastic project scheduling, fuzzy project scheduling, and robust (proactive) scheduling. The *time/cost trade-off* (or *activity crashing*) problems are related to the study in this chapter. Scholl (2001) and Gutjahr et al. (2000) formulate an expectation-based version of the stochastic linear time/cost trade-off problem as a scenario-based stochastic program. These problems have also recently been analyzed using chance-constrained formulations (Laslo, 2003; Golenko-Ginzburg & Gonik, 1999; Golenko-Ginzburg et al., 2000). We refer to Demeulemeester & Herroelen (2002) for a comprehensive discussion of contemporary stochastic project scheduling problems.

We consider the cases in which project managers can invest resources to either shorten task durations or prevent spikes in task durations due to uncertainty. For instance, task durations may be substantially delayed due to labor availability in construction, or shipping delays in logistics applications. In these settings, insuring tasks against delays may be accomplished by prehiring additional labor to keep in

reserve, or by paying additional money to guarantee timely delivery of goods. A decision maker would seek an optimal portfolio of resource investment, trading-off costs of insuring arcs with expected penalties associated with project deadline violation. We refer to this problem as the Stochastic Task Insurance Problem (STIP).

STIP is related to interdiction problems, which form another class of two-stage problems that often take place over networks. A typical network interdiction problem involves a network operator that wishes to minimize (without loss of generality) some objective over the network, such as a shortest path or minimum cost flow. The interdiction problem is set up as a Stackelberg game wherein an interdicting agent acts first to modify the characteristics of certain arcs (e.g., reducing or eliminating capacity) in order to maximize the operator's minimum cost. Some stochastic interdiction problems of note include those by Cormican et al. (1998) and Janjarassuk & Linderoth (2008).

In this chapter, we formulate STIP as a two-stage stochastic programming model amenable to Benders decomposition (Benders, 1962). Schultz (2003) provides a review of stochastic integer programming models and algorithms, and Chen et al. (2008) for multistage stochastic optimization models. Our primary contributions are summarized as follows. One, we propose a Benders decomposition framework for the solution of STIP in which lateness is penalized by a nondecreasing lower semi-continuous penalty function of project completion time. These functions are of significant practical importance, because they allow a decision-maker to capture discontinuities and/or smooth nonconvex portions of the penalty function. Discontinuities may arise because of fixed-charge fees due to lateness, and smooth nonconvexities may arise when penalty functions are concave functions that asymptotically approach a maximum value (e.g., due to project cancelation). Two, we demonstrate how to quickly recover coefficients for Benders cuts from the solution of a critical path problem, rather than requiring the direct solution of a more complex reformulation. Three, we cast STIP in the context of a

chance-constrained optimization problem, and demonstrate how our algorithms can be used to solve such instances.

We then conduct a computational study in Section 2.4 that both illustrates the efficiency of our procedures, and demonstrates the inherent difficulty of solving STIP. In particular, we examine two intuitive methods that managers may be tempted to use to determine which tasks should be insured. In Section 2.4.1.2, we consider the use of a simple rule in which tasks are insured in order of a nondecreasing ratio of task insurance cost to average time saved due to insurance. In Section 2.4.1.3, we solve a series of deterministic critical path insurance problems, in which the scenario outcomes are known a priori, to obtain an empirical estimate of how likely a task is to be insured in each scenario. (We refer to this as the *persistency* of a task (Bertsimas et al., 2006).) Hence, another intuitive statement may suppose that tasks having high persistence are more likely to be insured at optimality in STIP. However, we demonstrate that neither of these approaches are capable of reliably picking optimal tasks to insure in STIP. The vital implication is that STIP is too difficult to be solved by examining cost-to-benefit ratios as supposed in Section 2.4.1.2, and even too difficult to be solved by insuring those tasks that appear most often in the solution to a series of deterministic task insurance problems in which uncertain information is revealed before the insurance decisions take place.

Finally, note that the actual completion time of the project will be a function of the task insurance decisions and the outcome of the random task durations, and also of the penalty function that a manager may place on late completion times. We consider in Section 2.4.1.4 the case in which a decision-maker enforces a continuous two segment piecewise-linear penalty function on the late completion times. We observe that with convex penalties the average critical path length tends to be shorter than with concave penalties, explained by the fact that in the former case, severe penalty is imposed on

very late completion times. This observation is of particular interest to managers that are risk-averse and wish to mitigate worst-case scenarios.

The remainder of the chapter is organized as follows. Section 2.2 introduces the problem and provide a subgradient-based cutting-plane algorithm with respect to the convex penalty case. In Section 2.3, we first examine the case of piecewise-linear lower semi-continuous penalty functions, for which we employ the Reformulation-Linearization Technique (RLT) of Sherali and Adams (Sherali & Adams, 1990, 1994) to remodel the subproblem so that it is amenable to solution via Benders decomposition. We then extend our algorithm to handle general nondecreasing lower-semicontinuous penalty functions, and also solve a variation of STIP that we cast as a chance-constrained formulation. We employ the Sample Average Approximation (SAA) method (e.g., Shapiro & Homem-de-Mello (2000)) in Section 2.4 to solve instances having stochastic task durations.

2.2 Problem Statement and Convex Penalty Case

Let $G(\mathcal{N}, \mathcal{A})$ denote a directed graph representing the tasks to be completed in a complex project, with node set $\mathcal{N} = \{0, \ldots, n\}$, and arc set $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$, where $\mathcal{A}$ is topologically ordered such that $(i, j) \in \mathcal{A}$ only if $i < j$. Node 0 serves as the project starting point and node n as its completion point. We define $FS(i) = \{j : (i, j) \in \mathcal{A}\}$ as the set of nodes adjacent from node i, and $RS(i) = \{j : (j, i) \in \mathcal{A}\}$ as the set of nodes adjacent to node i, $\forall i \in \mathcal{N}$.

For each arc $(i, j) \in \mathcal{A}$, we represent the cost of insuring (i, j) by c_{ij}, and define binary decision variable x_{ij}, where $x_{ij} = 1$ if we insure arc (i, j) and $x_{ij} = 0$ otherwise. The set of possible finite scenarios is given by Ω, where for each scenario $s \in \Omega$, each arc $(i, j) \in \mathcal{A}$ is associated with an uninsured task duration d_{ij}^s and insured task duration g_{ij}^s, where $0 \leq g_{ij}^s \leq d_{ij}^s$. We use binary variables y_{ij}^s to denote whether arc (i, j) belongs to a critical path in scenario $s \in \Omega$, where $y_{ij}^s = 1$ if arc (i, j) is part of one identified critical path.

For our initial model, suppose that we have nondecreasing convex functions $\Theta^s : \mathbb{R} \mapsto \mathbb{R}$ that penalize the critical path length in each scenario. We define e^s as the probability of realizing a scenario $s \in \Omega$, and present the optimization problem as:

$$\begin{aligned} \textbf{CP: } \min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum_{s\in\Omega} e^s\Theta^s(\psi^s(x)) \\ \text{s.t.} \quad & x_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}, \end{aligned}$$

where $\Theta^s(\psi^s(x))$ is the penalty function in scenario $s \in \Omega$, and $\psi^s(x)$ is the critical path length with respect to x, given by

$$\begin{aligned} \textbf{CPM}^s(x)\textbf{: } \psi^s(x) = \max \quad & \sum_{(i,j)\in\mathcal{A}} (d^s_{ij} - (d^s_{ij} - g^s_{ij})x_{ij})y^s_{ij} && \text{(2–1a)} \\ \text{s.t.} \quad & \sum_{j\in FS(0)} y^s_{0j} = 1 && \text{(2–1b)} \\ & \sum_{j\in FS(i)} y^s_{ij} - \sum_{k\in RS(i)} y^s_{ki} = 0 \quad \forall i \in \mathcal{N} - \{0,n\} && \text{(2–1c)} \\ & 0 \le y^s_{ij} \le 1 \quad \forall (i,j) \in \mathcal{A}, && \text{(2–1d)} \end{aligned}$$

where Objective 2–1a maximizes the sum of task durations, Constraints 2–1b and 2–1c enforce flow-balance constraints for critical path contiguity, and Constraint 2–1d bounds the y-variables between 0 and 1.

Since $\psi^s(x)$ is convex in x and Θ^s is a nondecreasing convex function of $\psi^s(x)$, we have that $f^s(x) = \Theta^s(\psi^s(x))$ is convex in x. A standard Benders decomposition approach would create the following (relaxed) master problem:

$$\begin{aligned} \textbf{CP-MP: } \min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum_{s\in\Omega} e^s\eta^s \\ \text{s.t.} \quad & \eta^s \ge f^s(x^t) + [\partial f^s(x^t)]^T(x - x^t) \quad \forall t \in \mathbb{T}^s,\ s \in \Omega && \text{(2–2)} \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}, \end{aligned}$$

where $\mathbb{T}^s$ is the collection of optimality cuts under scenario s, x^t is the candidate solution in the t^{th} iteration, and $\partial f^s(x^t)$ is a subgradient of $f^s(x)$ at x^t.

To compute the corresponding parameters in Inequalities 2–2, rather than requiring the direct dual solution of a subproblem reformulation, note that $f^s(x^t)$ is directly obtainable from solving a critical path problem CPM$^s(x^t)$ and setting $f^s(x^t) = \Theta^s(\psi^s(x^t))$. Let $\Theta^{s,t}$ be the slope of $\Theta^s(\cdot)$ at $\psi^s(x^t)$, and given x^t, let $y^{s,t}$ be the optimal solution of CPM$^s(x^t)$. We have that $\{-(d^s_{ij} - g^s_{ij})y^{s,t}_{ij}\}$ is the $(i,j)^{th}$ element of a subgradient of $\psi^s(x)$ at x^t, and so

$$\partial f^s_{ij}(x^t) = -\Theta^{s,t}(d^s_{ij} - g^s_{ij})y^{s,t}_{ij} \quad \forall (i,j) \in \mathcal{A}. \tag{2–3}$$

2.3 Decomposition Algorithm for the Nonconvex Penalty Case

We analyze a class of problems involving nonconvex penalty functions, where the cuts generated using the subgradient method in Section 2.2 are no longer valid with respect to the nonconvex subproblems. For piecewise-linear lower semi-continuous penalty functions, we employ RLT to convexify the second-stage programs, and generate Benders cuts associated with the modified subproblems in Section 2.3.2. We then extend these techniques to handle more general nonconvex functions in Section 2.3.3, and employ our algorithms to solve a chance-constrained formulation of our problem in Section 2.3.4.

2.3.1 Reformulation-Linearization Technique

The RLT of Sherali and Adams (Sherali & Adams, 1990, 1994) enhances the solvability of zero-one programming problems by deriving a hierarchy of linear programming relaxations, which ultimately provide a convex hull representation of the problem. We briefly describe the RLT details most relevant to this chapter.

Consider a mixed-integer zero-one linear program whose feasible region X is defined over binary variables $x = (x_1, \dots, x_n)$ and nonnegative continuous variables $y = (y_{n+1}, \dots, y_m)$. Given any value of $d \in \{0, \dots, n\}$, we construct the relaxation of X at

level d by considering the following product terms:

$$F_d(J_1, J_2) = \prod_{j \in J_1} x_j \prod_{j \in J_2} (1 - x_j) \quad \forall J_1, J_2 \subseteq \{1, \dots, n\},\ J_1 \cap J_2 = \emptyset,\ |J_1 \cup J_2| = d, \qquad (2\text{–}4)$$

and by executing the following steps:

1. Reformulation. Multiply each constraint by each product $F_d(J_1, J_2)$ of order d to create a (nonlinear) polynomial mixed-integer zero-one program.

2. Linearization. Linearize the program by applying the identity $x_j^2 = x_j$ for all $j \in \{1, \dots, n\}$, substituting $w_J = \prod_{j \in J} x_j$, $\forall J \subseteq \{1, \dots, n\}$, $v_{kJ} = y_k \prod_{j \in J} x_j$, $\forall J \subseteq \{1, \dots, n\}$, $\forall k$, and relaxing integrality.

Thus, we linearize the reformulated problem into a higher-dimensional problem whose projection of X_d onto the original variables (x, y) is given by

$$X_{P_d} = \{(x, y) : (x, y, w, v) \in X_d\} \quad \forall d = 1, \dots, n.$$

Sherali & Adams (1990, 1994) show that $X_0 \equiv X_{P_0} \supseteq X_{P_1} \supseteq \cdots \supseteq X_{P_n} = conv(X)$, where $X_{P_0} \equiv X_0$ denotes the ordinary linear programming relaxation, and $conv(X)$ is the convex hull of X. We will adopt this strategy in the rest of this chapter to address STIPs having nonlinear cost functions.

2.3.2 Piecewise-linear Lower Semi-continuous Penalty Function

We begin by considering STIPs in which the completion time penalty is given by a piecewise-linear lower semi-continuous function, and develop a modified Benders decomposition method for the problem. We decompose STIP as a two-stage stochastic mixed-integer program that has binary x-variables and mixed integer recourse variables in the first and second stages, respectively, and independent subproblems for each scenario $s \in \Omega$.

Letting η_s denote the optimal objective value of the subproblem based on scenario s, we formulate the master problem as:

$$\textbf{MP-PW: min} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} + \sum_{s\in\Omega} e^s\eta_s \tag{2–5a}$$

$$\text{s.t.} \quad \eta_s + m^{s,l}x \geq n^{s,l} \quad \forall l \in L^s,\ s \in \Omega \tag{2–5b}$$

$$x_{ij} \in \{0,1\}\ \forall (i,j) \in \mathcal{A}, \quad \eta_s \geq \underline{\eta}^s\ \forall s \in \Omega, \tag{2–5c}$$

where L^s designates a set of Benders cuts derived for the s^{th} subproblem, having coefficients $m^{s,l}$ and $n^{s,l}$, as we describe below, and $\underline{\eta}^s$ denotes the smallest penalty that could be incurred in scenario s for any choice of x. (Note that the penalty associated with the critical path length when all task durations are set to the g^s-values is a valid lower bound for $\underline{\eta}^s$.)

Note that graph $G(\mathcal{N},\mathcal{A})$ is acyclic in a project management instance. Hence, given a first-stage solution $\hat{x}$, we can quickly compute the corresponding critical path in $O(|\mathcal{A}|)$ steps (e.g., Ahuja et al. (1993)), and let $\hat{y}^s_{ij} = 1$ if $(i,j) \in \mathcal{A}$ belongs to our identified critical path, and $\hat{y}^s_{ij} = 0$ otherwise. The main advantage of our second-stage subproblem formulation is that we simply separate a critical path problem from the whole subproblem model. We then recover all subproblem optimal dual values by using the critical path length, and further generate modified Benders cuts without actually solving the entire subproblem formulation. Hence, we only solve a critical path problem regarding each subproblem and save a large amount of computational effect.

For scenario $s \in \Omega$ we define Θ^s over intervals $1,\ldots,K^s$, where interval k is defined over $(\tau^s_k,\ \tau^s_{k+1}]$, for $k = 1,\ldots,K^s$, and where $\tau^s_{K^s+1}$ is a maximum possible critical path length in scenario s. For convenience, we assume that τ^s_1 is just smaller than $\underline{\eta}^s$ to allow all intervals to be open on the left side. For each scenario $s \in \Omega$, the piecewise-linear penalty function has slope m^s_k and intercept b^s_k over interval $k,\ \forall k = 1,\ldots,K^s$, where $m^s_k \geq 0,\ \forall k$, and $b^s_1 \geq 0$ to ensure that each function is nonnegative and nondecreasing.

We define binary variable z_k^s such that $z_k^s = 1$ if the project completion time belongs to interval k, and $z_k^s = 0$ otherwise, for $k = 1, \ldots, K^s$ and $s \in \Omega$. Define u_i^s to represent the length of a longest path from node 0 to node i, $\forall i = 0, \ldots, n$, $s \in \Omega$, given the values of $\hat{x}_{ij}$ from the first-stage master problem (where $u_0^s = 0$). Also, define variables f_k^s as the objective function contribution due to interval k in scenario s, i.e., $f_k^s = (m_k^s u_n^s + b_k^s) z_k^s$. Letting Q denote a large constant number, we formulate the subproblem as

$$\textbf{SP-LS}^s(\hat{x})\textbf{: min} \quad \sum_{k=1}^{K^s} f_k^s \tag{2–6a}$$

$$\text{s.t.} \quad f_k^s \geq m_k^s u_n^s + b_k^s - Q(1 - z_k^s) \quad \forall k = 1, \ldots, K^s \tag{2–6b}$$

$$u_n^s \geq \tau_k^s z_k^s \quad \forall k = 1, \ldots, K^s \tag{2–6c}$$

$$-u_n^s \geq -\tau_{k+1}^s - Q(1 - z_k^s) \quad \forall k = 1, \ldots, K^s \tag{2–6d}$$

$$u_j^s - u_i^s \geq d_{ij}^s - (d_{ij}^s - g_{ij}^s)\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{2–6e}$$

$$f_k^s \geq 0 \quad \forall k = 1, \ldots, K^s \tag{2–6f}$$

$$\sum_{k=1}^{K^s} z_k^s = 1 \tag{2–6g}$$

$$z_k^s \in \{0, 1\} \quad \forall k = 1, \ldots, K^s. \tag{2–6h}$$

We employ RLT to reformulate STIP into a form that is amenable to solution by Benders decomposition. (Sherali (2001) discusses related development on applying RLT to piecewise-linear lower semi-continuous functions.) By noticing that $\sum_{k=1}^{K^s} z_k^s = 1$, $\forall s \in \Omega$, we generate the convex hull of solutions to the overall STIP (containing all scenarios) in which z_k^s is binary valued $\forall k,\ s$, based on the Special Structures RLT of Sherali et al. (1998), and the risk-based decomposition strategy of Sherali & Smith (2009). We also refer to Sherali & Fraticelli (2002) for related work. The key is to multiply Ineq. 2–6b through 2–6g, as well as the bounds $x_{ij} \leq 1$ and $x_{ij} \geq 0$, by $z_k^s,\ \forall k = 1, \ldots, K^s$, and $x_{ij} \geq 0$ by $(1 - \sum_{k=1}^{K^s} z_k^s)$ (yielding an equality constraint), before decomposition. After doing so, there exists an optimal solution in which all z-variables are binary valued, given binary x-values, and the problem can thus be solved by Benders decomposition.

Observe that, by definition, $f_k^s z_k^s = f_k^s$, and $f_k^s z_l^s = 0,\ \forall l \neq k$. We then linearize by substituting $(z_k^s)^2 = z_k^s,\ \forall k,\ z_l^s z_k^s = 0,\ \forall l \neq k$; and by defining $v_{ik}^s \equiv u_i^s z_k^s,\ \forall i = 0, \dots, n,\ k = 1, \dots, K^s$, and $w_{ijk}^s \equiv x_{ij} z_k^s,\ \forall (i,j) \in \mathcal{A},\ k = 1, \dots, K^s$. We apply Special Structures RLT (Sherali et al., 1998) to SP-LS$^s(\hat{x})$, and decompose the formulation. The resulting subproblem is given by

$$\min \quad \sum_{k=1}^{K^s} f_k^s \tag{2–7a}$$

$$\text{s.t.} \quad -m_l^s v_{nk}^s + (Q - b_l^s) z_k^s \geq 0 \quad \forall l,\ k = 1, \dots, K^s,\ l \neq k \tag{2–7b}$$

$$-m_k^s v_{nk}^s - b_k^s z_k^s + f_k^s \geq 0 \quad \forall k = 1, \dots, K^s \tag{2–7c}$$

$$v_{nk}^s \geq 0 \quad \forall k = 1, \dots, K^s \tag{2–7d}$$

$$v_{nk}^s - \tau_k^s z_k^s \geq 0 \quad \forall k = 1, \dots, K^s \tag{2–7e}$$

$$-v_{nk}^s + (\tau_{l+1}^s + Q) z_k^s \geq 0 \quad \forall l,\ k = 1, \dots, K^s,\ l \neq k \tag{2–7f}$$

$$-v_{nk}^s + \tau_{k+1}^s z_k^s \geq 0 \quad \forall k = 1, \dots, K^s \tag{2–7g}$$

$$v_{jk}^s - v_{ik}^s - d_{ij}^s z_k^s + (d_{ij}^s - g_{ij}^s) w_{ijk}^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \dots, K^s \tag{2–7h}$$

$$-\sum_{k=1}^{K^s} w_{ijk}^s = -\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{2–7i}$$

$$-w_{ijk}^s + z_k^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \dots, K^s \tag{2–7j}$$

$$w_{ijk}^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \dots, K^s \tag{2–7k}$$

$$\sum_{k=1}^{K^s} z_k^s = 1 \tag{2–7l}$$

$$f_k^s \geq 0 \quad \forall k = 1, \dots, K^s \tag{2–7m}$$

$$z_k^s \geq 0 \quad \forall k = 1, \dots, K^s. \tag{2–7n}$$

First, we observe that Ineq. 2–7b is implied by 2–7n if $m_l^s = 0$, and otherwise ($m_l^s > 0$), 2–7b is implied by 2–7g, since $(Q - b_l^s)/m_l^s > \tau_{k+1}^s$. Next, note that Constraints 2–7d, 2–7f, and 2–7m are implied by Constraints 2–7e, 2–7g, and 2–7c, respectively. We thus obtain

SP-LS$^s(\hat{x})$-RLT:

$$
\begin{array}{lll}
\min & \sum_{k=1}^{K^s} f_k^s & \text{Duals} \\
\text{s.t.} & -m_k^s v_{nk}^s - b_k^s z_k^s + f_k^s \geq 0 \quad \forall k = 1, \ldots, K^s & A_k \\
& v_{nk}^s - \tau_k^s z_k^s \geq 0 \quad \forall k = 1, \ldots, K^s & B_k^+ \\
& -v_{nk}^s + \tau_{k+1}^s z_k^s \geq 0 \quad \forall k = 1, \ldots, K^s & B_k^- \\
& v_{jk}^s - v_{ik}^s - d_{ij}^s z_k^s + (d_{ij}^s - g_{ij}^s) w_{ijk}^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, K^s & C_{ijk} \\
& -\sum_{k=1}^{K^s} w_{ijk}^s = -\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} & D_{ij} \\
& -w_{ijk}^s + z_k^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, K^s & E_{ijk} \\
& \sum_{k=1}^{K^s} z_k^s = 1 & F \\
& w_{ijk}^s \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, K^s, \quad z_k^s \geq 0 \quad \forall k = 1, \ldots, K^s &
\end{array}
$$

Consider an optimal primal solution having objective function value $\hat{f}^s$ and completion time $\hat{u}_n^s$, such that $\hat{u}_n^s$ belongs to interval $k' \in \{1, \ldots, K^s\}$ (noting that $\hat{f}^s = m_{k'}^s \hat{u}_n^s + b_{k'}^s$). Define "slopes" ρ_k^L and ρ_k^R associated with interval k, where $\rho_k^L = (\hat{f}^s - (m_k^s \tau_k^s + b_k^s))/(\hat{u}_n^s - \tau_k^s)$, and $\rho_k^R = (\hat{f}^s - (m_k^s \tau_{k+1}^s + b_k^s))/(\hat{u}_n^s - \tau_{k+1}^s)$, $\forall k = 1, \ldots, K^s$. That is, ρ_k^L is the slope of the penalty from its current value to the value of the penalty function on the left interval value of segment k, and ρ_k^R is similarly defined for the right interval value of segment k. If $\hat{u}_n^s = \tau_{k'+1}^s$, we use the conventions $\rho_{k'}^R = m_{k'}^s$ and $\rho_{k'+1}^L = \infty$. Note that ρ_k^L and ρ_k^R are always nonnegative since all the penalty functions are nondecreasing. Define $\hat{\mathcal{A}}$ as a set containing all arcs in a critical path, and $\hat{X}^b = \{(i,j) \in \hat{\mathcal{A}} : \hat{x}_{ij} = b\}$, for $b = 0$ and 1.

Proposition 2.1. *An optimal dual solution to SP-LS$^s(\hat{x})$-RLT is given as follows.*

- $A_k = 1,\ \forall k = 1, \ldots, K^s$;
- $C_{ijk} = 0,\ \forall (i,j) \notin \hat{\mathcal{A}},\ k = 1, \ldots, K^s$;
- *for all* $k = 1, \ldots, k',\ C_{ijk} = \max\{\rho_k^L,\ \rho_k^R,\ m_{k'}^s\}$, *and for all* $k = k'+1, \ldots, K^s,\ C_{ijk} = \min\{\rho_k^L,\ \rho_k^R,\ m_{k'}^s\},\ \forall (i,j) \in \hat{\mathcal{A}}$. *If* $C_{ijk} \geq m_k^s,\ B_k^+ = 0,\ B_k^- = C_{ijk} - m_k^s$; *otherwise,* $B_k^- = 0,\ B_k^+ = m_k^s - C_{ijk}$;
- *for arc* $(i,j) \in \hat{X}^1,\ D_{ij} = \min_{k=k',\ldots,K^s}\{(d_{ij}^s - g_{ij}^s) C_{ijk}\}$, *and for arc* $(i,j) \in \hat{X}^0,\ D_{ij} = \max_{k=1,\ldots,k'-1}\{(d_{ij}^s - g_{ij}^s) C_{ijk}\}$;

- *for all* $k = 1, \ldots, K^s$*, if arc* $(i,j) \in \hat{X}^1$, $E_{ijk} = (d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij}$*, and if* $(i,j) \in \hat{X}^0$, $E_{ijk} = 0$*;*
- $F = \hat{f}^s + \sum_{(i,j)\in\hat{X}^1} D_{ij}$.

Proof. We demonstrate that the proposed dual solution is dual feasible and complementary slack to the primal solution. The dual feasibility conditions to SP-LS$^s(\hat{x})$-RLT are given by

$$-m_k^s A_k + B_k^+ - B_k^- + \sum_{i\in RS(n)} C_{ink} = 0 \quad \forall k = 1, \ldots, K^s \tag{2–8a}$$

$$\sum_{l\in RS(i)} C_{lik} - \sum_{j\in FS(i)} C_{ijk} = 0 \quad \forall i = 1, \ldots, n-1,\ k = 1, \ldots, K^s \tag{2–8b}$$

$$(d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij} - E_{ijk} \le 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, K^s \tag{2–8c}$$

$$A_k = 1 \quad \forall k = 1, \ldots, K^s \tag{2–8d}$$

$$-b_k^s A_k - \tau_k^s B_k^+ + \tau_{k+1}^s B_k^- - \sum_{(i,j)\in\mathcal{A}} d_{ij}^s C_{ijk} + \sum_{(i,j)\in\mathcal{A}} E_{ijk} + F \le 0 \quad \forall k = 1, \ldots, K^s \tag{2–8e}$$

$$A_k,\ B_k^+,\ B_k^-,\ C_{ijk},\ E_{ijk} \ge 0 \quad \forall (i,j) \in \mathcal{A},\ k = 1, \ldots, K^s, \tag{2–8f}$$

where Constraints 2–8a, 2–8b, 2–8c, 2–8d, and 2–8e are associated with primal v_{nk}, v_{ik} $(\forall i = 1, \ldots, n-1)$, w_{ijk}, f_k, and z_k-variables, respectively. We first verify the feasibility of our given dual values. Note that Constraints 2–8d is directly satisfied by our choice of $A_k = 1$, $\forall k$. Also, for each k, all nonzero C_{ijk} values are equal and correspond to arcs (i,j) in a critical path, which verifies feasibility with respect to Constraints 2–8b. Furthermore, since $\sum_{i\in RS(n)} C_{ink} = C_{i^*nk}$, where $(i^*, n) \in \hat{\mathcal{A}}$, and since we explicitly set B_k^+ and B_k^- such that

$$C_{i^*nk} = m_k^s - B_k^+ + B_k^- \quad \forall k = 1, \ldots, K^s,$$

then Constraints 2–8a is satisfied. Also, all A_k, B_k^+, B_k^-, and C_{ijk} values are clearly nonnegative, since ρ_k^L, ρ_k^R, and m_k^s are all nonnegative.

We next show that Constraints 2–8c is satisfied, and that all E_{ijk} values are nonnegative. First, for $(i,j) \in \hat{X}^1$, E_{ijk} is set to $(d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij}$ so that Constraints

2–8c holds as an equality. To show that $E_{ijk} \geq 0$ in this case, for $k = 1, \dots, k'$, we have $C_{ijk} = \max\{\rho_k^L,\ \rho_k^R,\ m_{k'}^s\} \geq m_{k'}^s$, and since $D_{ij} = \min_{k=k',\dots,K^s}\{(d_{ij}^s - g_{ij}^s)C_{ijk}\} \leq (d_{ij}^s - g_{ij}^s)C_{ijk'}$ for $(i,j) \in \hat{X}^1$, we get $(d_{ij}^s - g_{ij}^s)C_{ijk} \geq (d_{ij}^s - g_{ij}^s)m_{k'}^s \geq D_{ij}$, which verifies that $E_{ijk} = (d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij} \geq 0$. For $k = k'+1, \dots, K^s$, $(d_{ij}^s - g_{ij}^s)C_{ijk} \geq \min_{k=k',\dots,K^s}\{(d_{ij}^s - g_{ij}^s)C_{ijk}\} = D_{ij}$, which also yields $E_{ijk} = (d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij} \geq 0$.

Now, for $(i,j) \in \hat{X}^0$, if $k = 1, \dots, k'$, $(d_{ij}^s - g_{ij}^s)C_{ijk} \leq \max_{k=1,\dots,k'}\{(d_{ij}^s - g_{ij}^s)C_{ijk}\} = D_{ij}$, while if $k = k'+1, \dots, K^s$, $(d_{ij}^s - g_{ij}^s)C_{ijk} = (d_{ij}^s - g_{ij}^s)\min\{\rho_k^L,\ \rho_k^R,\ m_{k'}^s\} \leq (d_{ij}^s - g_{ij}^s)m_{k'}^s \leq \max_{k=1,\dots,k'}\{(d_{ij}^s - g_{ij}^s)C_{ijk}\} = D_{ij}$. Therefore, setting $E_{ijk} = 0,\ \forall (i,j) \in \hat{X}^0,\ k = 1, \dots, K^s$ satisfies the remaining inequalities of 2–8c, and ensures nonnegativity of $E_{ijk},\ \forall (i,j) \in \mathcal{A},\ k = 1, \dots, K^s$.

Next, we verify that our dual solution is feasible to Ineq. 2–8e. For all $k = 1, \dots, K^s$, we have $E_{ijk} = 0 = (d_{ij}^s - g_{ij}^s)C_{ijk}\hat{x}_{ij},\ \forall (i,j) \in \hat{X}^0$, and $E_{ijk} = (d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij},\ \forall (i,j) \in \hat{X}^1$. Hence,

$$\begin{aligned}\sum_{(i,j)\in\mathcal{A}} E_{ijk} &= \sum_{(i,j)\in\hat{X}^0} E_{ijk} + \sum_{(i,j)\in\hat{X}^1} E_{ijk} \\ &= \sum_{(i,j)\in\hat{\mathcal{A}}} (d_{ij}^s - g_{ij}^s)C_{ijk}\hat{x}_{ij} - \sum_{(i,j)\in\hat{X}^1} D_{ij} \quad \forall k = 1, \dots, K^s. \end{aligned} \tag{2–9}$$

Substituting $\sum_{(i,j)\in\mathcal{A}} E_{ijk}$ with Constraints 2–9, and noting Constraints 2–8a, 2–8d, $C_{ijk} = 0,\ \forall (i,j) \notin \hat{\mathcal{A}}$, $C_{ijk} = C_{i^*nk},\ \forall (i,j) \in \hat{\mathcal{A}}$, and $\sum_{(i,j)\in\hat{\mathcal{A}}}(d_{ij}^s - (d_{ij}^s - g_{ij}^s)\hat{x}_{ij}) = \hat{u}_n^s$, we write Ineq. 2–8e as

$$\begin{aligned}&-b_k^s - \tau_k^s B_k^+ + \tau_{k+1}^s B_k^- - \hat{u}_n^s C_{i^*nk} - \sum_{(i,j)\in\hat{X}^1} D_{ij} + F \leq 0 \quad \forall k = 1, \dots, K^s, \text{or} \\ &-b_k^s - \tau_k^s B_k^+ + \tau_{k+1}^s B_k^- - \hat{u}_n^s C_{i^*nk} + \hat{f}^s \leq 0 \quad \forall k = 1, \dots, K^s. \end{aligned} \tag{2–10}$$

If $C_{i^*nk} \geq m_k^s$, with respect to Constraints 2–8a, we substitute $B_k^+ = 0,\ B_k^- = C_{i^*nk} - m_k^s$, and Constraints 2–10 becomes

$$(\hat{f}^s - (m_k^s\tau_{k+1}^s + b_k^s)) - (\hat{u}_n^s - \tau_{k+1}^s)C_{i^*nk} \leq 0. \tag{2–11}$$

For $k = 1, \dots, k'-1$, $(\hat{u}_n^s - \tau_{k+1}^s) > 0$, we have $C_{i^*nk} \geq (\hat{f}^s - (m_k^s \tau_{k+1}^s + b_k^s))/(\hat{u}_n^s - \tau_{k+1}^s) = \rho_k^R$; for $k = k', \dots, K^s$, if $(\hat{u}_n^s - \tau_{k+1}^s) < 0$, Constraints 2–11 requires $C_{i^*nk} \leq \rho_k^R$. If $k = k'$, $\hat{u}_n^s = \tau_{k'+1}^s$, and Constraints 2–11 becomes $\hat{f}^s - (m_{k'}^s \tau_{k'+1}^s + b_{k'}^s) - 0 C_{i^*nk'} = 0 \leq 0$. All of the above cases are satisfied by the defined value of C_{i^*nk}.

Similarly, if $C_{i^*nk} < m_k^s$, substituting $B_k^- = 0$, $B_k^+ = m_k^s - C_{i^*nk}$ in Constraints 2–10 gives us

$$(\hat{f}^s - (m_k^s \tau_k^s + b_k^s)) - (\hat{u}_n^s - \tau_k^s) C_{i^*nk} \leq 0. \tag{2–12}$$

We have $(\hat{u}_n^s - \tau_k^s) > 0$ for $k = 1, \dots, k'$ and $(\hat{u}_n^s - \tau_k^s) < 0$ for $k = k'+1, \dots, K^s$, due to which we require $C_{i^*nk} \geq (\hat{f}^s - (m_k^s \tau_k^s + b_k^s))/(\hat{u}_n^s - \tau_k^s) = \rho_k^L$ and $C_{i^*nk} \leq \rho_k^L$, respectively. The given value of C_{i^*nk} satisfies both of the requirements. In particular, when $k = k'$, $\rho_{k'}^L = \rho_{k'}^R = m_{k'}^s$, and from the above analysis we note that $C_{i^*nk'} \leq \rho_{k'}^R$ and $C_{i^*nk'} \geq \rho_{k'}^L$. Therefore, $C_{i^*nk'} = m_{k'}^s$, and Ineq. 2–8e holds as an equality. Thus, all dual feasibility conditions are satisfied.

Finally, we demonstrate that the dual solution is complementary slack to the primal. Recall that in the primal solution, we have $v_{ik}^s = u_i^s z_k^s$, $\forall i, k$, $f_k^s = 0$, $\forall k \neq k'$ (i.e., if $z_k^s = 0$), and $f_{k'}^s = \hat{f}^s = m_{k'}^s u_n^s + b_{k'}^s$. Then Constraints 2–7c is always binding $\forall k$, and Constraints 2–7e and 2–7g are binding for any $k \neq k'$. Recall that when $k = k'$, we set $C_{ijk'} = m_{k'}^s$ and $B_{k'}^+ = B_{k'}^- = 0$, which ensures complementary slackness with respect to Constraints 2–7e and 2–7g. Since $w_{ijk}^s = \hat{x}_{ij} z_k^s$, primal constraints 2–7h are potentially not binding only for $(i,j) \notin \hat{\mathcal{A}}$, and we satisfy complementary slackness by setting $C_{ijk} = 0$, $\forall k$, corresponding to $(i,j) \notin \hat{\mathcal{A}}$. Next, note that $E_{ijk} > 0$ only when $\hat{x}_{ij} = 1$ (i.e., $(i,j) \in \hat{X}^1$) and thus $w_{ijk}^s = z_k^s$, ensuring that Constraints 2–7j is binding. Now, with respect to dual inequality 2–8c, note that whenever its corresponding primal variable $w_{ijk}^s = 1$, we must have $k = k'$ ($z_k^s = 1$) and $\hat{x}_{ij} = 1$, in which case we set $E_{ijk'} = (d_{ij}^s - g_{ij}^s) C_{ijk'} - D_{ij}$, and ensure that Ineq. 2–8c is binding in this case. Finally, the primal variables z_k^s associated with dual inequality 2–8e are all zero except for $z_{k'}^s = 1$, in which case we demonstrated in the proof of dual feasibility that Ineq. 2–8e is binding.

Hence, our dual solution is feasible and complementary slack to the primal feasible solution, which completes the proof. □

If η_s is obtained by solving the first-stage master problem for some $s \in \Omega$, such that $\eta_s < \hat{f}^s$, a Benders cut can be generated as:

$$\eta_s \geq \hat{f}^s + \sum_{(i,j)\in\hat{X}^1} D_{ij}(1 - x_{ij}) - \sum_{(i,j)\in\hat{X}^0} D_{ij}x_{ij}. \tag{2–13}$$

Remark 2.1. Observe that we can compute the optimal dual values in Proposition 2.1 based on the current critical path length for the given scenario and its corresponding penalty. Hence, we generate Benders cuts in each scenario via the solution of critical path problems and avoid the direct solution of SP-LS$^s(\hat{x})$-RLT. This efficient cut-generation scheme is critical in reducing computational effort as evident in the computational results of Section 2.4. Also, note that the dual D_{ij} for arc $(i,j) \in \hat{X}^1$ can be interpreted as an underestimate of the rate at which the penalty function would increase due to uninsuring arc (i,j), while D_{ij} for $(i,j) \in \hat{X}^0$ is an overestimate of the rate of penalty reduction due to insuring arc (i,j).

In fact, there exist several alternative optimal dual solutions to SP-LS$^s(\hat{x})$-RLT, which can yield different cutting planes for MP-PW, as shown in Proposition 2.2.

Proposition 2.2. *Denoting* $\Delta^s(\hat{x}) = (\tau^s_{k'+1}m^s_{k'+1} + b^s_{k'+1} - \hat{f}^s)$*, an alternative optimal dual solution to SP-LS*$^s(\hat{x})$*-RLT is given by modifying the dual values in Proposition 2.1 as follows.*

- *For all* $k = k'+1, \dots, K^s$*, set* $C_{ijk} = 0,\ \forall (i,j) \in \hat{\mathcal{A}}$*, and* $B^-_k = 0,\ B^+_k = m^s_k$*.*
- *Arbitrarily order the arcs* $(i,j) \in \hat{X}^1$*, and index them as* $H = ((i,j)_1, \dots, (i,j)_{|\hat{X}^1|})$*. Associating dual* D_{ij_h} *with arc* $(i,j)_h \in \hat{X}^1$*, set* $D_{(ij)_h} = \min\{(d^s_{(ij)_h} - g^s_{(ij)_h})m^s_{k'},\ \Delta^s(\hat{x}) - \sum_{l=1}^{h-1} D_{(ij)_l}\},\ h = 1, \dots, H$*, and* $E_{ijk} = (d^s_{ij} - g^s_{ij})C_{ijk} - D_{ij},\ \forall k = 1, \dots, k',\ E_{ijk} = 0,\ \forall k = k'+1, \dots, K^s$*.*

Proof. We can verify the dual feasibility of our solution to Constraints 2–8a, 2–8b, 2–8d, and the nonnegativity of $A_k,\ B_k^+,\ B_k^-,\ C_{ijk}$ values, using the same arguments in the proof of Proposition 2.1.

We next demonstrate that Ineq. 2–8c is satisfied and all E_{ijk} are nonnegative. For arc $(i,j) \in \hat{X}^1$, if $k = 1, \dots, k'$, E_{ijk} is set to $(d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij}$, and Ineq. 2–8c is satisfied as an equality. Also, we have $(d_{ij}^s - g_{ij}^s)C_{ijk} = (d_{ij}^s - g_{ij}^s)\max\{\rho_k^L,\ \rho_k^R,\ m_{k'}^s\} \geq (d_{ij}^s - g_{ij}^s)m_{k'}^s \geq D_{ij}$, and $E_{ijk} \geq 0$ in this case using the same proof in Proposition 2.1. For $k = k'+1, \dots, K^s$, since $C_{ijk} = E_{ijk} = 0$, we need to show that $D_{ij} \geq 0$. If $\sum_{(i,j)\in\hat{X}^1}(d_{ij}^s - g_{ij}^s)m_{k'}^s \leq \Delta^s(\hat{x})$, we have $D_{ij} = (d_{ij}^s - g_{ij}^s)m_{k'}^s \geq 0,\ \forall (i,j) \in \hat{X}^1$; else, the values of D_{ij} are nonnegative for $h = 1, \dots, h'$ such that $\sum_{h=1}^{h'} D_{(ij)_h} = \Delta^s(\hat{x})$, and $D_{(ij)_h} = 0$ for the remaining values of h, which verifies that $D_{ij} \geq 0,\ \forall (i,j) \in \hat{X}^1$. Now, for all arcs $(i,j) \in \hat{X}^0$ and $k = 1, \dots, K^s$, note that $D_{ij} = \max_{k=1,\dots,k'}\{(d_{ij}^s - g_{ij}^s)C_{ijk}\} \geq (d_{ij}^s - g_{ij}^s)C_{ijk}$ (since $C_{ijk} = 0,\ \forall (i,j) \in \hat{\mathcal{A}},\ k = k'+1, \dots, K^s$), i.e., $(d_{ij}^s - g_{ij}^s)C_{ijk} - D_{ij} \leq 0$, and thus setting $E_{ijk} = 0$ satisfies Ineq. 2–8c.

Next, we verify the feasibility of Ineq. 2–8e. First, for $k = 1, \dots, k'$, $E_{ijk} = 0 = (d_{ij}^s - g_{ij}^s)C_{ijk}\hat{x}_{ij}, \forall (i,j) \in \hat{X}^0$, and $E_{ijk} = (d_{ij}^s - g_{ij}^s)C_{ijk}\hat{x}_{ij} - D_{ij},\ \forall (i,j) \in \hat{X}^1$. Hence, we have satisfied both Ineq. 2–9 and 2–10 for $k = 1, \dots, k'$. By using the same argument in the proof of Proposition 2.1, one can verify the feasibility of Ineq. 2–8e in the case of $k = 1, \dots, k'$.

For $k = k'+1, \dots, K^s$, $C_{ijk} = E_{ijk} = 0,\ \forall (i,j) \in \mathcal{A}$, and based on Ineq. 2–8a, $B_k^+ = m_k^s,\ B_k^- = 0$, we now transform Ineq. 2–8e into

$$-b_k^s - \tau_k^s m_k^s + \hat{f}^s + \sum_{(i,j)\in\hat{X}^1} D_{ij} \leq 0,\ \forall k = k'+1, \dots, K^s. \tag{2–14}$$

Note that by sequentially setting $D_{(ij)_h} = \min\{(d_{ij}^s - g_{ij}^s)m_{k'}^s,\ \Delta^s(\hat{x}) - \sum_{l=1}^{h-1} D_{ij_l}\}$, for all $(i,j) \in \hat{X}^1,\ h = 1, \dots, H$, we have

$$\sum_{(i,j)\in\hat{X}^1} D_{ij} = \min\left\{\sum_{(i,j)\in\hat{X}^1}(d_{ij}^s - g_{ij}^s)m_{k'}^s,\ \Delta^s(\hat{x})\right\} \leq \Delta^s(\hat{x}) = \tau_{k'+1}^s m_{k'+1}^s + b_{k'+1}^s - \hat{f}^s$$

which satisfies Ineq. 2–14 (since $\tau^s_{k'+1} m^s_{k'+1} + b^s_{k'+1} - \hat{f}^s \leq \tau^s_k m^s_k + b^s_k - \hat{f}^s$, $\forall k = k'+1, \dots, K^s$), and thus Ineq. 2–8e is also satisfied in the case of $k = k'+1, \dots, K^s$.

Complementary slackness of the dual solution to the primal is verified by the same argument as in the proof of Proposition 2.1. This completes the proof. □

We compare cutting planes 2–13 generated based on the duals from Proposition 2.1 and 2.2 in Figure 2-1. Note that both Proposition 2.1 and 2.2 essentially promise a decrease of $(d^s_{ij} - g^s_{ij})q^L$ in η_s due to setting $x_{ij} = 1$ for $(i,j) \in \hat{X}^0$, where q^L is the minimum slope of a tangent that underestimates Θ^s from τ^s_1 to $\hat{u}^s_n$, and touches Θ^s at $u^s_n = \hat{u}^s_n$. However, duals generated from Proposition 2.1 will increase η_s by $(d^s_{ij} - g^s_{ij})q^R$ due to setting $x_{ij} = 0$ for $(i,j) \in \hat{X}^1$, where q^R is the maximum slope of a tangent that underestimates Θ^s from $\hat{u}^s_n$ to $\tau^s_{K^s+1}$, and touches Θ^s at $u^s_n = \hat{u}^s_n$. By contrast, Proposition 2.2 increases η^s by $(d^s_{ij} - g^s_{ij})m^s_{k'}$ due to setting $x_{ij} = 0$ for $(i,j) \in \hat{X}^1$, but only enforces the total increase in η^s up to a maximum of $\Delta^s(\hat{x})$. After this limit is reached, the rate $m^s_{k'}$ of increase in η^s is no longer necessarily valid. Duals set according to Proposition 2.2 limit the total increase of η_s by $\Delta^s(\hat{x})$ by enforcing that $\sum_{(i,j)\in\hat{X}^1} D_{ij} \leq \Delta^s(\hat{x})$, and by greedily assigning positive values to D_{ij} in the order prescribed by H, until a total weight of $\Delta^s(\hat{x})$ has been assigned to those values, or until all $D_{ij} = (d^s_{ij} - g^s_{ij})m^s_{k'}$. For instance, suppose that $\hat{X}^1 = \{(1,2), (2,3), (3,4)\}$, with $m^s_{k'} = 1$ and $(d^s_{12} - g^s_{12}) = 15$, $(d^s_{23} - g^s_{23}) = 13$, and $(d^s_{34} - g^s_{34}) = 10$. Then if $\Delta^s(\hat{x}) = 24$, we could obtain any of the following five sets of dual D_{ij} values for $(i,j) \in \hat{X}^1$: $(D_{12}, D_{23}, D_{34}) = (15, 9, 0), (15, 0, 9), (11, 13, 0), (1, 13, 10)$, or $(14, 0, 10)$ by different permutations of H.

In general, we could generate an exponential number of alternative optimal dual solutions according to Proposition 2.2, none of which dominates the another. Furthermore, many of these inequalities may have $D_{ij} = 0$ for several $(i,j) \in \hat{X}^1$, particularly for those (i,j) that appear late in the ordering H. Alternatively, if $m^s_{k'}$ and/or $\Delta^s(\hat{x})$ are relatively large, then we can essentially scale the duals D_{ij} for $(i,j) \in \hat{X}^1$

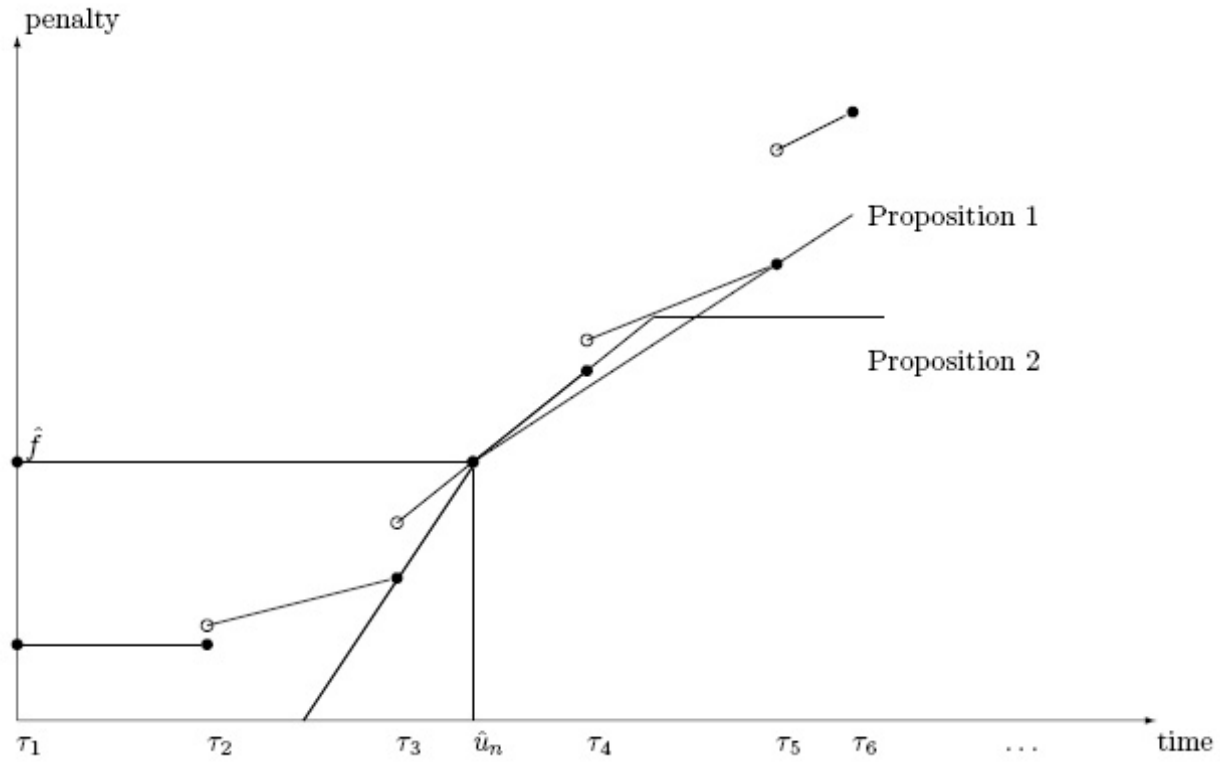


Figure 2-1. Comparison of cuts generated based on Proposition 2.1 and 2.2.

to ensure that duals $D_{ij},\ (i,j) \in \hat{X}^1$ set according to Proposition 2.2 would not be forced to zero (unless $(d^s_{ij} - g^s_{ij})m^s_{k'} = 0$). Note that no such scaling is necessary if $\Delta^s(\hat{x}) \geq \sum_{(i,j)\in\hat{X}^1}(d^s_{ij} - g^s_{ij})m^s_{k'}$, since in that case every permutation H would yield the same set of dual values, in which $D_{ij} = (d^s_{ij} - g^s_{ij})m^s_{k'},\ \forall (i,j) \in \hat{X}^1$. Also, if $\Delta^s(\hat{x}) = 0$, scaling would not be possible. Otherwise, if $0 < \Delta^s(\hat{x}) \leq \sum_{(i,j)\in\hat{X}^1}(d^s_{ij} - g^s_{ij})m^s_{k'}$, then setting $D_{ij} = (d^s_{ij} - g^s_{ij})m^s_{k'}[\Delta^s(\hat{x})/\sum_{(i,j)\in\hat{X}^1}(d^s_{ij} - g^s_{ij})m^s_{k'}]$ for all $(i,j) \in \hat{X}^1$ yields the following inequality:

$$\eta_s \geq \hat{f}^s - \sum_{(i,j)\in\hat{X}^0} D_{ij}x_{ij} + \frac{\Delta^s(\hat{x})}{\sum_{(i,j)\in\hat{X}^1}(d^s_{ij} - g^s_{ij})} \sum_{(i,j)\in\hat{X}^1} (d^s_{ij} - g^s_{ij})(1 - x_{ij}). \tag{2–15}$$

Remark 2.2. If $m^s_{k'} > q^R$, then according to Proposition 2.2, we greedily increase the coefficients associated with some arcs $(i,j) \in \hat{X}^1$, while ensuring that $\sum_{(i,j)\in\hat{X}^1} D_{ij}$ does not exceed $\Delta^s(\hat{x})$. In fact, we may generate an alternative cut of the form 2–13 by using

any slope value $\bar{q}^R \in [q^R, m^s_{k'}]$, in which we set $D_{(ij)_h} = \min\{(d^s_{(ij)_h} - g^s_{(ij)_h})\bar{q}^R, \Delta^s(\hat{x}) - \sum_{l=1}^{h-1} D_{(ij)_l}\}$, for all $h = 1, \ldots, H$. Let this affine function be described by $\bar{q}^R u^s_n + \bar{b}$.

Note that when $\bar{q}^R > q^R$, there exists a first "crossing point" $CP \geq \hat{u}^s_n$, such that there exists an $\varepsilon > 0$, where $\Theta^s(CP + \delta) < \bar{q}^R(CP + \delta) + \bar{b}, \forall\, 0 < \delta < \varepsilon$. We can underestimate Θ^s in this case by using a continuous three-segment piecewise-linear function: One from τ^s_1 to $\hat{u}^s_n$ having slope q^L (and intersecting Θ^s at $u^s_n = \hat{u}^s_n$), one from $\hat{u}^s_n$ to CP having slope $\hat{q}^R$, and the last from CP to $\tau^s_{K^s+1}$ having slope 0. Since the penalty function is nondecreasing, we ensure that this generated function underestimates Θ^s. We can then generate cuts of the form 2–13 based on an application of Proposition 2.2 to this three-segment underestimating function.

Define $\bar{\Delta}^s(\hat{x})$ as the difference between the values of crossing point TP^s and $(\hat{u}^s_n, \hat{f}^s)$ in the original penalty function. By employing the foregoing procedure, we increase the value of $\Delta^s(\hat{x})$ in Proposition 2.2 to $\bar{\Delta}^s(\hat{x})$. Consequently, we also need to increase the value of C_{ijk}, $k = k' + 1, \ldots, K^s$, $\forall (i,j) \in \hat{\mathcal{A}}$ to maintain dual feasibility regarding Ineq. 2–8e. Furthermore, note that by employing a different slope $\bar{q}^R \in [q^R, m^s_{k'}]$, the cutting plane touches the original penalty function at various values of TP^s, and thus we obtain different values of $\bar{\Delta}^s(\hat{x})$, as well as C_{ijk}, $k = k' + 1, \ldots, K^s$, $\forall (i,j) \in \hat{\mathcal{A}}$. Since the penalty function is nondecreasing, $\bar{\Delta}^s(\hat{x})$ and C_{ijk} increase as $\bar{q}^R$ decreases. In particular, when $\bar{q}^R = q^R$, we obtain the same duals as given in Proposition 2.1. Note that $\bar{q}^R \leq m^s_{k'}$, by replacing $\Delta^s(\hat{x})$ with $\bar{\Delta}^s(\hat{x})$, we underestimate Cut 2–13 in Proposition 2.2, and thus we verify the validity of these diversifications of Cut 2–13 based on Proposition 2.2.

2.3.3 General Nonconvex Penalty Function

In this section, we extend our cutting-plane algorithms to more general nonconvex penalty functions. We now assume only that Θ^s is lower semi-continuous, nondecreasing, and does not have infinite derivatives. Our approach essentially dynamically approximates the penalty function using linear or two-segment concave piecewise-linear functions,

based on the current critical path length and penalty function shape. We describe the details as follows.

1. Initialize the algorithm by setting $UB = \infty$, and by formulating MP-PW with no Benders inequalities.

2. Solve the master problem MP-PW and obtain first-stage solution $\hat{x}$ and lower bound value LB. Solve CPM$^s(\hat{x})$ and obtain a critical path length $\hat{u}_n^s$ for each $s \in \Omega$, which yields an upper bound of $c\hat{x} + \sum_{s\in\Omega} e^s\Theta^s(\hat{u}_n^s)$ on the optimal objective value. If UB is larger than this upper bound value, then set $UB = c\hat{x} + \sum_{s\in\Omega} e^s\Theta^s(\hat{u}_n^s)$. If $LB = UB$, then terminate with optimal solution $\hat{x}$; else, continue to the next step.

3. If $\eta_s < \Theta^s(\hat{u}_n^s)$ for some $s \in \Omega$, we can temporarily instate a piecewise-linear function that underestimates Θ^s. If a linear function $qu_n^s + q_0$ can underestimate Θ^s with $q\hat{u}_n^s + q_0 = \Theta^s(\hat{u}_n^s)$, then we generate a cut of the form 2–2. Otherwise, we underestimate Θ^s on the interval $[\tau_1^s, \hat{u}_n^s]$ with a linear function $q^L u_n^s + q_0^L$ and also on the interval $[\hat{u}_n^s, \tau_{K^s+1}^s]$ with a linear function $q^R u_n^s + q_0^R$, with $q^L > q^R$ and $q^L\hat{u}_n^s + q_0^L = q^R\hat{u}_n^s + q_0^R = \Theta^s(\hat{u}_n^s)$. We can then compute coefficients of Cut 2–13 according to Propositions 2.1 or 2.2. We add Cut 2–13 to MP-PW, and return to step 2.

Note that Ineq. 2–13 generated for scenario s exactly approximates $\Theta^s(\hat{u}_n^s)$ at $\hat{x}$. Since all x-variables are binary, there are a finite number of solutions to MP-PW, which ensures that this algorithm finitely reaches an optimal solution.

Example 1. Suppose that the penalty function of scenario s is of the form

$$\Theta^s(t) = \begin{cases} 1 & 0 \le t \le 1; \\ 2 + \frac{(t-1)^2}{15} & 1 < t \le 4; \\ 2\sqrt{t} & 4 < t \le 10; \\ 6 + \frac{(t-9)^2}{2} & 10 < t \le 11. \end{cases}$$

Given a critical path length $\hat{u}_n^s = 9$, a penalty $\Theta^s(9) = 6$ is incurred. We underestimate Θ^s on the interval $[0, 9]$ by passing an affine function through $\hat{u}_n^s = 9,\ \Theta^s(\hat{u}_n^s) = 6$, with the smallest possible slope q^L that underestimates the function. Next, we repeat this procedure over the interval $[9, 11]$, obtaining a maximum slope q^R that underestimates Θ^s over this interval. We compute these slopes as follows

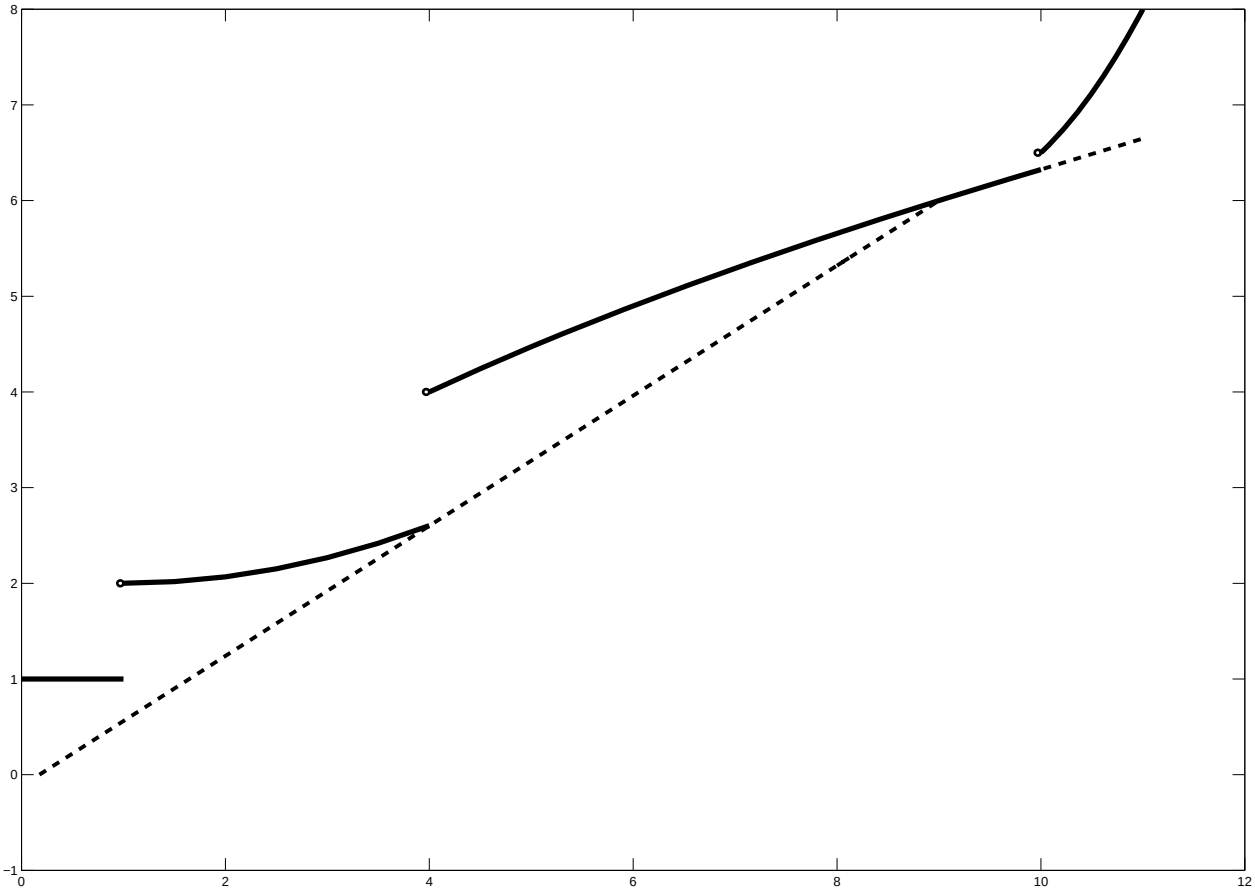


Figure 2-2. Illustration of cutting-plane algorithm for a nonconvex penalty function.

(illustrated in Figure 2-2).

$$q^L = \frac{\hat{f}^s - \Theta^s(4)}{\psi^s(\hat{x}) - 4} = \frac{6 - 2.6}{9 - 4} = 0.68,$$
$$q^R = \frac{\hat{f}^s - \Theta^s(10)}{\psi^s(\hat{x}) - 10} = \frac{6 - 2\sqrt{10}}{9 - 10} = (2\sqrt{10} - 6).$$

If $\eta^s < 6$ in the solution of MP-PW, then we generate a Benders cut as:

$$\eta_s \geq 6 + \sum_{(i,j)\in\hat{X}^1} (2\sqrt{10} - 6)(d_{ij}^s - g_{ij}^s)(1 - x_{ij}) - \sum_{(i,j)\in\hat{X}^0} 0.68(d_{ij}^s - g_{ij}^s)x_{ij}. \qquad (2\text{–}16)$$

2.3.4 Chance-constrained Problem

The previous analysis also permits us to consider a chance-constrained version of STIP as follows:

$$\textbf{CC}_\epsilon\textbf{:}\ \min\left\{\sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}\ :\ x\in X,\ Pr\{\psi(x,\xi)\le\mathcal{T}\}\ge 1-\epsilon\right\}, \tag{2–17}$$

where $X\subseteq B^{|\mathcal{A}|}$ forms a deterministic feasible region, ξ is a random vector with support $\xi\in\Xi\subset R^l$, and $\psi: R^n\times R^l\to R^m$ is a given constraint mapping that generates the critical path length given a first-stage decision x and ξ. Also, $\mathcal{T}$ is a random variable associated with the critical path length threshold, and ϵ is a risk level parameter chosen by the decision maker. Given a finite set of scenarios Ω, and ξ^s and $\mathcal{T}^s$ as the realization of ξ and $\mathcal{T}$ under scenario $s\in\Omega$, the chance-constrained can be rewritten as

$$\sum_{s\in\Omega}\mathbb{I}(\psi(x,\xi^s)\le\mathcal{T}^s)\ge(1-\epsilon)|\Omega|, \tag{2–18}$$

where $\mathbb{I}(A)$ denotes whether event A is true (i.e., $\mathbb{I}(A)=1$) or not (i.e., $\mathbb{I}(A)=0$). Define variables $p^s\in\{0,1\},\ \forall s\in\Omega$, such that $p^s=1$ if a critical path length is permitted to violate the project target time $\mathcal{T}^s$ in scenario s, and $p^s=0$ otherwise. Recalling that η_s is the s^{th} subproblem objective, we formulate the master problem of CC_ϵ as follows.

$$\begin{aligned}
\textbf{CC}_\epsilon\textbf{-MP:}\ \min\quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \\
\text{s.t.}\quad & m^{s,l}x+n^{s,l}\eta_s\ge o^{s,l}\quad \forall l\in L^s,\ s\in\Omega && \text{(2–19a)}\\
& \sum_{s\in\Omega}\eta_s\le\lfloor\epsilon|\Omega|\rfloor && \text{(2–19b)}\\
& x_{ij}\in\{0,1\}\ \forall(i,j)\in\mathcal{A},\quad \eta_s\ge 0\ \forall s\in\Omega, && \text{(2–19c)}
\end{aligned}$$

where $m^{s,l},\ n^{s,l}$, and $o^{s,l}$ represent the coefficients associated with the l^{th} Benders cut derived from the s^{th} subproblem, which is formulated as

$$\textbf{CC}_\epsilon\textbf{-SP}^s(\hat{x})\textbf{:}\ \min\quad p^s$$

$$\text{s.t.} \quad u_j^s - u_i^s \geq d_{ij}^s - (d_{ij}^s - g_{ij}^s)\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{2–20a}$$

$$u_n^s \leq \mathcal{T}^s + Qp^s \tag{2–20b}$$

$$p^s \in \{0,1\}, \tag{2–20c}$$

where Q is once again a large constant. We multiply Ineq. 2–20a and 2–20b, as well as the bounds $x_{ij} \leq 1$ and $x_{ij} \geq 0$, by p^s and by $(1-p^s)$, to generate the convex hull of solutions for which p^s is binary. We then linearize by substituting $(p^s)^2 = p^s,\ p^s(1-p^s) = 0$, and by defining $v_i^s \equiv u_i^s p^s,\ \forall i = 0,\ldots,n$, $w_{ij}^s \equiv x_{ij}p^s,\ \forall (i,j) \in \mathcal{A}$. After decomposition, the resulting subproblem is given by

CC$_\epsilon$-SP$^s(\hat{x})$-RLT:

$$\begin{aligned}
\min \quad & p^s \\
\text{s.t.} \quad & v_j^s - v_i^s + (d_{ij}^s - g_{ij}^s)w_{ij}^s - d_{ij}^s p^s \geq 0 \quad \forall (i,j) \in \mathcal{A} && \text{(2–21a)} \\
& u_j^s - v_j^s - u_i^s + v_i^s - (d_{ij}^s - g_{ij}^s)w_{ij}^s + d_{ij}^s p^s \\
& \qquad \geq d_{ij}^s - (d_{ij}^s - g_{ij}^s)\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} && \text{(2–21b)} \\
& -v_n^s + (\mathcal{T}^s + Q)p^s \geq 0 && \text{(2–21c)} \\
& -u_n^s + v_n^s - \mathcal{T}^s p^s \geq -\mathcal{T}^s && \text{(2–21d)} \\
& -w_{ij}^s \geq -\hat{x}_{ij} \quad \forall (i,j) \in \mathcal{A} && \text{(2–21e)} \\
& -w_{ij}^s + p^s \geq 0 \quad \forall (i,j) \in \mathcal{A} && \text{(2–21f)} \\
& w_{ij}^s - p^s \geq \hat{x}_{ij} - 1 \quad \forall (i,j) \in \mathcal{A} && \text{(2–21g)} \\
& w_{ij}^s \geq 0 \quad \forall (i,j) \in \mathcal{A}, \qquad p^s \geq 0. && \text{(2–21h)}
\end{aligned}$$

After solving CC$_\epsilon$-MP, if $\hat{\eta}_s = 1$, or if both $\hat{\eta}_s$ and the optimal objective value to CC$_\epsilon$-SP$^s(\hat{x})$-RLT equal zero, then no cut is generated. Otherwise, if $\hat{\eta}_s$ is less than the optimal objective to CC$_\epsilon$-SP$^s(\hat{x})$-RLT, we generate a Benders cut.

With respect to the structural constraints, we associate duals C_{ij}^-, C_{ij}^+, B^+, B^-, D_{ij}, E_{ij}^-, and E_{ij}^+ with Constraints 2–21a, 2–21b, 2–21c, 2–21d, 2–21e, 2–21f, and

2–21g, respectively. Recall that $\hat{\mathcal{A}}$ denotes a set containing all arcs in a critical path, and $\hat{X}^b = \{(i,j) \in \hat{A} : \ \hat{x}_{ij} = b\}$, for $b = 0$ and 1.

Proposition 2.3. *Given $\hat{x}$ from the master problem, for each scenario s with $\hat{p}^s = 1$ (i.e., $\hat{u}_n^s > \mathcal{T}^s$), if $\hat{\eta}_s < \hat{p}^s = 1$, an optimal solution to CC$_\epsilon$-SP$^s(\hat{x})$-RLT is given as:*

- $C_{ij}^- = C_{ij}^+ = 1/(\hat{u}_n^s - \mathcal{T}^s), \ \forall (i,j) \in \hat{\mathcal{A}}$;
- $C_{ij}^- = C_{ij}^+ = 0, \ \forall (i,j) \in \mathcal{A} \setminus \hat{\mathcal{A}}$;
- $B^+ = B^- = 1/(\hat{u}_n^s - \mathcal{T}^s)$;
- *for all* $(i,j) \in \hat{X}^1$, $E_{ij}^+ = E_{ij}^- = (d_{ij}^s - g_{ij}^s)/(\hat{u}_n^s - \mathcal{T}^s), \ D_{ij} = 0$;
- *for all* $(i,j) \in \hat{X}^0$, $E_{ij}^+ = E_{ij}^- = D_{ij} = 0$.

Proof. We demonstrate the dual feasibility and complementary slackness of the proposed dual solution. First, the dual feasibility conditions to CC$_\epsilon$-SP-$^s(\hat{x})$-RLT are given by:

$$\sum_{l \in RS(n)} (C_{ln}^- - C_{ln}^+) - B^+ + B^- = 0 \tag{2–22a}$$

$$\sum_{l \in RS(i)} (C_{li}^- - C_{li}^+) - \sum_{l \in FS(i)} (C_{ij}^- - C_{ij}^+) = 0 \quad \forall i = 1, \dots, n-1 \tag{2–22b}$$

$$\sum_{l \in RS(n)} C_{ln}^+ - B^- = 0 \tag{2–22c}$$

$$\sum_{l \in RS(i)} C_{li}^+ - \sum_{l \in FS(i)} C_{ij}^+ = 0 \quad \forall i = 1, \dots, n-1 \tag{2–22d}$$

$$(d_{ij}^s - g_{ij}^s)(C_{ij}^- - C_{ij}^+) - D_{ij} - E_{ij}^- + E_{ij}^+ \leq 0 \quad \forall (i,j) \in \mathcal{A} \tag{2–22e}$$

$$-\sum_{(i,j) \in \mathcal{A}} d_{ij}^s C_{ij}^- + \sum_{(i,j) \in \mathcal{A}} d_{ij}^s C_{ij}^+ + \sum_{(i,j) \in \mathcal{A}} (E_{ij}^- - E_{ij}^+) + (\mathcal{T}^s + Q)B^+ - \mathcal{T}^s B^- \leq 1 \tag{2–22f}$$

$$C_{ij}^-, \ C_{ij}^+, \ B^+, \ B^-, \ D_{ij}, \ E_{ij}^-, \ E_{ij}^+ \geq 0, \ \forall (i,j) \in \mathcal{A}, \tag{2–22g}$$

where Constraints 2–22a, 2–22b, 2–22c, 2–22d, 2–22e, and 2–22f are associated with primal v_n^s, v_i^s $(\forall i = 1, \dots, n-1)$, u_n^s, u_i^s $(\forall i = 1, \dots, n-1)$, w_{ij}^s, and p^s-variables, respectively. First, note that all C_{ij}^- and C_{ij}^+ values are equal corresponding to $(i,j) \in \hat{A}$, and $C_{ij}^- = C_{ij}^+ = 0$ for $(i,j) \in \mathcal{A} \setminus \hat{\mathcal{A}}$, so that Constraints 2–22b and 2–22d are both

satisfied. Constraint 2–22c is satisfied by setting $B^- = C^+_{i^*n}$, where arc (i^*, n) is the critical path arc entering node n. We also satisfy Eq. 2–22a by setting $B^+ = B^-$.

Constraint 2–22e, for $(i,j) \in \mathcal{A} \setminus \hat{\mathcal{A}}$, is satisfied by setting all $C^-_{ij},\ C^+_{ij},\ D_{ij},\ E^-_{ij}$, and E^+_{ij} values equal to zero. Recalling that $C^-_{ij} = C^+_{ij},\ \forall (i,j) \in \hat{\mathcal{A}}$, we satisfy Constraint 2–22e as an equality for $(i,j) \in \hat{X}^1$ by setting $E^+_{ij} = E^-_{ij} = (d^s_{ij} - g^s_{ij})/(\hat{u}^s_n - \mathcal{T}^s)$ and $D_{ij} = 0$. For $(i,j) \in \hat{X}^0$, we have Constraint 2–22e equaling zero since $E^+_{ij} = E^-_{ij} = D_{ij} = 0$. Also note that $E^+_{ij},\ E^-_{ij}$, and D_{ij} are all nonnegative for all $(i,j) \in \mathcal{A}$.

We verify that our dual solution is feasible for Constraints 2–22f. Given dual values of $E^+_{ij},\ E^-_{ij},\ C^-_{ij},\ C^+_{ij}, B^+$, and B^-, Eq. 2–22f becomes $-\sum_{(i,j)\in\mathcal{A}} d^s_{ij}(C^-_{ij} - C^+_{ij}) + 0 + (\mathcal{T}^s + Q - \mathcal{T}^s)/(\hat{u}^s_n - \mathcal{T}^s) = 1$ by assigning the compensating value $\hat{u}^s_n - \mathcal{T}^s \geq 0$ to Q, which is satisfied as an equality.

Finally, we demonstrate that the duals are also complementary slack to the the primal. Since $\hat{p}^s = 1$, Constraints 2–21a becomes $u^s_j - u^s_i \geq d^s_{ij} - (d^s_{ij} - g^s_{ij})\hat{x}_{ij},\ \forall (i,j) \in \mathcal{A}$, which are potentially not binding for $(i,j) \notin \hat{\mathcal{A}}$, and we satisfy the complementary slackness by setting $C^-_{ij} = 0$, corresponding to arcs $(i,j) \notin \hat{\mathcal{A}}$. Constraints 2–21b are always binding, and thus we require $C^+_{ij} \geq 0,\ \forall (i,j) \in \mathcal{A}$. For $D_{ij} = 0$ and $E^+_{ij} \geq 0,\ \forall (i,j) \in \mathcal{A}$, Constraints 2–21e and 2–21g are always binding since whenever a Benders cut needed to be generated, we have $\hat{p}^s = 1$ and $\hat{w}^s_{ij} = \hat{x}_{ij}$. If $\hat{x}_{ij} = 0$, (i.e., $(i,j) \in \hat{X}^0$), Constraints 2–21f holds as an inequality, and we have $E^-_{ij} = 0$ in this case. Next, Constraint 2–22e is always binding for the given dual values, and for $\hat{p}^s = 1$, we have already shown that Constraint 2–22f holds as an equality in the foregoing analysis. Therefore, the proposed values of duals are dual feasible and complementary slack to the primal. This completes the proof. □

For each scenario s with $\hat{p}^s = 1$, if $\hat{\eta}_s < \hat{p}^s$, a Benders cut is generated as:

$$\eta_s \geq \frac{\sum_{(i,j)\in\hat{\mathcal{A}}} d^s_{ij} - (d^s_{ij} - g^s_{ij})x_{ij}}{\hat{u}^s_n - \mathcal{T}^s} - \frac{\mathcal{T}^s}{\hat{u}^s_n - \mathcal{T}^s} + \sum_{(i,j)\in\hat{X}^1} \frac{(d^s_{ij} - g^s_{ij})}{\hat{u}^s_n - \mathcal{T}^s}(x_{ij} - 1)$$

$$= \frac{\sum_{(i,j)\in\hat{\mathcal{A}}} d_{ij}^s - \sum_{(i,j)\in\hat{X}^1} (d_{ij}^s - g_{ij}^s) - \mathcal{T}^s}{\hat{u}_n^s - \mathcal{T}^s} - \sum_{(i,j)\in\hat{\mathcal{A}}\setminus\hat{X}^1} \frac{(d_{ij}^s - g_{ij}^s)}{\hat{u}_n^s - \mathcal{T}^s} x_{ij}$$

$$= 1 - \sum_{(i,j)\in\hat{X}^0} \frac{(d_{ij}^s - g_{ij}^s)}{\hat{u}_n^s - \mathcal{T}^s} x_{ij}. \tag{2–23}$$

2.4 Computational Results

We demonstrate the computational efficacy of our cutting-plane algorithms for the expectation and chance-constrained problems by testing our algorithms on fifteen randomly generated network instances. Table 2-1 provides the parameters used to generate the instances, where $|\mathcal{N}|$ gives the number of nodes and **degree range** denotes the minimum and maximum degrees of each node allowed in the initial phase of graph generation. We generate each instance as a topologically ordered graph (i.e., where $(i,j) \in \mathcal{A}$ only if $i < j$) by implementing the following procedures. We begin by initializing $\mathcal{A} = \emptyset$, and then in a loop, we randomly generate two nodes $i,\ j \in \mathcal{N}$, where $i < j,\ (i,j) \notin \mathcal{A}$, and the degree of both i and j is strictly smaller than the maximum value of the degree range. We add arc (i,j) to the graph, and increase the degrees of i and j by one. We repeat this procedure until the degrees of all nodes lie within the degree range. After this initial phase is complete, we ensure that there exists at least one path from node 0 to node i, and one from node i to node n, for all $i = 1, \ldots, n-1$. If not, we artificially construct such path(s) from node 0 to node i or from node i to node n, and make sure that each path contains at least $\min\{0.2|\mathcal{N}|, \lceil 0.5i \rceil\}$ nodes for paths connecting 0 to i, and $\min\{0.2|\mathcal{N}|, \lceil 0.5(|\mathcal{N}| - i) \rceil\}$ nodes for paths connecting i to n. (Note that we potentially violate the maximum value of the degree range at some nodes after adding these additional paths.) We generate five such instances for each combination of $|\mathcal{N}|$ and degree range, and report the total number of arcs generated for each instance in Table 2-1.

For each arc $(i,j) \in \mathcal{A}$, we randomly generate a typical task duration value from a uniform distribution over the interval $[10, 300]$. To generate scenario data we examine

Table 2-1. Test instances.

$\|\mathcal{N}\|$	Degree range	$\|\mathcal{A}\|$ for each instance				
		1	2	3	4	5
30	[5, 15]	332	322	348	388	394
50	[10, 20]	850	814	870	802	856
70	[15, 25]	1640	1606	1638	1454	1490

the practical case in which a task is more likely to be delayed than completed earlier, and where the duration of delays exceeds the amount of time by which a task could be early. Hence, we generate d_{ij}^s in scenario s by multiplying the typical task duration for $(i,j) \in \mathcal{A}$ by a random value uniformly distributed over the interval $[0.9, 1.5]$. We randomly generate g_{ij}^s for arc $(i,j) \in \mathcal{A}$, scenario s, by generating g_{ij}^s from the uniformly distributed $[0.5d_{ij}^s, 0.7d_{ij}^s]$. The cost c_{ij} to insure arc $(i,j) \in \mathcal{A}$ is uniformly distributed over the interval $[25, 50]$. Finally, we round all the values of d_{ij}^s, g_{ij}^s and c_{ij} to the nearest integer values.

2.4.1 Expectation-based Penalty Function Cases

Our first experiment tests the computational efficacy of our procedures for the cases of convex penalty functions (Section 2.2) and nonconvex piecewise-linear lower semi-continuous penalty functions (Section 2.3.2). Here we employ the Sample Average Approximation (SAA) method (Kleywegt et al., 2001; Mak et al., 1999; Norkin et al., 1998) and Appendix A, an exterior sampling method designed to provide bounds on stochastic programs. In each case, we vary the sample size $N = |\Omega|$ to examine the tradeoff in narrowing the gap between computed statistical upper and lower bounds and increasing solution time by using relatively large values of N. We test all instances with sample sizes of $N = 50,\ 100$, and 200 scenarios. We use three and five penalty function segments, denoted as "3-segment" and "5-segment," respectively, for both convex and nonconvex penalty cases. We name all instances as **w-x-y-z**, where **w** = C or Nc corresponding to convex and nonconvex cases, respectively, and **x, y, z** are the number of function segments, number of nodes, and instance number, respectively.

For instance, Nc-5-30-3 is a nonconvex five-segment instance using the third 30-node network.

To generate the penalty functions, we first compute an original critical path length $\hat{u}_n^s$ according to the uninsured duration times d_{ij}^s. We compute the threshold values $\tau_1^s, \ldots, \tau_{K^s+1}^s$ by setting $\tau_i^s = (0.7 + 0.3(i-1)/K^s)\hat{u}_n^s$, $\forall i = 1, \ldots, K^s + 1$, so that $\tau_1^s = 0.7\hat{u}_n^s$, $\tau_{K^s+1}^s = \hat{u}_n^s$, and all intervals are evenly distributed. If completion time is within the target due time (i.e., less than τ_1^s), no penalty is incurred. The remainder of the penalty function is generated as follows.

For the convex penalty case, we first generate the left-hand-point function value $f_1^L = 0$ of segment 1, and then we generate an increasing series of slopes m_k^s of each segment k such that $0 < m_1^s < m_2^s < \cdots < m_{K^s}^s$ (we use $m_1^s = 1.2$, $m_k^s = 1.2m_{k-1}^s$ and $m_1^s = 1.1$, $m_k^s = 1.1m_{k-1}^s$ for the 3-segment and 5-segment cases, respectively, $\forall k = 2, \ldots, K^s$). We compute f_k^L at the left-hand-point of each segment k as $f_k^L = f_{k-1}^L + m_k^s(\tau_k - \tau_{k-1})$, $\forall k = 2, \ldots, K^s$, to ensure convexity of the penalty function. For the nonconvex case, f_1^L is uniformly distributed over the interval $[100, 400]$. We randomly generate the slopes m_k^s according to a uniform distribution over the interval $[0.2, 0.9]$. To ensure that the function is increasing, we set $f_k^L = 1.05(f_{k-1}^L + m_k^s(\tau_k^s - \tau_{k-1}^s))$, $\forall k = 2, \ldots, K^s$, and thus the increasing piecewise linear function becomes discontinuous and nonconvex.

For each instance, we use $M = 20$ as the number of samples, and the size of the reference sample is set to $N' = 10000$ scenarios. All decomposition algorithms are implemented using CPLEX 11.0 (ILOG, 2008) via ILOG Concert Technology 2.5, and all computations are performed on a SUN UltraSpace-III with a 900 MHz processor and 2.0 GB RAM. Computational times are reported in CPU seconds. We allow a one-hour (3600 seconds) time limit.

2.4.1.1 Computational results of expectation-based models

For the convex case, we use Cut 2–2, and for the nonconvex case, we compute Cut 2–13 using optimal dual values according to Proposition 2.1, and present the computational results in Tables 2-2 through 2-5. For these tables, t_{max}, t_{min}, and t_{avg} represent the maximum, minimum, and average CPU seconds for each instance over all $M = 20$ samples, respectively. If our algorithm fails to solve some sample within the time limit, we report "LIMIT" in t_{max}, and present the average CPU time of all solvable samples in t_{avg}. **Iter** and **Cuts** represent average number of times that we solve the master problem and the average number of Benders cuts 2–2 or 2–13 generated before achieving optimality, respectively. Columns labeled **LB** and **UB** denote the statistical lower bound and the best (minimum) upper bound of the optimal objective function value using the reference sample, respectively. **Gap** represents the difference between LB and UB as a percentage of the lower bound.

Comparing the results with respect to the convex and nonconvex cases, the latter requires more iterations and cuts generation, and thus increases the CPU time. We observe that the optimality gaps improve by increasing the sample size, N, of scenarios. However, increasing N also leads to an increase in CPU times and in cutting plane generation. For instance, using $N = 200$ scenarios, we reduce the optimality gaps associated with all fifteen instances to less than 1% in all computational experiments represented by Tables 2-2 through 2-5. The average number of cuts generated by each sample is approximately a linear function of scenarios, and the average CPU time increases by more than five times for some instances compared to the case of $N = 100$. Indeed, some samples of instances having 50 and 70 nodes cannot be solved within the time limit for the 5-segment nonconvex penalty case.

Remark 2.3. Since we only need to solve a critical path problem to obtain the Benders cutting planes required for our algorithm, we save significant computational effort compared to approaches that (a) directly solve the non-decomposed mixed integer

Table 2-2. Computational results for 3-segment convex instances.

$N = \|\Omega\|$	Instance	t_{max}	t_{min}	t_{avg}	Iter	Cuts	LB	UB	Gap (%)
	C-3-30-1	0.64	0.10	0.30	5.30	71.20	234.44	240.18	2.45
	C-3-30-2	0.65	0.07	0.26	4.50	58.10	44.81	47.20	5.33
	C-3-30-3	1.11	0.19	0.47	5.40	81.30	290.37	295.41	1.74
	C-3-30-4	1.13	0.19	0.59	5.65	72.80	285.46	289.24	1.32
	C-3-30-5	1.24	0.18	0.62	6.10	98.45	111.87	114.87	2.68
	C-3-50-1	3.28	0.27	1.14	5.30	74.20	167.22	172.14	2.94
	C-3-50-2	2.70	0.17	0.99	4.85	58.10	57.15	59.10	3.41
$N = 50$	C-3-50-3	3.85	0.29	1.56	5.05	95.05	188.03	191.18	1.68
	C-3-50-4	2.42	0.22	0.94	4.25	56.10	94.79	96.15	1.43
	C-3-50-5	3.21	0.25	1.16	5.15	76.30	170.37	174.90	2.66
	C-3-70-1	5.53	0.99	2.63	5.20	97.40	352.36	356.34	1.13
	C-3-70-2	5.45	1.00	2.52	5.30	107.60	311.12	319.69	2.75
	C-3-70-3	5.59	0.96	2.62	5.30	114.15	157.63	158.12	0.31
	C-3-70-4	4.47	0.84	2.03	4.95	72.50	239.35	245.89	2.73
	C-3-70-5	4.56	0.94	2.30	5.10	80.05	193.19	196.00	1.45
	C-3-30-1	1.25	0.23	0.55	4.95	139.75	238.89	242.07	1.33
	C-3-30-2	1.27	0.11	0.52	4.60	124.20	46.05	47.13	2.35
	C-3-30-3	2.08	0.27	0.96	5.30	161.45	290.25	293.41	1.09
	C-3-30-4	2.35	0.31	1.14	5.45	146.60	290.08	290.95	0.30
	C-3-30-5	2.64	0.30	1.39	6.00	194.70	108.50	110.77	2.09
	C-3-50-1	6.48	0.42	1.98	5.35	141.10	162.23	166.07	2.37
	C-3-50-2	4.85	0.31	1.77	4.80	114.25	57.55	58.10	0.96
$N = 100$	C-3-50-3	7.26	0.39	2.52	5.00	169.55	190.37	191.18	0.43
	C-3-50-4	4.46	0.31	1.68	4.05	110.65	93.56	94.54	1.05
	C-3-50-5	6.04	0.41	1.97	5.30	148.40	173.49	174.90	0.81
	C-3-70-1	15.40	2.25	6.86	5.25	181.65	353.47	356.34	0.81
	C-3-70-2	15.28	2.18	6.70	5.65	202.95	312.14	316.70	1.46
	C-3-70-3	17.01	2.23	7.86	5.35	217.50	155.06	156.97	1.23
	C-3-70-4	13.25	1.90	5.33	4.85	144.50	239.37	242.56	1.33
	C-3-70-5	14.23	1.97	6.11	5.05	155.00	193.33	194.47	0.59
	C-3-30-1	2.10	0.44	0.96	4.35	230.35	241.08	242.07	0.41
	C-3-30-2	3.08	0.24	1.12	4.25	241.25	46.81	47.13	0.68
	C-3-30-3	4.35	0.42	1.91	5.15	320.30	292.25	292.68	0.15
	C-3-30-4	5.06	0.42	1.94	5.40	292.60	290.72	290.95	0.08
	C-3-30-5	5.00	0.45	2.40	5.95	322.75	109.73	110.77	0.95
	C-3-50-1	12.47	0.65	4.74	5.20	263.35	162.34	163.12	0.48
	C-3-50-2	9.31	0.55	4.08	4.55	218.05	57.91	58.10	0.33
$N = 200$	C-3-50-3	13.42	0.60	4.91	5.10	322.45	190.26	190.72	0.24
	C-3-50-4	8.15	0.55	3.91	4.25	208.95	93.95	94.54	0.63
	C-3-50-5	11.89	0.73	4.81	5.15	279.50	174.52	174.63	0.06
	C-3-70-1	21.40	4.02	10.39	4.80	339.65	353.91	356.34	0.69
	C-3-70-2	23.03	4.23	9.60	5.15	397.55	314.77	316.70	0.61
	C-3-70-3	23.87	4.10	11.33	5.30	409.50	155.96	156.97	0.65
	C-3-70-4	17.52	3.28	7.54	4.85	280.35	241.21	242.56	0.56
	C-3-70-5	19.49	3.49	8.57	5.05	317.60	193.98	194.47	0.25

Table 2-3. Computational results for 5-segment convex instances.

$N = \|\Omega\|$	Instance	t_{max}	t_{min}	t_{avg}	Iter	Cuts	LB	UB	Gap (%)
	C-5-30-1	1.29	0.23	0.59	5.20	70.05	267.40	274.41	2.62
	C-5-30-2	1.27	0.24	0.48	4.00	58.05	78.36	80.32	2.50
	C-5-30-3	1.54	0.32	0.64	5.25	75.15	317.95	324.35	2.01
	C-5-30-4	1.50	0.28	0.48	4.40	67.30	328.31	338.09	2.98
	C-5-30-5	1.86	0.31	0.94	6.00	82.75	158.84	163.09	2.68
	C-5-50-1	4.67	0.48	1.44	5.10	71.85	148.53	152.42	2.62
	C-5-50-2	5.09	0.30	1.05	4.35	56.35	73.35	75.06	2.33
$N = 50$	C-5-50-3	8.49	0.58	1.66	5.00	87.10	187.97	191.29	1.77
	C-5-50-4	4.37	0.42	1.22	4.50	61.95	134.84	138.46	2.68
	C-5-50-5	6.56	0.47	2.31	5.20	73.30	164.25	168.91	2.84
	C-5-70-1	9.40	1.53	3.84	5.05	97.15	354.36	361.41	1.99
	C-5-70-2	10.71	1.40	3.97	5.25	103.05	316.45	323.53	2.24
	C-5-70-3	11.49	1.55	3.86	5.40	112.00	191.31	191.20	-0.06
	C-5-70-4	7.90	1.00	3.03	4.90	70.15	232.47	238.96	2.79
	C-5-70-5	8.77	1.13	3.11	5.05	78.45	230.65	235.73	2.20
	C-5-30-1	2.69	0.29	1.43	4.95	142.50	266.95	270.98	1.51
	C-5-30-2	2.48	0.30	1.17	4.05	119.30	73.08	74.53	1.98
	C-5-30-3	2.80	0.41	1.28	5.35	142.55	321.47	324.35	0.90
	C-5-30-4	2.62	0.39	1.00	4.80	135.25	332.40	334.51	0.63
	C-5-30-5	3.34	0.49	2.07	6.00	169.65	155.75	157.96	1.42
	C-5-50-1	9.77	0.92	3.04	5.20	140.45	145.86	148.37	1.72
	C-5-50-2	9.16	0.53	2.49	4.40	112.30	73.97	75.06	1.47
$N = 100$	C-5-50-3	16.47	1.07	4.38	5.15	170.50	185.38	186.95	0.85
	C-5-50-4	8.74	0.63	2.06	4.65	117.30	134.25	136.14	1.41
	C-5-50-5	13.18	0.87	3.65	5.25	141.60	166.40	168.91	1.51
	C-5-70-1	19.63	2.81	7.34	5.20	184.65	356.45	361.41	1.39
	C-5-70-2	19.92	3.35	8.44	5.20	203.55	312.72	317.67	1.58
	C-5-70-3	23.92	2.84	7.91	5.45	216.85	190.81	191.20	0.20
	C-5-70-4	17.72	1.93	5.74	4.95	135.20	231.41	235.03	1.56
	C-5-70-5	18.25	2.00	5.88	5.10	156.70	231.37	235.73	1.88
	C-5-30-1	7.90	1.23	3.26	4.90	238.65	270.01	270.98	0.36
	C-5-30-2	6.04	0.77	3.09	4.25	238.15	73.19	73.35	0.22
	C-5-30-3	7.32	0.93	3.44	5.00	280.65	324.22	324.35	0.04
	C-5-30-4	6.97	0.89	3.11	4.95	268.35	334.08	334.51	0.13
	C-5-30-5	9.73	1.25	5.24	5.90	328.10	157.05	157.96	0.58
	C-5-50-1	26.10	1.87	8.82	5.10	273.50	148.09	148.37	0.19
	C-5-50-2	24.01	1.65	7.32	4.35	214.65	73.39	73.88	0.67
$N = 200$	C-5-50-3	41.48	1.92	9.32	4.95	331.60	185.00	185.16	0.09
	C-5-50-4	23.23	1.25	7.34	4.45	212.75	134.96	135.24	0.21
	C-5-50-5	37.01	2.57	10.12	5.05	269.10	168.70	168.91	0.12
	C-5-70-1	82.86	7.95	21.40	4.95	346.60	359.40	360.09	0.19
	C-5-70-2	79.58	8.78	24.15	5.00	389.85	311.58	314.50	0.94
	C-5-70-3	95.13	8.29	22.98	5.25	407.95	190.94	191.20	0.14
	C-5-70-4	59.26	6.58	17.42	4.75	256.10	231.95	232.86	0.39
	C-5-70-5	63.72	6.21	18.29	5.00	286.70	231.72	233.24	0.66

Table 2-4. Computational results for 3-segment nonconvex instances.

$N = \|\Omega\|$	Instance	t_{max}	t_{min}	t_{avg}	Iter	Cuts	LB	UB	Gap (%)
	Nc-3-30-1	95.14	4.02	20.47	30.10	1229.40	514.66	518.32	0.71
	Nc-3-30-2	82.59	2.83	17.55	27.55	1083.50	312.70	317.21	1.44
	Nc-3-30-3	120.42	6.43	27.68	34.20	1399.90	569.97	569.70	-0.05
	Nc-3-30-4	153.03	6.27	32.88	35.40	1406.70	553.62	559.21	1.01
	Nc-3-30-5	172.30	7.91	35.62	34.75	1483.55	378.65	383.32	1.23
	Nc-3-50-1	200.23	8.65	37.80	29.35	1237.45	436.45	441.34	1.12
	Nc-3-50-2	154.37	4.82	30.45	27.10	1006.20	323.29	326.95	1.13
$N = 50$	Nc-3-50-3	260.59	9.14	43.68	34.55	1348.25	436.31	441.64	1.22
	Nc-3-50-4	152.45	5.03	29.87	26.90	1012.80	400.65	406.00	1.34
	Nc-3-50-5	229.80	8.98	40.57	31.45	1246.95	410.17	412.81	0.64
	Nc-3-70-1	87.43	19.48	46.33	33.55	1263.60	615.83	621.64	0.94
	Nc-3-70-2	89.37	17.25	44.79	34.80	1309.45	589.02	601.37	2.10
	Nc-3-70-3	82.56	14.31	41.11	33.15	1254.35	439.15	445.72	1.50
	Nc-3-70-4	77.83	13.96	32.87	30.65	1196.75	481.99	490.46	1.76
	Nc-3-70-5	74.70	10.23	35.64	30.90	1188.50	480.51	486.85	1.32
	Nc-3-30-1	254.35	11.89	58.99	29.55	2394.45	516.87	518.32	0.28
	Nc-3-30-2	222.49	10.21	34.83	27.40	2090.20	314.56	316.89	0.74
	Nc-3-30-3	370.88	15.89	69.56	33.50	2553.45	568.66	569.70	0.18
	Nc-3-30-4	346.90	14.37	79.18	34.95	2798.05	557.81	559.21	0.25
	Nc-3-30-5	403.64	17.65	87.68	33.85	2800.15	381.49	383.01	0.40
	Nc-3-50-1	437.24	20.67	74.26	29.20	2321.40	432.50	436.01	0.81
	Nc-3-50-2	342.95	14.30	62.48	26.50	1999.60	321.64	323.83	0.68
$N = 100$	Nc-3-50-3	528.47	21.52	83.20	33.25	2596.70	435.42	439.67	0.98
	Nc-3-50-4	326.00	14.21	59.61	27.15	2007.85	400.12	403.60	0.87
	Nc-3-50-5	495.38	19.73	81.77	30.80	2485.65	410.56	412.81	0.55
	Nc-3-70-1	306.38	34.62	167.82	32.60	2410.90	617.29	619.21	0.31
	Nc-3-70-2	357.65	37.28	178.45	33.75	2568.30	587.26	591.28	0.68
	Nc-3-70-3	317.68	35.91	157.18	32.85	2419.35	441.68	445.72	0.91
	Nc-3-70-4	299.14	25.04	139.21	30.20	2398.70	485.12	490.46	1.10
	Nc-3-70-5	276.53	26.73	133.69	31.05	2359.05	479.55	483.04	0.73
	Nc-3-30-1	942.50	22.80	206.10	30.35	5024.85	516.19	516.28	0.02
	Nc-3-30-2	852.00	28.69	90.62	26.90	4043.20	314.87	315.02	0.05
	Nc-3-30-3	999.57	20.17	200.14	34.00	4977.50	568.29	568.36	0.01
	Nc-3-30-4	1043.68	29.22	235.53	33.95	5431.10	558.04	558.28	0.04
	Nc-3-30-5	1125.41	36.35	289.64	34.30	5596.55	380.25	380.57	0.08
	Nc-3-50-1	1192.49	61.24	200.12	30.00	4500.80	434.50	435.24	0.17
	Nc-3-50-2	967.67	43.37	177.95	26.85	4040.15	323.23	323.83	0.19
$N = 200$	Nc-3-50-3	1521.38	59.41	239.02	32.65	4901.40	437.28	437.56	0.06
	Nc-3-50-4	910.55	37.69	169.34	27.55	3960.35	402.79	403.60	0.20
	Nc-3-50-5	1327.53	53.20	212.68	31.20	4652.65	412.59	412.81	0.05
	Nc-3-70-1	1278.52	98.42	459.77	33.10	4733.85	619.09	621.64	0.41
	Nc-3-70-2	1430.22	110.27	490.36	33.95	4902.15	587.63	591.28	0.62
	Nc-3-70-3	1350.06	106.52	445.80	32.35	4782.45	443.17	443.95	0.18
	Nc-3-70-4	1101.30	77.78	401.22	30.00	4459.25	487.84	490.46	0.54
	Nc-3-70-5	1062.79	81.93	392.76	30.75	4507.30	481.28	483.04	0.37

Table 2-5. Computational results for 5-segment nonconvex instances.

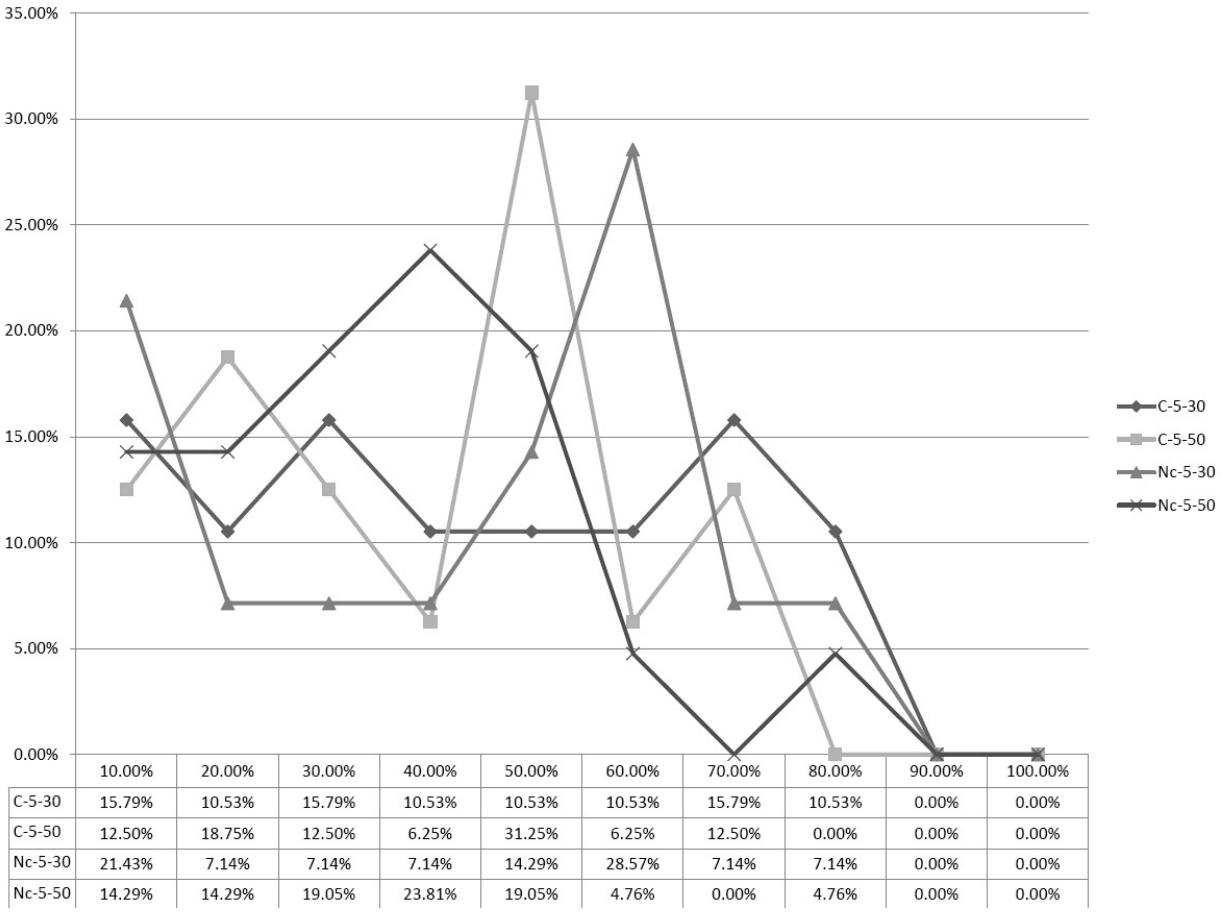
$N = \|\Omega\|$	Instance	t_{max}	t_{min}	t_{avg}	Iter	Cuts	LB	UB	Gap (%)
	Nc-5-30-1	276.29	4.90	61.62	38.85	1646.05	520.26	525.01	0.91
	Nc-5-30-2	222.87	3.76	53.27	30.25	1521.35	316.33	319.62	1.04
	Nc-5-30-3	298.64	5.12	68.36	39.35	1788.50	570.43	575.90	0.96
	Nc-5-30-4	336.78	6.22	82.19	38.10	1893.05	559.66	563.63	0.71
	Nc-5-30-5	359.63	7.03	88.74	40.35	2001.50	382.94	386.12	0.83
	Nc-5-50-1	594.37	8.28	178.65	42.65	1865.25	438.92	443.91	1.14
	Nc-5-50-2	637.69	5.04	100.30	34.30	1675.40	321.45	325.64	1.30
$N = 50$	Nc-5-50-3	787.42	9.54	189.61	45.75	2011.50	442.37	446.88	1.02
	Nc-5-50-4	435.62	5.56	93.17	43.00	1724.65	404.68	409.53	1.20
	Nc-5-50-5	677.34	8.95	143.55	43.70	2189.75	415.80	421.01	1.25
	Nc-5-70-1	1009.37	20.96	209.77	54.30	2219.55	620.17	625.58	0.87
	Nc-5-70-2	1191.64	23.58	219.47	56.65	2403.85	592.54	609.01	2.78
	Nc-5-70-3	1006.52	21.04	203.61	53.30	2283.35	479.38	485.27	1.23
	Nc-5-70-4	978.43	18.72	184.65	50.55	2154.40	475.24	483.33	1.70
	Nc-5-70-5	925.80	19.25	176.19	51.65	2199.15	480.28	482.87	0.54
	Nc-5-30-1	561.99	12.63	158.95	39.80	3250.20	517.42	523.42	1.16
	Nc-5-30-2	453.23	7.29	100.11	32.10	3008.65	318.21	319.29	0.34
	Nc-5-30-3	594.84	10.11	123.90	39.10	3506.90	574.85	575.90	0.18
	Nc-5-30-4	627.81	11.30	156.47	40.05	3582.95	563.89	565.31	0.25
	Nc-5-30-5	700.55	13.98	162.44	41.15	3996.85	385.81	386.58	0.20
	Nc-5-50-1	1236.28	22.91	394.87	43.10	3788.25	437.97	441.62	0.83
	Nc-5-50-2	1485.67	16.39	269.48	34.00	3349.60	322.96	325.64	0.83
$N = 100$	Nc-5-50-3	1732.10	25.88	413.26	45.65	4211.50	444.34	446.88	0.57
	Nc-5-50-4	1323.14	15.64	221.59	41.35	3405.60	402.38	405.15	0.69
	Nc-5-50-5	1568.62	24.30	347.63	42.10	4300.20	417.17	419.73	0.61
	Nc-5-70-1	LIMIT	237.45	1617.25	53.30	4183.25	621.85	625.58	0.60
	Nc-5-70-2	LIMIT	272.37	1638.91	56.70	4297.60	597.88	605.35	1.25
	Nc-5-70-3	LIMIT	214.58	1594.32	53.90	4164.30	477.12	481.23	0.86
	Nc-5-70-4	2947.81	190.46	1323.96	50.20	4005.95	475.98	483.33	1.54
	Nc-5-70-5	3245.89	192.31	1345.88	51.80	4017.30	477.84	480.29	0.51
	Nc-5-30-1	1629.64	49.53	394.75	35.50	6024.40	523.31	523.42	0.02
	Nc-5-30-2	1299.30	27.41	293.64	30.15	5996.70	318.80	319.29	0.15
	Nc-5-30-3	1701.52	31.27	360.11	36.35	6893.60	573.84	574.55	0.12
	Nc-5-30-4	1805.13	30.06	407.82	37.45	7018.25	564.13	564.37	0.04
	Nc-5-30-5	1950.68	33.28	459.35	38.70	7845.80	384.56	384.88	0.08
	Nc-5-50-1	3468.23	81.24	874.52	41.95	7412.65	437.65	440.21	0.58
	Nc-5-50-2	LIMIT	63.59	637.95	33.65	6700.15	323.00	323.93	0.29
$N = 200$	Nc-5-50-3	LIMIT	96.72	903.21	44.85	8395.30	446.47	446.88	0.09
	Nc-5-50-4	3329.45	60.47	502.13	40.05	6765.20	404.11	405.15	0.26
	Nc-5-50-5	3569.70	89.64	797.54	41.35	8527.40	418.34	419.73	0.33
	Nc-5-70-1	LIMIT	194.52	2002.26	52.95	7059.35	622.95	623.70	0.12
	Nc-5-70-2	LIMIT	188.64	2014.85	56.15	7446.90	604.73	609.01	0.71
	Nc-5-70-3	LIMIT	197.49	1972.58	53.75	7232.15	480.52	481.23	0.15
	Nc-5-70-4	LIMIT	177.31	1674.19	50.95	6884.50	478.36	480.45	0.44
	Nc-5-70-5	LIMIT	170.55	1625.74	51.45	6832.15	479.64	480.29	0.14

model, or (b) decompose the model but explicitly solve subproblems SP-LS$^s(\hat{x})$-RLT by linear programming to obtain the duals (as opposed to our dual recovery procedure given in Proposition 2.1). To illustrate the computational importance of our approach, we solved instance **Nc-3-30-1** with 50 scenarios by applying CPLEX to the non-decomposed mixed integer model. Each of the $M = 20$ samples took at least six CPU hours to solve using this approach, compared with an average of 20.47 seconds using our decomposition methodology. Using the same instance, we also decomposed the problem and solved the RLT-enhanced subproblem SP-LS$^s(\hat{x})$-RLT by linear programming rather than by our dual recovery technique, and none of the 20 samples were solved within the one-hour time limit.

2.4.1.2 Analysis of insured arc characteristics

In this part, we provide insights pertaining to optimal solutions obtained by our expectation-based STIP models. Let $\bar{d}_{ij}$ and $\bar{g}_{ij}$ respectively denote average uninsured and insured task durations of arc (i,j) over all scenarios; thus, $\bar{d}_{ij} - \bar{g}_{ij}$ represents the average duration-reduction value for arc (i,j) due to its insurance. Here, we examine the extent to which small ratios of an arc's cost-to-duration-reduction ratio $c_{ij}/(\bar{d}_{ij} - \bar{g}_{ij})$ influences whether or not the arc will be insured in the optimal solution we obtain. For a given instance, we order all arcs $(i,j) \in \mathcal{A}$ in nondecreasing order of their $c_{ij}/(\bar{d}_{ij} - \bar{g}_{ij})$ values, and examine the frequency in which arcs at different portions of this spectrum are insured in the obtained optimal solution.

We conduct this experiment on all **w-5-30-z** and **w-5-50-z** instances. In Figure 2-3, the arcs are partitioned into groups such that the top 10% of arcs ordered as above belong to the first group (labeled "10%"), followed by the next top 10% of arcs in the second group (labeled "20%"), and so on. These groups are depicted on the horizontal axis, and the vertical axis represents the percentage of arcs in each of the ten groups that are insured in the optimal solution obtained. We see that arcs (i,j) having very high values of $c_{ij}/(\bar{d}_{ij} - \bar{g}_{ij})$ relative to other arcs' ratios are not likely to be insured. Indeed,



	10.00%	20.00%	30.00%	40.00%	50.00%	60.00%	70.00%	80.00%	90.00%	100.00%
C-5-30	15.79%	10.53%	15.79%	10.53%	10.53%	10.53%	15.79%	10.53%	0.00%	0.00%
C-5-50	12.50%	18.75%	12.50%	6.25%	31.25%	6.25%	12.50%	0.00%	0.00%	0.00%
Nc-5-30	21.43%	7.14%	7.14%	7.14%	14.29%	28.57%	7.14%	7.14%	0.00%	0.00%
Nc-5-50	14.29%	14.29%	19.05%	23.81%	19.05%	4.76%	0.00%	4.76%	0.00%	0.00%

Figure 2-3. Arc-insuring trend with increasing values of $c/(\bar{d} - \bar{g})$.

no arcs in the upper "20%" of cost-to-duration-ratio were insured in optimal solutions to any of the instances tested here. However, no trend is evident regarding which of the remaining arcs will be selected in an optimal solution. This underscores the difficulty of the problem and the necessity of using sophisticated approaches for their solution.

2.4.1.3 The persistency of the first-stage optimal solution

In this part, we test the notion that one may be able to anticipate which arcs will be insured at optimality by solving a series of deterministic task-insurance instances, one corresponding to each possible scenario. Specifically, for each scenario $s \in \Omega$, we could solve a deterministic problem as $\min\{cx + f(\psi(x, \xi^s)) : x \in \{0,1\}^{|A|}\}$, and obtain its optimal first-stage solution as $x^*(\xi^s)$. For each $(i,j) \in A$, we then compute the percentage of these $|\Omega|$ instances in which (i,j) is insured (i.e., given

by $(\sum_{s\in\Omega} x_{ij}^*(\xi^s))/|\Omega|$). The arcs that are insured with high frequency are said to be persistent. A closely related study was published by Bertsimas et al. (2006) for computing the persistency of binary variables (i.e., the probability that the variable will equal 1 at optimality) in discrete optimization problems under objective uncertainty with only partial information on the distribution of the objective coefficients. Here we empirically investigate whether persistent arcs correspond to those that are insured in the optimal STIP solution.

We test this hypothesis on instances Nc-3-30-1 and Nc-5-30-1 with a sample size of $N = 200$, and present the results in Table 2-6. The top row, labeled *Arc No.*, gives the labels of arcs that were insured in at least one deterministic task-insurance instance (i.e., $x_{ij}^*(\xi^s) = 1$ for some $s \in \Omega$). The rows for Nc-3-30-1 and Nc-5-30-1 state the number of times that each arc appears in a deterministic task-insurance solution (out of 200 scenarios). For instance, for Nc-3-30-1, there are 50 scenarios in which arc 5 is insured out of the 200 deterministic task-insurance instances, and for Nc-5-30-1, there are 53 such scenarios in which arc 5 is insured. The arcs insured in the (unique, in both cases) optimal STIP solution we obtain are marked with * in each row. (Arcs not depicted in Table 2-6 were not insured in the optimal STIP solution.)

Table 2-6. Illustration of first-stage solution persistency.

Arc No.	5	129	169	197	209	257	321	329	331
Nc-3-30-1	50	52	56	10	*54	76	58	67	*47
Nc-5-30-1	53	81	*121	5	*153	97	33	*93	65

*: Arc was insured at optimality.

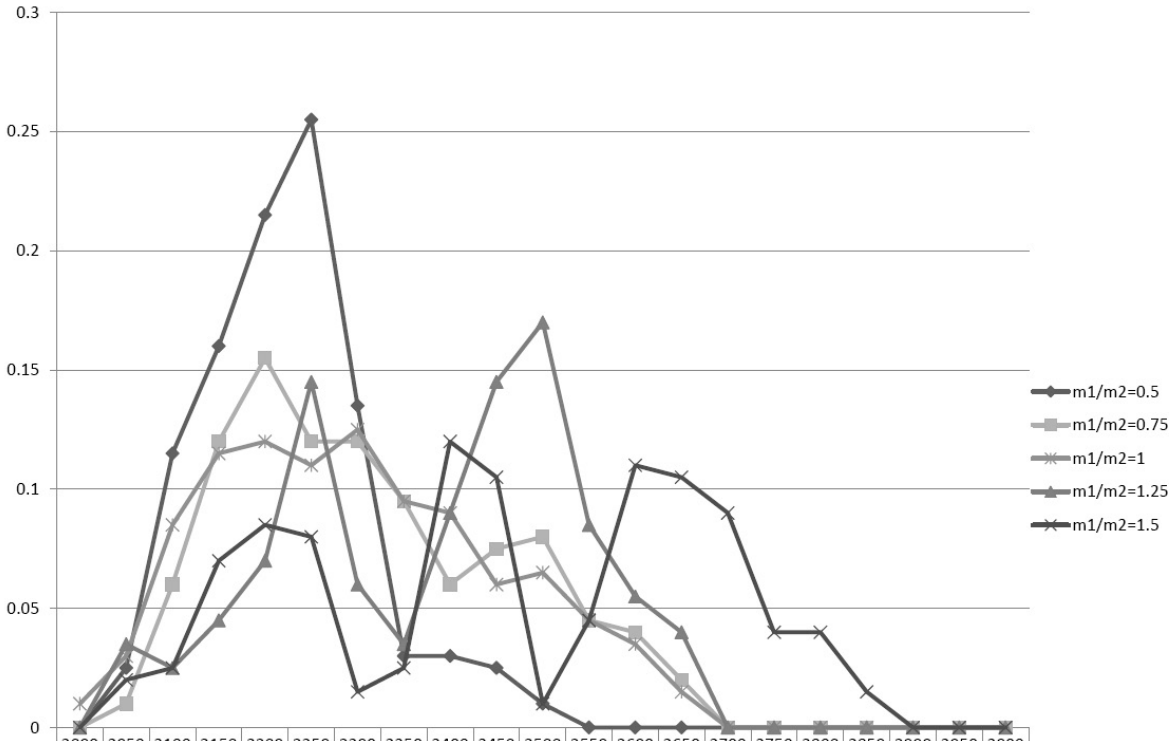
Observe that optimality of persistent arcs does not hold in general, in the sense that arcs insured in a high percentage of task-insurance instances do not necessarily appear in the optimal STIP solution. For instance, in Nc-3-30-1, arc 257 is insured in 76 scenarios (more than any other arc), but is not insured in the optimal STIP solution; in fact, none of the four most-frequently-insured arcs in the row for Nc-3-30-1 are insured in

the optimal STIP solution. However, Nc-5-30-1 displays a stronger correlation, in which three out of the top four most-frequently-insured arcs are insured in the optimal STIP solution.

2.4.1.4 The critical path length distribution

We also analyze the distribution of critical path lengths given different forms of the penalty function. In this experiment, we obtain an optimal solution x^*, compute $\psi^s(x^*)$ for each scenario $s \in \Omega$, and approximate the distribution of the critical path lengths with respect to different penalty functions. We consider the first 50-node graph in our data set, use a sample size of $N = 200$, and examine various two-segment continuous piecewise-linear penalty functions. Each penalty function has slope $m_1 = 1$ for the first segment, which has a penalty of 0 when the critical path length is 0. The second piece of the function begins when the critical path length equals 2300 (with a penalty of 2300), and has slope m_2. We consider ratio values m_1/m_2 in the set $\{0.5, 0.75, 1, 1.25, 1.5\}$, so that the first two ratios give a nonlinear convex penalty function, the ratio of 1 yields a linear penalty function, and the last two values give a nonlinear concave penalty function. We optimize STIP given each penalty function, and plot distributions of the resulting critical path lengths over the 200 scenarios in Figure 2-4. In particular, for each horizontal segment labeled with value t, the plot gives the percentage of 200 scenarios that have critical path length in the interval $[t, t + 50)$.

Figure 2-4 demonstrates that for functions having smaller ratios of m_1/m_2, the critical path lengths tend to be shorter on average, and exhibit smaller standard deviations compared to the critical path length distributions for functions having larger ratios of m_1/m_2. The actual means for critical path lengths given penalty functions having m_1/m_2 ratios of 0.5, 0.75, 1, 1.25, and 1.5 are 2247, 2328, 2319, 2409, and 2483, respectively, and their standard deviations are 90, 132, 152, 158, and 215, respectively. This result is intuitive, because a sharply increasing penalty function essentially acts as a barrier function and limits the number of scenarios in which the



	2000	2050	2100	2150	2200	2250	2300	2350	2400	2450	2500	2550	2600	2650	2700	2750	2800	2850	2900	2950	3000
m1/m2=0.5	0	0.025	0.115	0.16	0.215	0.255	0.135	0.03	0.03	0.025	0.01	0	0	0	0	0	0	0	0	0	0
m1/m2=0.75	0	0.01	0.06	0.12	0.155	0.12	0.12	0.095	0.06	0.075	0.08	0.045	0.04	0.02	0	0	0	0	0	0	0
m1/m2=1	0.01	0.03	0.085	0.115	0.12	0.11	0.125	0.095	0.09	0.06	0.065	0.045	0.035	0.015	0	0	0	0	0	0	0
m1/m2=1.25	0	0.035	0.025	0.045	0.07	0.145	0.06	0.035	0.09	0.145	0.17	0.085	0.055	0.04	0	0	0	0	0	0	0
m1/m2=1.5	0	0.02	0.025	0.07	0.085	0.08	0.015	0.025	0.12	0.105	0.01	0.045	0.11	0.105	0.09	0.04	0.04	0.015	0	0	0

Figure 2-4. Distributions of the critical path lengths given different penalty functions.

critical path length is allowed to become large. By contrast, when m_1/m_2 is large, and the marginal cost of finishing the project very late becomes relatively small, the distribution functions tend to spread out across the spectrum of possible completion times.

2.4.2 Chance-constrained Formulation Case

Regarding problem CC_ϵ, when $\epsilon = 0$, one can use a scenario approximation method to solve $\mathrm{CC}_{\epsilon=0}$ by solving the following approximation problem based on an independent Monte Carlo sample of random vectors $\xi^1, \ldots, \xi^N$:

$$\min \left\{ \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \ : \ x \in X, \ \psi(x, \xi^s) \leq \mathcal{T}^s \quad \forall s = 1, \ldots, N \right\}. \tag{2–24}$$

Luedtke & Ahmed (2008) approximate CC_ϵ with a general $\epsilon \geq 0$ by solving a sample approximation problem. Let $\xi^1, \ldots, \xi^N$ be an independent Monte Carlo sample of the random vector ξ, and for a fixed $\alpha \in [0, 1)$, consider the following sample approximation problem

$$\mathbf{CC}_\alpha^N\mathbf{:}\ \hat{z}_\alpha^N = \min \left\{ \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} : x \in X_\alpha^N \right\} \tag{2–25}$$

where

$$X_\alpha^N = \left\{ x \in X : \frac{1}{N} \sum_{s=1}^{N} \mathbb{I}(\psi(x, \xi^s) \leq \mathcal{T}^s) \geq 1 - \alpha \right\}. \tag{2–26}$$

Given ξ^s and first-stage binary variables $\hat{x}$, $\psi(\hat{x}, \xi^s)$ is given by $CPM^s(\hat{x})$. Note that when $\alpha = 0$, the sample approximation problems 2–25 and 2–26 are equivalent to the scenario approximation program 2–24. We examine in Section 2.4.2.1 the case in which CC_α^N yields feasible solutions for CC_ϵ, and then discuss in Section 2.4.2.2 how to determine lower bounds with different confidences when $\alpha = \epsilon$.

Here we only test the first instance of each graph size, named as **y-1**, where **y** represents the number of nodes (30, 50, or 70). We generate $\mathcal{T}^s$ from a uniform integer distribution over the interval $[0.7\hat{u}_n^s, 0.9\hat{u}_n^s]$. We again generate integer arc-insurance costs c_{ij}, $(i, j) \in \mathcal{A}$, uniformly over the interval $[25, 50]$.

2.4.2.1 Feasible solutions for CC_α^N

For a fixed value of $\alpha < \epsilon$, we wish to obtain a feasible solution to CC_ϵ with probability at least $1 - \delta$, for $\delta \in (0, 1)$. Since our feasible region X is finite, the result of Theorem 5 in Luedtke & Ahmed (2008) shows that it is sufficient to find a feasible solution to CC_α^N satisfying

$$N \geq \frac{1}{2(\alpha - \epsilon)^2} \log \frac{1}{\delta} + \frac{n}{2(\alpha - \epsilon)^2} \log U, \tag{2–27}$$

where in particular, when $\alpha = 0$, Theorem 7 suggests a sample size of

$$N \geq \frac{1}{\epsilon} \log \frac{1}{\delta} + \frac{n}{\epsilon} \log U, \tag{2–28}$$

and U is such that the number of feasible solutions obeys $|X| \leq U^n$. In this case, since $x \in X \subseteq \{0,1\}^n$, we use $U = 2$.

We choose $\delta = 0.001$, $M = 10$ samples, and a reference sample size of $N' = 10000$ scenarios for all three instances. We consider the cases of $\epsilon = 0.01$ and $\epsilon = 0.005$, and use $\alpha = 0$ and $\alpha = \epsilon$. Based on Ineq. 2–27 and 2–28, when $\alpha = 0$, we use $N = 50,\ 80$, and 100 for the case with $\epsilon = 0.01$, and $N = 100,\ 150$, and 200 for the case with $\epsilon = 0.005$. When $\alpha = \epsilon$, we use sample sizes of $N = 1000$ and 2000 scenarios. Next, we define the *violation risk* of a solution $\hat{x}$ as the percentage of reference scenarios for which the critical path (given insurance decisions $\hat{x}$) exceeds $\mathcal{T}^s$. We say that a solution is feasible if its violation risk does not exceed ϵ. Tables 2-7 and 2-8 report the objective function values of generated feasible solutions. In these tables t_{avg} and **Num** denote the average solution time and the total number of feasible solutions over all 10 samples of each instance, respectively.

Table 2-7. Feasible solution results for CC_ϵ sample problems with $\epsilon = 0.01$.

α	$N = \lvert\Omega\rvert$	Instances	t_{avg}	Solution Violation			Feasible Solution Costs			
				Max	Min	Avg	Num	Max	Min	Avg
$\alpha = 0$	50	30-1	0.27	0.103	0.003	0.037	2	159	154	156.50
		50-1	0.42	0.124	0.007	0.041	1	212	212	212
		70-1	1.23	0.152	0.012	0.052	0	-	-	-
	80	30-1	0.39	0.052	0.000	0.019	3	159	154	156.33
		50-1	0.67	0.076	0.000	0.024	2	212	209	210.50
		70-1	1.78	0.092	0.001	0.031	2	234	232	233.00
	100	30-1	0.51	0.032	0.000	0.012	4	153	150	151.00
		50-1	0.78	0.038	0.000	0.017	4	209	207	208.00
		70-1	1.94	0.041	0.000	0.022	3	234	230	232.00
$\alpha = \epsilon$	1000	30-1	2.58	0.041	0.000	0.012	6	143	140	141.17
		50-1	8.98	0.057	0.002	0.015	4	195	190	192.25
		70-1	63.99	0.023	0.005	0.013	4	223	220	221.50
	2000	30-1	8.29	0.018	0.000	0.004	8	140	140	140.00
		50-1	23.81	0.041	0.001	0.008	7	195	190	190.71
		70-1	93.28	0.023	0.000	0.007	7	223	220	221.29

Table 2-8. Feasible solution results for CC_ϵ sample problems with $\epsilon = 0.005$.

α	$N = \lvert\Omega\rvert$	Instances	t_{avg}	Solution Violation			Feasible Solution Costs			
				Max	Min	Avg	Num	Max	Min	Avg
$\alpha = 0$	100	30-1	0.51	0.032	0.000	0.012	2	153	151	152.00
		50-1	0.78	0.038	0.000	0.017	2	209	209	209.00
		70-1	1.94	0.041	0.000	0.022	1	234	234	234.00
	150	30-1	0.68	0.030	0.000	0.010	2	153	151	152.00
		50-1	1.32	0.031	0.000	0.009	3	209	207	208.33
		70-1	3.29	0.037	0.000	0.014	2	234	234	234.00
	200	30-1	1.21	0.025	0.000	0.007	4	153	151	151.50
		50-1	2.06	0.029	0.000	0.007	6	210	209	209.17
		70-1	5.46	0.032	0.000	0.009	5	234	234	234.00
$\alpha = \epsilon$	1000	30-1	4.13	0.013	0.000	0.006	7	151	151	151.00
		50-1	9.36	0.021	0.000	0.013	9	205	203	203.22
		70-1	109.23	0.017	0.001	0.004	8	227	227	227.00
	2000	30-1	11.09	0.007	0.000	0.002	9	151	151	151.00
		50-1	27.28	0.004	0.000	0.001	10	203	203	203.00
		70-1	132.56	0.008	0.001	0.002	9	227	227	227.00

In Table 2-7, considering instance **30-1**, when $\alpha = 0$ and $N = 50$, the minimum violation among all solutions given by the $M = 10$ samples is $0.003 < \epsilon = 0.01$, and thus it is a feasible solution. The maximum violation risk among these samples is 0.103. Two out of 10 samples yield feasible solutions with objective values 159 and 154. With N increasing to 80 and 100, the number of feasible solutions increases to three and four, respectively. When $\alpha = \epsilon = 0.01,\ N = 1000$, the minimum violation risk of all solutions is zero, and the maximum violation risk is 0.041 (which is not feasible). The number of feasible solutions increases to six. By setting $N = 2000$, there are eight feasible solutions, all of which yield an objective value of 140. In Table 2-8, we set $\epsilon = 0.005$, and **30-1** yields more feasible solutions with better solution quality in each combination of α and N. Thus, by using $\alpha = \epsilon$, all instances yield more feasible solutions compared with the case of using $\alpha = 0$. However, more computational time is required to solve each instance's samples, because Ineq. 2–27 requires larger values of N when $\alpha = \epsilon$. On the

other hand, by decreasing the value of ϵ, we obtain more feasible solutions with higher solution quality in each setting.

2.4.2.2 Lower bounds for CC_α^N

Theorem 4 of Luedtke & Ahmed (2008) provides a mechanism for obtaining a lower bound on CC_ϵ by solving CC_α^N with confidence $1-\delta$. Given $\alpha \in [0,1)$, we must choose positive integers $N,\ L$, and M such that $L \leq M$, and

$$\sum_{i=0}^{L-1} \begin{pmatrix} M \\ i \end{pmatrix} \rho(\alpha,\epsilon,N)^i (1-\rho(\alpha,\epsilon,N))^{M-i} \leq \delta, \tag{2–29}$$

where $\rho(\alpha,\epsilon,N)$ represents the probability of having at most $\lfloor \alpha N \rfloor$ "successes" in N independent trials, in which the probability of a success in each trial is ϵ. With $\alpha = \epsilon$, one can choose the value of M independent of N to obtain a lower bound with confidence $1-\delta$. Recalling that $M = 10$, if we take $L = 1$ (which corresponds to taking the minimum optimal solution over all $M = 10$ total runs, not only over the feasible solutions), we obtain a lower bound with $1-\delta = 0.999$ confidence. More generally, one can take a larger $L \in \{1,\ldots,M\}$ resulting in a lower bound with less confidence, but narrowing the optimality gap.

In Tables 2-9 and 2-10 we obtain lower bounds for the chance-constrained problems by taking $L = 1,2,3,4,5$, yielding corresponding confidence levels at least $0.999, 0.989, 0.945, 0.828, 0.623$. We use the minimum objective function value of all feasible solutions to serve as the upper bound, and report the gaps between the upper and lower bounds with respect to each confidence.

Table 2-9 shows that for instance **30-1**, setting $\alpha = \epsilon = 0.01,\ N = 1000$, and $L = 1$, the minimum objective function value, 137, over all 10 samples serves as a lower bound with confidence at least 0.999. We use the minimum objective function value 140 of all *feasible* solutions given in Table 2-7 as the upper bound, and the optimality gap is given as 2.19%. In Table 2-10, when $\alpha = \epsilon = 0.005$, we close the optimality gap by choosing

$L = 4$ (with at least 0.828 confidence). This result is consistent with the fact that we have seven feasible solutions out of 10 samples in Table 2-8, and when $L > 10 - 7 = 3$, the lower and upper bounds are equal.

Similar to the discussion of feasible solutions, we narrow the optimality gap faster by using larger values of N. Furthermore, by allowing smaller ϵ, we can close the optimality gap with a higher confidence, which is intuitive since $\epsilon = 0.005$ yields more feasible solutions for each instance than $\epsilon = 0.01$. For example, when $\epsilon = 0.01$ and $N = 2000$, Table 2-7 shows that we find eight feasible solutions out of 10 samples for instance **30-1**. In Table 2-9, the two infeasible solutions both provide a lower bound of 139 to the original problem. We then set $L = 3$, and claim that 140 is a lower bound with confidence at least 0.945, which eliminates the optimality gap. In Table 2-10, we claim optimality with higher confidence: When $N = 2000$ we close the optimality gap with confidence at least 0.989 for instance **30-1**, compared with 0.945 in the case of $\epsilon = 0.01$ in Table 2-9.

Table 2-9. Lower bound results for CC_ϵ sample problems with $\epsilon = 0.01$.

α	$N = \|\Omega\|$	Instances	Upper bound	Lower bound with confidence at least					Gap (%) with confidence at least				
				0.999	0.989	0.945	0.828	0.623	0.999	0.989	0.945	0.828	0.623
$\alpha = \epsilon$	1000	30-1	140	137	137	139	139	140	2.19	2.19	0.72	0.72	0.00
		50-1	190	186	188	188	188	189	2.15	1.06	1.06	1.06	0.53
		70-1	220	217	217	219	219	219	1.38	1.38	0.46	0.46	0.46
	2000	30-1	140	139	139	140	140	140	0.72	0.72	0.00	0.00	0.00
		50-1	190	188	189	189	190	190	1.06	0.53	0.53	0.00	0.00
		70-1	220	219	219	219	220	220	0.46	0.46	0.46	0.00	0.00

Table 2-10. Lower bound results for CC_ϵ sample problems with $\epsilon = 0.005$.

α	$N = \|\Omega\|$	Instances	Upper bound	Lower bound with confidence at least					Gap (%) with confidence at least				
				0.999	0.989	0.945	0.828	0.623	0.999	0.989	0.945	0.828	0.623
$\alpha = \epsilon$	1000	30-1	151	147	149	149	151	151	2.72	1.34	1.34	0.00	0.00
		50-1	203	201	203	203	203	203	1.00	0.00	0.00	0.00	0.00
		70-1	227	225	225	227	227	227	0.89	0.89	0.00	0.00	0.00
	2000	30-1	151	149	151	151	151	151	1.34	0.00	0.00	0.00	0.00
		50-1	203	203	203	203	203	203	0.00	0.00	0.00	0.00	0.00
		70-1	227	225	225	225	225	225	0.89	0.89	0.89	0.89	0.89

CHAPTER 3
CRITICAL NODE PROBLEMS ON TREES AND SERIES-PARALLEL GRAPHS

3.1 Problem Description and Literature Survey

In this chapter, let $G(\mathcal{V}, \mathcal{E})$ be a simple undirected graph with node set $\mathcal{V} = \{1, \dots, n\}$ and edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and let B be a positive integer node deletion budget. We seek an optimal strategy of deleting no more than B nodes (along with their incident edges) to impair the connectivity of G. This problem was explored in the work of Albert et al. (2000), who examined the attack tolerance of complex networks with respect to two distinct metrics: maximizing the number of components, and minimizing the maximum component size, induced by strategic node deletions. They employed a simple heuristic for attacking a network via node deletions, though, in lieu of an optimal approach. The present chapter seeks to satisfy this gap by providing dynamic programming (DP) algorithms for optimally attacking specially structured networks according to the metrics given above.

The node deletion problems that we consider fall under the category of Critical Node Problems (CNPs), which are of great significance in studying network properties and analyzing network reliability under disruptive events. To more precisely describe the problems we consider in this chapter, define a (maximal) *connected component* as a subgraph such that every pair of nodes in the subgraph is connected by a path, and no path exists between a node outside the subgraph and a node belonging to the subgraph. We consider the following network-connectivity metrics, which are applied to G after some subset of nodes has been deleted: (i) the number of components (ignoring deleted nodes), and (ii) the largest component size. The goal is to maximally disconnect the graph, as defined by either maximizing the number of components or minimizing the largest component in the remaining graph. We refer to these two problems as MaxNum and MinMaxC, respectively, which are closely related to the problem of identifying critical infrastructure components (Houck et al., 2004; Murray

et al., 2007), which arises in transportation (Jenelius et al., 2006; Matisziw & Murray, 2009), power grid construction (Carnal, 2005; Chien, 2006; Salmeron et al., 2004), homeland security (Brown et al., 2006), and telecommunication (Alevras et al., 1997; Resende & Pardalos, 2006) applications. For instance, Krebs (2001) describes a problem in which a commander wants to attack a terrorist network by identifying a limited number of individuals whose deletions result in the maximum breakdown of the network communication. Zhou et al. (2006) employ the CNP to provide a scope for containment of an epidemic outbreak. This situation is also considered by Cohen et al. (2003), who analyze an immunization network in which each individual is modeled as a node, and where immunized nodes cannot propagate viruses. The goal is to minimize the virus transmissibility by vaccinating a subset of critical nodes.

The CNP has also played an important role in network survivability analysis. For example, Wollmer (1964) maximizes the reduction of the maximum flow between a given pair of origin and destination by removing a set of arcs. Myung & Kim (2004) consider an undirected transportation network, and maximize the total amount of traffic lost due to edge failures. We refer to (Grötschel et al., 1995) and (Shier, 1991) for comprehensive reviews of this topic.

A variety of objectives have been used to access the consequences of deleting nodes or edges. One major category assumes that the network is constructed to perform certain functions (e.g., transporting goods or people), and evaluates network survivability based on the single/multiple-commodity maximum flow or the shortest path between given source-sink node pairs (e.g., Grubesic & Murray (2006); Matisziw & Murray (2009)). Another focus is on combinatorial properties, such as network connectivity after the removal of nodes or edges (e.g., Arulselvan et al. (2009); Dinh et al. (2010); Duque-Antón et al. (2009)). Grubesic et al. (2008) categorize CNP-related research into global measures (Grubesic & Murray, 2006) that primarily investigate graph properties associated with node interactions, and local nodal measures that

evaluate the vitality of certain nodes by studying their local characteristics (Borgatti & Everett, 2006; Freeman, 1978–1979), such as degree and centrality (e.g., the k-center problem (Hochbaum & Shmoys, 1985)).

Similar to our study, Arulselvan et al. (2009) and Dinh et al. (2010) seek a subset of nodes whose removal results in the minimum pairwise connectivity among the remaining nodes. A more general statement of this problem is to associate a weight (or "demand") with each node pair, and examine how node or edge deletions disrupt pairwise node demands. Demand between a node pair is disrupted in these studies when there does not exist a path connecting the node pair, or if at least one of the nodes in the demand pair is deleted. Di Summa et al. (2010) consider the problem of deleting a cardinality-constrained set of nodes that maximizes the sum of disrupted pairwise demands. In particular, they provide polynomial-time algorithms for optimally solving the problem on trees when unit demands exist between all node pairs. Similarly, the *sparsest cut problem* (Arora et al., 2005, 2010; Chawla et al., 2006; Hajiaghayi & Räcke, 2006; Matula & Shahrokhi, 1990) is an edge deletion problem, where each edge is associated with some cost of deletion. A feasible solution to this problem can be seen as a proper subset of nodes S, and a *cut set* given by the set of edges having exactly one incident node in S. Defining $c(S)$ as the sum of edge costs in the cut set, and $dem(S)$ as the sum of pairwise demands disrupted by removing edges in the cut set, the objective is to find an S that minimizes $c(S)/dem(S)$. This problem has applications in telecommunication and social network analysis (Bonsma, 2004; Mann et al., 2008). Moreover, a variation of MaxNum in which one deletes edges rather than nodes leads to the k-CUT problem (Garey & Johnson, 1979), which seeks a partition of an edge-weighted graph into k components, and minimizes the total edge weight between components. Goldschmidt & Hochbaum (1994) construct a polynomial-time algorithm for a fixed k by recursively solving min-cut problems. It is worth noting that the MaxNum and MinMaxC problems studied here are distinct from the pairwise-demand

studies given above, because neither the number of components nor the maximum size of components is reducible to (or from) pairwise-connectivity metrics.

Methodologies for solving CNP variants include simulation, heuristic, and mathematical programming approaches (Murray et al., 2008). Simulation- (Houck et al., 2004) and heuristic-based (Matisziw et al., 2009) methodologies are prevalent due to the complexity of solving these problems, and are often used for providing insights to large-scale problems. For instance, Tu (2000) proposes a greedy algorithm that first deletes a node having the largest degree in the current graph, updates the graph by eliminating the deleted node and all its adjacent edges, and reiterates until some maximum number of nodes have been deleted. Shen et al. (2011) provide counterexamples to show that this algorithm, while efficient, does not generally yield an optimal solution, nor does it yield a polynomial-time approximation scheme for solving either MaxNum or MinMaxC. An alternative to heuristic strategies or simulation is the use of mathematical programming techniques, which often require the solution of bilevel or trilevel programming models (Akgun, 2000; Arroyo, 2010; Brown et al., 2006; San Martin, 2007; Smith et al., 2007). Interdiction models (Arulselvan et al., 2009; Cormican et al., 1998; Lim & Smith, 2007; Wood, 1993) have been also studied for flow-based problems on general graphs, but require longer CPU time for obtaining optimal solutions.

In this chapter, we develop polynomial-time DP algorithms for solving MaxNum and MinMaxC on specially solvable cases of *trees* and *series-parallel graphs*. This analysis also permits us to solve problems on k*-hole subgraphs* (defined in Section 3.2.3). The complexity of our algorithms grows exponentially as a function of k, as expected due to the $\mathcal{NP}$-hardness of the problems we study on general graphs. We also demonstrate that the case in which each node is associated with a general integer deletion cost, and a knapsack constraint restricts the deletion of nodes, is also polynomially solvable for both problems. Moreover, the MinMaxC problem may be of interest in a weighted setting, where nodes represent entities having different weights and the goal is to

minimize the maximum-weighted component in the remaining graph. For the case in which each node is associated with a deletion cost and a weight, we show that MinMaxC becomes $\mathcal{NP}$-hard on trees, but give a polynomial-time algorithm for solving it on chain graphs (defined in Section 3.2.4.2).

The specific motivation for the problems considered in this chapter is summarized by the following observations.

- The problems we consider are of central importance in the study of social networks (Albert et al., 2000; Borgatti, 2006), which have received substantial attention in the past decade since the introduction of the "small-world network" concept (Watts & Strogatz, 1998). The term was originally proposed to demonstrate the small-world phenomenon arising in collaboration networks, but has also shown promise in describing complex systems in biology, sociology, economics, and medicine (Newman et al., 2002; Petreska et al., 2010). Latora & Marchiori (2005) and Crucitti et al. (2004a) emphasize the importance of studying vulnerability and protecting such networks. The MaxNum and MinMaxC objectives in this chapter help to better understand properties of vital actors in a social network, including cohesion, centrality, and connectivity.

- Most CNP variants having diverse connectivity metrics are shown to be $\mathcal{NP}$-complete on general graphs (Arulselvan et al., 2009; Di Summa et al., 2010; Dinh et al., 2010; Kerivin & Mahjoub, 2005). Shen et al. (2011) have shown that both MaxNum and MinMaxC are $\mathcal{NP}$-hard in the strong sense on general graphs following the results of Yannakakis (1978). By studying the problems on specially-structured graphs, our contribution thus helps to better define the boundary between hard and easy problem classes for MaxNum and MinMaxC.

- Trees and series-parallel graphs are of particular interest due to their prevalence in real network structures. A tree is the simplest connected graph, and sometimes implies a hierarchy of nodes, such as those seen in transportation and logistics systems. Tree structures are also commonly seen as backbones or subgraphs of a larger network. In addition, we examine in this chapter networks that are "almost" trees. This flexibility permits us to consider a broader class of graphs beyond those strictly classified as tree structures. Meanwhile, series-parallel graph structures often appear in electric circuits and telecommunication structures. The DP approach employed in this chapter permits polynomial-time solutions of these problems on subgraphs of a given general graph, parameterized by the number of node deletions.

The remainder of the chapter is organized as follows. In Section 3.2, we develop DP algorithms for solving MaxNum and MinMaxC as well as their variants on tree structures,

while Section 3.3 handles the problems on series-parallel graphs. Section 3.4 provides computational results that demonstrate the performance of the DP approaches, and conducts comparative analysis by varying parameters.

3.2 DP for Solving MaxNum and MinMaxC on Trees

Given an undirected tree $T(V, E)$, let r be the root node, and let l_i be the level of node i, $\forall i \in V$, i.e., the length of the path from i to r (where $l_r = 0$ in particular). Denote S_i as the set of all children nodes of i, and let T_i be the subtree rooted at node i. A key concept related to solutions over T_i is the *open set* of i, denoted as O_i, which consists of all nodes in the same component to which node i belongs. We denote the cardinality of O_i as o_i. (If node i is deleted, then O_i is empty and $o_i = 0$.) Our DP algorithm records a set of non-dominated node deletion solutions for each subtree T_i, starting from the leaf nodes, and proceeding up through the tree until terminating at the root node. We first establish the following proposition.

Proposition 3.1. *Suppose that $n \geq 3$ and $B < n$. There exists an optimal solution to all MaxNum and MinMaxC instances on tree graphs in which no leaf node is deleted.*

Proof. Consider a feasible node deletion solution $\widetilde{X}$ in which some leaf node l is deleted. Denote $p(l)$ as the parent node of l. If $p(l)$ is not deleted, we denote $C_{p(l)}$ as the component that contains $p(l)$ in the remaining graph. Consider a solution in which we delete $p(l)$ instead of l if $p(l)$ has not been deleted in $\widetilde{X}$, and in which we withdraw the deletion of l if $p(l)$ has already been deleted in $\widetilde{X}$. We discuss the impact of this operation on the MaxNum and MinMaxC cases as follows.

MaxNum: If $p(l)$ has not been deleted, then by deleting $p(l)$ instead of l, we either keep the same number of components (when $|C_{p(l)}| = 1$), or increase the number of components by at least 1 (when $|C_{p(l)}| > 1$). If $p(l)$ has already been deleted, then withdrawing the deletion of node l increases the number of components by 1.

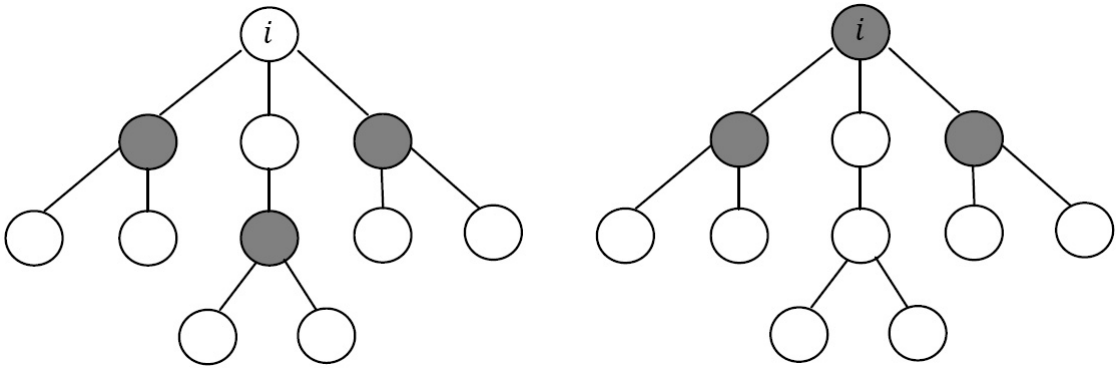
MinMaxC: If $p(l)$ has not been deleted, then deleting $p(l)$ instead of l creates a new one-node component consisting only of node l. If $|C_{p(l)}| = 1$, then component $C_{p(l)}$

vanishes; else, the deletion of node $p(l)$ splits $C_{p(l)}$ into $j \geq 1$ (disjoint) components $C^1, \ldots, C^j$ such that $\bigcup_{i=1}^{j} C^i = C_{p(l)} \setminus \{p(l)\}$, and thus $|C^i| \leq |C_{p(l)}|, \forall i = 1, \ldots, j$. If $p(l)$ has been deleted in $\widetilde{X}$, then withdrawing the deletion of node l creates a new one-node component consisting only of node l. Because $B < n$, the lower bound on the maximum component size is 1, and the creation of this new singleton component does not increase the objective value.

By repeating this operation for every leaf node, we obtain a solution that is no worse than $\widetilde{X}$. □

3.2.1 Solving MaxNum on Trees

Define $f_i(p_i, n_i)$ as the fewest number of deletions required on subtree T_i, given that (a) $p_i = 0$ if subtree root i is deleted, and $p_i = 1$ otherwise, and (b) such that n_i components are created in subtree T_i, not including O_i. (Figure 3-1 illustrates this definition.) Based on this notion, we describe some key concepts related to solution dominance that enable us to develop a DP algorithm.



A $f_i(1, 6) = 3$ B $f_i(0, 5) = 3$

Figure 3-1. Illustration of the definition of $f_i(p_i, n_i)$. A) Illustration of $f_i(p_i, n_i)$ when an open set is present; note that $n_i = 6$ because the open set itself is not counted in n_i. B) Illustration of $f_i(p_i, n_i)$ when no open set is present.

Definition 3.1. *Consider a subtree T_i of T for some $i \in V$, and node subsets $X^1, X^2 \subseteq T_i$. Suppose that if X^2 is part of an optimal node deletion solution $\overline{X}$ to MaxNum or MinMaxC (where $\overline{X}$ deletes no nodes in T_i except for X^2), then solution $X' =$*

$(\overline{X} \setminus X^2) \cup X^1$ *(i.e., we replace* X^2 *with* X^1*) must also be optimal. Then we say that* X^1 dominates X^2*. If several solutions dominate each other according to this criteria, then we arbitrarily choose one to be a non-dominated solution, with the remaining solutions being dominated.*

Lemma 1. *Consider a subtree* T_i*, for some* $i \in V$*, and node subsets* $X^1, X^2 \subseteq T_i$*,* $|X^1| \leq |X^2|$*, such that if* $i \in X^1$*, then* $i \in X^2$*. Suppose that for* $t = 1, 2$*, the deletion of nodes* X^t *from* T_i *results in* n_i^t *components in* T^i *(not including the open set), and that* $n_i^1 \geq n_i^2$*. Then* X^1 *dominates* X^2*.*

Proof. Consider solutions $\overline{X}$ and X' that respectively delete only nodes in X^2 and in X^1 over T_i. Under the assumptions in Lemma 1, solution X' splits the graph into at least as many components as in solution $\overline{X}$ without increasing the number of deleted nodes. In particular, if $i \notin X^1$, $i \in X^2$, and node i has a parent node that is deleted in the common solution to nodes in $V \setminus T_i$, then X' creates at least one more component than $\overline{X}$. Therefore, it is also optimal to delete nodes in X^1 instead of in X^2. □

Lemma 2. *Consider a subtree* T_i *for* $i \in V$*, and node subsets* $X^1, X^2 \subseteq T_i$*,* $|X^1| \leq |X^2|$*, such that* $i \in X^1$ *and* $i \notin X^2$*. Suppose that the deletion of subset* X^t *creates* n_i^t *components in* T_i *for* $t = 1, 2$*, where* $n_i^1 \geq n_i^2 + 1$*. Then* X^1 *dominates* X^2*.*

Proof. Again, we consider solutions $\overline{X}$ and X' that respectively delete X^2 and X^1 over T_i. First suppose that either $i = r$, or node i has a parent node, $p(i)$, that is deleted in X' (and $\overline{X}$). In this case, solution X' creates n_i^1 components within T_i, $\overline{X}$ creates $n_i^2 + 1$ components within T_i, and both X' and $\overline{X}$ create the same number of components in $V \setminus T_i$. Recalling that $n_i^1 \geq n_i^2 + 1$, solution X' creates at least as many components in T_i as does $\overline{X}$. Now, suppose instead that $p(i)$ exists and does not belong to X' (or $\overline{X}$). Define C_i as the component containing node i in solution X', and define $Q_i = T_i \cup C_i$. Note that after deleting nodes in both solutions $\overline{X}$ and X', no node in Q_i is connected to a node in $V \setminus Q_i$. We have that X' creates $n_i^1 + 1$ components within Q_i, $\overline{X}$ creates $n_i^2 + 1$

components within Q_i, and both solutions create the same number of components in $V \setminus Q_i$. Hence, in any case, X' is at least as good a solution as $\overline{X}$, and we could always replace X^2 with X^1 without losing solution optimality. □

Based on Lemmas 1 and 2, we will retain only non-dominated solutions in our approach (where if X^1 and X^2 dominate each other, we retain one such solution).
We now turn our attention to computing the f-values required by our algorithm. By Proposition 3.1, we can set $f_l(1, 0) = 0$ for all leaf nodes $l \in V$, because l is not deleted. To compute all other values of the form $f_i(p_i, n_i)$, we synthesize the solutions at child subtrees T_v for all $v \in S_i$, depending on whether or not node i is deleted. We examine recursive models for these two cases as follows. The recursion for updating f_i-elements when $p_i = 0$ is given by

$$f_i(0, n_i) = \min \quad \sum_{v \in S_i} f_v(p_v, n_v) + 1 \tag{3–1a}$$

$$\text{s.t.} \quad n_i = \sum_{v \in S_i} n_v + \sum_{v \in S_i} p_v, \tag{3–1b}$$

$$p_v \in \{0, 1\},\ n_v \in \{0, \dots, |T_v| - 1\} \quad \forall v \in S_i, \tag{3–1c}$$

where Objective 3–1a minimizes the sum of node deletions at all subtrees $T_v, \forall v \in S_i$, and adds an additional deletion at node i. Note that by deleting node i, any non-empty open set in a child subtree T_v becomes a new component. Hence, Equation 3–1b states that n_i is given by the summation of subtree components (not including the open sets), plus the number of non-empty open sets at all child subtrees T_v. Constraint 3–1c states in this problem the range of the variables p_v and $n_v, \forall v \in S_i$. Because these range values are implied by the definition of f_v, we omit them from our future optimization problem statements.

When $p_i = 1$, we use the following recursion:

$$f_i(1, n_i) = \min \quad \sum_{v \in S_i} f_v(p_v, n_v) \tag{3–1d}$$

$$\text{s.t.} \qquad n_i = \sum_{v \in S_i} n_v, \tag{3–1e}$$

where Equality 3–1e updates n_i by summing the number of all components at its child subtrees, excluding the open sets. (Note that all open sets merge into a single open set O_i in this case.)

Note that computing $f_i(p_i, n_i)$ via Formulation 3–1 may be infeasible in some cases, or exceed the budget B, or be dominated by another value as given by Lemmas 1 and 2. In these cases, we set $f_i(p_i, n_i) = \infty$ to ignore those values in future computations. (More precisely, the data structures in our implementation only store finite and non-dominated f-values.) For the remainder of this chapter, we occasionally refer to finite f_i-values as entries, with which backtracking solution information is also associated. We present an overview of the DP solution scheme for solving MaxNum on trees as follows.

Step 0 (Initialization). Initialize $f_l(1, 0) = 0$ at every leaf node $l \in V$. Mark all non-leaf nodes as "unexamined."

Step 1 (Examine a node). If all nodes have been examined, go to Step 3. Otherwise, pick any unexamined node $i \in V$ such that each node $v \in S_i$ has been examined, and proceed to Step 2.

Step 2 (Update solution). Update $f_i(p_i, n_i)$ at T_i according to Formulation 3–1. If Formulation 3–1 is not feasible, or has an optimal solution that exceeds the node deletion budget or is dominated, then set $f_i(p_i, n_i) = \infty$. For each finite entry of $f_i(p_i, n_i)$, record the arguments $(p_v, n_v), \forall v \in S_i$, that lead to the optimal value computed for $f_i(p_i, n_i)$.

Step 3 (Identify an optimal solution). Among the list of finite $f_r(p_r, n_r)$-values, report one having the largest value of $n_r + p_r$, which represents the number of components created by the solution corresponding to this entry. (Note that the inclusion of p_r accounts for the open set component, which is not included in n_r.) Backtrack to

determine an optimal set of node deletions as follows. If $p_r = 0$, then delete r, and otherwise do not. Then consider subtrees T_v for each $v \in S_r$, and examine the recorded arguments that computed an optimal $f_r(p_r, n_r)$-value. Recursively apply the backtracking approach to identify all deleted nodes.

3.2.1.1 Sequential DP steps

In the above solution scheme, the key step is the recursive computation of f-values in Formulation 3–1. Here, we provide a polynomial-time method that solves the recursions sequentially by synthesizing solutions from one child subtree matrix at a time.

First, we (arbitrarily) index all subtree nodes $v \in S_i$ as $\{i_1, \dots, i_{|S_i|}\}$, and construct a $2 \times |T_i|$ matrix $H^s(i)$ for each $s \in \{1, \dots, |S_i|\}$. Entries $h_i^s(p_i^s, n_i^s)$ of $H^s(i)$ are computed for $p_i^s \in \{0, 1\}$ and $n_i^s \in \{0, \dots, |T_i| - 1\}$, and represent minimum-cardinality node deletions over node i and nodes in subtrees $T_{i_1}, \dots, T_{i_s}$, corresponding to root deletion status p_i^s and number of (non-open) components n_i^s. In particular, $H^{|S_i|}(i)$ yields the desired f_i-values. Accordingly, for each subtree T_i, we divide Step 2 into $|S_i|$ steps: We initialize $H^1(i)$ by merging node i with solutions $f_{i_1}(p_{i_1}, n_{i_1})$ at T_{i_1}, and then update $H^s(i)$ by merging entries in $H^{s-1}(i)$ with $f_{i_s}(p_{i_s}, n_{i_s})$ entries corresponding to T_{i_s}, for all $s = 2, \dots, |S_i|$.

Step 2a (Initializing $H^1(i)$). We compute $h_i^1(p_i^1, n_i^1)$ as:

$$h_i^1(0, n_i^1) = 1 + \min\left\{f_{i_1}(1, n_i^1 - 1),\ f_{i_1}(0, n_i^1)\right\} \tag{3–2a}$$

$$h_i^1(1, n_i^1) = \min\left\{f_{i_1}(1, n_i^1),\ f_{i_1}(0, n_i^1)\right\}. \tag{3–2b}$$

Equation 3–2a computes the objective as the minimum number of deletions at node i_1 among cases where $p_{i_1} = 1$ or $p_{i_1} = 0$, plus the deletion at node i. When $p_{i_1} = 1$, we have that $n_i^1 = n_{i_1} + 1$ because the open set at node i_1 contributes to the total number of components after i is deleted; else, the number of components does not change

(i.e., $n_i^1 = n_{i_1}$). Equation 3–2b is the same as Equality 3–2a, but keeps $n_i^1 = n_{i_1}$ for all situations. For each case, we record the arguments (p_{i_1}, n_{i_1}) that minimize the objective.

Step 2b (Updating $H^s(i)$). We compute $h_i^s(p_i^s, n_i^s), \forall s = 2, \dots, |S_i|$, by merging matrices $H^{s-1}(i)$ with f_{i_s}-entries as follows.

$$h_i^s(0, n_i^s) = \min \quad h_i^{s-1}(0, n_i^{s-1}) + f_{i_s}(p_{i_s}, n_{i_s}) \tag{3–2c}$$
$$\text{s.t.} \quad n_i^s = n_i^{s-1} + n_{i_s} + p_{i_s}$$
$$h_i^s(1, n_i^s) = \min \quad h_i^{s-1}(1, n_i^{s-1}) + f_{i_s}(p_{i_s}, n_{i_s}) \tag{3–2d}$$
$$\text{s.t.} \quad n_i^s = n_i^{s-1} + n_{i_s}$$

In Equality 3–2c, we compute the number of components n_i^s as the sum of n_i^{s-1} and n_{i_s}, in addition to the open set at i_s if one exists, which would become a new (non-open) component. Similarly, when i is not deleted, Equality 3–2d seeks a minimum sum of $h_i^{s-1}(1, n_i^{s-1})$ (indicating that i is not deleted) and $f_{i_s}(p_{i_s}, n_{i_s})$, where n_i^s equals the sum of n_i^{s-1} and n_{i_s}. Note that if an open set exists in T_i, it merges with the one containing node i without creating a new component. Record all arguments $(p_{i_1}, n_{i_1}), \dots, (p_{i_s}, n_{i_s})$ that lead to the calculation of values in $H^s(i)$.

3.2.1.2 Complexity analysis

Computing $H^s(i)$ for $s = 1, \dots, |S_i|$ at each subtree T_i represents the bottleneck operation. We can compute Equality 3–2a and Equality 3–2b in constant time for each non-leaf node i, and so obtaining $h_i^1(1, n_i^1)$ and $h_i^1(0, n_i^1)$ for all non-leaf nodes i, and $n_i^1 = 0, \dots, |T_i| - 1$, can be done in $O(n^2)$ steps. To compute $H^s(i)$ via Equality 3–2c and Equality 3–2d for $s = 2, \dots, |S_i|$, we merge each entry of $H^{s-1}(i)$ with each finite-valued $f_{i_s}(p_{i_s}, n_{i_s})$. There are $O(|T_i|)$ entries in $H^{s-1}(i)$ and $O(|T_{i_s}|)$ values of $f_{i_s}(p_{i_s}, n_{i_s})$. Because both $|T_i|$ and $|T_{i_s}|$ are bounded by the total number of nodes, the computation of $H^s(i)$ takes $O(n^2)$ steps. The maximum number of such matrices is bounded by the number of nodes in the tree, and therefore the algorithmic complexity of solving MaxNum on trees is $O(n^3)$.

To compute the space complexity of our procedure, we analyze the maximum amount of storage required at any point of our algorithm. Storage is required for the $H^s(i)$-matrices, and for retaining partial solutions that lead to these entries. Note that even though we can delete $H^{s-1}(i)$ after computing $H^s(i)$, for $s = 2, \ldots, |S_i|$, we still must store $O(n)$ H-matrices simultaneously for backtracking purposes. Recall that for each entry of $H^{|S_i|}(i)$ (i.e., the f_i-values), we store the number of nodes corresponding to the entry and the (p_v, n_v) arguments in each subtree T_v, $\forall v \in S_i$, that led to the computation of the entry. This requires $O(2n(1 + 2|S_i|)) = O(n|S_i|)$ elements to be stored in $H^{|S_i|}(i)$. Summing over $i \in V$, and noting that $\sum_{i \in V} |S_i| = n - 1$, we have that the space complexity of our procedure is $O(n^2)$.

3.2.2 Solving MinMaxC on Trees

To solve MinMaxC, one approach is to define similar recursions as used for MaxNum. This approach would let $f_i(o_i, m_i)$ represent the fewest number of deletions on subtree T_i such that there is an open set of size o_i, and a maximum component size of m_i (excluding the set O_i). We would again compute the f_i-values by merging solutions from child subtrees of node i. Note here that if we record objectives for $f_i(o_i, m_i)$, we would store $O(n^2)$ values at each node (for the $O(n)$ possible values of o_i and m_i). Also, the merging procedure described in Section 3.2.1.1 would now require the pairwise comparison of elements in two $O(n^2)$ matrices at a time, leading to an $O(n^4)$ merging operation. The overall complexity of solving MinMaxC on trees by the procedure above would be $O(n^5)$. We provide the detailed descriptions in Appendix B. This algorithm, while still finishes in polynomial number of steps, is quite inefficient. Instead, we adopt the following binary search scheme.

Let τ be a fixed target size, such that $f_i(o_i, \tau)$ represents the fewest number of deletions required on the subtree T_i that generates an open set of size o_i where $o_i \leq \tau$, and a largest component size of no more than τ. We separately derive recursions based on whether or not a deletion takes place at node i (i.e., $o_i = 0$ or $o_i > 0$). For the case of

$o_i = 0$, we have

$$f_i(0,\tau) = \min \quad \sum_{v \in S_i} f_v(o_v,\tau) + 1, \tag{3–3a}$$

where Equality 3–3a accounts for the minimum total number of deletions at all child subtrees T_v, $\forall v \in S_i$, plus one deletion at node i. Because the maximum component size in any subtree, including its open set, is no more than τ, then deleting node i ensures that no component has more than τ nodes.

The recursion for updating elements for which $0 < o_i \leq \tau$ is given by

$$f_i\left(o_i,\tau\right) = \min \quad \sum_{v \in S_i} f_v(o_v,\tau) \tag{3–3b}$$

$$\text{s.t.} \quad o_i = \sum_{v \in S_i} o_v + 1 \leq \tau. \tag{3–3c}$$

Objective 3–3b is the same as Equality 3–3a, but omits a deletion at node i. Equation 3–3c updates o_i as the sum of all child open-set sizes in addition to one at node i (noting that O_i will include node i and all open sets O_v, $\forall v \in S_i$), and ensures that o_i is no larger than τ. Because the maximum component sizes existing at all subtrees are no more than τ, the maximum component size after merging is also not more than τ. We again set $f_i(o_i,\tau) = \infty$ if it is infeasible or requires more than B deletions. Additionally, if $f_i(o_i^1,\tau) \leq f_i(o_i^2,\tau)$ and $o_i^1 \leq o_i^2$ with at least one inequality being strict, then the $f(o_i^2,\tau)$ term is clearly dominated and can be ignored by setting it equal to ∞. As before, we set $f_l(1,\tau) = 0$ (and $f_l(0,\tau) = \infty$) at every leaf node $l \in V$.

The binary search approach initializes an upper bound $UB = n - B$ (because any B deletions guarantees a maximum component size of no more than $n - B$), a lower bound $LB = 1$, and an initial target size $\tau = \lfloor (n - B + 1)/2 \rfloor$. Let the incumbent solution be one in which any set of B nodes is deleted from the graph.

Step I (Solve MinMaxC for a fixed τ). Compute $f_i(o_i,\tau)$ at all subtrees T_i. If the list of finite $f_r(o_r,\tau)$-values is empty, then the MinMaxC objective exceeds LB, and we

set $LB = \tau + 1$. Otherwise, there exists a feasible solution satisfying the component size threshold τ. Backtrack to find one such solution that minimizes the number of deleted nodes, compute the actual maximum component size τ' ($\leq \tau$) corresponding to this solution, update $UB = \tau'$, and store the identified solution as the new incumbent. Go to Step II.

Step II (Update τ). If $LB = UB$, then the incumbent solution is optimal, and the procedure terminates. Otherwise, update $\tau = \lfloor (UB + LB)/2 \rfloor$, and go to Step I. □

For solving $f_i(o_i, \tau)$ in Step I, we propose a similar sequential DP procedure as in Section 3.2.1.1. We revise our definition of $h_i^s(o_i^s, \tau)$, which now represents solutions to T_i (including minimum number of nodes and optimal subtree arguments) after incorporating its subtrees $T_{i_1}, \ldots, T_{i_s}$. Note that the state spaces of both $f_i(o_i, \tau)$ and $H^s(i)$ are $O(n)$ for a fixed τ, and merging can be done in $O(n^2)$ steps. Hence, the complexity of Step I is $O(n^3)$. The maximum number of steps for performing binary search is $O(\log n)$, and thus the overall complexity of solving MinMaxC on trees is $O(n^3 \log n)$. Note that we only need to store all f- and h-values for the current τ-value in Step I, and thus the space complexity remains $O(n^2)$ by using the same arguments as for MaxNum.

3.2.3 Solving MaxNum and MinMaxC on k-hole-graphs

To accommodate more general network structures that arise in practice, we extend our analysis to a more general class of graphs that subsume trees. Define a hole of a graph as a set of nodes $v_1, \ldots, v_m$ such that an edge exists between v_i and v_j ($i < j$) if and only if $i = j - 1$ or $i = 1$ and $j = m$. A *k-hole-graph* is a connected graph that contains exactly k holes (e.g., Gross & Yellen (2003)). Next, we discuss how our approach can be modified to solve MaxNum and MinMaxC on k-hole-graphs.

We first describe an algorithm for solving our problems on 1-hole-graphs $T'(V, E)$. Denote M^1 as the hole, which consists of nodes $\{v_1, \ldots, v_{|M^1|}\} \subseteq V$. Consider Case 0, in which no nodes in M^1 are deleted and Case $i = 1, \ldots, |M^1|$, in which we delete

node v_i (possibly in addition to some other hole nodes). Note that all solutions belong to at least one of these cases. In Case 0 we treat the entire hole as a single root node that represents an $|M^1|$-node open set, and recover a tree structure. Our DP recursions are then executed as before, with a minor modification in the final merge operation to account for the $|M^1|$-node root hole, which cannot be deleted in this case. For MaxNum, this is handled by insisting that $p_r = 1$, and in the case of MinMaxC, the middle term of Equality 3–3c becomes $\sum_{v \in S_r} o_v + |M^1|$. In Case i we first delete hole node v_i to obtain a tree, and then execute our DP approach as before (accounting for the additional deletion of v_i in our objective). The best solution found over Cases $0, \dots, M^1$ is optimal to T'.

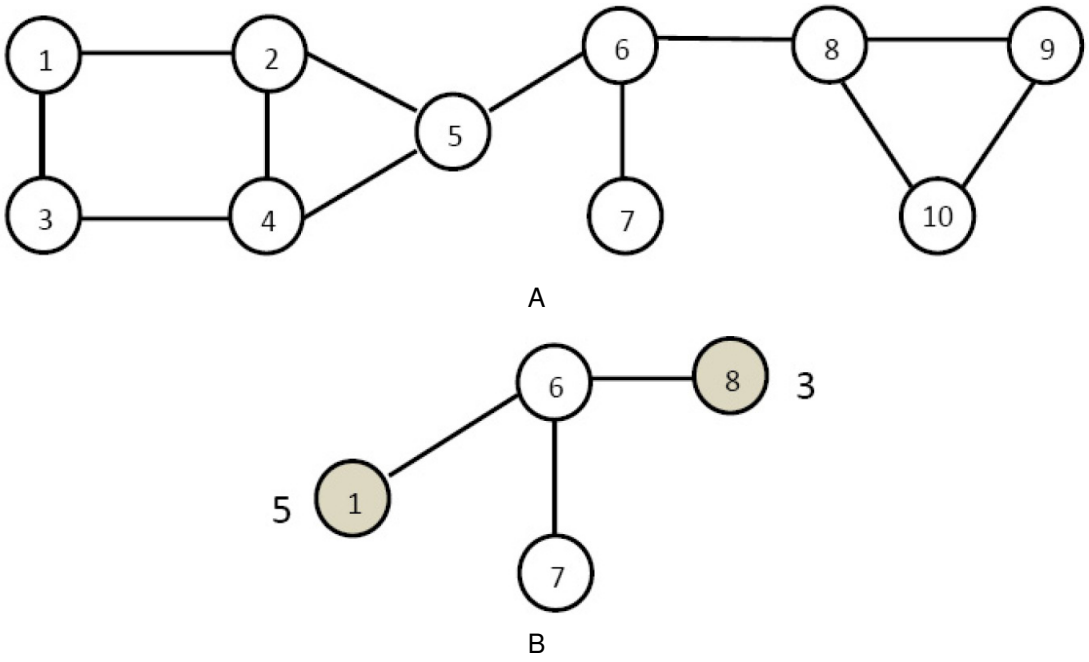


Figure 3-2. Illustration of node aggregation in hole-graphs. A) Original 3-hole graph. B) Aggregated graph, with shaded nodes representing aggregated nodes, and node weights given alongside aggregated nodes.

For the case of k-hole-graphs ($k \geq 2$), denote $M^1, \dots, M^k$ as the k holes in the graph, where nodes $\{v_1, \dots, v_q\}$ compose the union of the nodes in these holes. For any pair of holes that share common node(s), we aggregate them into one component, as depicted in Figure 3-2. First, consider Case 0 in which no node is deleted in any hole.

We treat every hole M^j as a *hole-node* that represents a size-$|M^j|$ component. This operation recovers a tree structure in which hole nodes have weights corresponding to the number of nodes they represent. We then execute the DP recursions as before, with a simple modification that prohibits deletion of aggregated nodes, and also accounts for their weights when calculating open-set sizes in subtrees for MinMaxC.

In Case i we delete node v_i for $i = 1, \dots, q$, and obtain a p-hole-graph such that $p < k$. We recursively solve the problem on the resulting p-hole-graph, where a 0-hole-graph (i.e., a tree) serves as the base case. Denoting $\Gamma(k)$ as the complexity of solving either node deletion problem on k-hole-graphs, we have that $\Gamma(k) = O(n\,\Gamma(k-1))$. Because $\Gamma(0)$ is $O(n^3)$ for MaxNum, $\Gamma(k)$ is $O(n^{3+k})$ for MaxNum, and by the same argument, $\Gamma(k)$ is $O(n^{3+k}\log n)$ for MinMaxC.

Remark 3.1. Note that we can interpret any connected graph G as a k-hole graphs for some integer value of k. However, the complexities of our algorithms increase exponentially as a function of k, even though the algorithms are polynomial-time for a fixed k. Thus, we envision that our algorithms are most useful for optimally solving MaxNum and MinMaxC on sparse graphs having relatively few holes.

3.2.4 Variants on Trees

In this section, we analyze various extensions of our problems on trees. First, we consider the case in which we associate a (positive integer) deletion cost with each node, and require that the total node deletion cost must be no more than B. The objectives for MaxNum and MinMaxC remain unchanged, now with a tie-breaking objective of minimizing total node deletion cost rather than the number of deleted nodes. Note that we can equivalently execute the foregoing algorithms by letting the number of deleted nodes be a state variable, and minimizing node deletion cost in the objective. We would require only a simple modification that accounts for node deletion costs in Formulation 3–1a, 3–2a, and 3–3a (instead of the "$+1$" that is currently present). The

complexity of both DP algorithms stays unchanged, and thus this case is polynomially solvable.

For MinMaxC, we may have the situation in which each node represents a larger entity, and is weighted by the value of that entity. For instance, if each node represents a city, a node's weight may represent its population. We refer to MinMaxC with (non-negative integer) node deletion costs and node weights as MinMaxC-CW. We show that MinMaxC-CW is $\mathcal{NP}$-hard in general, but that for the special case of chain graphs, MinMaxC-CW can be solved in polynomial time.

3.2.4.1 MinMaxC-CW on general trees

Given a tree $T(V, E)$, let c_i and w_i be the deletion cost and the weight associated with node $i \in V$, respectively. Denote $\overline{\text{MinMaxC-CW}}$ as the decision version of MinMaxC-CW, which seeks a subset of nodes to be deleted having a total deletion cost of B, such that the resulting maximum component weight is no more than $\mathcal{W}$.

Theorem 3.1. *$\overline{\text{MinMaxC-CW}}$ is $\mathcal{NP}$-complete, even on trees.*

Proof. We first show that $\overline{\text{MinMaxC-CW}}$ belongs to $\mathcal{NP}$. Given a node deletion solution, we can compute all component weights by running a polynomial-time search algorithm on the remaining graph. We can verify whether the associated total node deletion cost is no more than B, and whether all weighted component sizes are no more than $\mathcal{W}$.

Next, we show that $\overline{\text{MinMaxC-CW}}$ is $\mathcal{NP}$-complete by using a transformation from the PARTITION problem (Garey & Johnson, 1979) stated as follows: Given a finite set $A = \{1, \ldots, n\}$ and a positive integer size a_i for each $i \in A$, is there a subset $A' \subseteq A$ such that $\sum_{i \in A'} a_i = \sum_{i \in A} a_i/2$? We transform a PARTITION instance by creating a root node 0, nodes associated with each element $i \in A$, and edges $(0, i)$ for all $i = 1, \ldots, n$. We set $w_0 = 0$ and $c_0 = \sum_{i \in A} a_i$, and we have $w_i = c_i = a_i$ for each leaf node i. Set $B = \mathcal{W} = \sum_{i \in A} a_i/2$. We show that a PARTITION instance has a solution if and only if the $\overline{\text{MinMaxC-CW}}$ instance has one.

First, suppose that a PARTITION instance has a solution A'. By deleting all nodes associated with elements in A', the total deletion cost is $\sum_{i\in A'} c_i = \sum_{i\in A'} a_i = \sum_{i\in A} a_i/2 = B$, and the only remaining component consists of node 0 and all nodes in $A \setminus A'$, which has a total weight of $w_0 + \sum_{i\in A\setminus A'} w_i = \sum_{i\in A\setminus A'} a_i = \mathcal{W}$. Hence, the $\overline{\text{MinMaxC-CW}}$ instance also has a solution.

Now, consider any solution to a $\overline{\text{MinMaxC-CW}}$ instance. Node 0 is not deleted in this solution due to the budget constraint. Letting A' be the set of all nodes that are deleted, we have $\sum_{i\in A'} a_i = \sum_{i\in A'} c_i = B = \sum_{i\in A} a_i/2$. All remaining nodes connecting to root node 0 form the only component with a total weight of $w_0 + \sum_{i\in A\setminus A'} w_i = \mathcal{W} = \sum_{i\in A} a_i/2$. Thus, A' and $A \setminus A'$ form a valid partition, and the PARTITION instance also has a solution. This completes the proof. □

Remark 3.2. Observe that Theorem 3.1 shows that MinMaxC-CW is $\mathcal{NP}$-hard in the ordinary sense, and that a pseudopolynomial-time algorithm may exist to solve these problems. Indeed, a simple weighted version of MinMaxC applied to MinMaxC-CW requires a state space of size $O(\sum_{i=1}^{n} w_i)$ at each node. Hence, a weighted version of the MinMaxC algorithm is of pseudopolynomial complexity, which verifies that MinMaxC-CW is not strongly $\mathcal{NP}$-hard. □

3.2.4.2 A polynomial algorithm for MinMaxC-CW on a chain graph

We now provide a polynomial-time shortest-path algorithm for solving MinMaxC-CW on a chain, i.e., $V = \{1, \dots, n\}$ and $E = \{(i, i+1) \mid i = 1, \dots, n-1\}$. We first create a graph transformation from the chain as follows. We create dummy nodes 0 and $n+1$, which will serve as origin and destination nodes, respectively, and define $c_0 = c_{n+1} = 0$. Arcs are only constructed from lower-indexed nodes to higher-indexed nodes, where traversing arc (i, j) represents a decision of deleting both nodes i and j, but not nodes $i+1, \dots, j-1$. Let $\mathcal{P}$ denote the set of all such arcs (i, j), $\forall 0 \le i < j \le n$. Let $d_{ij} = c_j$ be the cost associated with arc (i, j), $\forall (i, j) \in \mathcal{P}$. Also, define $g_{ij} = \sum_{k=i+1}^{j-1} w_k$ as the weight associated with each arc $(i, j) \in \mathcal{P}$.

Again, we use a binary search approach that fixes a target maximum component size value τ at each iteration. Given a fixed τ, let $\kappa(\tau)$ denote the minimum cost (in terms of d-values) of a set of node deletions, such that the largest component weight is no more than τ. (If no solution exists, then $\kappa(\tau) = \infty$.) Define $\mathcal{P}(\tau) = \{(i,j) \in \mathcal{P} : g_{ij} \leq \tau\}$. Then $\kappa(\tau)$ is the shortest distance of any path from 0 to $n+1$ using arcs in $\mathcal{P}(\tau)$.

We can execute a binary search algorithm to identify the smallest value of τ for which $\kappa(\tau) \leq B$. Alternatively, if $\sum_{i=1}^{n} w_i$ is very large, we could first sort the g_{ij} values in non-decreasing order, and then perform our binary search procedure over the elements in this sorted list. Sorting the g_{ij} values requires $O(n^2 \log(n^2))$ steps, noting that $|\mathcal{P}|$ is $O(n^2)$. The binary search algorithm would then examine no more than the minimum of $\log(\sum_{i=1}^{n} w_i)$ and $\log(n^2)$ times of τ.

Note that there are $O(n)$ nodes and m (which is $O(n^2)$) arcs in the transformed shortest-path graph, and all distance values between each pair of nodes are non-negative. We can solve each shortest-path instance in $O(m + n \log n)$ time by using a Fibonacci heap implementation of Dijkstra's algorithm (e.g., Ahuja et al. (1993)). Treating $m = O(n^2)$ here, the overall complexity of our algorithm is $O(n^2 \log(\min\{\sum_{i=1}^{n} w_i, n^2\}))$, thus verifying that our algorithm is strongly polynomial.

3.3 Series-Parallel Graph Analysis

In this section, we give the definition of a series-parallel graph and its associated concepts in Section 3.3.1, and describe DP algorithms for MaxNum and MinMaxC in Section 3.3.2 and 3.3.3.

3.3.1 Series-Parallel Graphs and Tree Decomposition

A *two-terminal graph* (TTG) $G(s,t)$ has nodes s and t that represent *source* and *sink*, respectively. Given two TTGs $G_1(s_1,t_1)$ and $G_2(s_2,t_2)$, a *series composition* creates a new TTG by merging the sink t_1 and the source s_2, where s_1 becomes the new source and t_2 becomes the new sink. We denote the series operation as $G = S(G_1, G_2)$. A *parallel composition* creates a new TTG by merging the two sources s_1 and s_2 into

the new source node, and the two sinks t_1 and t_2 into the new sink node. The parallel operation is denoted by $G = P(G_1, G_2)$. A *series-parallel graph* (SPG) is a graph that is constructed by a sequence of series and parallel compositions starting from a set of single-edge graphs, each of which we refer to as K_2.

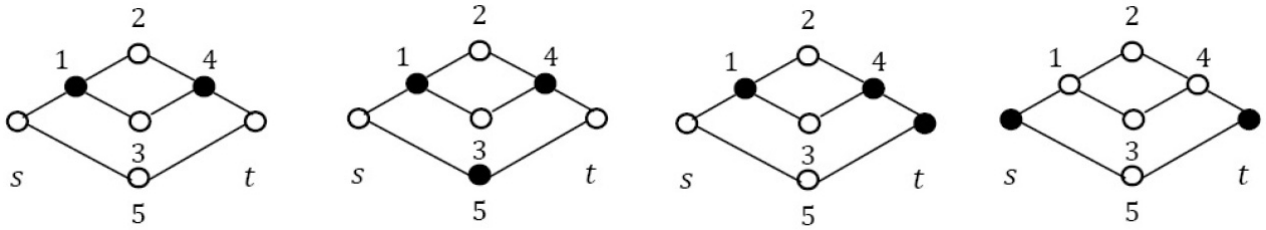
An SPG $G(s, t)$ can be decomposed into several K_2 base graphs. This decomposition is specified by a binary decomposition tree $T(G)$ whose nodes represent subgraphs of G. Each non-leaf node of the tree has two child subgraphs and an associated operation (series or parallel) that combines these two children to form the parent subgraph. (All leaf nodes represent K_2.) It has been shown in (Takamizawa et al., 1982; Valdes et al., 1982) that $T(G)$ can be constructed in linear time in terms of the size of G.

3.3.2 Solving MaxNum on SPGs

We begin solving MaxNum on some SPG $\overline{G}$ by constructing $T(\overline{G})$. Our approach begins from the leaf nodes of $T(\overline{G})$ and works its way up to the root via a set of series and parallel merging operations. Recall that for the MaxNum objective on trees, we let p_i indicate the binary deletion status at node i, where $p_i = 0$ if node i is deleted, and $p_i = 1$ otherwise. For any SPG subgraph $G(s, t)$ of $\overline{G}$, we specify the same status values p_s and p_t for s and t, respectively. We will also need a binary state variable c_{st}, which equals 1 if s and t belong to the same component in $G(s, t)$, and 0 otherwise. Note that $c_{st} = 0$ whenever $p_s + p_t \leq 1$. Finally, the open sets are defined as before, where O_s (or O_t) denotes the component that includes source s (or sink t). Given an SPG $G(s, t)$, the function $f_G(p_s, p_t, c_{st}, n_G)$ gives the fewest number of deletions in G corresponding to state variables p_s, p_t, and c_{st}, such that there exist n_G components in G, not including O_s and O_t. Figure 3-3 illustrates these concepts.

We initialize $f_{K_2}(p_s, p_t, c_{st}, n_{K_2})$ for each leaf subgraph K_2 in $T(\overline{G})$ as follows, corresponding to the four possible combinations of deleting or not deleting s and t:

$$f_{K_2}(1, 1, 1, 0) = 0,\ f_{K_2}(0, 1, 0, 0) = 1,\ f_{K_2}(1, 0, 0, 0) = 1,\ f_{K_2}(0, 0, 0, 0) = 2. \qquad (3\text{–}4)$$



A $f(1,1,1,2)=2$ B $f(1,1,0,2)=3$ C $f(1,0,0,2)=3$ D $f(0,0,0,2)=2$

Figure 3-3. Illustration of the definition of $f_G(p_s, p_t, c_{st}, n_G)$. A) The source and sink are connected by path s–5–t. Nodes 2 and 3 are two singleton components. Node 5 belongs to the open set at nodes s and t. B) The source and sink are not connected. Nodes 2 and 3 are two singleton components. C) Since node t is deleted, there is no path connecting s and t. Node 5 belongs to the open set at node s having a cardinality of 2. D) No path connects nodes s and t. Nodes 1, 2, 3, and 4 form a 4-node component. Node 5 is a singleton.

3.3.2.1 The series operation

Consider the computation of $f_G(p_s, p_t, c_{st}, n_G)$ when $G = S(G_1, G_2)$. First, we have:

$$f_G(1,1,1,n_G) = \min \quad f_{G_1}(1,1,1,n_{G_1}) + f_{G_2}(1,1,1,n_{G_2}) \qquad \text{(3–5a)}$$
$$\text{s.t.} \quad n_G = n_{G_1} + n_{G_2}.$$

To create an SPG with $p_s = p_t = 1$ and $c_{st} = 1$ for a series operation, we must have that the two terminals are connected (and not deleted) in both G_1 and G_2. Open sets O_{s_1} and O_{s_2} merge in this case without creating any new component, and so $n_G = n_{G_1} + n_{G_2}$. Next, consider the recursion:

$$f_G(1,1,0,n_G) = \min \begin{cases} \min & f_{G_1}(1,1,c_{s_1t_1},n_{G_1}) + f_{G_2}(1,1,c_{s_2t_2},n_{G_2}) \\ \text{s.t.} & c_{s_1t_1} + c_{s_2t_2} \le 1 \\ & n_G = n_{G_1} + n_{G_2} + (1-c_{s_1t_1})(1-c_{s_2t_2}); \\ \min & f_{G_1}(1,0,0,n_{G_1}) + f_{G_2}(0,1,0,n_{G_2}) - 1 \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2}. \end{cases} \qquad \text{(3–5b)}$$

Here again, we have that $p_s = p_t = 1$, but $c_{st} = 0$, and so there are multiple characteristics of G_1 and G_2 that could be present when they merge to form G. The first minimization subproblem in Formulation 3–5b handles the case in which neither t_1 nor s_2 is deleted. In this case, a path will exist from $s = s_1$ to $t = t_2$ unless either $c_{s_1 t_1} = 0$ or $c_{s_2 t_2} = 0$, and hence we constrain the sum of those values to be no more than 1. Observe that the number of components will now be the sum of the components from G_1 and G_2, plus 1 if $c_{s_1 t_1} = c_{s_2 t_2} = 0$. The additional component in this case consists of nodes in $O_{t_1} \cup O_{s_2}$, which do not belong to O_s or O_t in G, but were not counted in the components in G_1 or G_2 because they belonged to one of the open sets. The second minimization subproblem in Formulation 3–5b handles the case in which both t_1 and s_2 are deleted. Note that the deletion of node $t_1 = s_2$ is double-counted when adding f_{G_1} and f_{G_2}, which is why we correct the sum by subtracting 1. No additional components are generated, and hence $n_G = n_{G_1} + n_{G_2}$.

The remaining recursions are given as:

$$f_G(0,1,0,n_G) = \min \begin{cases} \min & f_{G_1}(0,1,0,n_{G_1}) + f_{G_2}(1,1,c_{s_2 t_2},n_{G_2}) \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2} + (1 - c_{s_2 t_2}); \\ & \\ \min & f_{G_1}(0,0,0,n_{G_1}) + f_{G_2}(0,1,0,n_{G_2}) - 1 \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2} \end{cases} \tag{3–5c}$$

$$f_G(1,0,0,n_G) = \min \begin{cases} \min & f_{G_1}(1,1,c_{s_1 t_1},n_{G_1}) + f_{G_2}(1,0,0,n_{G_2}) \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2} + (1 - c_{s_1 t_1}); \\ & \\ \min & f_{G_1}(1,0,0,n_{G_1}) + f_{G_2}(0,0,0,n_{G_2}) - 1 \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2} \end{cases} \tag{3–5d}$$

$$f_G(0,0,0,n_G) = \min \begin{cases} \min & f_{G_1}(0,1,0,n_{G_1}) + f_{G_2}(1,0,0,n_{G_2}) \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2} + 1; \\ & \\ \min & f_{G_1}(0,0,0,n_{G_1}) + f_{G_2}(0,0,0,n_{G_2}) - 1 \\ \text{s.t.} & n_G = n_{G_1} + n_{G_2}. \end{cases} \tag{3–5e}$$

The recursions given by 3–5c, 3–5d, and 3–5e are similar to Equality 3–5b, with the following exceptions. One, Equality 3–5c requires that $p_s = 0$ and $p_t = 1$, and hence all merging operations require $p_{s_1} = 0$ and $p_{t_2} = 1$. Also, for the case in which $p_{t_1} = p_{s_2} = 1$, recall that we must count an extra component when $c_{s_1 t_1} = c_{s_2 t_2} = 0$. In this case, noting that $p_{s_1} = 0$ implies that $c_{s_1 t_1} = 0$, we account for the extra component simply when $c_{s_2 t_2} = 0$. Similar logic holds for Equality 3–5d, and for Equality 3–5e as well by noting that both $c_{s_1 t_1} = 0$ and $c_{s_2 t_2} = 0$ are implied when merging the case of $p_{s_1} = p_{t_2} = 0$.

3.3.2.2 The parallel operation

Consider the computation of $f_G(p_s, p_t, c_{st}, n_G)$ when $G = P(G_1, G_2)$. The resulting number of components n_G is the sum of n_{G_1} and n_{G_2} in all cases: A parallel merge cannot serve to join any components except for those in the open sets at either terminal (which are not counted in n_G), and cannot create new components. The recursions are given as follows, first for the case of $p_s = p_t = 1$:

$$f_G(1,1,0,n_G) = \min \quad f_{G_1}(1,1,0,n_{G_1}) + f_{G_2}(1,1,0,n_{G_2}) \tag{3–6a}$$
$$\text{s.t.} \quad n_G = n_{G_1} + n_{G_2}$$
$$f_G(1,1,1,n_G) = \min \quad f_{G_1}(1,1,c_{s_1 t_1},n_{G_1}) + f_{G_2}(1,1,c_{s_2 t_2},n_{G_2}) \tag{3–6b}$$
$$\text{s.t.} \quad c_{s_1 t_1} + c_{s_2 t_2} \geq 1$$
$$n_G = n_{G_1} + n_{G_2}.$$

In parallel synthesis operations, we must have $p_{s_1} = p_{s_2}$ and $p_{t_1} = p_{t_2}$ for the merge to be valid. Observe in Equality 3–6a that if $c_{s_1 t_1} = c_{s_2 t_2} = 0$, then s and t will be disconnected

after merging, because all s–t paths in the merged graph consist entirely of edges in G_1, or of edges in G_2 (but not a mixture of their edges). However, if $c_{s_1t_1} = 1$ or $c_{s_2t_2} = 1$ (as in Equality 3–6b), then s and t would remain connected by the same path as before, and so $c_{st} = 1$.

For the remaining cases, we have the following recursions:

$$f_G(0,1,0,n_G) = \min \quad f_{G_1}(0,1,0,n_{G_1}) + f_{G_2}(0,1,0,n_{G_2}) - 1 \tag{3–6c}$$
$$\text{s.t.} \quad n_G = n_{G_1} + n_{G_2}$$
$$f_G(1,0,0,n_G) = \min \quad f_{G_1}(1,0,0,n_{G_1}) + f_{G_2}(1,0,0,n_{G_2}) - 1 \tag{3–6d}$$
$$\text{s.t.} \quad n_G = n_{G_1} + n_{G_2}$$
$$f_G(0,0,0,n_G) = \min \quad f_{G_1}(0,0,0,n_{G_1}) + f_{G_2}(0,0,0,n_{G_2}) - 2 \tag{3–6e}$$
$$\text{s.t.} \quad n_G = n_{G_1} + n_{G_2}.$$

Note that $c_{st} = 0$ in these cases, because at least one of s or t is deleted. These recursions account for the fact that the deletion of node s (in Equality 3–6c), or of node t (in Equality 3–6d), or of both (in Equality 3–6e) is double-counted in the objective, and thus correct the sums accordingly. We now state the following lemmas related to the solution dominance issue.

Lemma 3. *Consider an SPG G, and two node deletion subsets X^1 and X^2 that respectively correspond to entries $f_G(p_s^1, p_t^1, c_{st}^1, n_G^1)$ and $f_G(p_s^2, p_t^2, c_{st}^2, n_G^2)$. If $p_s^1 = p_s^2,\ p_t^1 = p_t^2,\ c_{st}^1 \leq c_{st}^2,\ n_G^1 \geq n_G^2$, and $|X^1| \leq |X^2|$, then solution X^1 dominates X^2.*

Proof. When $c_{st}^1 = c_{st}^2$, Lemma 3 is obvious. When $c_{st}^1 = 0$ and $c_{st}^2 = 1$, O_s and O_t can be parts of two distinct components in a solution that deletes nodes X^1 over G, while O_s and O_t must belong to the same component in any solution in which X^2 is deleted over G. Hence, replacing X^2 with X^1 over G cannot reduce the total number of components in $\overline{G}$. □

Lemma 4. *Consider an SPG G, and two deletion solutions X^1 and X^2 that respectively correspond to entries $f_G(p_s^1, p_t^1, c_{st}^1, n_G^1)$ and $f_G(p_s^2, p_t^2, c_{st}^2, n_G^2)$. If $p_s^1 = p_s^2 = 1,\ p_t^1 = p_t^2 = 1,\ c_{st}^1 = 1,\ c_{st}^2 = 0,\ n_G^1 \geq n_G^2 + 1$, and $|X^1| \leq |X^2|$, then X^1 dominates X^2.*

Proof. Consider a node deletion solution $\overline{X}$ for $\overline{G}$, where $X^2 \subseteq \overline{X}$, and all nodes in G that are deleted in $\overline{X}$ belong to X^2, and an alternative solution $X' = (\overline{X} \setminus X^2) \cup X^1$. Let $\overline{C}_i$ denote the component to which $i \in V$ belongs after the deletion of nodes in $\overline{X}$. Note that in either solution $\overline{X}$ or X', $G \cup \overline{C}_s \cup \overline{C}_t$ is disconnected from the rest of graph $\overline{G}$, and thus solutions $\overline{X}$ and X' both induce the same number of components in $\overline{G} \setminus (G \cup \overline{C}_s \cup \overline{C}_t)$. Deleting X^2 creates at most $n_G^2 + 2$ components over $(G \cup \overline{C}_s \cup \overline{C}_t)$, while deleting X^1 instead creates $n_G^1 + 1$ components. Noting that $n_G^1 + 1 \geq n_G^2 + 2$ by assumption, X' creates at least as many components as $\overline{X}$ over $\overline{G}$, with $|X'| \leq |\overline{X}|$, and so X^1 dominates X^2. □

Based on Lemmas 3 and 4, we discard dominated solutions in our procedure, except for one arbitrary solution among each set of those that dominate each other.

3.3.2.3 Solution scheme and complexity analysis

The overall DP solution scheme for solving MaxNum on SPGs is given as follows.

Step 0 (Initialization). Given an SPG $\overline{G}$, execute the algorithm of Valdes et al. (1982) to obtain a corresponding binary decomposition tree $T(\overline{G})$. For each leaf node G in $T(\overline{G})$ (representing K_2), initialize $f_G(p_s, p_t, c_{st}, n_G)$ according to Equality 3–4. Label all non-leaf nodes as unexamined.

Step 1 (Examine a Node). If all nodes in $T(\overline{G})$ have been examined, go to Step 3. Otherwise, pick any unexamined node $G \in T(\overline{G})$ whose children have been examined.

Step 2 (Update Solution Matrices). Let G_1 and G_2 be the child nodes of G in $T(\overline{G})$. Compute $f_G(p_s, p_t, c_{st}, n_G)$ via Equality 3–5 or Equality 3–6 if $G = S(G_1, G_2)$ or $G = P(G_1, G_2)$, respectively, for all possible state value combinations of $p_s,\ p_t,\ c_{st}$, and n_G. For entries that are not feasible, or exceed the deletion budget B, or are dominated,

set their values to ∞. For each finite-valued entry $f_G(p_s, p_t, c_{st}, n_G)$, record arguments $(p_{s_1}, p_{t_1}, c_{s_1t_1}, n_{G_1})$ and $(p_{s_2}, p_{t_2}, c_{s_2t_2}, n_{G_2})$ that are used to compute the entry.

Step 3 (Report an Optimal Solution). At the root node $G_r \in T(\overline{G})$, choose a solution that minimizes node deletions among those that maximize the number of components. Observe that a finite-valued entry $f_{G_r}(p_{s_r}, p_{t_r}, c_{s_rt_r}, n_{G_r})$ corresponds to a solution yielding $n_{G_r} + p_{s_r} + p_{t_r} - c_{s_rt_r}$ components. Backtrack to determine an optimal set of node deletions as follows. Delete s_r (t_r) if $p_{s_r} = 0$ ($p_{t_r} = 0$). Then consider child node graphs G_{r_1} and G_{r_2} of G_r in $T(\overline{G})$, and examine the recorded arguments that result in an optimal $f_{G_r}(p_{s_r}, p_{t_r}, c_{s_rt_r}, n_{G_r})$-value. Note that if $G_r = P(G_{r_1}, G_{r_2})$, we do not need to delete any more nodes because $s_r = s_{r_1} = s_{r_2}$ and $t_r = t_{r_1} = t_{r_2}$. If $G_r = S(G_{r_1}, G_{r_2})$, then if $p_{t_{r_1}} = p_{s_{r_2}} = 0$, we delete node $t_{r_1} = s_{r_2}$, and otherwise we do not. Recursively apply this procedure to identify all deleted nodes. □

Note that at each node G in $T(\overline{G})$, we update recursive functions f_G by merging its children solutions. Values of n_G are bounded by the number of nodes, n. Because there exist at most $5n$ combinations of parameters $p_s,\ p_t,\ c_{st}$, and n_G, computing each value of $f_G(p_s, p_t, c_{st}, n_G)$ from its two child nodes takes $O(n^2)$ steps. The corresponding decomposition tree $T(\overline{G})$ can be obtained in linear time, and thus the number of nodes G in $T(\overline{G})$ is $O(n)$. The total algorithmic complexity is therefore $O(n^3)$.

To determine the DP space complexity, observe that for each $G \in T(\overline{G})$, we store the f-entries and arguments that compute these optimal values. This requires $O(n)$ elements to be stored for each $G \in T(\overline{G})$. Because $T(\overline{G})$ has $O(n)$ nodes, the space complexity of our procedure is $O(n^2)$.

3.3.3 Solving MinMaxC on SPGs

Defining similar recursions used for solving MaxNum on SPGs, we can develop an $O(n^7)$ algorithm for solving MinMaxC, whose details are given in Appendix C. Instead, we propose a binary search algorithm for solving MinMaxC on SPGs that is similar to the one we prescribed for trees. Let τ represent a fixed target for the maximum

component size, and let $f_{G_i}(o_{s_i}, o_{t_i}, c_{s_i t_i}, \tau)$ denote the fewest number of deletions that achieves an open set size of o_{s_i} at source s_i, size of o_{t_i} at sink t_i with connection parameter $c_{s_i t_i}$, and the largest component size (including the open sets) that is no larger than τ. Accordingly, we initialize the leaf node graphs K_2 of the decomposition tree as

$$f_{K_2}(2,2,1,\tau) = 0,\ f_{K_2}(0,1,0,\tau) = 1,\ f_{K_2}(1,0,0,\tau) = 1,\ f_{K_2}(0,0,0,\tau) = 2, \tag{3–7}$$

assuming that $\tau \geq 2$, and set all other f-values as ∞. (If $\tau = 1$, then $f_{K_2}(2,2,1,1) = \infty$ for all leaf node graphs K_2 of $T(\overline{G})$.)

3.3.3.1 The series operation

When $G = S(G_1, G_2)$, we provide the following recursions to update f_G-elements. Equations 3–8a through 3–8c describe the cases of both terminals being deleted, only source s being deleted, and only sink t being deleted, respectively. These three recursions take the minimum of two minimization subproblems: One in which node t_1 $(= s_2)$ is deleted, and the other in which it is not. The recursions are given by:

$$f_G(0,0,0,\tau) = \min \begin{cases} \min & f_{G_1}(0,0,0,\tau) + f_{G_2}(0,0,0,\tau) - 1; \\ \min & f_{G_1}(0,o_{t_1},0,\tau) + f_{G_2}(o_{s_2},0,0,\tau) \\ \text{s.t.} & o_{t_1} + o_{s_2} - 1 \leq \tau \end{cases} \tag{3–8a}$$

$$f_G(0,o_t,0,\tau) = \min \begin{cases} \min & f_{G_1}(0,0,0,\tau) + f_{G_2}(0,o_{t_2},0,\tau) - 1 \\ \text{s.t.} & 1 \leq o_t = o_{t_2}; \\ \min & f_{G_1}(0,o_{t_1},0,\tau) + f_{G_2}(o_{s_2},o_{t_2},c_{s_2 t_2},\tau) \\ \text{s.t.} & 1 \leq o_t = o_{t_2} + (o_{t_1} - 1)c_{s_2 t_2} \leq \tau \\ & o_{t_1} + o_{s_2} - 1 \leq \tau \end{cases} \tag{3–8b}$$

$$f_G(o_s, 0, 0, \tau) = \min \begin{cases} \min & f_{G_1}(o_{s_1}, 0, 0, \tau) + f_{G_2}(0, 0, 0, \tau) - 1 \\ \text{s.t.} & 1 \le o_s = o_{s_1}; \\ \\ \min & f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1 t_1}, \tau) + f_{G_2}(o_{s_2}, 0, 0, \tau) \\ \text{s.t.} & 1 \le o_s = o_{s_1} + (o_{s_2} - 1) c_{s_1 t_1} \le \tau \\ & o_{t_1} + o_{s_2} - 1 \le \tau. \end{cases} \tag{3–8c}$$

In the objectives of Equality 3–8a through 3–8c, we subtract 1 from the summation of f_{G_1} and f_{G_2} if $o_{t_1} = o_{s_2} = 0$ due to the double-counted deletion of node $t_1 = s_2$. Also, when node $t_1 = s_2$ is not deleted, the merging operation forms a new component that has size $o_{t_1} + o_{s_2} - 1$ in the case of Equality 3–8a due to the combination of O_{t_1} and O_{s_2}. In Equality 3–8b (or Equality 3–8c), this merged component is part of O_t (O_s) when s_2 and t_2 (s_1 and t_1) are connected.

For the case in which there exist non-empty open sets at the source and the sink of the merged graph, we explore the case in which s_1 is disconnected from t_2 separately from the case in which the two are connected. First, if s_1 is disconnected from t_2, we have (similar to Equality 3–8b and Equality 3–8c):

$$f_G(o_s, o_t, 0, \tau) = \min \begin{cases} \min & f_{G_1}(o_{s_1}, 0, 0, \tau) + f_{G_2}(0, o_{t_2}, 0, \tau) - 1; \\ \text{s.t.} & 1 \le o_s = o_{s_1} \\ & 1 \le o_t = o_{t_2}; \\ \\ \min & f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1 t_1}, \tau) + f_{G_2}(o_{s_2}, o_{t_2}, c_{s_2 t_2}, \tau) \\ \text{s.t.} & c_{s_1 t_1} + c_{s_2 t_2} \le 1 \\ & 1 \le o_s = o_{s_1} + (o_{s_2} - 1) c_{s_1 t_1} \le \tau \\ & 1 \le o_t = o_{t_2} + (o_{t_1} - 1) c_{s_2 t_2} \le \tau \\ & o_{t_1} + o_{s_2} - 1 \le \tau \\ & o_{s_2},\ o_{t_1} \ge 1. \end{cases} \tag{3–8d}$$

The first minimization subproblem in Equality 3–8d handles the case in which $t_1 = s_2$ is deleted, and the second one considers the case in which $t_1 = s_2$ is not deleted. Note that in the second case, a path exists from $s = s_1$ to $t = t_2$ if and only if $c_{s_1 t_1} = c_{s_2 t_2} = 1$, and thus we constrain $c_{s_1 t_1} + c_{s_2 t_2}$ to be no more than 1. The remaining constraints are similar to those in Equality 3–8b and 3–8c.

If s_1 is connected to t_2, then

$$\begin{aligned} f_G(o_s, o_s, 1, \tau) = \min \quad & f_{G_1}(o_{s_1}, o_{s_1}, 1, \tau) + f_{G_2}(o_{s_2}, o_{s_2}, 1, \tau) \\ \text{s.t.} \quad & 1 \leq o_s = o_{s_1} + o_{s_2} - 1 \leq \tau. \end{aligned} \tag{3–8e}$$

Here we require $c_{s_1 t_1} = c_{s_2 t_2} = 1$ to ensure that s and t are connected. Note that $o_{s_1} = o_{t_1}$ ($o_{s_2} = o_{t_2}$) when $c_{s_1 t_1} = 1$ ($c_{s_2 t_2} = 1$). Since $c_{st} = 1$, the new open sets O_s and O_t are identical given by $O_{s_1} \cup O_{s_2}$, and have $o_{s_1} + o_{s_2} - 1$ nodes.

3.3.3.2 The parallel operation

Now consider the computation of $f_G(o_s, o_t, c_{st}, \tau)$ when $G = P(G_1, G_2)$. We give the recursions for updating elements for which $0 \leq o_s \leq \tau,\ 0 \leq o_t \leq \tau$ as below, first for the cases in which at least one of s and t is deleted (and thus $c_{st} = 0$):

$$f_G(0, 0, 0, \tau) = \min \quad f_{G_1}(0, 0, 0, \tau) + f_{G_2}(0, 0, 0, \tau) - 2 \tag{3–9a}$$

$$\begin{aligned} f_G(o_s, 0, 0, \tau) = \min \quad & f_{G_1}(o_{s_1}, 0, 0, \tau) + f_{G_2}(o_{s_2}, 0, 0, \tau) - 1 \\ \text{s.t.} \quad & 1 \leq o_s = o_{s_1} + o_{s_2} - 1 \leq \tau \\ & o_{s_1},\ o_{s_2} \geq 1 \end{aligned} \tag{3–9b}$$

$$\begin{aligned} f_G(0, o_t, 0, \tau) = \min \quad & f_{G_1}(0, o_{t_1}, 0, \tau) + f_{G_2}(0, o_{t_2}, 0, \tau) - 1 \\ \text{s.t.} \quad & 1 \leq o_t = o_{t_1} + o_{t_2} - 1 \leq \tau \\ & o_{t_1},\ o_{t_2} \geq 1. \end{aligned} \tag{3–9c}$$

These recursions subtract 2 from the objective of Equality 3–9a, and 1 from the objectives of Equality 3–9b and Equality 3–9c, to avoid double-counting terminal node

deletions. We merge O_{s_1} and O_{s_2} to form O_s in Equality 3–9b, and perform a similar operation for O_t in Equality 3–9c.

When neither source s nor sink t is deleted, we consider the following two cases. If s and t are not connected, we have (similar to Equality 3–9b and Equality 3–9c)

$$\begin{aligned} f_G(o_s, o_t, 0, \tau) = \min \quad & f_{G_1}(o_{s_1}, o_{t_1}, 0, \tau) + f_{G_2}(o_{s_2}, o_{t_2}, 0, \tau) \\ \text{s.t.} \quad & 1 \le o_s = o_{s_1} + o_{s_2} - 1 \le \tau \\ & 1 \le o_t = o_{t_1} + o_{t_2} - 1 \le \tau \\ & o_{s_1},\ o_{s_2},\ o_{t_1},\ o_{t_2} \ge 1. \end{aligned} \tag{3–9d}$$

Note that to impose $c_{st} = 0$, we have $c_{s_1t_1} = c_{s_2t_2} = 0$. If node s is connected to t, then

$$\begin{aligned} f_G(o_s, o_s, 1, \tau) = \min \quad & f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1t_1}, \tau) + f_{G_2}(o_{s_2}, o_{t_2}, c_{s_2t_2}, \tau) \\ \text{s.t.} \quad & c_{s_1t_1} + c_{s_2t_2} \ge 1 \\ & 1 \le o_s = o_{s_1} + o_{s_2} - 2 + o_{t_1}(1 - c_{s_1t_1}) + o_{t_2}(1 - c_{s_2t_2}) \le \tau \\ & o_{s_1},\ o_{s_2},\ o_{t_1},\ o_{t_2} \ge 1. \end{aligned} \tag{3–9e}$$

Recursion 3–9e requires either $c_{s_1t_1} = 1$ or $c_{s_2t_2} = 1$ so that $c_{st} = 1$, which also implies that $o_s = o_t$. We consider three possible combinations of $c_{s_1t_1}$ and $c_{s_2t_2}$ values. If $c_{s_1t_1} = 0$ and $c_{s_2t_2} = 1$, then $O_{s_2} = O_{t_2}$, and O_{s_1} is merged with O_{s_2} and O_{t_1}. The size of this component is $o_{s_1} + o_{s_2} + o_{t_1} - 2$ in this case, because s_1 belongs to both O_{s_1} and O_{s_2}, and t belongs to both O_{s_2} and O_{t_1}. The case of $c_{s_1t_1} = 1$ and $c_{s_2t_2} = 0$ is symmetric to the one above. Finally, if $c_{s_1t_1} = c_{s_2t_2} = 1$, then $O_{s_1} = O_{t_1}$ and $O_{s_2} = O_{t_2}$. We merge O_{s_1} with O_{s_2} in this case, and subtract 2 from $o_{s_1} + o_{s_2}$ because s and t belong to both O_{s_1} and O_{s_2}.

In order to avoid updating unnecessary f_i-values, we now state a solution dominance criteria as follows, whose proof is obvious and thus omitted.

Lemma 5. *Consider an SPG G, and two deletion node subsets X^1 and X^2 that respectively correspond to entries $f_G(o_s^1, o_t^1, c_{st}^1, \tau)$ and $f_G(o_s^2, o_t^2, c_{st}^2, \tau)$, such that $c_{st}^1 = c_{st}^2,\ o_s^1 \leq o_s^2,\ o_t^1 \leq o_t^2$, and $|X^1| \leq |X^2|$. Then X^1 dominates X^2.*

Note that Lemma 5 requires $c_{st}^1 = c_{st}^2$. Perhaps surprisingly, this dominance criteria does not extend to the case in which $c_{st}^1 \neq c_{st}^2$. For entries that are not feasible, or require more than B deletions, or are dominated as indicated in Lemma 5, we again set the corresponding values of $f_G(o_s, o_t, c_{st}, \tau)$ to ∞.

3.3.3.3 Binary search scheme complexity analysis

We employ a binary search algorithm for solving MinMaxC on SPGs with the following modifications to the one presented in Section 3.2.2. In Step I, we construct a decomposition tree $T(\overline{G})$, and compute $f_G(o_s, o_t, c_{st}, \tau)$ at every node $G \in T(\overline{G})$. This computation follows similar DP steps for solving MaxNum on SPGs in Section 3.3.2.3 with the following exceptions. In Step 0, we initialize leaf graphs in $T(\overline{G})$ according to Equality 3–7. In Step 2, we compute $f_G(o_s, o_t, c_{st}, \tau)$ by using Equality 3–8 for series operations and Equality 3–9 for parallel operations. We recover incumbent solutions via Step 3.

There are at most $O(n^2)$ $f_G(o_s, o_t, c_{st}, \tau)$ entries for a fixed τ (for the $O(n)$ possible values of o_s and o_t), and thus merging solutions at each $G \in T(\overline{G})$ takes $O(n^4)$ steps. Because there are $O(n)$ nodes in $T(\overline{G})$, Step I in Section 3.2.2 requires $O(n^5)$ steps for a given τ. The maximum number of binary search steps is $O(\log n)$, leading to an $O(n^5 \log n)$ time complexity algorithm for solving MinMaxC on SPGs.

To compute the space complexity, for a fixed τ, we need to store $O(n^2)$ $f_G(o_s, o_t, c_{st}, \tau)$ entries at each of the $O(n)$ subgraph nodes $G \in T(\overline{G})$, including the six arguments that lead to the computation of these values. The total space complexity is thus $O(n^3)$.

3.4 Computational Results

We test the efficacy of our MaxNum and MinMaxC DP algorithms on randomly-generated tree and SPG instances. We intend to determine: (i) the average tree- and SPG-sizes

our algorithms can optimize within a stipulated time limit; (ii) how the average CPU time of each case is affected by varying B-values; (iii) how much CPU time can be saved by applying Lemmas 1 through 5. For solving MinMaxC on SPGs, we also report the details of binary search steps, to better understand the effects of varying B-values on CPU time.

To generate trees, we specify the number of levels in the tree. We generate twenty tree instances having $10,\ 11$, and 12 levels each, and refer to a tree instance containing **w** levels as **T-w**. For each instance, we start from a root node, and randomly generate one, two, or three child nodes to the root. We then recursively apply this procedure at each child node, until reaching leaf nodes at level w-1. The generated 10-, 11-, and 12-level tree instances contain an average of 1004.39, 2083.16, and 4102.45 nodes, respectively.

To generate SPGs, we first specify the number of nodes contained in an instance. We use 500-, 1000-, and 2000-node graphs, and generate twenty instances for each size. Let **SPG-y** refer to the instances having **y** nodes. We declare two nodes to be the SPG terminals, and randomly generate an integer from 1 to 4, representing the number of branches at each terminal node. We then evenly distribute all remaining nodes into each branch (noting that some branches may contain one more node than others). For instance, for a 30-node instance, let node 1 and node 30 be the terminal nodes, and suppose that we generate four branches between node 1 and node 30. We assign node sets $\{2, \dots, 8\},\ \{9, \dots, 15\},\ \{16, \dots, 22\}$, and $\{23, \dots, 29\}$ to these branches, and recursively build SPGs for the seven-node instance in each branch. There are two special cases in this procedure: If there are no nodes in a branch, then we connect the two terminals associated with that branch. If there is one node in the branch, we connect the two terminals to this node.

3.4.1 The CPU Time for Various B-Values

With respect to different instance sizes, we first present the computational results of solving MaxNum and MinMaxC on trees and SPGs in Tables 3-1 and 3-2, by examining various budget values B. Each entry in column **A** and column **W** represents the average and the worst CPU seconds required to obtain an optimal solution among all twenty instances generated for the corresponding size-varied tree (or SPG). All algorithms were implemented by using the C++ programming language, and all computations were performed on a Dell PowerEdge 2600 UNIX machine with two Pentium 4 3.2 GHz (1M Cache) processors, 6.0 GB memory, and Red Hat Version 5.0 installed.

Table 3-1. The average and worst CPU seconds for solving MaxNum and MinMaxC on trees.

	MaxNum						MinMaxC					
B	T-10		T-11		T-12		T-10		T-11		T-12	
	A	W	A	W	A	W	A	W	A	W	A	W
1	0.00	0.01	0.01	0.02	0.03	0.05	0.03	0.05	0.08	0.12	0.23	0.35
2	0.00	0.01	0.02	0.03	0.05	0.08	0.05	0.09	0.16	0.23	0.39	0.64
3	0.00	0.03	0.02	0.04	0.07	0.10	0.05	0.09	0.20	0.31	0.56	0.75
4	0.00	0.02	0.03	0.05	0.10	0.14	0.07	0.10	0.24	0.35	0.74	1.03
5	0.01	0.03	0.03	0.04	0.13	0.20	0.09	0.16	0.37	0.52	0.89	1.22
6	0.01	0.03	0.04	0.06	0.15	0.23	0.11	0.18	0.43	0.59	1.07	1.69
7	0.01	0.03	0.04	0.06	0.17	0.29	0.13	0.21	0.47	0.66	1.27	1.72
8	0.01	0.02	0.04	0.08	0.20	0.34	0.12	0.21	0.49	0.69	1.38	1.89
9	0.01	0.02	0.04	0.07	0.25	0.32	0.14	0.20	0.54	0.77	1.67	2.57
10	0.01	0.03	0.06	0.09	0.26	0.34	0.17	0.23	0.60	0.81	2.04	3.20
11	0.02	0.03	0.05	0.07	0.27	0.34	0.19	0.27	0.64	0.86	2.18	3.06
12	0.02	0.04	0.06	0.08	0.29	0.38	0.20	0.29	0.74	1.03	2.58	3.42
13	0.02	0.04	0.07	0.11	0.30	0.40	0.24	0.38	0.91	1.21	2.75	3.76
14	0.02	0.03	0.08	0.13	0.29	0.38	0.27	0.36	1.12	1.53	2.97	3.98

We observe that solving MaxNum takes less CPU time than solving MinMaxC on tree instances of the same size, and that for either MaxNum and MinMaxC, our algorithms consume more CPU time for solving instances on SPGs than on trees. These observations are consistent with the worst-case algorithmic complexity analysis of each case. Next, we note that the CPU time generally increases as we increase the budget, since the corresponding f-matrix becomes denser as B increases, and thus

Table 3-2. The average and worst CPU seconds for solving MaxNum and MinMaxC on SPGs.

B	MaxNum						MinMaxC					
	SPG-500		SPG-1000		SPG-2000		SPG-500		SPG-1000		SPG-2000	
	A	W	A	W	A	W	A	W	A	W	A	W
1%$\|\mathcal{V}\|$	0.06	0.08	0.49	0.83	1.53	1.88	4.35	5.23	15.75	19.23	84.48	106.83
2%$\|\mathcal{V}\|$	0.11	0.13	0.74	1.00	4.39	5.31	3.76	4.47	18.52	21.55	137.33	162.12
3%$\|\mathcal{V}\|$	0.17	0.21	1.03	1.17	7.61	8.84	3.75	4.24	22.59	25.78	221.12	262.76
4%$\|\mathcal{V}\|$	0.24	0.29	1.54	1.75	12.31	14.34	3.75	4.33	28.71	32.86	240.76	282.66
5%$\|\mathcal{V}\|$	0.35	0.41	2.18	2.59	20.67	23.77	3.65	4.42	27.40	31.79	256.03	294.75
6%$\|\mathcal{V}\|$	0.43	0.53	2.99	3.50	24.33	30.28	3.66	4.31	28.62	32.69	264.46	298.63
7%$\|\mathcal{V}\|$	0.51	0.57	3.72	4.33	28.85	32.99	3.67	4.25	24.44	28.75	252.42	306.85
8%$\|\mathcal{V}\|$	0.60	0.69	4.70	5.27	33.07	36.98	3.53	4.08	22.99	25.55	242.39	299.61
9%$\|\mathcal{V}\|$	0.71	0.79	5.72	6.39	36.12	40.90	3.14	3.59	21.01	23.90	209.11	248.25
10%$\|\mathcal{V}\|$	0.80	0.97	5.54	6.40	46.91	53.45	3.07	3.47	21.97	24.95	190.66	218.50
11%$\|\mathcal{V}\|$	0.87	1.00	6.25	7.03	51.63	58.54	2.81	3.14	19.31	22.41	172.22	191.47
12%$\|\mathcal{V}\|$	0.94	1.08	7.29	8.17	59.04	65.97	2.65	2.99	18.66	21.51	160.36	193.45
13%$\|\mathcal{V}\|$	1.07	1.20	7.80	9.11	69.81	79.28	2.57	2.90	19.00	23.57	157.89	178.35
14%$\|\mathcal{V}\|$	1.07	1.22	8.90	9.97	70.76	81.69	2.47	2.81	17.14	19.13	149.01	170.73

our DP algorithms must expend more effort in examining more pairs of finite-valued f-entries. One exception to this trend appears to arise in the performance of MinMaxC on SPGs, where the CPU time required to solve these instances increases with B initially, and then decreases after B becomes 7%–10% of the number of nodes.

Remark 3.3. Note that in general DP solution times primarily depend on the number of active states involved during the updates of recursive functions. To see the effects of using Lemmas 1 through 5, we also test algorithms having no dominated-state identification procedures on the same tree- and SPG-instances. The results show that the domination criteria presented in these lemmas are critical in improving the efficacy of our DP algorithms. For instance, without Lemmas 1 and 2, our DP algorithm solves MaxNum on 1000-node (10-level) tree-instances in an average of 1.5–2.5 seconds, while it solves the same instances in 0.01–0.03 seconds in the worst case (shown in Table 3-1) with the use of these lemmas. Lemmas 3 and 4 help to decrease the CPU time required to solve MaxNum on 500-node SPGs from an average of 7–10 seconds to an average of 0.1–1.2 seconds. The effect of applying Lemma 5 is the most significant.

For instance, when B is in the range of 7%–10% of the number of nodes, our approach solves 2000-node SPGs within 150–250 seconds by using this lemma, but can only solve 120-node SPGs in 600–800 seconds without the lemma.

3.4.2 Binary Search Details for Solving MinMaxC on SPGs

In order to determine why the CPU time does not increase as B increases when solving MinMaxC on SPGs, we compute the CPU time (in seconds) required at each binary search step for instance SPG-2000-1 with $B = 20,\ 100,\ 180$, and 260, reported in column **S-Time** (illustrated in Table 3-3). We also record the lower bound *LB*, the upper bound *UB*, and the τ-value associated with each step. Column **Found** indicates whether or not a solution is found whose maximum component size is no more than τ.

Table 3-3. Computational details in solving MinMaxC on SPG-2000-1.

Step	B=20 (CPU = 92s, Obj = 62)					Step	B=100 (CPU = 234s, Obj = 14)				
	LB	UB	τ	S-Time	Found		LB	UB	τ	S-Time	Found
1	1	1980	990	46	yes	1	1	1900	950	182	yes
2	1	928	464	24	yes	2	1	934	467	38	yes
3	1	254	127	10	yes	3	1	423	211	9	yes
4	1	127	64	3	yes	4	1	203	102	3	yes
5	1	64	32	0	no	5	1	84	42	1	yes
6	33	64	48	1	no	6	1	31	16	0	yes
7	49	64	56	1	no	7	1	16	8	0	no
8	57	64	60	1	no	8	9	16	12	0	no
9	61	64	62	1	yes	9	13	16	14	0	yes
10	61	62	61	1	no	10	13	14	13	0	no
Step	B=180 (CPU = 187s, Obj = 6)					Step	B=260 (CPU = 163s, Obj = 6)				
	LB	UB	τ	S-Time	Found		LB	UB	τ	S-Time	Found
1	1	1820	910	146	yes	1	1	1780	890	130	yes
2	1	900	450	29	yes	2	1	885	442	24	yes
3	1	437	218	7	yes	3	1	437	218	6	yes
4	1	203	102	2	yes	4	1	215	108	2	yes
5	1	100	50	1	yes	5	1	100	50	1	yes
6	1	30	15	0	yes	6	1	29	15	0	yes
7	1	15	8	0	yes	7	1	14	7	0	yes
8	1	8	4	0	no	8	1	7	4	0	no
9	5	8	6	0	yes	9	5	7	6	0	yes
10	5	6	5	0	no	10	5	6	5	0	no

In Table 3-3, we note that there are 10 binary search steps required for each of these instances. At each step, note that as we increase B, the target maximum component size τ tends to decrease. This behavior is intuitive, since a larger B typically allows us to obtain a more disconnected network, which accelerates the rate at which *UB* is decreased. Hence, the additional effort incurred due to larger B-values (which in turn tend to increase the number of non-dominated f-values in the DP procedure) is offset somewhat by the smaller τ-values. Our results indicate that after B becomes 7%–10% of the number of nodes, the effort required to execute the binary search steps tends to decrease due to the decreased τ-values.

CHAPTER 4
CRITICAL NODE PROBLEMS ON GENERAL GRAPHS

4.1 Problem Description and Literature Survey

Considering the node deletion problems we considered in Chapter 3, we extend our discussion of solving MaxNum and MinMaxC on specially-structured graphs to *general* undirected graphs $G(\mathcal{V}, \mathcal{E})$. Moreover, we in this chapter consider a third network-connectivity metric as the minimum cost to connect the graph after node deletions, given a set of edge construction costs. We refer to this problem as MaxMinLR. If there exist alternative optima for these problems, we define an optimal solution as one requiring the fewest node deletions. In our analysis, we begin by assuming that deleted nodes are ignored in the network-connectivity metrics, and discuss the modification required to handle the case in which deleted nodes must also be reconnected.

As previously reviewed, applications related to the problems that we study in this chapter have been conducted in areas of telecommunication (Resende & Pardalos, 2006), social network activities (Borgatti, 2006), homeland security (Brown et al., 2006), and epidemic control (Zhou et al., 2006). In particular, node-deletion problems on general graphs have significant relevance to analyzing the attack tolerance of complex networks (Alderson, 2008), and of small-world networks (Watts & Strogatz, 1998), which have received substantial attention in the past decade. (Appendix D reviews literature on the evolution of studies on this topic.)

An intuitive approach to solving the problems we consider in this chapter is to greedily delete a node having the largest degree in the current graph along with its adjacent edges, and reiterate until B nodes have been deleted (e.g., Albert et al. (2000); Tu (2000)). However, this algorithm does not generally yield even a constant-factor polynomial-time approximation scheme. Figure 4-1 depicts a MaxNum instance with $B = 1$, in which the greedy algorithm removes the gray node, resulting in one component (excluding the deleted node). However, the optimal solution removes the

black node, yielding four components. Figure 4-2 depicts a MinMaxC instance with $B = 1$, where the greedy algorithm deletes the highest-degree node (colored gray), leaving a largest component that has 16 nodes. However, the optimal solution deletes the black node, which results in a smaller largest-component size of five.

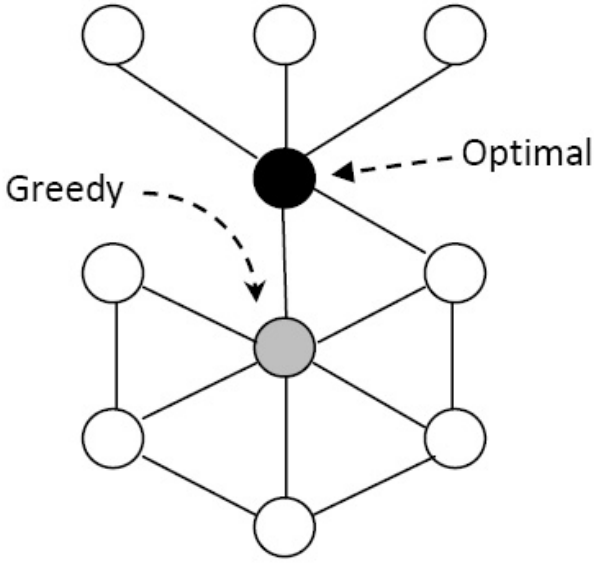


Figure 4-1. Suboptimality of the greedy algorithm in MaxNum for $B = 1$.

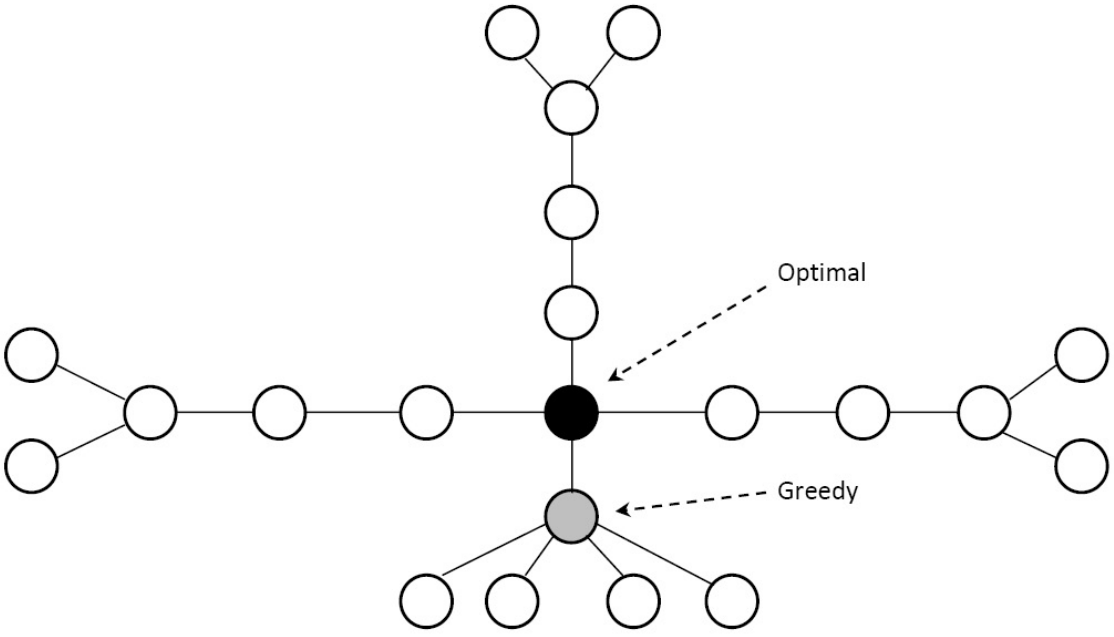


Figure 4-2. Suboptimality of the greedy algorithm in MinMaxC for $B = 1$.

By contrast, we explore an exact optimization algorithm for these problems in this chapter. Our central approach for these problems is inspired by bilevel optimization techniques used in network interdiction models (e.g., Arroyo (2010); Brown et al. (2006); Cormican et al. (1998); Lim & Smith (2007); San Martin (2007); Smith et al. (2007); Smith & Lim (2008); Wood (1993)). Of particular relevance to our study, Akgun (2000) analyzes an interdiction algorithm that minimizes the maximum pairwise sums of flows from K node groups on an undirected graph $G(N, A)$ having capacitated edges. In their problem, a set of K disjoint node groups exists with $N'_1 \cup \cdots \cup N'_K \subseteq N$, and the maximum flow objective is given by $\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \delta_{ij}$, where δ_{ij} is the maximum flow possible from nodes in N'_i to nodes in N'_j. Their approach employs multi-commodity flow interdiction analysis to solve a min-max bilevel programming problem.

Most node deletion problems are $\mathcal{NP}$-complete on general graphs (e.g., Arulselvan et al. (2009); Dinh et al. (2010); Di Summa et al. (2010); Kerivin & Mahjoub (2005)). Due to Theorem 1 in Yannakakis (1978), MinMaxC is $\mathcal{NP}$-hard in the strong sense. We show in this chapter that both MaxNum and MaxMinLR are strongly $\mathcal{NP}$-hard as well.(They become polynomially solvable when restricted to certain special graph types as we demonstrated in Chapter 3.)

In this chapter, we formulate MaxNum, MinMaxC, and MaxMinLR on general graphs as two-stage network interdiction models. We then transform each model into an integrated mixed integer program (MIP). For MaxNum and MinMaxC, we derive a class of valid inequalities designed to improve the solvability of our MIPs, based on polynomial-time DP solutions of subgraphs derived from G (e.g., Hartman et al. (2010)).

The remainder of the chapter is organized as follows. We begin by formulating MIP models for MaxNum, MinMaxC, and MaxMinLR in Section 4.2. Section 4.3 demonstrates how to utilize a DP approach for solving MaxNum and MinMaxC on k-hole subgraphs of G to obtain objective bounds and valid inequalities for the corresponding

MIP models. We examine the computational performance of our approach in Section 4.4.

4.2 Complexity Analysis and MIP Formulations

We formulate the three node deletion problems as bilevel min-max (or max-min) programs, and prove that each is strongly $\mathcal{NP}$-hard. We then demonstrate how to obtain an integrated MIP model for each node deletion problem.

4.2.1 MaxNum

We begin by considering MaxNum, which maximizes the number of components after deleting a subset of nodes. (Recall that deleted nodes are not treated as their own components.) We first show that this problem is $\mathcal{NP}$-hard in the strong sense.

Denote $\overline{\text{MaxNum}}$ as the decision version of MaxNum, which seeks to delete a subset of no more than B nodes, such that the number of components in G (not including nodes that are deleted) is at least some given integer target value $\mathcal{T}$.

Theorem 4.1. *$\overline{\text{MaxNum}}$ is $\mathcal{NP}$-complete in the strong sense.*

Proof. $\overline{\text{MaxNum}}$ clearly belongs to $\mathcal{NP}$: Given a subset of (no more than B) nodes that have been deleted from G, we eliminate all of their incident edges. We then compute the number of components in this remaining graph using a polynomial-time search algorithm (Ahuja et al., 1993), and check whether the number of components is at least $\mathcal{T}$.

Next, we show that $\overline{\text{MaxNum}}$ is $\mathcal{NP}$-complete by using a transformation from INDEPENDENT SET (IS) (Garey & Johnson, 1979) stated as follows: Given a graph $G'(V', E')$, does there exist a subset $S \subseteq V'$ having k nodes, such that no pair of nodes in S is adjacent? We transform an IS instance into a $\overline{\text{MaxNum}}$ instance with exactly the same graph $G'(V', E')$. Let node deletion budget $B = |V'| - k$, and the target $\mathcal{T} = k$.

Suppose that the IS instance has a solution S with $|S| = k$. We construct a solution to $\overline{\text{MaxNum}}$ by deleting all nodes in $V' \setminus S$. Because S is an independent set, no edge exists between any pair of nodes in S, and thus no edges remain when all nodes in

$V' \setminus S$ are deleted. Each remaining node is a singleton component, and thus there are k components in G'.

Now suppose that the IS instance has no solution. For any set of nodes $Q \subseteq V'$ with $|Q| \leq B$ that we delete (along with their incident edges), there exists at least one edge in the remaining graph (or else $V' \setminus Q$ is a feasible solution to IS). The total number of components is then at most $k - 1 < \mathcal{T}$.

Because IS is $\mathcal{NP}$-complete in the strong sense, and we have used only polynomially-bounded numerical data in our transformation, we have shown that $\overline{\text{MaxNum}}$ is $\mathcal{NP}$-complete in the strong sense as well. □

Remark 4.1. Note that an alternative version of MaxNum, which retains all deleted nodes as isolated components in the remaining graph, is also strongly $\mathcal{NP}$-hard. We can modify Theorem 4.1 for this case by using the same transformation, keeping $B = |V'| - k$, and setting $T = |V'|$. □

Now, to formulate MaxNum as a MIP, define binary variables $x_i, \ \forall i \in \mathcal{V}$, such that $x_i = 1$ if node i is *not* deleted, and $x_i = 0$ otherwise. Also, define $y_{ij} \in \{0, 1\}, \ \forall (i, j) \in \mathcal{E}$, such that $y_{ij} = 0$ if edge $(i, j) \in \mathcal{E}$ is deleted (due to the deletions of nodes i, j, or both), and $y_{ij} = 1$ otherwise. Note that $y_{ij} = x_i x_j$, i.e., an edge is not deleted if and only if both of its incident nodes are not deleted. Let $\eta(x, y)$ be the number of components remaining in G given a deletion solution (x, y). We give MaxNum as

$$\begin{aligned}
\max \quad & \eta(x, y) - \frac{1}{n} \sum_{i \in \mathcal{V}} (1 - x_i) && \text{(4–1a)} \\
\text{s.t.} \quad & \sum_{i \in \mathcal{V}} (1 - x_i) \leq B && \text{(4–1b)} \\
& x_i + x_j - 1 \leq y_{ij} \quad \forall (i, j) \in \mathcal{E} && \text{(4–1c)} \\
& x_i \in \{0, 1\} \quad \forall i \in \mathcal{V} && \text{(4–1d)} \\
& 0 \leq y_{ij} \leq 1 \quad \forall (i, j) \in \mathcal{E}. && \text{(4–1e)}
\end{aligned}$$

In Objective 4–1a, we introduce a penalty term $-1/n \sum_{i \in \mathcal{V}} (1 - x_i)$ such that if there exists more than one solution that maximizes the number of components, a solution having the fewest deleted nodes is chosen as an optimal solution. Constraint 4–1b limits the number of deleted nodes to be no more than B, and Constraints 4–1c force $y_{ij} = 1$ if $x_i = x_j = 1$, for all $(i, j) \in \mathcal{E}$. Otherwise, if $x_i = 0$ or $x_j = 0$, y_{ij} will correctly take on a value of 0 in some optimal solution, and thus no further constraints are required to enforce the condition that $y_{ij} = x_i x_j$.

We next formulate the problem of calculating $\eta(x, y)$ using a MIP model on an auxiliary network, and show that the linear programming (LP) relaxation of this formulation yields a convex hull representation of the problem, given binary y-values.

Let $\widetilde{G}(\mathcal{V} \cup \{0\}, \mathcal{A})$ denote a transformed directed network, where node 0 will act as a dummy source node. Set $\mathcal{A}$ consists of two directed arcs (i, j) and (j, i) for all edges $(i, j) \in \mathcal{E}$. Our approach requires that a path must exist in $\widetilde{G}$ from node 0 to each node i, for every $i \in \mathcal{V} : x_i = 1$. Define $\widetilde{\mathcal{V}} = \{i \in \mathcal{V} : x_i = 1\}$ as the set of active nodes, and let $\{(0, i),\ \forall i \in \mathcal{V}\}$ be a set of potential arcs, each of which costs one unit to construct. Let $\mathcal{A}^*$ be a minimum-cardinality potential arcs, such that there exists a path from node 0 to $i, \forall i \in \widetilde{\mathcal{V}}$, using arcs in $\mathcal{A} \cup \mathcal{A}^*$. Then the number of graph components equals $|\mathcal{A}^*|$.

Define $FS(i) = \{j : (i, j) \in \mathcal{A}\}$ as the set of nodes adjacent from node i, and $RS(i) = \{j : (j, i) \in \mathcal{A}\} \cup \{0\}$ as the set of nodes adjacent to node $i,\ \forall i \in \mathcal{V}$. Also, $FS(0) = \mathcal{V}$ and $RS(0) = \emptyset$. We define binary variables z_i for all $i \in \mathcal{V}$, such that $z_i = 1$ if we construct arc $(0, i)$ (i.e., if we will use $(0, i)$ in a path from 0 to some node in $\widetilde{\mathcal{V}}$), and $z_i = 0$ otherwise. The goal is to minimize the number of arcs $(0, i)$ constructed over all $i \in \mathcal{V}$, subject to the restriction that at least one path can be routed from node 0 to every active node $k \in \widetilde{\mathcal{V}}$. We associate a different commodity with each node pair $(0, k), \forall k \in \widetilde{\mathcal{V}}$, and define f_{ijk} as the multi-commodity flow variable on arc $(i, j) \in \mathcal{A}$ that routes a path from node 0 to node $k \in \mathcal{V}$. Additionally, define parameters $a_0 = 1$ and

$a_i = 0,\ \forall i \in \mathcal{V}$. The MaxNum subproblem is given by

$$\eta(x, y) = \min \quad \sum_{i \in \mathcal{V}} z_i \tag{4–2a}$$

$$\text{s.t.} \quad \sum_{j \in FS(i)} f_{ijk} - \sum_{j \in RS(i)} f_{jik} \geq a_i x_k \quad \forall i \in \{0\} \cup \mathcal{V},\ k \in \mathcal{V},\ i \neq k \tag{4–2b}$$

$$f_{0jk} \leq z_j \quad \forall j,\ k \in \mathcal{V} \tag{4–2c}$$

$$f_{ijk} \leq y_{ij} \quad \forall (i, j) \in \mathcal{A},\ k \in \mathcal{V} \tag{4–2d}$$

$$z_i \in \{0, 1\} \quad \forall i \in \mathcal{V} \tag{4–2e}$$

$$f_{ijk} \geq 0 \quad \forall (i, j) \in \mathcal{A},\ k \in \mathcal{V}. \tag{4–2f}$$

Constraints 4–2b are multi-commodity flow balance equations that require one unit of flow to originate at node 0 and terminate at node k, for each $k \in \widetilde{\mathcal{V}}$. Note that Constraints 4–2b do not require any flow to reach node $k \in \mathcal{V}$ if it is not active (i.e., if $x_k = 0$, then $f_{ijk} = 0,\ \forall (i, j) \in \mathcal{A}$, and $f_{0ik} = 0,\ \forall i \in \mathcal{V}$, is feasible). Constraints 4–2c indicate that no flow is permitted on arc $(0, i),\ \forall i \in \mathcal{V}$, if the arc is not constructed, and Constraints 4–2d prevent flow on arc $(i, j) \in \mathcal{A}$ if it is deleted (where $y_{ij} \equiv y_{ji},\ \forall (i, j) \in \mathcal{A}$). Constraints 4–2e and 4–2f require z to be binary, and f to be nonnegative, respectively.

We illustrate a feasible solution in Figure 4-3 in which three components remain in G after some other nodes (not shown) have been deleted. In this graph, only edges $(1, 2)$, $(2, 3)$, and $(4, 5)$ remain after node deletion. An optimal solution to Formulation 4–2 constructs potential arcs $(0, 1)$, $(0, 4)$, and $(0, 6)$, and results in 6-commodity flows illustrated alongside the dashed arrows. All other f-values are zero. The three arcs constructed correspond to the number of components.

Next, we show that solving Formulation 4–2 is equivalent to solving its LP relaxation.

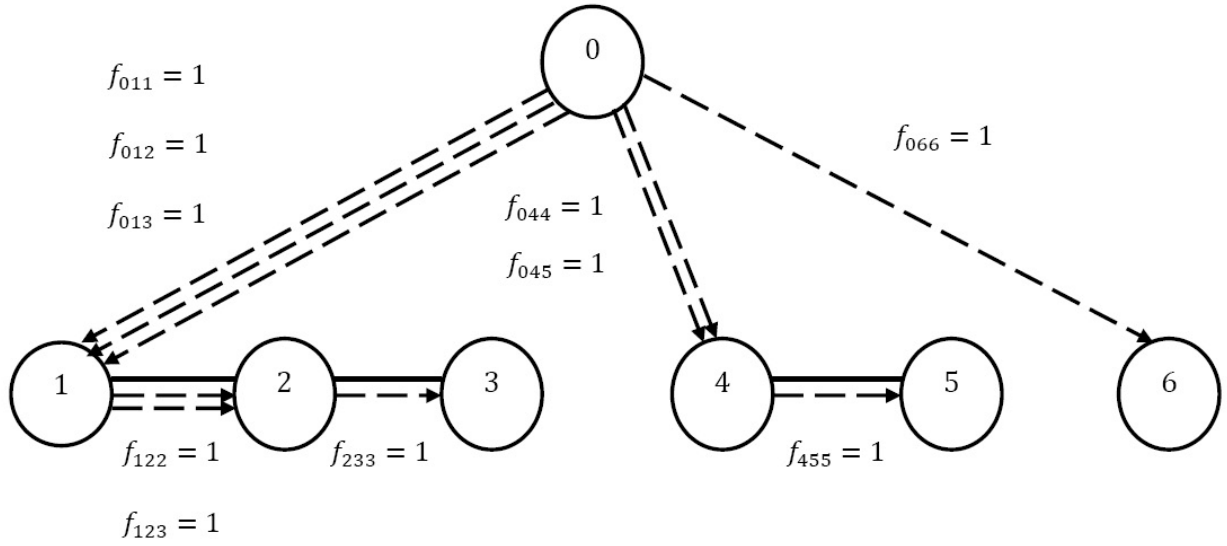


Figure 4-3. Example solution to Formulation 4–2.

Proposition 4.1. *Let P be the feasible region of the LP relaxation of Formulation 4–2 (in which Constraints 4–2e are eliminated). Given binary x and y-values, a subproblem solution (f, z) with a fractional z- or f-value is not an extreme point of P.*

Proof. We first show that any solution $(\bar{f}, \bar{z}) \in P$ with $0 < \bar{z}_i < 1$ for some $i \in \mathcal{V}$ can be represented as a strict convex combination of two distinct points $(\bar{f}^1, \bar{z}^1)$ and $(\bar{f}^2, \bar{z}^2)$ in P, and thus is not an extreme-point solution. Define sets I^+ and I^0 such that $j \in I^+$ if the $(j,k)^{\text{th}}$ constraints of 4–2c are not binding for all $k \in \mathcal{V}$, and $j \in I^0$ if at least one such constraint is binding for some k. First, suppose that $i \in I^+$, and let $\epsilon_i = \min_{k \in \widetilde{\mathcal{V}}} \left\{ \bar{z}_i - \bar{f}_{0ik} \right\}$. We set $\bar{f}^1 = \bar{f}^2 = f$ and $\bar{z}^1 = \bar{z}^2 = z$, with the exception of setting $\bar{z}^1_i = \bar{z}_i + \epsilon_i$ and $\bar{z}^2_i = \bar{z}_i - \epsilon_i$. Noting that $\epsilon_i > 0$ because $\bar{z}_i > \bar{f}_{0ik},\ \forall k \in \widetilde{\mathcal{V}}$, solutions $(\bar{f}^1, \bar{z}^1)$ and $(\bar{f}^2, \bar{z}^2)$ are feasible and distinct. Thus, $(\bar{f}, \bar{z}) = 1/2(\bar{f}^1, \bar{z}^1) + 1/2(\bar{f}^2, \bar{z}^2)$ cannot be an extreme point in this case.

Now, suppose that $i \in I^0$, and denote C_i as the component to which node i belongs. Define $\epsilon' = \min_{j \in C_i,\ k \in \widetilde{\mathcal{V}}} \left\{ \bar{f}_{0jk} : \bar{f}_{0jk} > 0 \right\}$, i.e., ϵ' is the smallest possible positive flow that traverses any arc $(0, j)$, over all $j \in C_i$. Note that $0 < \epsilon' \leq \bar{z}_j,\ \forall j \in C_i$, where the second inequality holds due to Ineq. 4–2c.

Next, we design the values of $(\bar{f}^1, \bar{z}^1)$ and $(\bar{f}^2, \bar{z}^2)$ as follows. First, define $\mathcal{K}_i^+ = \{k \in \widetilde{\mathcal{V}} : \bar{f}_{0ik} > 0\}$. (Note that $\mathcal{K}_i^+ \neq \emptyset$, because $i \in I^0$.) Let $\bar{z}_i^1 = \bar{z}_i - \epsilon'$, and $\bar{f}_{0ik}^1 = \bar{f}_{0ik} - \epsilon'$, $\forall k \in \mathcal{K}_i^+$. Consider any $k \in \mathcal{K}_i^+$, and note that because $\bar{f}_{0ik} < 1$, there exists at least one node $l \in C_i$ ($l \neq i$) such that $\bar{f}_{0lk} > 0$. Let $\bar{f}_{0lk}^1 = \bar{f}_{0lk} + \epsilon'$ and $\bar{z}_l^1 = \bar{z}_l + \epsilon'$. Also, denote $\overline{P}_{uvk}$ as the set of all arcs in some path from u to v, using only arcs (i, j) in $\mathcal{A}$ for which $\bar{f}_{ijk} > 0$. We set $\bar{f}_{abk}^1 = \bar{f}_{abk} - \epsilon'$ for all arcs $(a, b) \in \overline{P}_{ikk}$, and set $\bar{f}_{cdk}^1 = \bar{f}_{cdk} + \epsilon'$ for all arcs $(c, d) \in \overline{P}_{lkk}$. Repeat the foregoing procedure for all nodes k in I^0, compensating for the lack of flow to node k via node i by increasing flow to node k via node l, and set all other elements of $(\bar{f}^1, \bar{z}^1)$ equal to their corresponding values in $(\bar{f}, \bar{z})$.

These operations clearly retain feasibility to the flow balance constraints 4–2b without violating bound restrictions 4–2d and 4–2f. Constraints 4–2c are satisfied by decreasing $\bar{z}_i^1$ at the same rate as $\bar{f}_{0ik}^1$, and increasing $\bar{z}_l^1$ at the same rate as $\bar{f}_{0lk}^1$, $\forall k \in \mathcal{K}_i^+$.

To design solution $(\bar{f}^2, \bar{z}^2)$, we reverse the perturbation from $(\bar{f}, \bar{z})$ to $(\bar{f}^1, \bar{z}^1)$, thus setting $(\bar{f}^2, \bar{z}^2) = (\bar{f}, \bar{z}) + \left[(\bar{f}, \bar{z}) - (\bar{f}^1, \bar{z}^1)\right]$. (The arguments to verify the feasibility of $(\bar{f}^2, \bar{z}^2)$ are the same as above.) Solutions $(\bar{f}^1, \bar{z}^1)$ and $(\bar{f}^2, \bar{z}^2)$ are again distinct because $\epsilon' > 0$. Note that $(\bar{f}, \bar{z}) = 1/2(\bar{f}^1, \bar{z}^1) + 1/2(\bar{f}^2, \bar{z}^2)$, which is a strict convex combination of two distinct feasible points in P. Thus, we conclude that all extreme points to P have binary z-values.

Finally, if z is a binary vector, then the subproblem decomposes into $|\mathcal{V}|$ separable path flow problems in terms of the f-variables, and thus f must also be binary-valued in any extreme point solution. This completes the proof. □

According to Proposition 4.1, we reformulate subproblem 4–2 by replacing Ineq. 4–2e with $z_i \geq 0$ for all $i \in \mathcal{V}$, where the upper bounds $z_i \leq 1$ are unnecessary noting that no z-value exceeds 1 in any optimal solution. Let π_{ik}, $-\alpha_{0ik}$, and $-\alpha_{ijk}$ be dual variables associated with primal constraints 4–2b, 4–2c, and 4–2d, respectively. The

dual of Formulation 4–2 is given by:

$$\eta(x,y) = \max \quad \sum_{k\in\mathcal{V}} x_k\pi_{0k} - \sum_{(i,j)\in\mathcal{A}}\sum_{k\in\mathcal{V}} y_{ij}\alpha_{ijk} \tag{4–3a}$$

$$\text{s.t.} \quad (\pi_{0k} - \pi_{ik}) - \alpha_{0ik} \le 0 \quad \forall i,\ k \in \mathcal{V},\ i \ne k \tag{4–3b}$$

$$(\pi_{ik} - \pi_{jk}) - \alpha_{ijk} \le 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V} \tag{4–3c}$$

$$\sum_{k\in\mathcal{V}} \alpha_{0ik} \le 1 \quad \forall i \in \mathcal{V} \tag{4–3d}$$

$$\alpha_{0ik} \ge 0,\ \forall i,\ k \in \mathcal{V};\ \alpha_{0ii} = 0,\ \forall i \in \mathcal{V};$$

$$\alpha_{ijk} \ge 0,\ \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V}$$

$$\pi_{0k},\ \pi_{ik} \ge 0 \quad \forall i,\ k \in \mathcal{V}, \tag{4–3e}$$

where Ineq. 4–3b, 4–3c, and 4–3d are dual constraints respectively associated with primal variables f_{0ik}, f_{ijk}, and z_i. By replacing $\eta(x,y)$ with Ineq. 4–3a in Formulation 4–1, we obtain a bilinear mixed-integer program with nonlinear terms of $x_k\pi_{0k}$ and $y_{ij}\alpha_{ijk}$ existing in the objective function. Note that π_{0k} and α_{ijk} indicate shadow prices associated with the $(0,k)^{\text{th}}$ constraint of Ineq. 4–2b and the $((i,j),k)^{\text{th}}$ constraint of Ineq. 4–2d, respectively. For each unit increase in the right-hand-side (RHS) of Ineq. 4–2b or 4–2d, the number of components will increase by no more than one. To see this, an increase in the RHS of Ineq. 4–2b corresponds to changing $a_i x_k$ from 0 to 1 (implying that $i = 0$ and node k is now active), and hence requiring that a new path must now be established from 0 to k. This can be done at a cost of no more than 1, by simply setting $z_k = f_{0kk} = 1$. An increase in the RHS of Ineq. 4–2d can only occur when $-y_{ij}$ is changed from -1 to 0, and thus removes links (i,j) and (j,i) in $\mathcal{A}$. It can create at most one new component in the graph, and so its shadow price cannot exceed 1.

Therefore, we have that $\pi_{0k} \le 1$ and $\alpha_{ijk} \le 1$. Because all x- and y-variables are binary-valued, by letting $\beta_{0k} \equiv x_k\pi_{0k}$ and $\gamma_{ijk} \equiv y_{ij}\alpha_{ijk}$, we linearize these bilinear terms using the following inequalities:

$$\beta_{0k} \le x_k,\ \beta_{0k} \le \pi_{0k} \quad \forall k \in \mathcal{V} \tag{4–4a}$$

$$\gamma_{ijk} \geq y_{ij} + \alpha_{ijk} - 1,\ \gamma_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V}, \tag{4–4b}$$

where we omit constraints $\beta_{0k} \geq 0,\ \beta_{0k} \geq x_k + \pi_{0k} - 1$, $\gamma_{ijk} \leq y_{ij}$, and $\gamma_{ijk} \leq \alpha_{ijk}$ because they will not be violated by any optimal solution. The integrated MaxNum formulation is then given by

$$\textbf{MaxNum: } \max \quad \sum_{k \in \mathcal{V}} \beta_{0k} - \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{V}} \gamma_{ijk} - \frac{1}{n} \sum_{i \in \mathcal{V}} (1 - x_i) \tag{4–5}$$

s.t. Constraints 4–1b through 4–1e, 4–3b through 4–3e, 4–4a through 4–4b.

4.2.2 MinMaxC

Our next objective is to minimize the largest component size in the graph after node deletions. The MinMaxC objective function is similar to the one for MaxNum:

$$\min \left\{ \eta'(y) + \frac{1}{n} \sum_{i \in \mathcal{V}} (1 - x_i) : \text{Constraints 4–1b through 4–1e} \right\}, \tag{4–6}$$

where $\eta'(y)$ represents the largest component size given y. (Note that we can equivalently consider $\eta'(y)$ as the largest component size in G *including* all deleted nodes, because each deleted node is a singleton component. These cardinality-1 components do not increase the objective function value unless $B \geq |\mathcal{V}|$ and the problem is trivial.)

Define variables $\sigma_{ik} \in \{0, 1\}$ such that $\sigma_{ik} = 1$ if nodes i and k belong to the same component, and $\sigma_{ik} = 0$ otherwise. (In particular, $\sigma_{kk} = 1,\ \forall k \in \mathcal{V}$.) Letting $\lambda = \eta'(y)$ be a variable that represents the largest component size, we give an integrated MIP formulation as:

$$\textbf{MinMaxC: } \min \quad \lambda + \frac{1}{n} \sum_{i \in \mathcal{V}} (1 - x_i) \tag{4–7a}$$

s.t. Constraints 4–1b through 4–1e

$$\sigma_{kk} = 1 \quad \forall k \in \mathcal{V} \tag{4–7b}$$

$$\sigma_{jk} - \sigma_{ik} \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V} \tag{4–7c}$$

$$\lambda \geq \sum_{i \in \mathcal{V}} \sigma_{ik} \quad \forall k \in \mathcal{V} \tag{4–7d}$$

$$\sigma_{ik} \geq 0 \quad \forall i,\ k \in \mathcal{V}. \tag{4–7e}$$

Constraints 4–7d indicate that $\lambda \geq \max_{k \in \mathcal{V}} \left\{ \sum_{i \in \mathcal{V}} \sigma_{ik} \right\}$, where $\sum_{i \in \mathcal{V}} \sigma_{ik}$ calculates the cardinality of the component to which node k belongs, $\forall k \in \mathcal{V}$. At optimality, λ takes on its minimum feasible value, and thus equals $\max_{k \in \mathcal{V}} \left\{ \sum_{i \in \mathcal{V}} \sigma_{ik} \right\}$. Next, we establish the following proposition to show that other constraints in 4–7 guarantee feasible and binary σ-values as defined.

Proposition 4.2. *Given binary y-values, there exists an optimal solution to Formulation 4–7 in which $\sigma_{ik} = 1$ if and only if node i and node k belong to the same component, and $\sigma_{ik} = 0$ otherwise.*

Proof. In Constraints 4–7b, we impose $\sigma_{kk} = 1,\ \forall k \in \mathcal{V}$. Constraints 4–7c state that when $i = k$, we have that $\sigma_{jk} - \sigma_{kk} \geq y_{kj} - 1,\ \forall (k, j) \in \mathcal{A}$, and thus enforce all σ_{jk}-values to equal 1 if an edge exists between nodes j and k (i.e., $y_{kj} = 1$). These constraints propagate through the component and force $\sigma_{ik} = 1$ for all nodes i that are connected to node k. Note that for nodes $l \in \mathcal{V}$ that are not connected to k, it is feasible to set $\sigma_{lk} = 0$. Noting Ineq. 4–7d, there must exist an optimal solution in which all σ-values take on their smallest values allowed by Ineq. 4–7c and 4–7e, and therefore, σ_{ij} either takes on a value of 0 or 1. □

Remark 4.2. We can also prove that Proposition 4.2 is valid by formulating a similar network design model to the one presented for the MaxNum case. In the MinMaxC case, this network flow problem would once again establish a set of potential arcs that could emanate from a dummy node, but with the restriction that only one potential arc can be built at optimality. Each of the original nodes $i \in \mathcal{V}$ has a maximum demand of x_i, and the objective of the model is to send as much flow across a potential arc as possible subject to the restriction that no node receives more flow than its demand. This problem is equivalent to identifying the maximum component size in the graph. To

guarantee the integrality of the network design problem, we can then apply the Special Structures RLT of Sherali & Adams (Sherali et al., 1998) to ensure that all extreme points have integer-valued variables. One can then apply a series of model substitutions and simplifications to obtain Ineq. 4–7, which would equivalently establish the claim in Proposition 4.2. Appendix E describes the procedure details.

Figure 4-4 illustrates an optimal solution to Formulation 4–7, such that λ is computed as the maximum of $\sum_{i\in\mathcal{V}} \sigma_{ik},\ \forall k = 1, \dots, 6$, given by $\sum_{i\in\mathcal{V}} \sigma_{i1} = \sum_{i\in\mathcal{V}} \sigma_{i2} = \sum_{i\in\mathcal{V}} \sigma_{i3} = 3$.

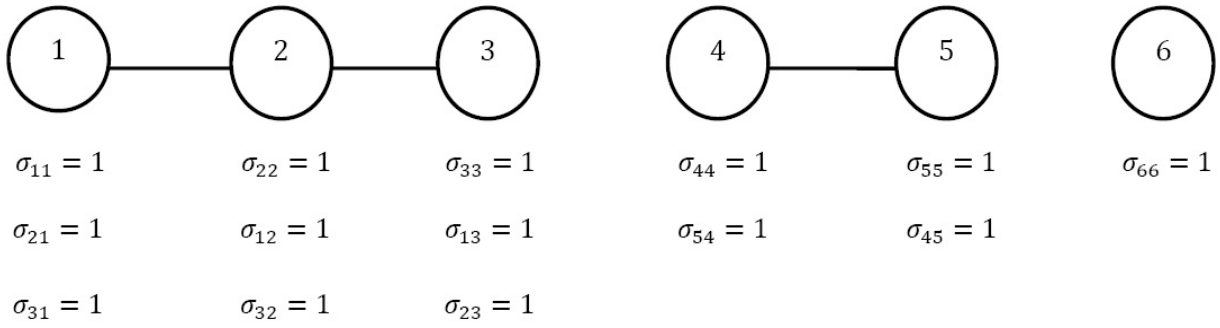


Figure 4-4. Example solution to Formulation 4–7.

4.2.3 MaxMinLR

The last connectivity metric that we consider regards the minimum link construction cost for reconnecting a network. Our motivation for considering this problem is that it represents the case in which a two-stage Stackelberg game is played between an interdicting agent and some network operator. The network operator will reconnect all surviving nodes after an attack at minimum cost, while the interdictor's goal is to maximize the minimum cost that the operator incurs. Let $\widetilde{\mathcal{E}}$ be the set of edges that can be built to reconnect the graph after edge deletions, and c_{ij} be the link construction cost associated with edge $(i,j),\ \forall (i,j) \in \widetilde{\mathcal{E}}$.

We begin by showing that MaxNum is a special case of MaxMinLR. Note that any MaxMinLR instance in which $\widetilde{\mathcal{E}} = \mathcal{E}$, and $c_{ij} = 1,\ \forall (i,j) \in \widetilde{\mathcal{E}}$, is identical to a MaxNum instance on the same graph: The number of components in an optimal MaxNum

solution is one fewer than the number of edges required to reconnect the graph. (This observation is true regardless of whether we treat deleted nodes as entities that need to be reconnected in the MaxMinLR problem.) Thus, because MaxNum is a special case of MaxMinLR, and MaxNum is strongly $\mathcal{NP}$-hard, then MaxMinLR is also strongly $\mathcal{NP}$-hard. By contrast to MaxNum and MinMaxC, Theorem 4.2 shows that when deleted nodes do not need to be reconnected, MaxMinLR remains hard even when G is a tree.

Theorem 4.2. *MaxMinLR (without reconnecting deleted nodes) is strongly $\mathcal{NP}$-hard, even if c_{ij} is binary-valued for all $(i,j) \in \widetilde{\mathcal{E}}$, and if G is a tree.*

Proof. Denote $\overline{\text{MaxMinLR}}$ as the decision version of MaxMinLR, which seeks to delete a subset of no more than B nodes, such that the minimum link construction cost for reconnecting all remaining nodes in G is at least a given target $\mathcal{C}$. We first show that $\overline{\text{MaxMinLR}}$ belongs to $\mathcal{NP}$: given a graph G' after nodes have been deleted, we can execute Prim's algorithm (e.g., Ahuja et al. (1993)), starting with graph G' and adding edges from among those in $\widetilde{\mathcal{E}}$, to identify a minimum-cost set of edges that reconnects the graph. Because Prim's algorithm runs in polynomial time, $\overline{\text{MaxMinLR}} \in \mathcal{NP}$.

Next, we show that $\overline{\text{MaxMinLR}}$ is $\mathcal{NP}$-complete using transformation from IS on a graph $G^I(V^I, E^I)$, in which we seek an independent set $S \subseteq V^I$ having k nodes (assuming that $k \geq 2$). We transform an IS instance into a $\overline{\text{MaxMinLR}}$ instance having a graph $\mathcal{G}(\mathcal{V}^0, \mathcal{A}^0)$, where $\mathcal{V}^0$ contains a dummy node 0 and nodes corresponding to every node in V^I. Define set $\mathcal{A}^0 = \{(0,j) : j \in V^I\}$, $\widetilde{\mathcal{E}} = \{(i,j) : i,j \in V^I\}$, and the interdiction budget $B = |V^I| + 1 - k$. For every edge $(i,j) \in E^I$, the link construction cost $c_{ij} = 0$, and we let $c_{ij} = 1$ for any other node pair $(i,j) \in \widetilde{\mathcal{E}} \setminus E^I$. The cost target $\mathcal{C}$ is given as $k-1$.

Suppose that the IS instance has a solution S with $|S| = k$. A solution to the $\overline{\text{MaxMinLR}}$ instance can be constructed by deleting node 0 and all nodes in $V^I \setminus S$, which leaves all nodes in $\mathcal{V}^0$ as singletons. We construct a subset of node pairs in $\widetilde{\mathcal{E}}$ to reconnect the remaining nodes, which are just nodes in S. Because S is an independent set, $c_{ij} = 1 \; \forall i,\ j \in S$, and the network operator thus incurs a cost of $k-1$ for building

$k-1$ edges to reconnect all nodes in S. Hence, there is also a solution to the $\overline{\text{MaxMinLR}}$ instance.

Now suppose that there are no solutions to the IS instance. First observe that if node 0 is not deleted in the $\overline{\text{MaxMinLR}}$ instance, then any set of attacks on nodes corresponding to V^I yields a graph that remains connected, with a corresponding construction cost of zero. Hence, assume that node 0 has been deleted in any solution to the $\overline{\text{MaxMinLR}}$ problem. Now, for every subset Q of V^I with $|Q| = k$, there exists at least one edge $(i,j) \in E^I$ such that both nodes i and j belong to Q. The network operator can build edge (i,j) at zero cost, and can connect the remaining nodes in Q at a cost of at most $k-2 < \mathcal{C}$. The transformed $\overline{\text{MaxMinLR}}$ instance is thus a no-instance as well.

Because IS is $\mathcal{NP}$-complete in the strong sense, data used in our transformation is polynomially bounded (with binary c-values), and $\mathcal{G}(\mathcal{V}^0, \mathcal{A}^0)$ is constructed as a tree, we have shown that MaxMinLR on tree graphs with binary construction costs is strongly $\mathcal{NP}$-hard. □

We model MaxMinLR as a MIP by creating a directed auxiliary graph $\widetilde{G}(\mathcal{V}, \mathcal{A})$, in which two directed arcs (i,j) and (j,i) appear in $\mathcal{A}$ associated with each edge $(i,j) \in \mathcal{E}$. Meanwhile, two directed arcs (i,j) and (j,i) appear in a potential arc set $\widetilde{\mathcal{A}}$ for each edge $(i,j) \in \widetilde{\mathcal{E}}$ that may be used to reconnect the graph. Recall that $\widetilde{\mathcal{V}}$ is the set of all active nodes. We formulate the problem of determining the minimum connection cost on $\widetilde{G}(\mathcal{V}, \mathcal{A})$ as a multi-commodity network design problem, which routes $(|\widetilde{\mathcal{V}}| - 1)$ paths from some active node q to all other active nodes (one path each from node q to node i, $\forall i \in \widetilde{\mathcal{V}} \setminus \{q\}$). Without loss of generality, we specify source node q as the smallest-indexed active node. Define binary variables w_i, such that $w_i = 1$ if node i is the smallest-indexed node in $\widetilde{\mathcal{V}}$, and $w_i = 0$ otherwise. In the master problem, we establish

the definition of w by using the following inequalities:

$$w_i \leq x_i \quad \forall i \in \mathcal{V} \tag{4–8a}$$

$$w_i \leq 1 - x_k \quad \forall i,\ k \in \mathcal{V} :\ k < i \tag{4–8b}$$

$$\sum_{i \in \mathcal{V}} w_i = 1, \tag{4–8c}$$

where Constraints 4–8a ensure $w_i = 0$ if node i has been deleted, and Constraints 4–8b enforce $w_i = 0$ if any node k having a smaller index than i (i.e., $k < i$) has not been deleted. Constraint 4–8c forces one eligible w_i to take a value of 1. The master problem of MaxMinLR is similar to Formulation 4–1 except that the objective maximizes $\eta''(w, x, y) - 1/n \sum_{i \in \mathcal{V}} (1 - x_i)$, where $\eta''(w, x, y)$ is the minimum link construction cost given binary values w, x, and y, and the constraint set includes Ineq. 4–8a, 4–8b, and Eq. 4–8c.

Define $FS'(i) = \{j : (i, j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}\}$ as the set of nodes potentially adjacent from node i, and $RS'(i) = \{j : (j, i) \in \mathcal{A} \cup \widetilde{\mathcal{A}}\}$ as the set of nodes potentially adjacent to i, $\forall i \in \mathcal{V}$. Let f_{ijk} be the flow on arc $(i, j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}$ corresponding to paths from node q to node k, $\forall k \in \mathcal{V}$. Define binary variables $z_{ij} \in \{0, 1\}$, such that $z_{ij} = 1$ if we construct arc (i, j), and $z_{ij} = 0$ otherwise, $\forall (i, j) \in \widetilde{\mathcal{A}}$. The MaxMinLR subproblem is given by

$$\eta''(w, x, y) = \min \quad \sum_{(i,j) \in \widetilde{\mathcal{A}}} c_{ij} z_{ij} \tag{4–9a}$$

$$\text{s.t.} \quad \sum_{j \in FS'(i)} f_{ijk} - \sum_{j \in RS'(i)} f_{jik} \geq w_i - (1 - x_k) \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{4–9b}$$

$$f_{ijk} \leq z_{ij} + y_{ij} \quad \forall (i, j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{4–9c}$$

$$z_{ij} \leq y_{ij} \quad \forall (i, j) \in \widetilde{\mathcal{A}} \tag{4–9d}$$

$$z_{ij} \in \{0, 1\} \quad \forall (i, j) \in \widetilde{\mathcal{A}} \tag{4–9e}$$

$$f_{ijk} \geq 0 \quad \forall (i, j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V}, \tag{4–9f}$$

where Constraints 4–9b represent multi-commodity flow balance conditions at each node $i \in \widetilde{\mathcal{V}}$, which require the existence of a path from node q to node k for all active

$k \in \widetilde{\mathcal{V}}$. Note that these constraints are only stated for all $i,\ k \in \mathcal{V},\ i < k$, because if $w_i = 1$, then $x_1 = \cdots = x_{i-1} = 0$ due to Ineq. 4–8b. Constraints 4–9c force $f_{ijk} = 0$ if $(i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}$ is neither connected (i.e., $y_{ij} = 0$) nor constructed (i.e., $z_{ij} = 0$), where $y_{ij} \equiv 0$ if $(i,j) \notin \mathcal{A}$ and $z_{ij} \equiv 0$ if $(i,j) \notin \widetilde{\mathcal{A}}$. Constraints 4–9d stipulate that no arc incident to any deleted node is constructed. Constraints 4–9e and 4–9f state logical conditions on the variables.

Remark 4.3. Consider the MaxMinLR problem in which the network operator reconnects all nodes (including those that are deleted) in G. We retain the same master problem as before, but delete Constraints 4–8 and simply set $w_1 = 1$, and $w_i = 0,\ \forall i \in \mathcal{V} \setminus \{1\}$. This simplification is valid because we now must force the entire graph to be reconnected after node deletions. Also, we retain the objective function and all constraints of 4–9, except that we remove Constraints 4–9d, and replace Ineq. 4–9b with

$$\sum_{j \in FS'(i)} f_{ijk} - \sum_{j \in RS'(i)} f_{jik} = w_i \quad \forall i,\ k \in \mathcal{V},\ i \neq k. \tag{4–10}$$

Proposition 4.3. *Let P' be the feasible region of the LP relaxation of Formulation 4–9 in which integrality Constraints 4–9e are eliminated. Given binary values of w and y, a subproblem solution (f, z) with fractional f or z is not an extreme point of P'.*

The proof of Proposition 4.3 is similar to the proof of Proposition 4.1 and is hence omitted. Therefore, to formulate MaxMinLR as a single MIP, we replace $\eta''(w, x, y)$ in the master problem by the dual of the LP relaxation of Formulation 4–9, and rewrite all bilinear terms as a series of linear inequalities similar to Ineq. 4–4a and 4–4b.

Let τ_{ik}, $-\theta_{ijk}$, and $-\mu_{ij}$ be dual variables associated with 4–9b, 4–9c, and 4–9d, respectively. In particular, define $\tau_{ik} \equiv 0$ for all $i,\ k \in \mathcal{V},\ i \geq k$. We present the integrated MaxMinLR MIP model as follows.

$$\max \quad \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{V}, k > i} \left(\alpha'_{ik} + \beta'_{ik} - \tau_{ik}\right)$$

$$- \sum_{(i,j)\in\mathcal{A}\cup\widetilde{\mathcal{A}}} \sum_{k\in\mathcal{V}} \gamma'_{ijk} - \sum_{(i,j)\in\widetilde{\mathcal{A}}} \delta'_{ij} - \frac{1}{n}\sum_{i\in\mathcal{V}} (1 - x_i) \tag{4–11a}$$

s.t. Constraints 4–1b through 4–1e, 4–8a through 4–8c

$$\tau_{jk} - \tau_{ik} - \theta_{ijk} \leq 0 \quad \forall (i,j) \in \mathcal{A}\cup\widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{4–11b}$$

$$\sum_{k\in\mathcal{V}} \theta_{ijk} - \mu_{ij} \leq c_{ij} \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{4–11c}$$

$$\alpha'_{ik} \leq w_i,\ \alpha'_{ik} \leq \tau_{ik} \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{4–11d}$$

$$\beta'_{ik} \leq x_k,\ \beta'_{ik} \leq \tau_{ik} \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{4–11e}$$

$$\gamma'_{ijk} \geq y_{ij} + \theta_{ijk} - 1,\ \gamma'_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A}\cup\widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{4–11f}$$

$$\delta'_{ij} \geq y_{ij} + \mu_{ij} - 1,\ \delta'_{ij} \geq 0 \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{4–11g}$$

$$\tau_{ik} \geq 0 \quad \forall i,\ k \in \mathcal{V},\ i < k; \quad \theta_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A}\cup\widetilde{\mathcal{A}},\ k \in \mathcal{V};$$

$$\mu_{ij} \geq 0 \quad \forall (i,j) \in \widetilde{\mathcal{A}}. \tag{4–11h}$$

Constraints 4–11b and 4–11c are dual constraints associated with primal variables $f_{ijk},\ \forall (i,j) \in \mathcal{A}\cup\widetilde{\mathcal{A}},\ k \in \mathcal{V}$ and $z_{ij},\ \forall (i,j) \in \widetilde{\mathcal{A}}$, respectively. We define $\alpha'_{ik} \equiv w_i\tau_{ik}$, $\beta'_{ik} \equiv x_k\tau_{ik}$, $\gamma'_{ijk} \equiv y_{ij}\theta_{ijk}$, and $\delta'_{ij} \equiv y_{ij}\mu_{ij}$ in the objective, and enforce these relationships via Constraints 4–11d, 4–11e, 4–11f, and 4–11g, respectively, as before.

4.3 Bounds and Inequalities for MaxNum and MinMaxC

We now discuss strategies for bounding the optimal objective function values for MaxNum and MinMaxC, and using the obtained bounds to tighten the MIP formulations for these problems. Recall the polynomial-time optimal DP algorithms for MaxNum and MinMaxC on k-hole-graphs (where k is a constant) prescribed in Chapter 3. We do not carry out this analysis for MaxMinLR, as Theorem 4.2 indicates that it is not solvable (in general) in polynomial time, even when G is a tree, unless $\mathcal{P} = \mathcal{NP}$.

We begin by describing mechanisms for bounding the connectivity objectives, i.e., the number of components in MaxNum, and the largest component size in MinMaxC. Denote x_G and x'_G as optimal solutions to MaxNum and MinMaxC on G, respectively. Also, let $\eta_G(x)$ equal the number of components, and $\eta'_G(x)$ equal the largest component

size, after we delete a subset of nodes (and their incident edges) associated with solution x on graph G. We first establish the following proposition.

Lemma 6. *For any subgraph $G_S(\mathcal{V}, \mathcal{E}_S)$ of G, we have:*

$$\eta_G(x_{G_S}) \leq \eta_G(x_G) \leq \eta_{G_S}(x_{G_S}) \tag{4–12a}$$

$$\eta'_G(x'_{G_S}) \geq \eta'_G(x'_G) \geq \eta'_{G_S}(x'_{G_S}). \tag{4–12b}$$

Proof. The first inequality in 4–12a holds because x_{G_S} is feasible to the master problem 4–1 on G, yielding a valid lower bound $\eta_G(x_{G_S})$ on $\eta_G(x_G)$. The second inequality in 4–12a holds because $\mathcal{E}_S \subseteq \mathcal{E}$, and any optimal MaxNum solution on G_S generates at least as many components as it does on G. The same arguments hold for Ineq. 4–12b, because any feasible solution to MinMaxC provides an upper bound on $\eta'_G(x'_G)$, and an optimal solution on any subgraph yields a lower bound on $\eta'_G(x'_G)$. This completes the proof. □

In our implementation, we specify *a priori* some value $k_{\max}$ that denotes the maximum number of holes that we allow in any induced subgraph of G, and generate a series of m subgraphs of G, denoted as $H^1, \ldots, H^m$, in which H^l is a k_l-hole-graph ($k_l \leq k_{\max}$), $\forall l = 1, \ldots, m$. According to Lemma 6, the following bounds are valid:

$$\max_{l=1,\ldots,m} \{\eta_G(x_{H^l})\} \leq \eta_G(x_G) \leq \min_{l=1,\ldots,m} \{\eta_{H^l}(x_{H^l})\} \tag{4–13a}$$

$$\min_{l=1,\ldots,m} \{\eta'_G(x'_{H^l})\} \geq \eta'_G(x'_G) \geq \max_{l=1,\ldots,m} \{\eta'_{H^l}(x'_{H^l})\}. \tag{4–13b}$$

Given the bounds established in Cuts 4–13a and 4–13b, we now turn our attention to employing these bounds within a graph partitioning strategy that generates valid inequalities for MaxNum and MinMaxC. Given $G(\mathcal{V}, \mathcal{E})$, we partition $\mathcal{V}$ into m nonempty subsets $V_1, \ldots, V_m$, such that $V_i \cap V_j = \emptyset$ for all $i, j = 1, \ldots, m$, $i \neq j$ and $\bigcup_{i=1}^m V_i = \mathcal{V}$. Each partition V_i yields a subgraph $G_i(V_i, E_i)$, in which $E_i = \{(u, v) \in \mathcal{E} \mid u, v \in V_i\}$ is induced by nodes in V_i. (Note that if G is connected and $m \geq 2$, then $\bigcup_{i=1}^m E_i \subset \mathcal{E}$.) Let k_i be the number of holes in each subgraph G_i, $\forall i = 1, \ldots, m$.

We then execute the DP algorithm in Chapter 3 on each k_i-hole subgraph G_i given a node deletion budget of B. As is typical in DP schemes, we obtain the optimal connectivity objective values corresponding to every node deletion budget value $B_i = 0, \dots, B$. These values allow us to construct functions that reflect relationships between the connectivity objective and budgets in MaxNum and MinMaxC over G_i, which yield valid inequalities for their respective MIPs (Hartman et al., 2010).

We first present the development of our valid inequalities in the context of the MaxNum problem. Let $\eta_i(B_i)$ be the maximum number of components that we can obtain in G_i given deletion budget $B_i = 0, \dots, B$. Let variable η represent the optimal connectivity objective for solving MaxNum on G, and let η_i be a variable representing the optimal connectivity objective for solving MaxNum on G_i, $\forall i = 1, \dots, m$. We construct a piecewise-linear concave envelope function $g_i(B_i)$ of $\eta_i(B_i)$, such that $\eta_i(B_i) \leq g_i(B_i)$ for all $B_i = 0, \dots, B$. We use the tightest such function possible, in which every linear segment of $g_i(B_i)$ touches at least two points on the function $\eta_i(B_i)$, assuming that $B \geq 1$. We append the following system of valid inequalities into the MaxNum formulation, where B_i is now released as a variable representing the number of node deletions that take place over V_i.

$$\eta - \sum_{i=1}^{m} \eta_i \leq 0 \tag{4–14a}$$

$$\eta_i - g_i(B_i) \leq 0 \quad \forall i = 1, \dots, m \tag{4–14b}$$

$$B_i = \sum_{j \in V_i} (1 - x_j) \quad \forall i = 1, \dots, m, \tag{4–14c}$$

where Ineq. 4–14a is due to the fact that the partition procedure automatically eliminates all edges between each pair of G_i, $\forall i = 1, \dots, m$, and thus it creates at least as many components as solving the original problem on G. Constraints 4–14b are nonlinear constraints that can be substituted by a set of linear functions, each of which corresponds to a segment of $g_i(B_i)$, for all $i = 1, \dots, m$. Constraints 4–14c define B_i as

the number of node deletions taking place in set V_i. We demonstrate this procedure in Example 1.

Example 1. Given a 20-node graph G and a node deletion budget $B = 10$, we consider solving the MaxNum variation in which we count each deleted node as a singleton component. We first partition G into two subgraphs G_1 and G_2 with 10 nodes in each partition, and execute the MaxNum DP algorithms on both G_1 and G_2 for $B = 10$.

We obtain a series of optimal objective values $\eta_1(B_1)$ and $\eta_2(B_2)$ for each possible budget value B_1 and $B_2 = 0, \ldots, 10$, indicated by the solid lines in Figure 4-5. The horizontal axes represent possible B_i-values, for $i = 1, 2$. We also illustrate the piecewise-linear concave envelope functions $g_1(B_1)$ and $g_2(B_2)$ by dashed lines, whose values provide upper bounds for their corresponding $\eta_i(B_i)$ functions for $i = 1, 2$ (as indicated by the tables in the figures). For $i = 1$, we replace Ineq. 4–14b with the following linear valid inequalities, each of which is associated with one segment of $g_1(B_1)$:

$$\eta_1 \leq 2B_1 + 1, \qquad \eta_1 \leq B_1 + 4, \qquad \eta_1 \leq 10. \tag{4–15a}$$

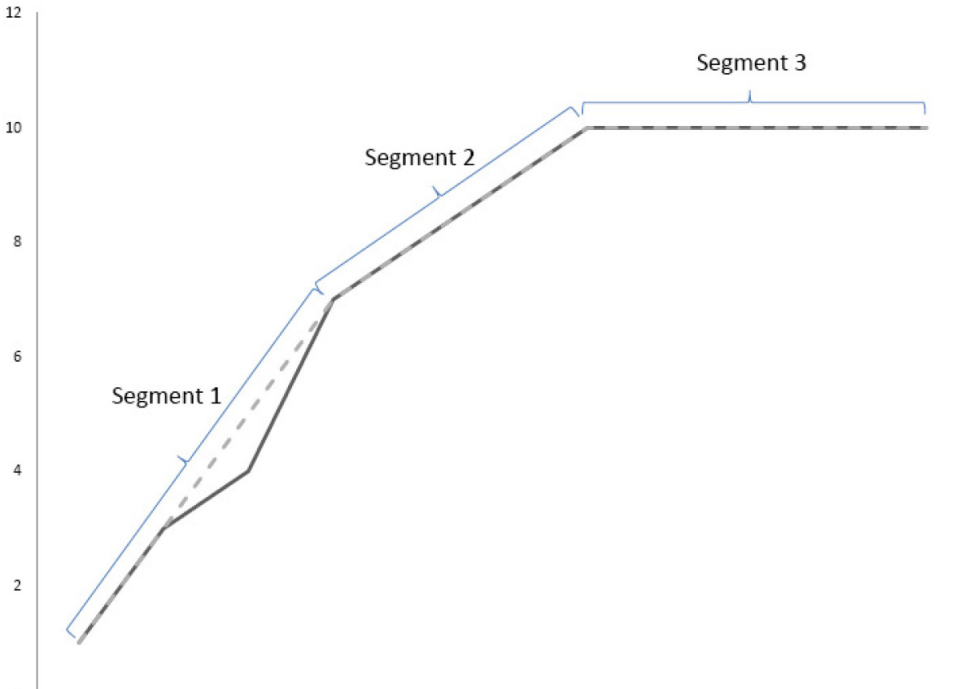
For $i = 2$, we substitute Ineq. 4–14b by using

$$\eta_2 \leq (4/3)B_2 + 1, \qquad \eta_2 \leq B_2 + 3, \qquad \eta_2 \leq 10. \tag{4–15b}$$

The procedure is similar for solving MinMaxC, except that we define $\eta'_i(B_i)$ as the minimum largest component size we can obtain over G_i in terms of budget value $B_i = 0, \ldots, B$, and we approximate it by using its *convex* envelope function $g'_i(B_i)$. We replace Constraints 4–14a and 4–14b with

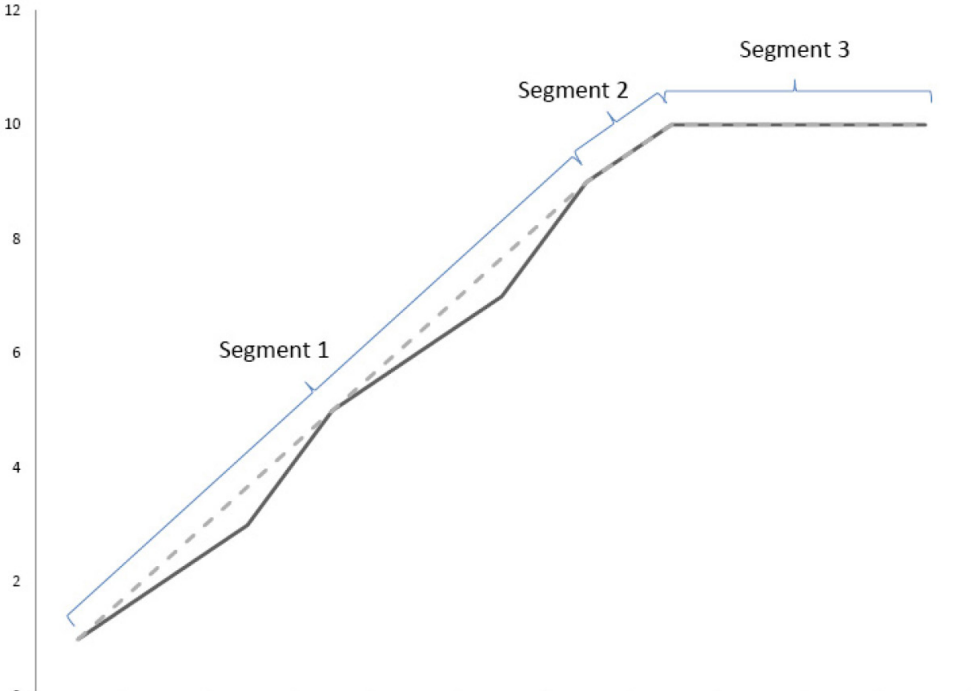
$$\eta' - g'_i(B_i) \geq 0 \quad \forall i = 1, \ldots, m, \tag{4–16}$$

where η' is defined analogously to η in MaxNum. Constraints 4–16 are linearized as before, except that the inequalities linearizing Ineq. 4–16 define its convex envelope, as illustrated in the following example.



B_1	0	1	2	3	4	5	6	7	8	9	10
η_1	1	3	4	7	8	9	10	10	10	10	10
g_1	1	3	5	7	8	9	10	10	10	10	10

A $g_1(B_1)$



B_2	0	1	2	3	4	5	6	7	8	9	10
η_2	1	2	3	5	6	7	9	10	10	10	10
g_2	1	2.33	3.67	5	6.33	7.67	9	10	10	10	10

B $g_2(B_2)$

Figure 4-5. Concave envelope function $g_i(B_i)$. A) Illustration of $g_1(B_1)$. B) Illustration of $g_2(B_2)$.

Example 2. Considering the same graph instance and budget value given in Example 1, we first solve MinMaxC by using DP algorithms on G_1 and G_2 for $B = 10$. The solid lines in Figure 4-6 illustrate the values of $\eta'_i(B_i)$ for $B_i = 0, \dots, 10,\ i = 1, 2$, while the dashed lines provide their corresponding convex envelope functions $g'_i(B_i)$, each of which yields lower bound values for $\eta'_i(B_i)$.

We use the following five- and three-segment functions to respectively describe $g'_i(B_i)$, for $i = 1, 2$:

$$\eta' \geq -3B_1 + 10,\ \eta' \geq -2B_1 + 9,\ \eta' \geq -B_1 + 6,\ \eta' \geq -0.5B_1 + 4,\ \eta' \geq 1 \quad \text{(4–17a)}$$

$$\eta' \geq -1.5B_2 + 10,\ \eta' \geq -B_2 + 8,\ \eta' \geq 1. \quad \text{(4–17b)}$$

Remark 4.4. Note that we can provide tighter inequalities by formulating exact representations for $\eta_i(B_i)$ and $\eta'_i(B_i)$. Define binary variables b_i^l, such that $b_i^l = 1$ if $B_i = l$, and $b_i^l = 0$ otherwise, for all $i = 1, \dots, m$ and $l = 0, \dots, B$. In the MaxNum case, for instance, we keep Constraints 4–14a, but could replace Constraints 4–14b and 4–14c with

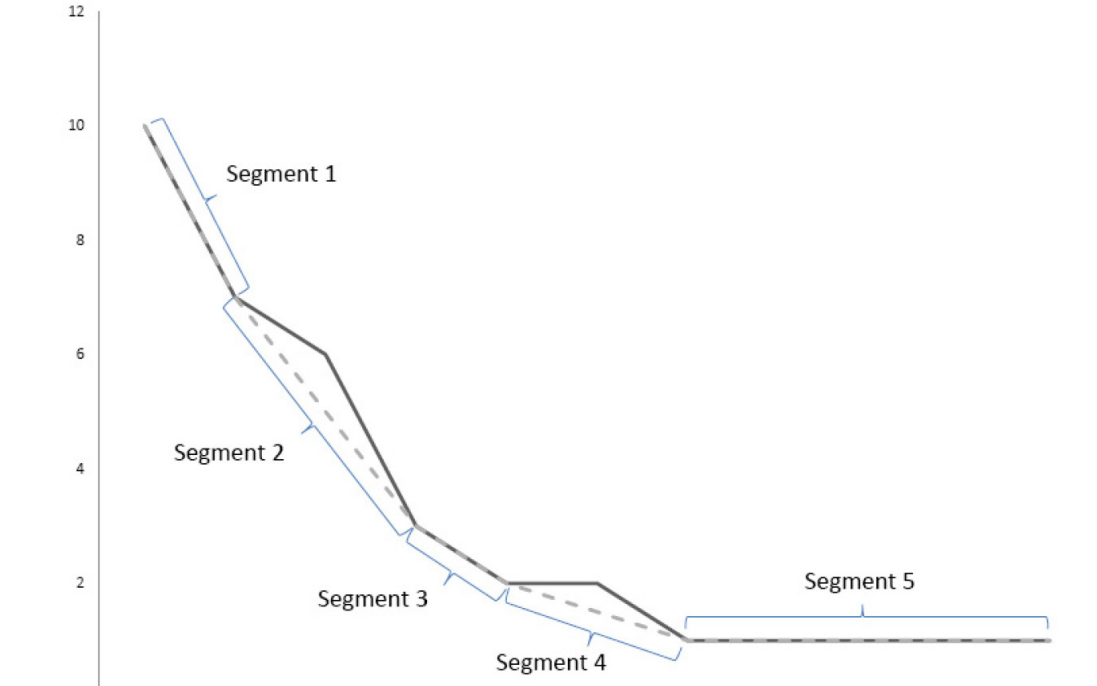
$$\eta_i = \sum_{l=0}^{B} \eta_i(l) b_i^l \quad \forall i = 1, \dots, m \quad \text{(4–18a)}$$

$$\sum_{l=0}^{B} l b_i^l = \sum_{j \in V_i} (1 - x_j) \quad \forall i = 1, \dots, m \quad \text{(4–18b)}$$

$$\sum_{l=0}^{B} b_i^l = 1 \quad \forall i = 1, \dots, m \quad \text{(4–18c)}$$

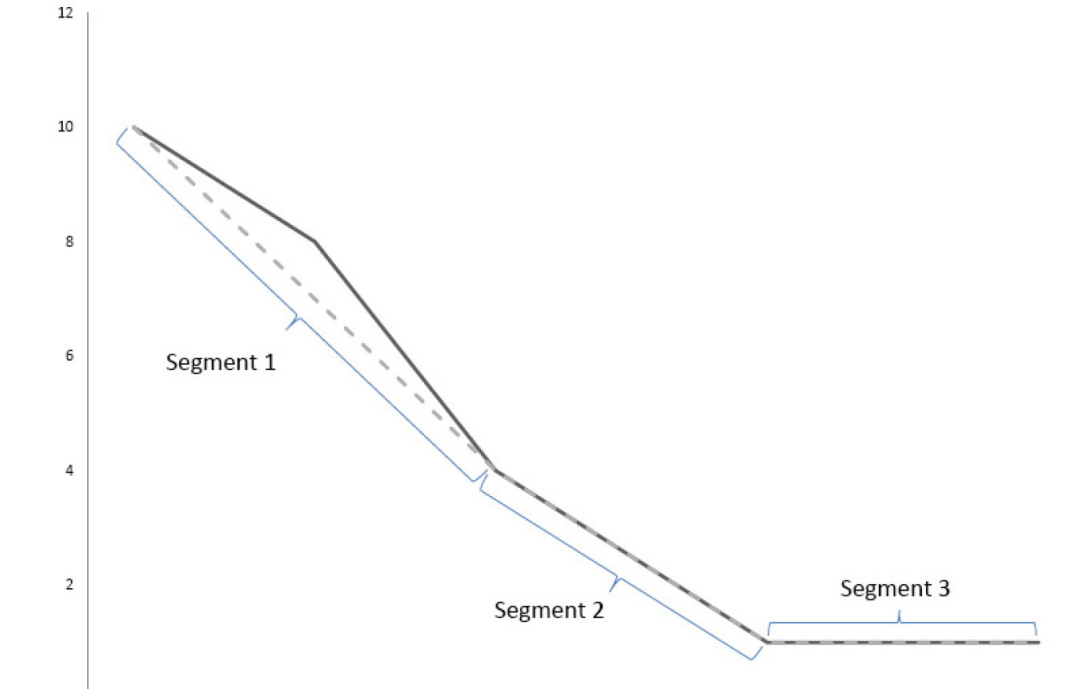
$$b_i^l \in \{0, 1\} \quad \forall i = 1, \dots, m,\ l = 0, \dots, B, \quad \text{(4–18d)}$$

where Constraints 4–18a enforce $\eta_i(l)$ to be the maximum number of components obtained over G_i if $B_i = l$, for all $i = 1, \dots, m$. Constraints 4–18b define $B_i = l$ as the number of node deletions over V_i if $b_i^l = 1$, for all $i = 1, \dots, m$, and for each $i = 1, \dots, m$, we can pick only one such l-value from $\{0, \dots, B\}$ according to Ineq. 4–18c.



B_1	0	1	2	3	4	5	6	7	8	9	10
η_1'	10	7	6	3	2	2	1	1	1	1	1
g_1'	10	7	5	3	2	1.5	1	1	1	1	1

A $g_1'(B_1)$



B_2	0	1	2	3	4	5	6	7	8	9	10
η_2'	10	9	8	6	4	3	2	1	1	1	1
g_2'	10	8.5	7	5.5	4	3	2	1	1	1	1

B $g_2'(B_2)$

Figure 4-6. Convex envelope function $g_i'(B_i)$. A) Illustration of $g_1'(B_1)$. B) Illustration of $g_2'(B_2)$.

However, computational experience indicates that the use of Ineq. 4–18 impedes the performance of the integer programming solver we use, especially when B is relatively large, since the number of binary variables increases linearly with B. □

4.4 Computational Results

This section investigates the computational efficacy of solving our MIP models with and without the valid inequalities presented in Section 4.3 (where appropriate), and examine the quality of the bounds obtained for these models. In our experiments, we first directly solve the MIP models of MaxNum, MinMaxC, and MaxMinLR. For MaxNum and MinMaxC, we then generate connectivity objective bounds by solving DPs on a set of randomly constructed k-hole subgraphs and employing Cut 4–13a or Cut 4–13b, which we augment by using a greedy-based heuristic. We then test the effectiveness of including the valid inequality systems derived from the graph partition strategy within the MIP models for MaxNum and MinMaxC. All MIP models and algorithms were implemented using the C++ programming language and CPLEX 11.0 (ILOG, 2008) via ILOG Concert Technology 2.5, and all computations were performed on a Dell PowerEdge 2600 UNIX machine with two Pentium 4 3.2 GHz (1M Cache) processors, 6.0 GB memory, and Red Hat Version 5.0 installed. We present CPU times in seconds, allow a one-hour (3600 seconds) time limit, and report LIMIT on all instances that could not be solved within the time limit.

4.4.1 Experimental Setup

We randomly generate twenty test instances for MaxNum, MinMaxC, and MaxMinLR, comprising five 10-, 20-, 30-, and 40-node instances. We refer to the i^{th} n-node instance as "n-i". To generate these graphs, we start with a set of 10-node base graphs, and connect nodes $i \in \mathcal{V}$ and $j \in \mathcal{V}$ $(i \neq j)$ with probability 0.6. After edges are generated in this fashion, we check to ensure that the graph is connected. If not, then we randomly add an edge that connects two components until the generated graph becomes connected.

Next, we randomly pick any two, three, and four such 10-node base graphs to construct 20-, 30-, and 40-node instances, respectively. (By generating graphs in this manner, we can seed the partitioning strategy in Section 4.3 with these densely connected base components that comprise each graph.) We retain all nodes and edges in the base graphs, and randomly generate edges between each pair of nodes that belong to different base graphs. For each such pair (i, j), we construct edge (i, j) with probability 0.1. If the resulting graph is not connected, then we again randomly connect its components as described above until a connected graph is obtained. For the MaxMinLR instances, we set $\widetilde{\mathcal{E}} = \mathcal{E}$, and randomly choose each edge construction cost c_{ij} from the integer set $\{1, \ldots, 10\}$ for all $(i, j) \in \widetilde{\mathcal{E}}$.

4.4.2 Experimental Results

We first examine the impact of using various budget values B in solving MaxNum, MinMaxC, and MaxMinLR. (We consider problem versions that ignore deleted nodes when computing the connectivity objective values.) Tables 4-1 and 4-2 report optimal solutions and CPU times for solving the MIP models derived for these problems in Section 4.2 on the 20-node and 30-node instances, respectively. The column labeled **Del.** represents the number of nodes deleted, the column labeled **Time** represents the CPU seconds required to obtain an optimal solution, and the columns labeled **NumC**, **MaxC**, and **MinLRC** represent the optimal connectivity objective values in the three problems.

Note that, in general, solution times for MaxNum, MinMaxC, and MaxMinLR first increase as we increase B, and then decrease when we continue to increase B above a threshold of approximately $0.25|\mathcal{V}|$. (None of the 40-node instances could be optimized within the time limit, for any of the three problems.) When $B = 12$, an optimal solution to any of the three problems removes all edges in the 20-node instances except in instance 20-4; for each 30-node instance, all edges are destroyed when $B = 18$. Also, when $B = 1, 2$ in the 20-node instances and $B = 2$ in the 30-node instances, the

Table 4-1. CPU times for solving MIPs on 20-node instances.

	Instance	MaxNum			MinMaxC			MaxMinLR		
		Time	Del.	NumC	Time	Del.	MaxC	Time	Del.	MinLRC
$B = 1$	20-1	5.71	1	1	1.98	1	19	7.40	0	0.00
	20-2	6.23	1	1	4.28	1	19	14.17	0	0.00
	20-3	10.14	1	1	3.15	1	19	9.54	0	0.00
	20-4	11.29	1	1	4.79	1	19	22.63	0	0.00
	20-5	14.03	1	1	6.24	1	19	10.72	0	0.00
$B = 2$	20-1	13.97	2	1	6.16	2	18	23.67	0	0.00
	20-2	13.01	2	1	10.00	2	18	24.36	0	0.00
	20-3	31.13	2	1	6.48	2	18	26.17	0	0.00
	20-4	32.86	2	1	9.45	2	18	72.21	0	0.00
	20-5	19.43	2	1	9.39	2	18	80.52	0	0.00
$B = 5$	20-1	42.19	5	3	10.33	5	8	124.31	5	4.92
	20-2	26.32	5	3	17.13	5	11	72.63	5	5.23
	20-3	36.90	5	3	17.75	5	10	119.06	5	5.06
	20-4	94.64	5	2	40.82	5	14	259.46	5	3.27
	20-5	117.25	5	2	41.57	5	12	260.28	5	3.74
$B = 9$	20-1	2.54	7	9	2.49	8	3	80.84	8	13.08
	20-2	25.19	9	5	9.02	8	4	249.74	9	7.49
	20-3	8.42	9	6	2.98	9	3	51.32	9	9.21
	20-4	35.72	9	4	46.67	8	6	961.59	9	6.33
	20-5	16.12	9	5	11.05	9	4	461.27	9	7.58
$B = 12$	20-1	0.81	10	10	0.92	10	1	28.80	10	15.62
	20-2	1.76	12	8	0.78	12	1	13.42	12	11.43
	20-3	1.14	11	9	0.86	11	1	20.87	11	13.34
	20-4	2.77	12	6	19.94	12	2	101.00	10	8.80
	20-5	1.15	12	8	1.39	12	1	197.95	12	12.01

optimal reconnection costs are all zero for solving MaxMinLR due to the fact that no solution could create more than one component without considering the deleted nodes as singleton components. It is worth noting that the optimization techniques presented here are unnecessary for very small B-values. Enumerating all possible combinations of node deletions is clearly more efficient than our approaches when B = 1 or 2.

Next, we examine the quality of the objective bounds obtained for MaxNum and MinMaxC, as given by Cuts 4–13a and 4–13b. Recall that $k_{\max}$ is the maximum number of holes that we will permit in any generated subgraph of $G(\mathcal{V}, \mathcal{E})$. Our procedure for generating subgraphs of G that have no more than $k_{\max}$ holes associates a priority parameter ν_{ij}, $\forall (i,j) \in \mathcal{E}$, such that ν_{ij} equals the degree of node i (i.e., the number

Table 4-2. CPU times for solving MIPs on 30-node instances.

	Instance	MaxNum			MinMaxC			MaxMinLR		
		Time	Del.	NumC	Time	Del.	MaxC	Time	Del.	MinLRC
$B = 2$	30-1	233.12	2	1	171.82	2	28	216.39	0	0.00
	30-2	173.28	2	1	170.01	2	28	446.63	0	0.00
	30-3	184.56	2	1	231.13	2	28	698.19	0	0.00
	30-4	135.01	2	1	205.42	2	28	313.06	0	0.00
	30-5	167.24	2	1	185.09	2	28	482.48	0	0.00
$B = 5$	30-1	743.65	5	4	382.67	5	18	LIMIT	–	–
	30-2	501.33	5	4	586.27	5	21	LIMIT	–	–
	30-3	734.59	5	4	135.98	5	15	LIMIT	–	–
	30-4	496.32	5	4	565.79	5	21	LIMIT	–	–
	30-5	613.76	5	5	408.46	5	17	LIMIT	–	–
$B = 10$	30-1	113.45	10	9	108.67	10	5	LIMIT	–	–
	30-2	128.61	10	9	53.61	10	4	LIMIT	–	–
	30-3	984.36	10	7	141.25	10	6	LIMIT	–	–
	30-4	105.92	10	8	69.34	10	5	3450.29	10	14.18
	30-5	165.87	10	7	102.93	10	5	LIMIT	–	–
$B = 15$	30-1	3.23	15	12	14.17	13	2	630.98	15	19.26
	30-2	2.46	15	13	24.11	13	2	2908.23	15	17.03
	30-3	6.79	15	12	16.9	13	2	221.87	15	16.25
	30-4	5.93	15	13	13.48	14	2	432.18	15	17.68
	30-5	6.02	15	13	21.11	13	2	490.47	15	16.92
$B = 18$	30-1	1.02	18	12	1.00	18	1	212.46	17	18.44
	30-2	1.10	17	13	0.75	17	1	132.91	18	20.24
	30-3	2.64	17	13	1.65	17	1	69.38	18	19.57
	30-4	1.75	18	12	0.92	18	1	73.65	17	18.60
	30-5	1.91	17	13	1.10	17	1	121.09	17	21.32

–: Not applicable.

edges in $\mathcal{E}$ incident to node i) plus the degree of node j. Let $\nu_{\max}$ and $\nu_{\min}$ be the maximum and minimum ν_{ij}-values among all $(i,j) \in \mathcal{E}$. Define $p_{ij} = (\nu_{ij} - \nu_{\min})/(\nu_{\max} - \nu_{\min})$ as a conditional probability that edge $(i,j) \in \mathcal{E}$ appears in a constructed subgraph. (In particular, if $\nu_{ij} = \nu_{\min}$, we adjust p_{ij} as 0.01 instead of being zero. Also, if $\nu_{\min} = \nu_{\max}$, then $p_{ij} = 0.5,\ \forall (i,j) \in \mathcal{E}$.) We order all edges arbitrarily, and examine them one by one to see if they will be added to the subgraph: If the addition of the edge creates more than $k_{\max}$ holes in the subgraph, then the edge will not be added to the subgraph, and otherwise, the edge is added with probability p_{ij}. After each edge has been tested for inclusion into the subgraph, we repeat the process with another round of potential edge additions until the addition of some edge will create more than $k_{\max}$ holes. We ensure

that the subgraph is connected by randomly adding edges in $\mathcal{E}$ to the subgraph that connect its components.

We examine values of $k_{\max} = 0,\ 0.1|\mathcal{V}|$, and $0.2|\mathcal{V}|$ in our test instances, and set $m = 5$ in both Cuts 4–13a and 4–13b. We also derive feasible solutions for both MaxNum and MinMaxC by implementing the greedy heuristic algorithm described in Section 4.1. Table 4-3 displays bound and objective data for MaxNum and MinMaxC on instances 10-1, 20-1, 30-1, 40-1, in which the column labeled **Prob.** indicates problem type. To obtain bounds for MaxNum, we execute the DP algorithm on a k-hole subgraph of G, record the optimal objective function value to the subgraph (yielding a valid upper bound on the objective), and then examine the actual number of components created when this solution is applied to the original graph G (yielding a valid lower bound on the objective). Let columns **SF-**$k_{\max}$ represent the largest lower bound from among the *feasible* solutions obtained, and columns **SO-**$k_{\max}$ represent the smallest upper bound from the best subgraph *optimal* objective values. Note that because MinMaxC is a minimization problem, SF-$k_{\max}$ becomes an upper bound (the smallest obtained among the $m = 5$ subgraphs) and SO-$k_{\max}$ becomes a lower bound (the largest one obtained). We also report the greedy-heuristic-based objective values in the column labeled **Greedy**. The column labeled **Opt.** gives the optimal objective values for each instance.

Note that the maximum (minimum) value among columns SF-0, SF-$0.1|\mathcal{V}|$, SF-$0.2|\mathcal{V}|$, and Greedy is a lower (upper) bound for MaxNum (MinMaxC) instances, and the minimum (maximum) value among columns SO-0, SO-$0.1|\mathcal{V}|$, and SO-$0.2|\mathcal{V}|$ is an upper (lower) bound for MaxNum (MinMaxC) instances. In general, the CPU times of solving the DPs on k-hole subgraphs range from 0.01 second to 0.1 second among all instances. For larger values of $k_{\max}$, the DP algorithm begins to take a prohibitively long time to solve. For example, when $k_{\max} = |\mathcal{V}|$ the DP algorithms cannot solve any 30- or 40-node subgraph instances within 100 seconds.

Table 4-3. Connectivity bounds for MaxNum and MinMaxC.

	Instance	Prob.	SF-0	SF-$0.1\|\mathcal{V}\|$	SF-$0.2\|\mathcal{V}\|$	Greedy	SO-0	SO-$0.1\|\mathcal{V}\|$	SO-$0.2\|\mathcal{V}\|$	Opt.
$B = 0.1\|\mathcal{V}\|$	10-1	MaxNum	1	1	1	1	2	2	2	1
		MinMaxC	9	9	9	9	5	6	6	9
	20-1	MaxNum	1	1	1	1	4	3	3	1
		MinMaxC	18	18	18	18	5	7	7	18
	30-1	MaxNum	1	1	1	1	4	4	3	1
		MinMaxC	27	27	27	27	8	10	10	27
	40-1	MaxNum	1	1	1	1	8	8	8	1
		MinMaxC	36	36	36	36	6	6	6	36
$B = 0.2\|\mathcal{V}\|$	10-1	MaxNum	1	2	2	2	3	3	3	2
		MinMaxC	8	6	6	5	3	3	4	5
	20-1	MaxNum	3	3	4	3	6	6	5	5
		MinMaxC	16	16	12	14	4	4	8	12
	30-1	MaxNum	3	3	4	4	7	7	7	6
		MinMaxC	18	16	14	16	4	6	6	14
	40-1	MaxNum	5	6	7	7	12	12	10	–
		MinMaxC	24	22	20	16	4	4	8	–
$B = 0.4\|\mathcal{V}\|$	10-1	MaxNum	4	4	5	5	5	5	5	5
		MinMaxC	4	3	2	2	2	2	2	2
	20-1	MaxNum	5	6	7	8	8	8	8	8
		MinMaxC	7	6	4	3	2	2	2	3
	30-1	MaxNum	8	8	9	8	13	13	12	10
		MinMaxC	9	7	5	6	2	2	2	4
	40-1	MaxNum	6	6	7	8	13	13	13	–
		MinMaxC	13	11	8	10	2	2	2	–

–: Not applicable.

Moreover, as expected, we find that the bounds are relatively tighter on smaller instances, because our generated subgraphs contain more of the original graph's edges. The bounds are also the tightest when the node deletion budget $B = 0.2|\mathcal{V}|$, as opposed to when $B = 0.1|\mathcal{V}|$ or $B = 0.4|\mathcal{V}|$. A possible reason for this behavior is that when B is too small, it is possible to create multiple components by deleting a few nodes in the generated subgraph, but not in the original graph. On the other hand, when B is large, we can isolate virtually all nodes in a subgraph, but not in the original graph. Lastly, note that bounds yielded by Greedy and SF-$0.2|\mathcal{V}|$ do not dominate each other, and can be effectively used in combination with one another to provide objective bounds.

For our last experiment, we investigate the impact of solving MaxNum and MinMaxC MIP models with the valid inequalities given by 4–14 and 4–16, respectively. Tables 4-4 through 4-6 compare CPU times for solving the problems on 20-, 30-, and

40-node instances. The column labeled **Orig.** represents CPU times of solving the problems on original graphs by using MIP models. For each 20- (30-) node instance, we partition the graph into two (three) 10-node base graphs, and for each 40-node instance, we partition the graph into either two 20-node subgraphs, or four 10-node subgraphs. We use optimal DP solutions obtained by solving the problems on the original subgraphs to generate valid inequality systems 4–14 and 4–16; this process required 0.1–0.2 CPU seconds for all 10-node partitions, and 100–120 CPU seconds for all 20-node partitions. We use "[·]" to mark any solution that is worse than the original MIP solution (in terms of CPU time or optimality gap).

Table 4-4. CPU times for 20-node instances using 2-partition inequalities.

Instance	Prob.	$B = 4$		$B = 8$	
		Orig.	2-Partition	Orig.	2-Partition
20-1	MaxNum	24.62	[34.52]	5.94	[16.85]
	MinMaxC	16.56	8.15	1.27	[1.90]
20-2	MaxNum	49.67	43.28	79.48	42.52
	MinMaxC	8.17	6.53	16.22	12.53
20-3	MaxNum	51.94	44.24	16.34	[33.84]
	MinMaxC	19.55	15.66	13.57	[19.29]
20-4	MaxNum	41.77	[88.48]	36.81	34.13
	MinMaxC	30.71	24.06	15.26	7.72
20-5	MaxNum	71.06	54.73	21.55	[34.65]
	MinMaxC	33.40	22.19	14.76	14.49

Note that the partition-based valid inequalities decrease solution times in most instances for $B = 4$, but in a much smaller portion of instances for $B = 8$. This decrease in effectiveness is due to the fact that the relaxations 4–14 and 4–16 become weaker as B increases, because the envelope function $g_i(B)$ (or $g'_i(B)$) more loosely approximates the true value function $\eta_i(B)$ (or $\eta'_i(B)$) as B increases.

Table 4-6 reports optimality gaps for solving 40-node instances. In addition to the same observations from Tables 4-4 and 4-5, note that the impact of our valid inequalities

Table 4-5. CPU times for 30-node instances using 3-partition inequalities.

Instance	Prob.	$B = 4$		$B = 8$	
		Orig.	3-Partition	Orig.	3-Partition
30-1	MaxNum	467.92	384.43	289.14	235.28
	MinMaxC	462.93	391.20	166.24	[204.12]
30-2	MaxNum	467.93	452.96	209.58	[218.07]
	MinMaxC	331.22	[334.29]	98.64	87.35
30-3	MaxNum	502.85	479.30	725.49	623.58
	MinMaxC	217.05	172.45	117.54	[121.11]
30-4	MaxNum	516.72	446.82	202.18	183.71
	MinMaxC	345.67	[351.84]	94.25	[96.36]
30-5	MaxNum	432.40	328.66	189.62	171.55
	MinMaxC	479.24	443.74	143.82	143.30

is relatively weaker if generated from four rather than two partitions. This result is intuitive because the 2-partition inequalities are based on subgraphs that jointly contain more of the original graph edges than the 4-partition subgraphs contain.

Table 4-6. Solution gaps for 40-node instances using 2- and 4-partition inequalities.

Instance	Prob.	$B = 4$			$B = 8$		
		Orig.	2-Partition	4-Partition	Orig.	2-Partition	4-Partition
40-1	MaxNum	131.39%	58.12%	131.35%	87.82%	48.07%	87.79%
	MinMaxC	27.82%	11.11%	27.85%	62.47%	32.82%	[62.49%]
40-2	MaxNum	124.51%	124.51%	110.29%	84.68%	33.78%	74.97%
	MinMaxC	26.19%	6.95%	20.42%	58.52%	21.10%	[58.68%]
40-3	MaxNum	122.56%	44.99%	112.14%	85.94%	85.92%	[88.85%]
	MinMaxC	25.92%	25.77%	25.85%	58.17%	57.09%	47.38%
40-4	MaxNum	114.68%	59.95%	[128.20%]	95.47%	49.98%	86.80%
	MinMaxC	27.93%	27.93%	27.87%	61.52%	47.38%	47.50%
40-5	MaxNum	125.26%	44.99%	120.01%	84.15%	53.76%	[100.18%]
	MinMaxC	26.25%	26.21%	26.20%	59.17%	44.65%	51.80%

CHAPTER 5
BROADCAST DOMINATION NETWORK DESIGN PROBLEM

5.1 Problem Description and Literature Survey

Let $G(V, E)$ be an undirected graph with node set $V = \{1, \ldots, n\}$, and edge set $E \subset V \times V$, $|E| = m$. A *dominating set* (DS) S is a subset of V, such that every node in V is *dominated* by at least one node in S. (A node in S dominates itself and all nodes to which it is adjacent in G.) Domination began as a problem on a chessboard (e.g., de Jaenisch, 1862), in which the minimum number of queens is placed on a chessboard so that every square can be reached by a queen in no more than a single move. A standard form of the domination problem seeks a DS that has the minimum node cardinality (Berge, 1962; Ore, 1967). Domination-related problems received much attention after the 1970s due to their importance in solving military, production, medicine, and many other practical problems. Over one thousand papers have been published on domination and its related topics, such as independence and covering (Haynes et al., 1998b). We refer readers to Haynes et al. (1998b,a) and Shen (2011) for comprehensive reviews of theories and algorithms that have been studied regarding fundamental and advanced topics of domination.

One recent study by Erwin (2004) introduces a domination variation called *broadcast domination* (BD), which assigns integer powers to each node. A node i having a positive power dominates itself and all nodes whose distances from i are no more than the power assigned to node i. The goal is to minimize the sum of all power values in the DS. By allowing power values to be larger than one, BD accommodates properties of some practical applications that cannot be accommodated in traditional DS settings. For instance, sensor towers having various signal strengths can be built at certain locations to monitor a network that represents some geographical area. BD is solvable in polynomial time on unweighted and undirected graphs (Heggernes & Lokshtanov, 2006), but its complexity remains open on weighted graphs (Lokshtanov, 2007). Blair

et al. (2004) develop polynomial-time algorithms for solving BD on trees, interval graphs, and series-parallel graphs. Blair & Horton (2005) and Dunbar et al. (2006) have studied several variants of BD and their relationships.

The problem that we consider in this chapter seeks an optimal edge construction solution that minimizes the weighted sum of edge construction costs, and the corresponding minimum BD cost over the augmented network. We refer to this problem as the Broadcast Domination Network Design problem (BDND). The idea is motivated by applications arising in areas such as homeland security, social networks, and medical resource allocation. We describe two such examples as follows.

- Recall the sensor tower allocation problem mentioned above. Now assume that we can also decide to install intermediate receivers between certain location pairs to enhance the signal transmission, which is equivalent to building edges that shorten geographical distances between node pairs in BDND settings. The installation of signal receivers could potentially decrease the minimum BD cost, but will incur extra operational cost, and the problem at hand is to determine an optimal tradeoff between these two types of costs.
- To distribute medical resources in an epidemic-outbreak scenario, a decision maker builds a set of size-diversified resource centers, each of which is able to reach contagious points within its ability. Meanwhile, a set of point pairs can be connected to decrease resource distribution difficulties, e.g., by building roads or local epidemic-control centers. The decision maker aims to minimize the total resource allocation and distribution costs.

In this chapter, we consider BDND on both unweighted and edge-weighted graphs, and show that it is NP-hard in the strong sense. We first formulate the problem as a monolithic mixed-integer program (MIP), but the formulation is weak and difficult to solve. As an alternative, we derive a decomposition strategy and associated cutting-plane algorithm. We prescribe modified Benders cut coefficients by proving bounds on the reduction in cost that results from adding new edges to BD subproblem. Relevant to the techniques we propose, modified Benders cuts for solving general two-stage MIPs have been examined in several different forms. Hooker (2000) and Hooker & Ottosson (2003) generalize the subproblem LP dual to an “inference”

dual, whose solution takes the form of a logical deduction that yields Benders cuts. The logic-based Benders cuts are designed to be valid using a proof of infeasibility or optimality when solving the subproblem, and their computational efficacy is demonstrated in general by Cambazard et al. (2004) and Cambazard & Jussien (2005), and particularly in planning and scheduling problems by Hooker (2005a,b, 2007). Codato & Fischetti (2006) later propose a type of combinatorial Benders cuts based on finding minimal sets of 0-1 variable assignments that create a conflict in the subproblem, which is a special case of the logic-based Benders cuts. Penuel et al. (2010) develop a cutting-plane algorithm for solving a two-stage stochastic facility location-activiation problem, in which binary decision variables appear in both stages of the problem. Their cuts are at least as strong as the LL cuts, although they are derived specifically for the facility location-activation problem and do not appear to be extendable to general two-stage mixed-integer programming problems.

Our cutting planes are derived by supposing that an edge is added to the graph, adjusting the solution to the modied subproblem, and then constructing a feasible solution to the original graph without the edge being added. The value of this feasible solution provides valid coefficients for the Benders cut. As we will show in Section 5.4, our method converges in a finite number of iterations. Also, we demonstrate that our cuts are at least as strong as the general-purpose LL inequality.

The remainder of the chapter is organized as follows. In Section 5.2, we introduce BD notation and solution concepts relevant to our current study. In Section 5.3, we show that BDND is $\mathcal{NP}$-hard, and provide a MIP formulation for the problem. We then decompose the problem, and generate valid inequalities for solving BDND for unweighted graphs in Section 5.4, and for weighted graphs in Section 5.5. Section 5.6 discusses a computational analysis of our proposed approaches.

5.2 BD Notation and Solution Concepts

Denote $N(i) = \{j \in V \mid (i,j) \in E\}$, and define set $N(V_S) = \cup_{i \in V_S}(N(i) \setminus V_S)$ as the *open neighborhood* of $V_S \subseteq V$. Let G_{V_S} be a subgraph of G induced by V_S, which contains all nodes in V_S and all edges in E incident to two nodes in V_S. Let d_{ij} be the distance between any pair of nodes i and j in G, defined as $d_{ii} = 1,\ \forall i \in V$, and $d_{ij} =$ length of a shortest path between node i and node j, for all $i,\ j \in V,\ i \neq j$. A *broadcast* on G is a set of "power values" f_i for all $i \in V$, and we define node set $V_f = \{i \in V \mid f_i > 0\}$. A broadcast f is called a BD if for any node $j \in V$ there exists a node $i \in V_f$ such that $d_{ij} \leq f_i$. Finally, denote $rad(G) = \min_{i \in V}\{\max_{j \in V,\ j \neq i}\{d_{ij}\}\}$ as the *radius* of G, and note that it is a valid upper bound on the optimal BD cost for any connected graph G.

We illustrate the notation in Figure 5-1 on an unweighted graph. The open neighborhood of node 3 is $N(3) = \{2, 4, 5\}$. The subgraph $G_{\{3\} \cup N(3)}$ is characterized by node set $\{2, 3, 4, 5\}$ and edge set $\{(2, 3), (3, 4), (3, 5)\}$. The radius $rad(G) = 4$ can be obtained either by $\max_{j \in V \setminus \{5\}}\{d_{5j}\}$ or by $\max_{j \in V \setminus \{6\}}\{d_{6j}\}$. The optimal BD solution is $f_3 = 2,\ f_9 = 1$, and $f_i = 0$ for all other nodes $i \in V$, corresponding to the optimal BD cost of 3.

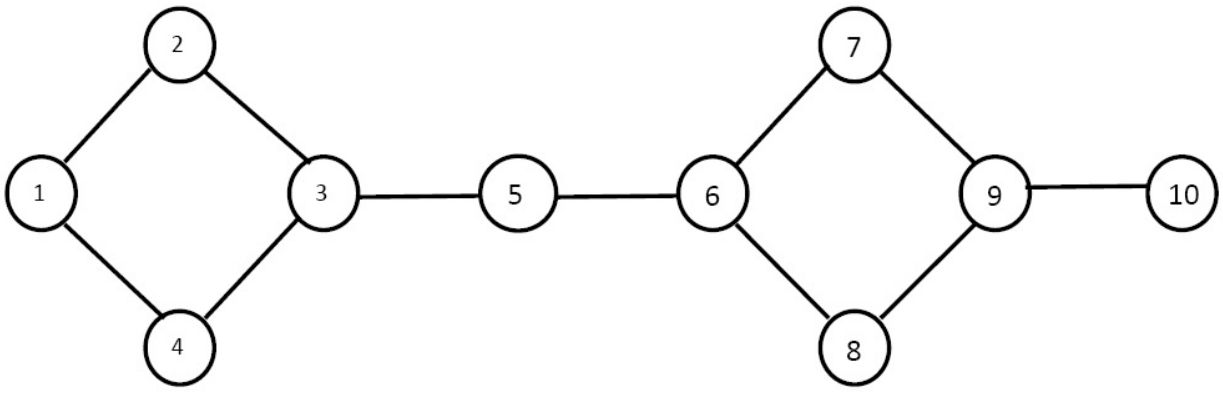


Figure 5-1. Illustration of notation and definitions.

Heggernes & Lokshtanov (2006) develop a polynomial-time algorithm for optimally solving BD on unweighted and undirected connected graphs, which we employ later in

the chapter. While we refer the readers to Heggernes & Lokshtanov (2006) for the full details of the algorithm, some of the concepts behind their solution are particularly relevant to our chapter. For all $i \in V_f$, define $B(i, f_i)$ as a *ball* of node i, which corresponds to the set of all nodes that are within a distance from f_i to node i. Node i is said to be the center node of $B(i, f_i)$ when $f_i > 0$. A BD solution f is *efficient* if $B(i, f_i) \cap B(j, f_j) = \emptyset$ for all $i, j \in V_f$, $i \neq j$. Define a domination graph $G_f = (V_f, E_f)$ corresponding to a solution f where $E_f = \{(i, j) \mid N(B(i, f_i)) \cap B(j, f_j) \neq \emptyset\}$. Heggernes & Lokshtanov (2006) show that there always exists an optimal BD solution f that is efficient on any unweighted graph G. This result leads to the proof that an optimal broadcast solution exists such that the degree of any node in its associated domination graph G_f is no more than 2, and thus G_f is either a path or a cycle. This clever insight permits the authors to derive an $O(n^6)$ optimal BD algorithm.

The polynomial-time algorithm of Heggernes & Lokshtanov (2006) constructs a directed node-weighted auxiliary graph $\mathcal{G}_u$ for any given node $u \in V$ as follows. For each node $v \in V$ and weight value $p \in \{1, \ldots, rad(G)\}$, there exists a node (v, p) in $\mathcal{G}_u$ either (a) if $u \in B(v, p)$ and $G(V \setminus B(v, p))$ is connected or empty, or (b) if $u \notin B(v, p)$ and $G(V \setminus B(v, p))$ has at most two connected components. They partition the node set of $\mathcal{G}_u$ into four subsets: $A_u = \{(v, p) : G(V \setminus B(v, p))$ is connected and $u \in B(v, p)\}$, $B_u = \{(v, p) : G(V \setminus B(v, p))$ has two connected components and $u \notin B(v, p)\}$, $C_u = \{(v, p) : G(V \setminus B(v, p))$ is connected and $u \notin B(v, p)\}$, and $D = \{(v, p) : B(v, p) = V\}$. For instance, if we let $u = 1$ in the graph depicted by Figure 5-1, nodes $(1, 1), \ldots, (1, 4)$ would belong to A_1. Node $(5, 1)$ is one node that would belong to B_1, and $(9, 1)$ is one node that would belong to C_1. Subset D would consist of nodes $(5, 4)$ and $(6, 4)$.

Moreover, for each vertex (v, p), denote $L_u(v, p)$ (similarly, $R_u(v, p)$) as the possibly empty connected component of $G(V \setminus B(v, p))$ that contains (does not contain) node u. A directed edge from (v, p) to (w, q) exists in $\mathcal{G}_u$ if and only if it meets the following

three conditions: (a) $B(v,p) \cap B(w,q) = \emptyset$ in G; (b) $R_u(v,p) \neq \emptyset$ and $L_u(w,q) \neq \emptyset$; (c) $B(v,p)$ contains all neighbors of $B(w,q)$ in $L_u(w,q)$, and $B(w,q)$ contains all neighbors of $B(v,p)$ in $R_u(v,p)$. For instance in Figure 5-1, an arc from $(3,2)$ to $(9,1)$ would exist in $\mathcal{G}_1$. Note that the role of u is to define an endpoint of a path corresponding to a BD solution, all vertices in A_u have zero indegree, all vertices in C_u have zero outdegree, and every vertex in D defines a radial BD on its own. Hence, finding a shortest path from node sets $A_u \cup D$ to $C_u \cup D$ in $\mathcal{G}_u$, over all possible choices of $u \in V$, will yield a minimum BD path (or a singleton node in D) on G.

Finally, we describe Heggernes and Lokshtanov's algorithm as follows. Given an unweighted graph $G(V,E)$, set an initial BD cost to its upper bound value $rad(G)+1$. For every vertex $x \in V$ and a possible power value $k = 1, \ldots, rad(G)$, if the remaining subgraph $G'' = G(V \setminus B(x,k))$ is connected or empty, the algorithm solves a minimum BD path problem on G'' by using the previous procedures. If the minimum BD path cost on G'' plus power k is smaller than the current BD cost, the algorithm assigns power k at vertex x, updates the current optimal BD solution to G'', and changes the current BD cost to the sum value. Note that this algorithm will yield a minimum BD solution whose balls are connected as a path or a cycle in $\mathcal{G}_u$, depending on whether $B(x,k)$ is connected with only one or two balls in BD path solution to G''.

In this chapter, we iteratively solve a relaxed master problem that constructs graph edges in the first stage, and then generate valid inequalities based on optimal BD solutions to the current graph in the second stage. For the unweighted graph case, we calculate optimal BD solutions by using the polynomial-time algorithm described above. The complexity of solving the weighted BD remains open (Lokshtanov, 2007), and we solve BD subproblem by formulating it as a MIP.

5.3 BDND Formulations and Complexity

Let $W = \{(i,j) \mid i, j \in V,\ i \neq j,\ (i,j) \notin E\}$ be a set of potential edges, given as input, that can be built in G. For every node pair $(i,j) \in W$, we associate a binary

variable q_{ij} such that $q_{ij} = 1$ if we construct (i, j), and $q_{ij} = 0$ otherwise. Let c_{ij} be the (nonnegative) cost of constructing link $(i, j) \in W$, and $G(q) = G(V, E \cup \{(i, j) \in W \mid q_{ij} = 1\})$ denote the resulting graph after we add all edges that correspond to q. The goal is to minimize the sum of the construction cost $\sum_{(i,j)\in W} c_{ij} q_{ij}$ and the corresponding BD cost $\delta \sum_{i \in V} f_i$ over graph $G(q)$, where δ is some given positive weight.

We show in Section 5.3.1 that BDND is $\mathcal{NP}$-hard in the strong sense. In Section 5.3.2 we provide a MIP formulation for BDND, and then set up foundational concepts related to our proposed decomposition approaches in Section 5.3.3.

5.3.1 BDND Complexity

Letting $\mathcal{L}$ be a given cost target, $\overline{\text{BDND}}$ is the decision version of BDND that seeks a solution (f, q) having a total cost not more than $\mathcal{L}$. We show that $\overline{\text{BDND}}$ is $\mathcal{NP}$-complete by using a transformation from exact cover by 3-sets (X3C) (Garey & Johnson (1979)).

Theorem 5.1. *$\overline{\text{BDND}}$ is $\mathcal{NP}$-complete in the strong sense, even if $G(V, E)$ is unweighted and undirected.*

Proof. It is easy to see that $\overline{\text{BDND}}$ belongs to $\mathcal{NP}$ by simply examining whether a guessed solution $(\hat{f}, \hat{q})$ is feasible and has a cost that does not exceed $\mathcal{L}$.

Next, we show that $\overline{\text{BDND}}$ is $\mathcal{NP}$-complete by a transformation from X3C. Given a set $X = \{x_1, \ldots, x_n\}$ with $n = 3b$ for some positive integer b, and a collection $C = \{C_1, \ldots, C_m\}$ of 3-element subsets of X, X3C seeks an exact cover for X, i.e., a subset $C' \subseteq C$ with $|C'| = b$ such that every element of X occurs in exactly one member of C'. We transform any arbitrary X3C instance to a $\overline{\text{BDND}}$ instance, and show that the X3C instance has a solution if and only if the transformed $\overline{\text{BDND}}$ instance has a solution. Without loss of generality, we assume that $b \geq 2$ and each element x_i is included in at least one clause in C.

We create one node for each element x_i, $\forall i = 1, \ldots, n$, and one for each clause C_j, $\forall j = 1, \ldots, m$. We call these *element nodes* $v_{x_1}, \ldots, v_{x_n}$ and *clause nodes* $v_{C_1}, \ldots, v_{C_m}$. We also create five dummy nodes denoted as $v_{d_1}, \ldots, v_{d_5}$. Let E be given by $\{(v_{C_j}, v_{C_k})$

$| C_j,\ C_k \in C,\ j \neq k\} \cup \{(v_{C_j}, v_{x_i}) \mid C_j \in C,\ x_i \in C_j\} \cup \{(v_{d_i}, v_{d_{i+1}}) \mid i = 1, \ldots, 4\}$. That is, E consists of a clique of edges between clause nodes, edges between each clause node and its three corresponding element nodes, and edges between adjacent dummy nodes. Let $W = \{(v_{d_3}, v_{C_j}) \mid j = 1, \ldots, m\}$, and let the cost of building edges between v_{d_3} and each clause node be 1. This transformation is illustrated in Figure 5-2, where $X = \{x_1, \ldots, x_{12}\}$, $C_1 = \{x_1, x_2, x_7\}$, $C_2 = \{x_3, x_4, x_{12}\}$, $C_3 = \{x_8, x_9, x_{10}\}$, $C_4 = \{x_2, x_4, x_7\}$, $C_5 = \{x_5, x_6, x_{11}\}$, and $C_6 = \{x_3, x_6, x_9\}$. The solid lines represent edges in W that we choose to build, and the dashed ones illustrate the coverage of all nodes by v_{d_3} in an optimal broadcast domination in which $f_{v_{d_3}} = 2$. A feasible solution $C' = \{C_1,\ C_2,\ C_3,\ C_5\}$ to X3C corresponds to the illustrated BDND solution.

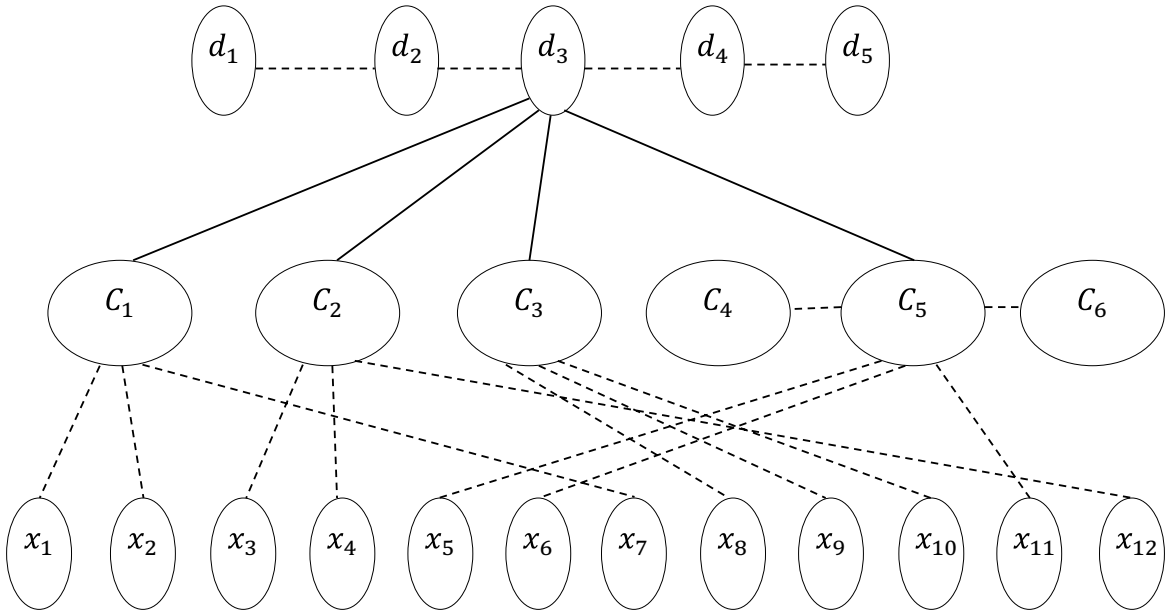


Figure 5-2. An example transformation from X3C to an $\overline{\text{BDND}}$ instance.

Set $\delta = b + 1$ and $\mathcal{L} = 3b + 2$, and suppose that the X3C instance has a solution $C' \subseteq C$ with $|C'| = b$. We can recover a corresponding solution to $\overline{\text{BDND}}$ by building

edges between v_{d_3} and each $v_{C_j} \in C'$ at a cost of b. As a result, a power of 2 assigned to node v_{d_3} covers all nodes in the graph, and the total cost is $3b + 2 = \mathcal{L}$.

Next, we show that a feasible solution to X3C can be obtained from a feasible solution to $\overline{\text{BDND}}$. First, observe that even if we construct all potential edges in W, the unique optimal BD solution has objective function value 2 with $f^*_{v_{d_3}} = 2$ and $f^*_i = 0$ for all other nodes $i \in V$. Solution f^* is a BD since v_{d_3} reaches dummy nodes $v_{d_1}, \ldots, v_{d_5}$ using the path $v_{d_1} - v_{d_2} - \cdots - v_{d_5}$, all clause nodes $v_{C_j},\ j = 1, \ldots, m$ (which would be adjacent to v_{d_3}), and all element nodes $v_{x_i},\ i = 1, \ldots, n$, using a length-of-two path v_{d_3}–v_{C_j}–v_{x_i}, for a clause C_j that includes x_i (assumed to exist). Also, a distinct feasible BD solution having an objective value of 2 must set either $f_{v_{d_1}} = 1$ or $f_{v_{d_2}} = 1$ to cover node v_{d_1}, and set either $f_{v_{d_4}} = 1$ or $f_{v_{d_5}} = 1$ to cover node v_{d_5}. However, no such solution could cover a node $v_{x_i},\ i = 1, \ldots, n$ (or even a node v_{C_j}), and so only one solution having objective 2 exists. Next, note that in a feasible $\overline{\text{BDND}}$ solution, it is impossible for a BD cost to exceed 2; otherwise the target cost $\mathcal{L} = 3b + 2$ is violated by setting the broadcast cost to 3 or more. Therefore, any feasible solution to the $\overline{\text{BDND}}$ instance includes $f_{v_{d_3}} = 2$, and $f_i = 0$ for all other nodes $i \in V$.

Now, if $\overline{\text{BDND}}$ has a solution, the solution must build exactly b edges of the form $(v_{d_3}, v_{C_j}),\ j = 1, \ldots, m$. (If there are fewer than b such edges, then v_{d_3} would not be able to cover at least three element nodes with $f_{v_{d_3}} = 2$, and if we build $k > b$ edges, the objective value of $k + 2\delta = 2b + 2 + k$ exceeds $\mathcal{L}$.) Let C' be the X3C clauses corresponding to the b clause nodes in the $\overline{\text{BDND}}$ solution that are adjacent to v_{d_3}. Note that if v_{C_j} is not connected to v_{d_3}, it is covered by node v_{d_3} via the path v_{d_3}–v_{C_i}–v_{C_j} for some $C_i \in C'$. Also, since the $\overline{\text{BDND}}$ solution covers all element nodes, each element node is adjacent to some node v_{C_i} with $C_i \in C'$. Thus, C' is a solution to the X3C because its cardinality is b and it contains all n elements in X.

Because all numerical data in the proof is polynomially bounded by the input size, and X3C is $\mathcal{NP}$-complete in the strong sense, $\overline{\text{BDND}}$ is also $\mathcal{NP}$-complete in the strong sense. □

5.3.2 MIP Formulation for BDND

We now formulate a MIP for solving BDND on general graphs, in which w_{ij} is a positive integer weight associated with every edge $(i,j) \in E \cup W$. (We assume that $w_{ij} \geq 1$ for all $(i,j) \in E \cup W$.) For all $i,\ j \in V$, let x_{ij} be a binary variable such that $x_{ij} = 1$ if node j is covered by node i, and $x_{ij} = 0$ otherwise. For all $i,\ j \in V,\ i \neq j,\ (u,v) \in E \cup W$, define variables $g_{uv}^{ij} \in \{0,1\}$ as "edge-covering" indicators, such that $g_{uv}^{ij} = 1$ if node i covers node j through a path that includes edge (u,v); otherwise, $g_{uv}^{ij} = 0$. Note that $|V|$ is a valid upper bound for the optimal BD cost (because $f_i = 1,\ \forall i \in V$, is always feasible). Alternatively, let $G_c,\ \forall c \in \mathcal{C}$ be the set of maximally connected components of G. Then $\sum_{c \in \mathcal{C}} rad(G_c)$ is also a valid upper bound on the optimal BD cost. Define $UB = \min\{|V|,\ \sum_{c \in \mathcal{C}} rad(G_c)\}$. We then add an artificial edge to E having a weight of $UB + 1$ for each node pair $(i,j),\ i \neq j$. Finally, define $A(i) = \{j : (i,j) \in E \cup W\}$ as the set of nodes adjacent or potentially adjacent to node i. The MIP model is

$$\textbf{BMIP:}\ \min \quad \sum_{(i,j) \in W} c_{ij} q_{ij} + \delta \sum_{i \in V} f_i \tag{5–1a}$$

$$\text{s.t.} \quad \sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \tag{5–1b}$$

$$f_i - x_{ii} \geq 0 \quad \forall i \in V \tag{5–1c}$$

$$\sum_{l \in A(i)} g_{il}^{ij} = 1 \quad \forall i,\ j \in V,\ i \neq j \tag{5–1d}$$

$$\sum_{l \in A(u)} g_{ul}^{ij} - \sum_{l \in A(u)} g_{lu}^{ij} = 0 \quad \forall i,\ j \in V,\ i \neq j,\ \forall u \in V - \{i,j\} \tag{5–1e}$$

$$\sum_{(u,v) \in E \cup W} w_{uv} g_{uv}^{ij} - f_i + (UB+1) x_{ij} \leq UB + 1 \quad \forall i,\ j \in V,\ i \neq j \tag{5–1f}$$

$$g_{uv}^{ij} \leq q_{uv} \quad \forall i,\ j \in V,\ i \neq j,\ (u,v) \in W \tag{5–1g}$$

$$f,\ g \geq 0,\ q,\ x \text{ binary}, \tag{5–1h}$$

where objective function 5–1a minimizes the sum of the edge construction and BD costs. Constraints 5–1b require that every node is covered by some node, and Constraints 5–1c allow a node to cover itself only if it has a positive power. Constraints 5–1d and 5–1e are flow balance constraints that enforce a covering path from i to j. Moreover, Constraints 5–1f ensure that if node i is designated to cover node j, the total distance of the chosen path from i to j is no more than f_i; otherwise, the right-hand-side of Constraints 5–1f becomes $(UB + 1)$, and the g-value associated with the artificial edge (i, j) can be set to one to satisfy the constraint. Constraints 5–1g ensure that for each $(u, v) \in W$, g^{ij}_{uv} can be set to one only if edge (u, v) has been constructed. Finally, Constraints 5–1h impose binary requirements on variables q and x.

The binary requirement on g-variables are relaxed as $g \geq 0$ indicated in Constraints 5–1h, because all extreme-point solutions have binary-valued g-variables. To see this, note that given any binary-valued $\hat{q}$ and $\hat{x}$, BMIP seeks a shortest path between all node pairs i and j such that $\hat{x}_{ij} = 1$. Then for all $(u, v) \in E \cup W$, $g^{ij}_{uv} = 1$ if (u, v) belongs to this shortest path, and $g^{ij}_{uv} = 0$ otherwise. Moreover, there also exists an integer f-solution at optimality by letting $f_i = \max\{0, \max_{j \in V : \hat{x}_{ij}=1} \{\sum_{(u,v)\in E\cup W} w_{uv} g^{ij}_{uv}\}\}, \forall i \in V$ (noting by assumption that all w-values are positive integers), and thus Constraints 5–1h only enforce $f \geq 0$.

5.3.3 A Decomposition Approach for BDND

The approach we take in this chapter decomposes BMIP into two stages, and employs cutting-plane algorithms for solving BDND. The master problem is given by

$$\text{BDND-M:} \quad \min \quad \sum_{(i,j)\in W} c_{ij} q_{ij} + \delta\eta \tag{5–2a}$$

$$\text{s.t.} \quad \eta + m^l q \geq n^l \quad \forall l \in L \tag{5–2b}$$

$$\eta \geq \underline{\eta}, \qquad q \text{ binary}, \tag{5–2c}$$

where $\eta = \min_f \sum_{i\in V} f_i$ is the minimum BD cost on $G(q)$. Set L designates a set of valid inequalities derived for the subproblem, where Ineq. $l \in L$ has coefficients m^l and n^l.

The inequalities in Constraints 5–2b must essentially replace the constraints in BMIP, and must force η to take on the optimal BD objective function on $G(q)$. We denote $\underline{\eta}$ as the smallest broadcast cost that could be incurred for any choice of q. Note that $\underline{\eta} = 1$ is always valid, but a stronger bound can be obtained (in polynomial time for unweighted graphs) by computing $\underline{\eta}$ as the optimal BD objective when all edges in W are added to G.

As in a typical Benders decomposition setup, we will need to solve a subproblem to generate Inequalities 5–2b. The idea is to relax Constraints 5–2b, and call the resulting formulation the relaxed master problem (RMP). We then solve RMP and obtain an optimal solution $(\bar{q}, \bar{\eta})$. The subproblem determines if the optimal BD objective on $G(\bar{q})$ matches $\bar{\eta}$. If so, then $(\bar{q}, \bar{\eta})$ is optimal. Otherwise, an inequality is added to constraint set 5–2b, the RMP is re-solved, and the process continues. The algorithm converges if the inequalities in 5–2b must eventually force η to equal the optimal BD objective, given q. In a traditional Benders algorithm, the subproblem is a linear program (or at least a convex optimization problem) for which duality information can be obtained, leading to inequalities that force the algorithm to converge.

However, no LP formulation is yet known as a function of q for either the unweighted or weighted BDND. One potential starting point for our decomposition algorithm considers the following MIP subproblem:

$$\begin{aligned} \text{BDND-S}(\hat{q}): \quad \min \quad & \sum_{i \in V} f_i \\ \text{s.t.} \quad & \text{Constraints 5–1b through 5–1h with } q \text{ fixed to } \hat{q}. \end{aligned}$$

While Benders inequalities cannot be obtained from BDND-S($\hat{q}$) due to the presence of integer x-variables in the formulation, it is still possible to relax the subproblem and generate valid inequalities based on the relaxation. Let λ_j, β_i, γ_{ij}, π^{ij}_u, θ_{ij}, and τ^{ij}_{uv} be the dual variables associated with Constraints 5–1b, 5–1c, 5–1d, 5–1e, 5–1f, and 5–1g. The dual formulation of the subproblem LP relaxation of BDND-S($\hat{q}$) is given as follows.

D-BDND-S($\hat{q}$):

$$\max \quad \sum_{j\in V}\lambda_j + \sum_{i\in V}\sum_{j\in V,\ j\neq i}[\gamma_{ij} + (UB+1)\theta_{ij}] + \sum_{(u,v)\in W}\hat{q}_{uv}\sum_{i\in V}\sum_{j\in V,\ j\neq i}\tau^{ij}_{uv} \tag{5–4a}$$

$$\text{s.t.} \quad \lambda_i - \beta_i \leq 0 \quad \forall i \in V \tag{5–4b}$$

$$\lambda_j + (UB+1)\theta_{ij} \leq 0 \quad \forall i,\ j \in V,\ i \neq j \tag{5–4c}$$

$$\beta_i - \sum_{j\in V,\ j\neq i}\theta_{ij} \leq 1 \quad \forall i \in V \tag{5–4d}$$

$$\gamma_{ij} - \pi^{ij}_v + w_{iv}\theta_{ij} + \tau^{ij}_{iv} \leq 0 \quad \forall i,\ j \in V,\ i \neq j,\ v \in A(i) \tag{5–4e}$$

$$\pi^{ij}_u - \pi^{ij}_v + w_{uv}\theta_{ij} + \tau^{ij}_{uv} \leq 0 \quad \forall i,\ j \in V,\ i \neq j,\ u \neq i,\ j,\ (u,v) \in E \cup W \tag{5–4f}$$

$$\beta \geq 0,\ \theta,\ \tau \leq 0,\ \lambda,\ \gamma,\ \pi \text{ unrestricted}, \tag{5–4g}$$

where $\tau^{ij}_{uv} \equiv 0$ for all $(u,v) \in E$ for notational convenience. Note that Constraints 5–4b, 5–4c, 5–4d, 5–4e, and 5–4f are dual constraints associated with primal variables x_{ii}, x_{ij} $(i \neq j)$, f_i, g^{ij}_{iv} $(v \in A(i))$, and g^{ij}_{uv} $(u \neq i,\ j,\ (u,v) \in E \cup W)$, respectively. At each iteration, given a first-stage solution $\hat{q}$, we solve D-BDND-S($\hat{q}$) to optimality, yielding an optimal dual solution $\hat{\lambda}$, $\hat{\beta}$, $\hat{\gamma}$, $\hat{\pi}$, $\hat{\theta}$, and $\hat{\tau}$. (An optimal dual solution must exist, because the optimal primal objective is bounded between 0 and UB regardless of q, and hence always has an optimal solution.) Noting that variables q only appear in the objective function of D-BDND-S(q), we obtain the following Benders cut:

$$\eta - \sum_{(u,v)\in W}\left(\sum_{i\in V}\sum_{j\in V,\ j\neq i}\hat{\tau}^{ij}_{uv}\right)q_{uv} \geq \sum_{j\in V}\hat{\lambda}_j + \sum_{i\in V}\sum_{j\in V,\ j\neq i}\left[\hat{\gamma}_{ij} + (UB+1)\hat{\theta}_{ij}\right]. \tag{5–5}$$

However, while cuts 5–5 are indeed valid, the use of these inequalities in a Benders decomposition setting will only force η to converge to the optimal LP relaxation value of BDND-S($\hat{q}$), not necessarily to the actual integer BD objective. Hence, these cuts may be useful in early iterations of our decomposition approach, but must be augmented with more inequalities for the algorithm to converge. We explore such inequalities in the following sections.

Remark 5.1. If Cut 5–5 is a "*low-intensity cut*," i.e., if the cut contains few nonzero coefficients of the q-variables, we may choose to apply a covering cut bundle (CCB) generation strategy (Saharidis et al., 2010b). This strategy generates valid inequalities that include more nonzero coefficients of the q-variables, and has been shown to accelerate the convergence of the Benders decomposition approach when the subproblems are continuous. The CCB procedure is summarized as follows in the context of BDND. First, a variable q_{uv} is defined to be T-covered in a Benders cut of the form 5–5 if the magnitude of its coefficient obeys

$$\left|\left(\sum_{i\in V}\sum_{j\in V,\ j\neq i}\hat{\tau}^{ij}_{uv}\right)\right| \geq T\mathrm{Max}_{(u,v)\in W}\left|\left(\sum_{i\in V}\sum_{j\in V,\ j\neq i}\hat{\tau}^{ij}_{uv}\right)\right|,$$

where T is a given parameter from interval $[0,1]$. A set of cuts are called T-CCB if *every* variable q_{uv} is T-covered at least in one cut in a bundle. Thus, in lieu of generating a single cut 5–5 in every iteration, we could accumulate a set of Benders cuts by examining the T-covered property of every q-variable in the previously generated cuts. If some q_{uv} is not T-covered, we impose bounds on its corresponding τ_{uv}-variables and re-compute D-BDND-S($\hat{q}$), yielding alternative cuts in which q_{uv} is T-covered. We repeat this process until either a T-CCB is generated, or the number of bundle cuts generated exceeds a given maximum number. The efficacy of performing CCB in the early phase of our computation, while cuts of the form 5–5 are still being generated, will be demonstrated in Section 5.6.

5.4 BDND Inequalities for Unweighted Graphs

Consider an RMP in which only a certain subset of Inequalities 5–2b are present, and suppose that we have obtained an optimal RMP solution $\hat{q}$. Define $\widehat{Q}^b = \{(i,j) \in W :\ \hat{q}_{ij} = b\}$, for $b \in \{0,1\}$. Denote $z(\widehat{Q})$ as the optimal BD cost on $\overline{G}(V, E \cup \widehat{Q})$. We aim to derive values $\alpha_{ij}(\widehat{Q}^1) \geq 0$ such that

$$z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widehat{Q}^c}\alpha_{ij}(\widehat{Q}^1)$$

holds true for *any* subsets $\widehat{Q}^1 \subseteq W$ and $\widehat{Q}^c \subseteq \widehat{Q}^0$, and therefore we can generate valid inequalities 5–2b of the following form:

$$\eta \geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widehat{Q}^0} \min\{\alpha_{ij}(\widehat{Q}^1), z(\widehat{Q}^1) - \underline{\eta}\}\ q_{ij}, \tag{5–6}$$

where $(z(\widehat{Q}^1) - \underline{\eta})$ are essentially coefficients given by the Laporte-Louveaux (LL) inequality (Laporte & Louveaux, 1993). Inequality 5–6 is stronger than the LL inequality when $\alpha_{ij}(\widehat{Q}^1) < (z(\widehat{Q}^1) - \underline{\eta})$ for some $(i,j) \in \widehat{Q}^0$.

Remark 5.2. Note that Inequality 5–6 is not generally valid when $\alpha_{ij}(\widehat{Q}^1) = z(\widehat{Q}^1) - z(\widehat{Q}^1 \cup (i,j)),\ \forall (i,j) \in \widehat{Q}^0$.

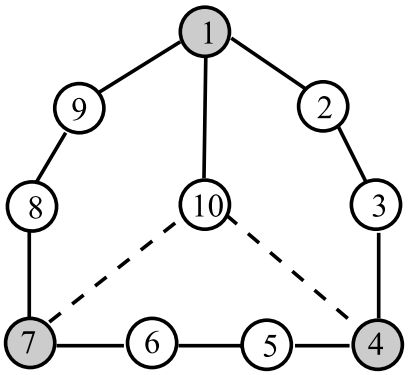


Figure 5-3. Illustration of invalid cut coefficients.

Consider the unweighted graph given in Figure 5-3 in which $|\widehat{Q}^c| = 2$, where the solid lines represent edges $(E \cup \widehat{Q}^1)$ and the dashed lines represent the edges in $\widehat{Q}^c$. The cost $z(\widehat{Q}^1) = 3$ by letting $f_1 = f_4 = f_7 = 1$, and $f_i = 0$ for all other nodes i. Adding either edge $(4, 10)$ or $(7, 10)$ to the graph, the optimal objective function value stays unchanged. However, if we add both $(4, 10)$ and $(7, 10)$, the optimal BD cost $z(\widehat{Q}^1 \cup \widehat{Q}^c) = 2$ with $f_{10} = 2$, and $f_i = 0$ for all other nodes i, and $z(\widehat{Q}^1 \cup \widehat{Q}^c) = 2 < 3 = z(\widehat{Q}^1) - 0 - 0$. □

The remainder of this section discusses two methods for calculating valid α-values in Inequality 5–6, which we obtain from an optimal BD solution on $G(\widehat{Q}^1)$.

5.4.1 Cuts Generated by Conservative Coefficient Estimation

We begin by stating the following lemma, which establishes a general upper bound for α-values.

Lemma 7. *Given an unweighted and undirected graph $G(V, E)$, and edges $\widehat{Q}^1$ constructed in the first-stage BDND solution, the following inequality*

$$z(\widehat{Q}^1 \cup (i,j)) \geq \min\left\{z(\widehat{Q}^1) - 1,\ \left\lfloor (z(\widehat{Q}^1)/2 \right\rfloor + 1\right\} \tag{5–7}$$

is valid for any $(i,j) \in \widehat{Q}^0$.

Proof. Let f_k be an integer power value at node k in any efficient feasible BD solution f on graph $\overline{G}(V, E \cup \widehat{Q}^1)$. At each node $k \in V$, let v_k be the center node of the ball covering node k (i.e., the node having a positive power value in the ball containing k), and define $p_k = f_{v_k} - d_{v_k k}$ as the *potential* associated with each node $k \in V$. (Hence, if $f_k \geq 1$, then $p_k = f_k$.)

Considering a graph $G'(V, E \cup \widehat{Q}^1 \cup (i,j))$ constructed by adding edge $(i,j) \in \widehat{Q}^0$ to $\overline{G}$, we denote p'_k and f'_k, $\forall k \in V$, as the respective potentials and power assignments in an optimal BD solution on G'. Next, we show how to recover a feasible solution (f, p) to $\overline{G}$ from solution (f', p') by analyzing the following cases.

- If nodes i and j are covered by different center nodes (i.e., $v_i \neq v_j$) in the solution f', then by letting $f_k = f'_k$, $\forall k \in V$, f remains feasible to $\overline{G}$, and thus $z(\widehat{Q}^1 \cup (i,j)) = \sum_{k\in V} f'_k = \sum_{k\in V} f_k \geq z(\widehat{Q}^1)$. (In fact, since $E \cup \widehat{Q}^1 \subset E \cup \widehat{Q}^1 \cup (i,j)$, this implies that $z(\widehat{Q}^1 \cup (i,j)) = z(\widehat{Q}^1)$.)

- If node i and node j are covered by a common node $v \in V$ in the solution f' for G' (i.e., $v = v_i = v_j$), then we consider the following two situations.

  1. In G', if there exists a shortest path from v to i that does not contain node j, and a shortest path from v to j that does not contain i, then the same logic in the first case shows that $z(\widehat{Q}^1 \cup (i,j)) = z(\widehat{Q}^1)$.
  2. Now suppose that edge (i,j) belongs to a shortest path in G' from v to some node l in $B(v, f'_v)$. Without loss of generality, we assume $p'_i = p'_j + 1$, and thus $f'_j = 0$. (Note that v and i could be the same node). After deleting (i,j), we will not necessarily be able to cover nodes j through l on this shortest path by

simply using f'. First, we assume that $p'_j > 0$. Letting $f_k = f'_k,\ \forall k \in V,\ k \neq j$ and $f_j = p'_j > 0$, we obtain a feasible solution to $\overline{G}(V, E \cup \widehat{Q}^1)$ because we set f_j the same as the previous potential at node j. Therefore, there exists a feasible BD solution on $\overline{G}(V, E \cup \widehat{Q}^1)$ having objective value

$$z(\widehat{Q}^1)^{feas} = \sum_{k \in V} f'_k + p'_j = z(\widehat{Q}^1 \cup (i,j)) + p'_j.$$

The optimal BD objective over $\widehat{Q}^1$ is then bounded as:

$$\begin{aligned} z(\widehat{Q}^1) &\leq z(\widehat{Q}^1)^{feas} = z(\widehat{Q}^1 \cup (i,j)) + p'_j = z(\widehat{Q}^1 \cup (i,j)) + p'_i - 1 \\ &\leq z(\widehat{Q}^1 \cup (i,j)) + z(\widehat{Q}^1 \cup (i,j)) - 1. \end{aligned} \tag{5–8}$$

The last inequality holds true because $p'_i \leq z(\widehat{Q}^1 \cup (i,j)) = \sum_{k \in V} f'_k$.
Based on valid inequality 5–8 and the integrality of $z(\widehat{Q}^1 \cup (i,j))$, we have $z(\widehat{Q}^1 \cup (i,j)) \geq \lceil (z(\widehat{Q}^1) + 1)/2 \rceil = \lfloor z(\widehat{Q}^1)/2 \rfloor + 1$.
Otherwise, if $p'_j = 0$, then we can obtain a feasible solution on $\overline{G}$ by letting $f_k = f'_k,\ \forall k \in V,\ k \neq j$, and $f_j = 1$. Thus, we have

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas} = z(\widehat{Q}^1 \cup (i,j)) + f_j \ \Rightarrow\ z(\widehat{Q}^1 \cup (i,j)) \geq z(\widehat{Q}^1) - 1.$$

Summarizing all cases above, the following inequality holds true:

$$\begin{aligned} z(\widehat{Q}^1 \cup (i,j)) &\geq \min\left\{ z(\widehat{Q}^1),\ z(\widehat{Q}^1) - 1,\ \left\lfloor z(\widehat{Q}^1)/2 \right\rfloor + 1 \right\} \\ &= \min\left\{ z(\widehat{Q}^1) - 1,\ \left\lfloor z(\widehat{Q}^1)/2 \right\rfloor + 1 \right\}. \end{aligned}$$

This completes the proof. □

We propose a valid inequality based on Lemma 7 in the following theorem.

Theorem 5.2. *For any subset $\widehat{Q}^1 \subseteq W$, and for any $\widehat{Q}^c \subseteq \widehat{Q}^0$, we have*

$$z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \left(\max\left\{ 1,\ \left\lceil z(\widehat{Q}^1)/2 \right\rceil - 1 \right\} \right) |\widehat{Q}^c|, \tag{5–9a}$$

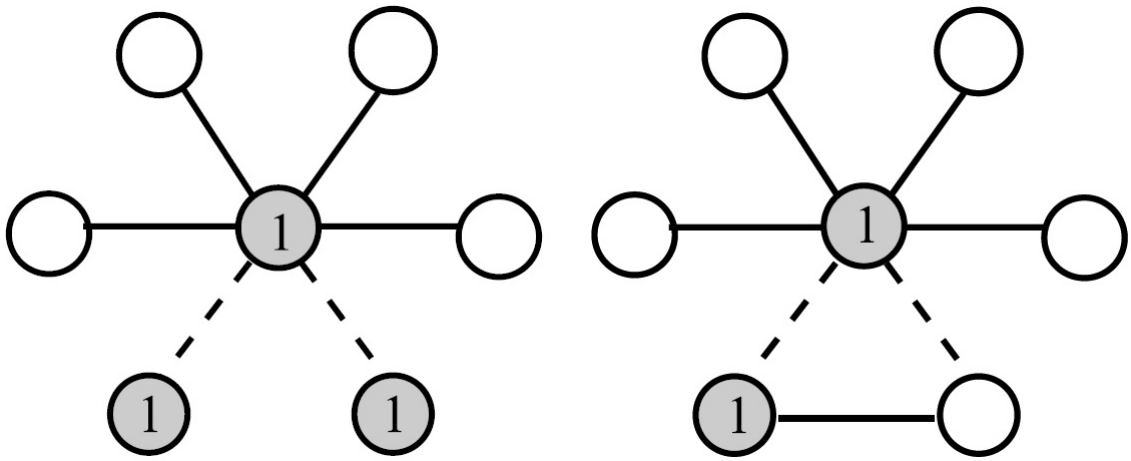
and therefore

$$\eta \geq z(\widehat{Q}^1) - \sum_{(i,j) \in \widehat{Q}^0} \left(\max\left\{ 1,\ \left\lceil z(\widehat{Q}^1)/2 \right\rceil - 1 \right\} \right) q_{ij} \tag{5–9b}$$

is valid to Formulation 5–2.

Proof. When $|\widehat{Q}^c| = 1$, Inequality 5–9a is true due to Lemma 7. Also, note that Inequality 5–9a must hold true if $|\widehat{Q}^c| \geq 3$ or $|\widehat{Q}^c| = 2$ with $z(\widehat{Q}^1)$ being odd, because the right-hand-side becomes less than or equal to 1 in those cases, and because $\eta \geq 1$ in any feasible solution.

Now, suppose that $|\widehat{Q}^c| = 2$ with $z(\widehat{Q}^1)$ being even. Note that $\lceil z(\widehat{Q}^1)/2 \rceil - 1 = 0$ when $z(\widehat{Q}^1) = 2$, and so $\max\{1, \lceil z(\widehat{Q}^1)/2 \rceil - 1\} = 1$. Hence, if $z(\widehat{Q}^1) = 2$, then Inequality 5–9a is implied by the bound $\eta \geq 1$ when $|\widehat{Q}^c| \geq 1$ in this case. Otherwise, if $z(\widehat{Q}^1)$ is an even number greater than 2, then the right-hand-side of Inequality 5–9a becomes 2 when $|\widehat{Q}^c| = 2$. Given that $z(\widehat{Q}^1) \geq 4$, it is impossible for $z(\widehat{Q}^1 \cup \widehat{Q}^c)$ to be 1 after adding the two edges in $\widehat{Q}^c$ (Figure 5-4).



A $z(\widehat{Q}^1) = 3$ B $z(\widehat{Q}^1) = 2$

Figure 5-4. Examples of possible $z(\widehat{Q}^1)$-values, given $z(\widehat{Q}^1 \cup \widehat{Q}^c) = 1$, $|\widehat{Q}^c| = 2$, and shadowed nodes having broadcast powers of 1. A) Case of $z(\widehat{Q}^1) = 3$. B) Case of $z(\widehat{Q}^1) = 2$.

To see this, note that an optimal BD value of 1 is possible only when one node is adjacent to all other nodes in the graph. By deleting two edges in $\widehat{Q}^c$, at most two nodes (say, j_1 and j_2) are now not adjacent to i. Then a feasible BD solution would exist with $f_i = f_{j_1} = f_{j_2} = 1$, contradicting the assumption of $z(\widehat{Q}^1) \geq 4$. This completes the proof. □

5.4.2 Cuts Generated by Using a Modified Feasible BD Solution

Now we propose an alternative way of computing valid α-coefficients in Cut 5–6. Recall that d_{ij} is the length of a shortest path between any node i and node j on G (with no edges added from W), where $d_{ij} = \infty$ if no path connects i and j in G. We first establish the following lemma.

Lemma 8. *Given an unweighted and undirected graph $G(V, E)$, and edges $\widehat{Q}^1$ constructed in the first stage, the following inequality*

$$z(\widehat{Q}^1 \cup (i,j)) \geq z(\widehat{Q}^1) - \lfloor d_{ij}/2 \rfloor \tag{5–10}$$

is valid for any $(i,j) \in \widehat{Q}^0$.

Proof. The lemma is trivially true if $d_{ij} = \infty$; hence, we focus on the case in which $d_{ij} < \infty$. We refer to the proof of Lemma 7 for the definitions of $f_k,\ p_k$ and $f'_k,\ p'_k$ for all $k \in V$. We next show how to recover a feasible BD solution (f, p) on $\overline{G}(V, E \cup \widehat{Q}^1)$ from an optimal BD solution (f', p') on $G'(V, E \cup \widehat{Q}^1 \cup (i,j))$.

In Cases 1 and 2a, as defined in the proof of Lemma 7, $z(\widehat{Q}^1 \cup (i,j)) = z(\widehat{Q}^1)$ by letting $f = f'$. Now consider Case 2b, for which we assume that node i and node j are covered by a center node v through a shortest path $\rho = v - \cdots - i - j$, and $p'_i = p'_j + 1$. We show how to rebuild a BD by modifying power assignments.

Denote $\overline{\rho}$ as the walk given by ρ, but where edge (i,j) is replaced by a shortest path from i to j in G. Note that $\overline{\rho}$ is given by four segments (some of which might be empty):

- Segment 1: A path from v to some node w
- Segment 2: A path from w to node i
- Segment 3: A path from node i back to w using the opposite route of Segment 2
- Segment 4: A path from w to j,

where the edges used in Segments 1, 2, and 4 are distinct. Let e_l denote the number of edges in Segment $l = 1, \ldots, 4$. Figure 5-5 illustrates a shortest path example and its

four segments. Observe that the shortest path from i to j in $\overline{G}$ uses the path given by Segments 3 and 4, and so $d_{ij} = e_3 + e_4$. After edge (i, j) is deleted, the potential change at node j is at least $e_2 + 1 - e_4$.

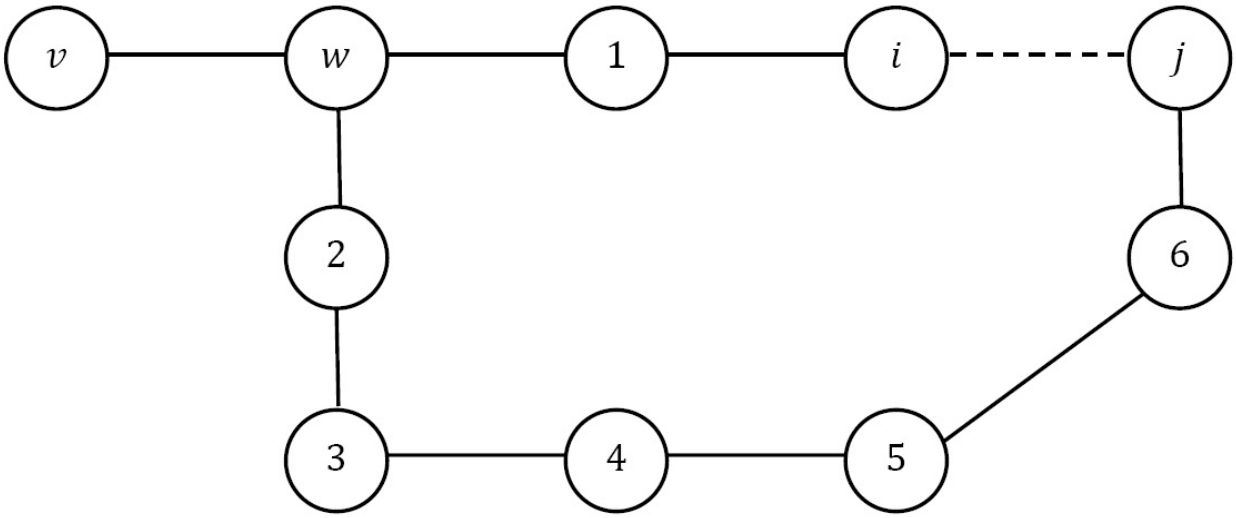


Figure 5-5. Illustration of the definition of the four segments, where Segment 1 is path v–w, Segment 2 is path w–1–i, Segment 3 is path i–1–w, and Segment 4 is path w–2–3–4–5–6–j (with $e_1 = 1$, $e_2 = e_3 = 2$, and $e_4 = 6$).

Let v^* be the terminal node that is $\lfloor d_{ij}/2 \rfloor$ edges away from node v on $\overline{\rho}$. Now suppose that the center node moves one edge at a time on $\overline{\rho}$. When the center moves on an edge in Segment 1, the potential at node j increases by 1. When the center moves on an edge in Segment 2, it is actually moving farther from node j, which results in a decrease of 1 in node j's potential. Moving the center on an edge in Segments 3 and 4 increase node j's potential by 1. Thus, the total change in node j's potential from moving the center by $\lfloor d_{ij}/2 \rfloor$ edges to node v^* on $\overline{\rho}$ is thus at least $\lfloor d_{ij}/2 \rfloor - e_2$.

Letting $f_k = f'_k$, $\forall k \in V - \{v, v^*\}$, $f_v = 0$, and $f_{v^*} = f'_v + \lfloor d_{ij}/2 \rfloor$, the potentials at nodes v and j are given as

$$\begin{aligned} p_v &\geq f_{v^*} - \lfloor d_{ij}/2 \rfloor = f'_v \text{ and} \\ p_j &\geq p'_j + (e_2 + 1 - e_4) + (\lfloor d_{ij}/2 \rfloor - e_2) + \lfloor d_{ij}/2 \rfloor \\ &= p'_j + 1 - e_4 + 2\lfloor d_{ij}/2 \rfloor \end{aligned}$$

$$\geq \quad p'_j + 1 - e_4 + (e_3 + e_4 - 1) \geq p'_j,$$

and the potentials at all other nodes either remain unchanged or are increased. Thus, f is a feasible BD solution on $\overline{G}(V, E \cup \widehat{Q}^1)$, with an objective value of

$$\begin{aligned} z(\widehat{Q}^1)^{feas} &= \sum_{k \in V \setminus \{v^*\}} f_k + f_{v^*} = \sum_{k \in V} f'_k - f'_v + (f'_v + \lfloor d_{ij}/2 \rfloor) \\ &= z(\widehat{Q}^1 \cup (i,j)) + \lfloor d_{ij}/2 \rfloor , \end{aligned}$$

which yields the following inequality

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas} = z(\widehat{Q}^1 \cup (i,j)) + \lfloor d_{ij}/2 \rfloor .$$

Thus, $z(\widehat{Q}^1 \cup (i,j)) \geq \min\{z(\widehat{Q}^1),\ z(\widehat{Q}^1) - \lfloor d_{ij}/2 \rfloor\} = z(\widehat{Q}^1) - \lfloor d_{ij}/2 \rfloor$, which completes the proof. □

Theorem 5.3. *For any subset $\widehat{Q}^1 \subseteq W$, and for any $\widehat{Q}^c \subseteq \widehat{Q}^0$, we have*

$$z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \sum_{(i,j) \in \widehat{Q}^c} \lfloor d_{ij}/2 \rfloor, \tag{5–11a}$$

and therefore

$$\eta \geq z(\widehat{Q}^1) - \sum_{(i,j) \in \widehat{Q}^0} \lfloor d_{ij}/2 \rfloor \, q_{ij} \tag{5–11b}$$

is valid to Formulation 5–2.

Proof. When $|\widehat{Q}^c| = 1$, Inequality 5–11a is true due to Lemma 8, and if $d_{ij} = \infty$ for some $(i,j) \in \widehat{Q}^c$, then Inequality 5–11a trivially holds.

When $|\widehat{Q}^c| \geq 2$ and $d_{ij} < \infty,\ \forall (i,j) \in \widehat{Q}^c$, we consider an optimal BD solution f' on $G'(V, E \cup \widehat{Q}^1 \cup \widehat{Q}^c)$ and its optimal objective value $z(\widehat{Q}^1 \cup \widehat{Q}^c)$, and show that there exists a solution f on $\overline{G}(V, E \cup \widehat{Q}^1)$ with objective $z(\widehat{Q}^1) \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j) \in \widehat{Q}^c} \lfloor d_{ij}/2 \rfloor$.

We begin by decomposing the graph into the balls formed by f', which we assume (without loss of generality) to be efficient. If ball $B(v, f'_v)$ (with $f'_v > 0$) does not contain any pair of nodes i and j such that $(i,j) \in \widehat{Q}^c$, then the solution given by f'_k for all k

belonging to $B(v, f'_v)$ remains feasible over this ball when edges $(i,j) \in \widehat{Q}^c$ are removed from the graph. If the ball contains exactly one such edge, then we can recover the potentials at all nodes in the associated ball $B(v, f'_v)$ by retaining the same f'-values if (i,j) falls under Case 2a of Lemma 7, or by setting $f_{v^*} = f'_v + \lfloor d_{ij}/2 \rfloor$ and $f_k = 0$ for all other nodes k in $B(v, f'_v)$ if (i,j) falls under Case 2b of Lemma 7.

Now consider a ball $B(v, f'_v)$ containing multiple Case 2b edges $(i_1, j_1), \ldots, (i_n, j_n) \in \widehat{Q}^c$, where all Case 2a edges can again be ignored. Suppose that we create a shortest path tree over this ball (using edges in $E \cup \widehat{Q}^1 \cup \widehat{Q}^c$) from node v to all other nodes. Edges $(i_1, j_1), \ldots, (i_n, j_n)$ are ordered such that $p'_{i_k} = p'_{j_k} + 1, \ \forall k = 1, \ldots, n$, and $p'_{i_k} \geq p'_{i_{k+1}}, \ \forall k = 1, \ldots, n-1$. Let v_1^* be the node that is $\lfloor d_{i_1 j_1}/2 \rfloor$ edges away from v, on the walk from v to j_1 that uses the path in the shortest path tree from v to j_1, but where edge (i_1, j_1) is replaced with the shortest path edges that connect i_1 and j_1 in G. Then, sequentially let v_m^* be the node that is $\lfloor d_{i_m j_m}/2 \rfloor$ edges away from v_{m-1}^* on the walk from v_{m-1}^* to j_m that uses the edges in the shortest path tree given above, but replaces all edges $(i_1, j_1), \ldots, (i_m, j_m)$ with their shortest paths in G (of lengths $d_{i_1 j_1}, \ldots, d_{i_m j_m}$, respectively), for all $m = 2, \ldots, n$.

We first delete (i_1, j_1), and move the center node from v to v_1^* by setting $f_{v_1^*} = f'_v + \lfloor d_{i_1 j_1}/2 \rfloor$ and $f_v = 0$. Following the same arguments in Lemma 8, the potentials at all nodes in $B(v, f'_v)$ are nondecreasing. We update the new ball as $B(v_1^*, f_{v_1^*})$, and note that it contains all nodes in $B(v, f'_v)$. Repeat the foregoing procedure for each (i_m, j_m) from $m = 2$ to n, in which we move the center node from v_{m-1}^* to v_m^*. We build a feasible BD solution as $f_{v_n^*} = f'_v + \sum_{m=1}^{n} \lfloor d_{i_m j_m}/2 \rfloor$ and $f_k = 0$ for all other nodes k in $B(v_n^*, f_{v_n^*})$, after deleting all edges $(i_m, j_m), \ m = 1, \ldots, n$.

Because each deleted edge (i,j) across all of the balls results in an objective increase of no more than $\lfloor d_{ij}/2 \rfloor$, the objective function value $z(\widehat{Q}^1)^{feas}$ corresponding to solution f satisfies

$$z(\widehat{Q}^1)^{feas} \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j) \in \widehat{Q}^c} \lfloor d_{ij}/2 \rfloor. \tag{5–12}$$

We thus conclude that $z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widehat{Q}^c} \lfloor d_{ij}/2 \rfloor$ (because $z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas}$) for any $\widehat{Q}^1 \subseteq W$, $\widehat{Q}^c \subseteq \widehat{Q}^0$, and thus 5–11b is a valid inequality to Formulation 5–2. □

Theorem 5.4. *Given an edge set $\widehat{Q}^1$, a valid inequality 5–6 is generated by letting*

$$\alpha_{ij}\left(\widehat{Q}^1\right) = \min\left\{\max\left\{1,\ \left\lceil z(\widehat{Q}^1)/2 \right\rceil - 1\right\},\ \lfloor d_{ij}/2 \rfloor\right\}, \tag{5–13}$$

for all $(i,j) \in \widehat{Q}^0$.

Proof. For any subset $\widehat{Q}^1 \subseteq W$, and for any $\widehat{Q}^c \subseteq \widehat{Q}^0$, we consider an optimal BD solution f' on $G'(V, E \cup \widehat{Q}^1 \cup \widehat{Q}^c)$ and its optimal objective value $z(\widehat{Q}^1 \cup \widehat{Q}^c)$. Our goal is to show that there exists a solution f on $\overline{G}(V, E \cup \widehat{Q}^1)$ with objective

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j)\in\widehat{Q}^c} \alpha_{ij}(\widehat{Q}^1), \tag{5–14}$$

given α-values in Theorem 5.4. First, when $|\widehat{Q}^c| = 1$, the result is straightforward due to Theorems 5.2 and 5.3.

When $|\widehat{Q}^c| \geq 2$ and $d_{ij} < \infty$, $\forall (i,j) \in \widehat{Q}^c$, we only need to focus on edges in $\widetilde{E} = \{$all Case 2b edges contained in $\widehat{Q}^c\}$. We partition $\widetilde{E}$ into two edge subsets $\widetilde{A}$ and $\widetilde{B}$, such that edge $(i,j) \in \widetilde{E}$ belongs to $\widetilde{A}$ if $\max\{1, \lceil z(\widehat{Q}^1)/2 \rceil - 1\} < \lfloor d_{ij}/2 \rfloor$, and belongs to $\widetilde{B}$ otherwise. We re-write Inequality 5–14 as

$$\begin{aligned} z(\widehat{Q}^1 \cup \widehat{Q}^c) &\geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widetilde{A}} \max\left\{1, \left\lceil \frac{z(\widehat{Q}^1)}{2} \right\rceil - 1\right\} - \sum_{(i,j)\in\widetilde{B}} \left\lfloor \frac{d_{ij}}{2} \right\rfloor - \sum_{(i,j)\in\widehat{Q}^c\setminus\widetilde{E}} \alpha_{ij}(\widehat{Q}^1) \\ &= z(\widehat{Q}^1) - \max\left\{1, \left\lfloor \frac{z(\widehat{Q}^1) - 1}{2} \right\rfloor\right\} |\widetilde{A}| - \sum_{(i,j)\in\widetilde{B}} \left\lfloor \frac{d_{ij}}{2} \right\rfloor - \sum_{(i,j)\in\widehat{Q}^c\setminus\widetilde{E}} \alpha_{ij}(\widehat{Q}^1) \end{aligned} \tag{5–15}$$

First, note that Inequality 5–15 must be true when $|\widetilde{A}| \geq 2$ by the same analysis in the proof of Theorem 5.2, and Inequality 5–15 must be true if $|\widetilde{A}| = 0$ by Theorem 5.3.

Thus, we study the case in which there is only one edge (i', j') in $\widetilde{A}$. Now let $f_{j'} = 1$ if $1 > \lfloor (z(\widehat{Q}^1) - 1)/2 \rfloor$, and $f_{j'} = p'_{i'} - 1$ otherwise. In the latter case, following the same argument in the proof of Lemma 7, the power increase is no more than

$\lfloor (z(\widehat{Q}^1) - 1)/2 \rfloor$ under the assumption that (i', j') is the only edge added. Hence, $f_{j'} = p'_{i'} - 1 \leq \lfloor (z(\widehat{Q}^1) - 1)/2 \rfloor$ must be true after more edges in $\widetilde{B}$ and $\widehat{Q}^c \setminus \widetilde{E}$ are added (because adding more edges potentially decreases BD cost $z(\widehat{Q}^1 \cup \widehat{Q}^c)$ and the potential $p'_{i'}$). We then implement the same procedures in the proof of Theorem 5.3 for all edges in $\widetilde{B}$ to maintain a feasible solution f revised from f'. (The center node is unchanged after the previous procedure for (i', j').) Finally, f has an objective value

$$z(\widehat{Q}^1)^{\textit{feas}} \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \max\left\{1, \left\lfloor \frac{z(\widehat{Q}^1) - 1}{2} \right\rfloor\right\} + \sum_{(i,j)\in\widetilde{B}} \left\lfloor \frac{d_{ij}}{2} \right\rfloor,$$

and Inequality 5–15 is true because $z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{\textit{feas}}$, and $\alpha_{ij}(\widehat{Q}^1) \geq 0$ for all other edges $(i, j) \in \widehat{Q}^c \setminus \widetilde{E}$. This completes the proof. □

5.5 BDND Inequalities for Weighted Graphs

To solve BDND on weighted graphs, we employ the same master problem BDND-M and subproblem BDND-S($\hat{q}$). However, no polynomial-time algorithm is yet known for computing an optimal BD on a weighted graph, and we solve the subproblem as a MIP at each iteration. Moreover, we must adjust our analysis for identifying valid inequality coefficients.

Given a first-stage solution $\hat{q}$, first, Lemma 7 and Theorem 5.2 hold on weighted graphs due to the assumption that $w_{ij} \geq 1,\ \forall (i, j) \in E \cup W$. We tighten the coefficients in Cut 5–9b by establishing the following results.

Lemma 9. *Given a weighted and undirected graph $G(V, E)$, and $\widehat{Q}^1 \subset W$ (with $\widehat{Q}^0 = W \setminus \widehat{Q}^1$), inequality*

$$z(\widehat{Q}^1 \cup (i, j)) \geq \min\left\{z(\widehat{Q}^1) - 1,\ \left\lceil (z(\widehat{Q}^1) + w_{ij})/2 \right\rceil\right\} \tag{5–16}$$

is valid for any $(i, j) \in \widehat{Q}^0$.

Proof. The proof is similar to the one for Lemma 7, with the modification of $p'_i = p'_j + w_{ij}$ in Case 2b. □

Theorem 5.5. *For any subset $\widehat{Q}^1 \subseteq W$, and for any $\widehat{Q}^c \subseteq \widehat{Q}^0$, we have*

$$z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widehat{Q}^c} \left(\max\left\{1,\ \left\lfloor (z(\widehat{Q}^1) - w_{ij})/2 \right\rfloor\right\}\right), \tag{5–17a}$$

and therefore

$$\eta \geq z(\widehat{Q}^1) - \sum_{(i,j)\in\widehat{Q}^0} \left(\max\left\{1,\ \left\lfloor (z(\widehat{Q}^1) - w_{ij})/2 \right\rfloor\right\}\right) q_{ij} \tag{5–17b}$$

is valid to Formulation 5–2.

Proof. Consider an optimal BD solution f' on $G'(V, E \cup \widehat{Q}^1 \cup \widehat{Q}^c)$ and its optimal objective value $z(\widehat{Q}^1 \cup \widehat{Q}^c)$. Let $\widetilde{E}$ be the set of all Case 2b edges contained in $\widehat{Q}^c$, and let $p'_i = p'_j + w_{ij},\ \forall (i,j) \in \widetilde{E}$. If $|\widetilde{E}| \leq 1$, then Constraints 5–17a (and 5–17b) follow by noticing that edges in $\widehat{Q}^c \setminus \{\widetilde{E}\}$ can be ignored, and by applying Lemma 9 if $|\widetilde{E}| = 1$. Hence, we focus on the case in which $|\widetilde{E}| \geq 2$.

We partition $\widetilde{E}$ into two subsets, denoted as E^1 and E^0, such that $p'_j > 0$ for all $(i,j) \in E^1$ and $p'_j = 0$ for all $(i,j) \in E^0$. After deleting all edges in $\widetilde{E}$, we construct f by letting $f_j = p'_j = p'_i - w_{ij},\ \forall j :\ (i,j) \in E^1$, and by letting $f_j = 1,\ \forall j :\ (i,j) \in E^0$. Also, let $f_v = f'_v$ at all other nodes v in V. By using the same analysis in the proof of Lemma 9, f is a feasible BD, and has an objective value of $z(\widehat{Q}^1)^{feas}$ such that

$$\begin{aligned} z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas} &= z(\widehat{Q}^1 \cup \widehat{Q}^c) + |E^0| + \sum_{(i,j)\in E^1} (p'_i - w_{ij}) \\ &\leq |E^0| + (1 + |E^1|) z(\widehat{Q}^1 \cup \widehat{Q}^c) - \sum_{(i,j)\in E^1} w_{ij}, \end{aligned}$$

because $z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq f'_{v_i} \geq p'_i,\ \forall i : (i,j) \in E^1$. This inequality yields

$$(1 + |E^1|) z(\widehat{Q}^1) \leq |E^0| + (1 + |E^1|) z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j)\in E^1} \left(z(\widehat{Q}^1) - w_{ij}\right)$$

by adding $|E^1|z(\widehat{Q}^1)$ on both sides of the inequality. Recalling our assumption of $|E^0| + |E^1| \geq 2$, we have:

$$\begin{aligned}
(1+|E^1|)z(\widehat{Q}^1 \cup \widehat{Q}^c) &\geq (1+|E^1|)z(\widehat{Q}^1) - |E^0| - \sum_{(i,j)\in E^1} \left(z(\widehat{Q}^1) - w_{ij}\right) \\
z(\widehat{Q}^1 \cup \widehat{Q}^c) &\geq z(\widehat{Q}^1) - \frac{|E^0|}{1+|E^1|} - \sum_{(i,j)\in E^1} \frac{z(\widehat{Q}^1) - w_{ij}}{1+|E^1|} && \text{(5–18a)} \\
&\geq z(\widehat{Q}^1) - |E^0| - \sum_{(i,j)\in E^1} \left\lfloor \frac{z(\widehat{Q}^1) - w_{ij}}{2} \right\rfloor && \text{(5–18b)} \\
&\geq z(\widehat{Q}^1) - \sum_{(i,j)\in \widetilde{E}} \left(\max\left\{1, \left\lfloor \frac{z(\widehat{Q}^1) - w_{ij}}{2} \right\rfloor \right\} \right) \\
&\geq z(\widehat{Q}^1) - \sum_{(i,j)\in \widehat{Q}^c} \left(\max\left\{1, \left\lfloor \frac{z(\widehat{Q}^1) - w_{ij}}{2} \right\rfloor \right\} \right).
\end{aligned}$$

We justify the step from Constraint 5–18a to Constraint 5–18b as follows. Observe that $|E^0| \geq |E^0|/(1+|E^1|)$, and $\lfloor (z(\widehat{Q}^1) - w_{ij})/2 \rfloor \geq (z(\widehat{Q}^1) - w_{ij})/(1+|E^1|)$ for all $(i,j) \in E^1$ when $|E^1| \geq 2$. Hence, Inequality 5–18b is true when $|E^1| \geq 2$ or when $|E^1| = 0$. When $|E^1| = 1$ (and so $|E^0| \geq 1$), then letting $E^1 = \{(u,v)\}$, we have that

$$\begin{aligned}
&\left(|E^0| + \left\lfloor \frac{z(\widehat{Q}^1) - w_{uv}}{2} \right\rfloor\right) - \left(\frac{|E^0|}{2} + \frac{z(\widehat{Q}^1) - w_{uv}}{2}\right) \\
= &\left(|E^0| - \frac{|E^0|}{2}\right) - \left(\frac{z(\widehat{Q}^1) - w_{uv}}{2} - \left\lfloor \frac{z(\widehat{Q}^1) - w_{uv}}{2} \right\rfloor\right) \\
\geq &\frac{1}{2}(1-1) = 0,
\end{aligned}$$

and so Inequality 5–18b holds true in this case as well. This completes the proof. □

Next, we derive another valid lower bound analogous to the one in Lemma 8.

Lemma 10. *Given a weighted and undirected graph $G(V,E)$, edge $(i,j) \in W$, and set $\widehat{Q}^1 \subseteq W \setminus (i,j)$, suppose that a center node v_j covers j through some shortest path over $\overline{G}(V, E \cup \widehat{Q}^1)$ consisting of nodes $P_{v_j j}$ in an optimal BD solution. Let s_{ij} be the shortest distance between nodes i and j in $\overline{G}(V, E \cup \widehat{Q}^1)$, for all $i, j \in V$, $i \neq j$. Define*

$C_{v_j j} = \{k \in P_{v_j j} - \{v_j, j\} : \ 2s_{v_j k} \geq d_{ij} - w_{ij}\}$. *The following inequality is valid:*

$$z(\widehat{Q}^1 \cup (i,j)) \geq z(\widehat{Q}^1) - \min\left\{\min_{k \in C_{v_j j}} \left\{s_{v_j k}\right\},\ (d_{ij} - w_{ij})\right\}, \tag{5–19}$$

where $\min_{k \in C_{v_j j}}\{s_{v_j k}\} = \infty$ *if* $C_{v_j j} = \emptyset$.

Proof. Recall that f_k and p_k are respectively the power assignment and the potential at node k in a feasible BD solution on $\overline{G}(V, E \cup \widehat{Q}^1)$, for all $k \in V$, and f' and p' correspond to an optimal BD solution on $G'(V, E \cup \widehat{Q}^1 \cup (i,j))$. Next, we show how to recover (f, p) from (f', p').

Let v_l be a node designated to cover node l in a BD solution, and let $P_{v_l l}$ be nodes in the corresponding covering path. First, if $v_i \neq v_j$ (Case 1 in the proof of Lemma 7), or if $v_i = v_j$ but nodes i and j are covered by two different paths (Case 2a in the proof of Lemma 7), then $z(\widehat{Q}^1 \cup (i,j)) = z(\widehat{Q}^1)$ by letting $f_k = f'_k,\ \forall k \in V$.

Now consider the case in which nodes i and j are covered by the same node $v = v_i = v_j$ through path $v - \cdots - i - j$ such that $p'_j = p'_i - w_{ij}$, i.e., Case 2b in the proof of Lemma 7. (Note that $d_{uv} > w_{uv},\ \forall (u,v) \in W$, by assumption; otherwise the deletion of edge (i,j) will not affect the objective value, and we can let $f_k = f'_k,\ \forall k \in V$.) After deleting edge (i,j), the potential at node j decreases by at most $p'_i - d_{ij} \leq p'_i - w_{ij}$. Letting $f_k = f'_k$ for all other nodes $k \in V,\ k \notin P_{vj}$, we modify the remaining f'-values as follows.

Note that we can always set $f_v = f' + (d_{ij} - w_{ij})$ if $d_{ij} < \infty$, such that p_j is also increased by at least $(d_{ij} - w_{ij})$ (if Segments 2 and 3 do not exist in the covering path, the potential at node j is increased by exactly $(d_{ij} - w_{ij})$), yielding a feasible BD on $\overline{G}(V, E \cup \widehat{Q}^1)$. Alternatively, consider a node $k \in C_{vj}$, as illustrated by Figure 5-6. (However, a shortest path between nodes i and j can share some common edges with the path from v to i, forming Segments 2 and 3 as we illustrate in Figure 5-5. We revise the definition of e_l as the sum of all edge weights associated with Segment $l = 1, \ldots, 4$.)

Now let $f_k = f'_v + s_{vk}$, $f_v = 0$, and $f_l = f'_l$ for all other nodes $l \in P_{vj}$, yielding $p_v \geq f_k - s_{vk} = f'_v$. Because $p_v \geq f'_v$, all nodes in $B(v, f_v)$, except perhaps for those

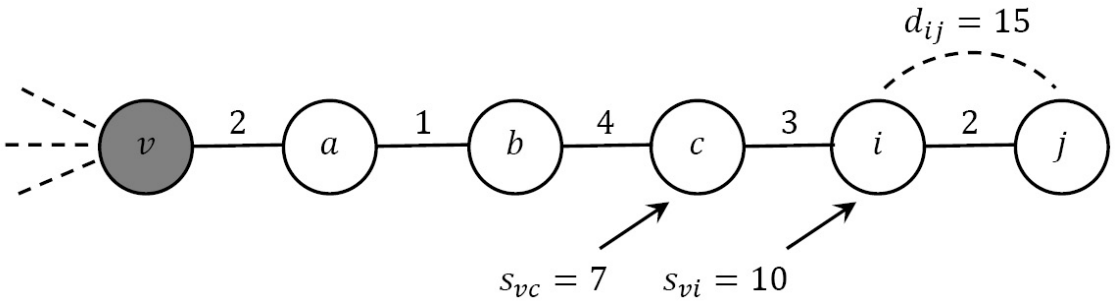


Figure 5-6. Example of $C_{vj} = \{c, i\}$ given $2s_{vi} > 2s_{vc} > d_{ij} - w_{ij} = 15 - 2 = 13$.

previously covered by a path that includes (i, j), are still covered in the revised solution. To verify that node k covers the remaining nodes in this ball, we need to verify that $p_j \geq p'_j$. After removing edge (i, j) from the graph, if we retain solution f', the potential at node j is at least $p'_j + (e_2 - e_4 + w_{ij})$. Because node k is on a shortest path from v to j using edges in $E \cup \widehat{Q}^1$, by replacing the broadcast power of f'_v at node v with $f'_v + s_{vk}$ at node k, the potential at node j increases by at least $s_{vk} + (s_{vk} - e_2)$. Hence,

$$p_j \geq p'_j + (e_2 - e_4 + w_{ij}) + (s_{vk} + (s_{vk} - e_2)) \geq p'_j - d_{ij} + w_{ij} + 2s_{vk} \geq p'_j,$$

because $d_{ij} = e_3 + e_4 \geq e_4$ and then $2s_{vk} \geq d_{ij} - w_{ij}$. Thus, all nodes in the ball are covered in the revised solution. The process of obtaining a revised solution is illustrated in Figure 5-7.

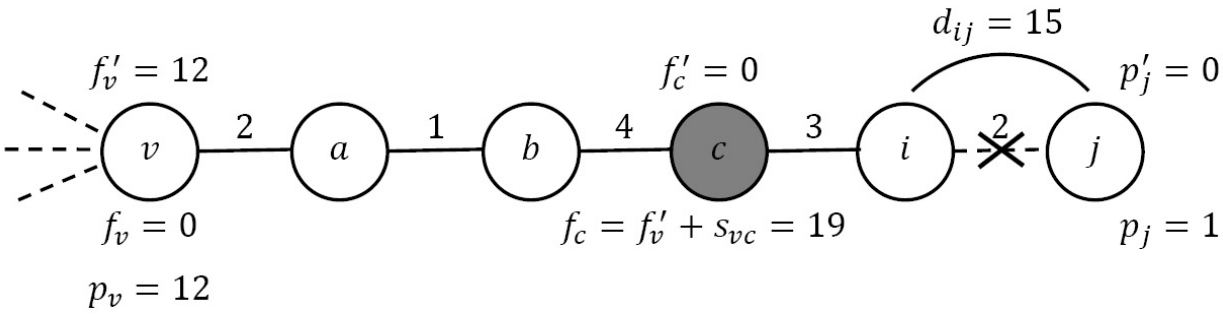


Figure 5-7. Illustration of a revised BD solution f by moving and increasing the power from v to $c \in C_{vj}$ when edge (i, j) is deleted.

The corresponding BD solution on $\overline{G}(V, E \cup \widehat{Q}^1)$ has an objective value

$$\begin{aligned} z(\widehat{Q}^1)^{feas} &= \sum_{i \in V,\ i \neq k} f_i + f_k = \sum_{i \in V} f'_i - f'_v + (f'_v + s_{vk}) \\ &= z(\widehat{Q}^1 \cup (i,j)) + s_{vk}. \end{aligned} \tag{5–20}$$

By examining the best of all possible feasible solutions shown in this proof, the following inequality

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas} = z(\widehat{Q}^1 \cup (i,j)) + \min \left\{ \min_{k \in C_{vj}} \{s_{vk}\},\ (d_{ij} - w_{ij}) \right\} \tag{5–21}$$

is valid for any $(i,j) \in \widehat{Q}^0$. This completes the proof. □

According to Lemma 10, we prescribe a class of valid inequalities 5–6 for the weighted BDND.

Theorem 5.6. *For any subset $\widehat{Q}^1 \subset W$, nonempty subset $\widehat{Q}^c \subseteq \widehat{Q}^0$, and $(i^*, j^*) \in \widehat{Q}^c$, we have*

$$z(\widehat{Q}^1 \cup \widehat{Q}^c) \geq z(\widehat{Q}^1) - \min \left\{ \min_{k \in C_{v_{j^*} j^*}} \left\{ s_{v_{j^*} k} \right\}, (d_{i^* j^*} - w_{i^* j^*}) \right\} - \sum_{(i,j) \in \widehat{Q}^c - \{(i^*, j^*)\}} (d_{ij} - w_{ij}), \tag{5–22a}$$

where s_{ij} is the shortest distance between any pair of nodes i and j on $\overline{G}(V, E \cup \widehat{Q}^1)$. Therefore,

$$\eta \geq z(\widehat{Q}^1) - \min \left\{ \min_{k \in C_{v_{j^*} j^*}} \left\{ s_{v_{j^*} k} \right\}, (d_{i^* j^*} - w_{i^* j^*}) \right\} q_{i^* j^*} - \sum_{(i,j) \in \widehat{Q}^0 - \{(i^*, j^*)\}} (d_{ij} - w_{ij}) q_{ij} \tag{5–22b}$$

is valid to Formulation 5–2 for any edge (i^*, j^*) in $\widehat{Q}^0$.

Proof. When $|\widehat{Q}^c| = 1$, Inequality 5–22a is true due to Lemma 10, regardless of whether $\widehat{Q}^c = \{(i^*, j^*)\}$. Hence, we examine the case of $|\widehat{Q}^c| \geq 2$. Consider an optimal BD solution f' on $G'(V, E \cup \widehat{Q}^1 \cup \widehat{Q}^c)$ and its optimal objective value $z(\widehat{Q}^1 \cup \widehat{Q}^c)$. We show

that there exists a solution f on $\overline{G}(V, E \cup \widehat{Q}^1)$ whose objective

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \min \left\{ \min_{k \in C_{v_{j^*} j^*}} \left\{ s_{v_{j^*} k} \right\}, (d_{i^* j^*} - w_{i^* j^*}) \right\} + \sum_{(i,j) \in \widehat{Q}^c - \{(i^*, j^*)\}} (d_{ij} - w_{ij}), \tag{5–23}$$

for any edge (i^*, j^*) in $\widehat{Q}^c$.

Recall that $\widetilde{E}$ represents the edge subset that consists of all Case 2b edges in $\widehat{Q}^c$. If $\widetilde{E} = \emptyset$, then $z(\widehat{Q}^1) = z(\widehat{Q}^1 \cup \widehat{Q}^c)$ by letting $f_k = f'_k$, $\forall k \in V$, and Inequality 5–23 clearly holds true.

Otherwise, if $\widetilde{E} \neq \emptyset$, we decompose the graph into the balls formed by f', and recover a solution such that every node in ball $B(v, f'_v)$ is covered by a node in the same ball, for all $v \in V_{f'}$. (Note that f' needs not be efficient in the weighted case.) We create a shortest path tree T (using edges $E \cup \widehat{Q}^1 \cup \widehat{Q}^c$) over each ball $B(v, f'_v)$, $\forall v \in V_{f'}$, from node v to all other nodes, and analyze the following two cases. First, we consider the case in which $(i^*, j^*) \notin \widetilde{E}$. To recover node potentials after the deletion of every edge $(i, j) \in \widetilde{E}$, we then increase the modified power at node v_j by $(d_{ij} - w_{ij})$ for all $(i, j) \in \widetilde{E}$. (Note that we might increase the power at some node multiple times if it covers multiple Case 2b edges in its ball.) Letting $f_k = f'_k$ for all other nodes $k \in V$, solution f yields a feasible BD cost

$$\begin{aligned} z(\widehat{Q}^1)^{\textit{feas}} &= z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j) \in \widetilde{E}} (d_{ij} - w_{ij}) \\ &\leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j) \in \widehat{Q}^c - \{(i^* j^*)\}} (d_{ij} - w_{ij}) + \min \left\{ \min_{k \in C_{v_{j^*} j^*}} \left\{ s_{v_{j^*} k} \right\}, (d_{i^* j^*} - w_{i^* j^*}) \right\}, \end{aligned}$$

and so Theorem 5.6 holds in this case.

Now consider the case in which $(i^*, j^*) \in \widetilde{E}$. If $\min_{k \in C_{v_{j^*} j^*}} \{ s_{v_{j^*} k} \} \geq (d_{i^* j^*} - w_{i^* j^*})$, then the theorem holds by the same analysis when $(i^*, j^*) \notin \widetilde{E}$. Otherwise, suppose that ball $B(v, f'_v)$ contains a total of n Case 2b edges, denoted as $(i_1, j_1), \ldots, (i_n, j_n)$, where $(i^*, j^*) = (i_1, j_1)$ without loss of generality. Increase the power at node v by

setting $f_v'' = f_v' + \sum_{m=2}^{n} (d_{i_m j_m} - w_{i_m j_m})$. With this revised power, all nodes in $B(v, f_v')$ are now covered by node v, except for any node l such that the path from v to l in T uses edge (i^*, j^*). To guarantee that all such nodes are also covered, we select $k^* \in \operatorname{argmin}_{k \in C_{v_{j_1}}} \{s_{vk}\}$ on a shortest path from v to j_1 over edges in $E \cup \widehat{Q}^1$. Now, set $f_v = 0,\ f_{k^*} = f_v'' + s_{vk^*}$, and $f_k = 0$ for all other nodes k in the ball. Then, by the same argument presented in the proof of Lemma 10, f_{k^*} covers all nodes in $B(v, f_v')$. For all other balls $B(v, f_v')$, we again set f_v to $f_v' + \sum_{(i,j) \in B(v, f_v') \cap \widetilde{E}} (d_{ij} - w_{ij})$ as necessary for all Case 2b edges contained in $B(v, f_v')$. The foregoing procedure yields a feasible BD cost

$$z(\widehat{Q}^1)^{feas} \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + s_{vk^*} + \sum_{(i,j) \in \widehat{Q}^c - \{(i^* j^*)\}} (d_{ij} - w_{ij}).$$

Because $z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{feas}$, Inequality 5–22a holds in both of the cases discussed above, and this completes the proof. □

Note that given a first-stage decision $\hat{q}$, a series of inequalities 5–22b can be generated by picking different edges (i^*, j^*) in $\widehat{Q}^0$.

Theorem 5.7. *In valid inequalities 5–6 for solving the weighted BDND, we have that*

$$\alpha_{i^* j^*}(\widehat{Q}^1) = \min \left\{ \max \left\{ 1,\ \left\lfloor (z(\widehat{Q}^1) - w_{ij})/2 \right\rfloor \right\},\ \min \left\{ \min_{k \in C_{v_{j^* j^*}}} \left\{ s_{v_{j^*} k} \right\}, (d_{i^* j^*} - w_{i^* j^*}) \right\} \right\}, \tag{5–24}$$

for a designated edge $(i^*, j^*) \in \widehat{Q}^0$*, and*

$$\alpha_{ij}(\widehat{Q}^1) = \min \left\{ \max \left\{ 1,\ \left\lfloor (z(\widehat{Q}^1) - w_{ij})/2 \right\rfloor \right\},\ (d_{ij} - w_{ij}) \right\}, \tag{5–25}$$

for $(i, j) \in \widehat{Q}^0 - \{i^*, j^*\}$.

Proof. We again show that Inequality 5–14 holds for the weighted graphs, given α-values in Theorem 5.7. Similarly, when $|\widehat{Q}^c| = 1$, the result is true due to Theorems 5.5 and 5.6.

When $|\widehat{Q}^c| \geq 2$ and $d_{ij} < \infty,\ \forall (i, j) \in \widehat{Q}^c$, we again focus on all Case 2b edges in $\widetilde{E} \subseteq \widehat{Q}^c$, and partition the set into two edge subsets $\widetilde{A}$ and $\widetilde{B}$, such that $\alpha_{ij}(\widehat{Q}^1)$ takes the

value proposed by Theorem 5.5 for all $(i,j) \in \widetilde{A}$, and the value proposed by Theorems 5.6 for all $(i,j) \in \widetilde{B}$. Note that $\widetilde{A}$ consists of two subsets E^1 and E^0, defined as in the proof of Theorem 5.5. We first revise powers on nodes that are adjacent to edges in $\widetilde{A}$ by executing the same procedures in the proof of Theorem 5.5, and then revise powers on nodes that are adjacent to edges in $\widetilde{B}$ using the procedures in Theorem 5.6. (Note that all power modification procedures in Theorem 5.5 only increase f_j, but do not move the center node v in each ball $B(v, f'_v)$; this allows us to perform procedures in Theorem 5.6 on top of the revised solution obtained from Theorem 5.5.) By the same analysis in the proofs of both theorems, we obtain a feasible BD solution f having an objective value

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{\textit{feas}} = z(\widehat{Q}^1 \cup \widehat{Q}^c) + |E^0| + \sum_{(i,j)\in E^1} (p'_i - w_{ij}) + \sum_{(i,j)\in \widetilde{B}} (d_{ij} - w_{ij}).$$

(For notation convenience, we present $(d_{ij} - w_{ij})$ for all edges $(i,j) \in \widetilde{B}$. This term needs to be adjusted if the designated edge (i^*, j^*) is in $\widetilde{B}$. We refer readers to Theorem 5.6 for details.)

Now examine Inequalities 5–18a and 5–18b in the proof of Theorem 5.5. Both of the inequalities hold after we respectively add $-\sum_{(i,j)\in\widetilde{B}} (d_{ij} - w_{ij})/(1 + |E^1|)$ and $-\sum_{(i,j)\in\widetilde{B}} (d_{ij} - w_{ij})$ on the right-hand-sides. (Inequality 5–18b is true because $1/(1 + |E^1|) \leq 1, \forall |E^1| \geq 0$, $d_{ij} - w_{ij} > 0, \forall (i,j) \in \widetilde{B}$, and the same analysis for other aspects in the proof of Theorem 5.6.) By adding $-\alpha_{ij}(\widehat{Q}^1)$ for all $(i,j) \in \widehat{Q}^c \setminus \{\widetilde{E}\}$ on the right-hand-side, we preserve the validity of the inequality, and thus

$$z(\widehat{Q}^1) \leq z(\widehat{Q}^1)^{\textit{feas}} \leq z(\widehat{Q}^1 \cup \widehat{Q}^c) + \sum_{(i,j)\in\widehat{Q}^c} \alpha_{ij}(\widehat{Q}^1),$$

for the given α-values. This completes the proof. □

Remark 5.3. Note that the α-values in Theorem 5.7 are expensive to compute, because we need to solve problem BDND-S($\hat{q}$) as a MIP in the weighted case in order to

find $z(\widehat{Q}^1)$. Denote $\underline{z}(\widehat{Q})$ as a lower bound value of an optimal BD objective (e.g., obtained by solving the LP relaxation of BDND-S($\hat{q}$)). Alternative valid inequalities can be generated by replacing $z(\widehat{Q}^1)$ with $\underline{z}(\widehat{Q}^1)$ in Cut 5–6, and by using $\underline{z}(\widehat{Q}^1)$ to compute the corresponding $\alpha_{ij}(\widehat{Q}^1)$ in Theorem 5.7. These cuts are also valid because $\underline{z}(\widehat{Q}^1) \leq z(\widehat{Q}^1)$, but will not generally be sufficient to force the decomposition algorithm to converge.

5.6 Computational Results

In this section, we discuss computational results of solving BDND on randomly generated graph instances. We tested the MIP model and decomposition approach on 45 undirected graphs for both unweighted and weighted cases, comprising fifteen 15-, 30-, and 45-node instances. We further classify all instances into *sparse*, *medium*, and *dense* instances, referred to as "$|V|$-s-l," "$|V|$-m-l," and "$|V|$-d-l," respectively. Here, $|V|$ represents the node cardinality and $l = 1, \ldots, 5$ is the instance number of each graph type. We generate each instance as a connected undirected graph by implementing the following procedures. We first pre-assign a **degree range** for each type, denoting the minimum and maximum degrees of each node allowed in the graph generation process.

Table 5-1 provides the degree ranges we used and the range of the number of edges contained in each graph type over all $l = 1, \ldots, 5$. For instance, among all five 15-node sparse graph instances, we require that each node is connected to at least two, and at most five, other nodes, and the number of edges generated varies from 27 to 32. Our random graph generation procedure begins by initializing $E = \emptyset$, and then in a loop, randomly generates two nodes $i, j \in V$, where $(i, j) \notin E$, and the degrees of both i and j are less than the maximum degree. We add edge (i, j) to the graph, and repeat this procedure until the degrees of all nodes lie within the degree range. After this initial phase, if the graph is not yet connected, we randomly connect node pairs from two different components until a connected graph is obtained. For each edge $(i, j) \in E$ in a weighted instance, we also randomly generate an edge distance by uniformly selecting

an integer from $\{1, 2, 3, 4\}$. Let $W = \{(i, j) \mid i,\ j \in V,\ i \neq j\} \setminus E$. We generate a link construction cost $c_{ij},\ \forall (i, j) \in W$ from a uniform distribution over the interval [0.1, 0.3] for all unweighted graph instances, and over the interval [0.2, 0.8] for all weighted graph instances.

Table 5-1. Test instances.

$\|V\|$	Degree ranges			Range of $\|E\|$ for each density type		
	s	m	d	s	m	d
15	2–5	4–8	6–13	[27, 32]	[46, 52]	[67, 80]
30	2–6	5–10	7–15	[53, 58]	[87, 96]	[154, 158]
45	2–8	5–12	12–24	[109, 126]	[179, 190]	[398, 434]

We then solve BDND for both unweighted and weighted cases by using the BMIP formulation 5–1 and by our proposed decomposition approach. We employ Benders cuts 5–5 at early iterations of our decomposition approach, and later change to optimality cuts 5–6, whose coefficients are stated by Theorem 5.4 for the unweighted case and by Theorem 5.7 for the weighted case. All MIP models and decomposition algorithms were implemeted using CPLEX 11.0 via ILOG Concert Technology 2.5. We perform all computations on a Dell PowerEdge 2600 UNIX machine with two Pentium 4 3.2 GHz (1M Cache) processors, 6.0 GB memory, and Red Hat Version 5.0 installed. We report computational times in CPU seconds, and allow a one-hour (3600 seconds) time limit.

We first examine the computational efficacy of using different criteria to change from Benders cuts to optimality cuts. At the end of each Benders iteration, we compute the gap between the optimal objective values of the master problem and the current subproblem, and change to optimality cuts if either (i) the gap is less than a pre-given threshold ϵ or (ii) we have generated more than 50 Benders cuts. Letting $\epsilon = 30\%,\ 20\%$, and 10%, we report the results for solving BDND on the first instance of each graph type in Table 5-2. Columns **# Cuts** give the number of Benders cuts generated before the gap becomes less than the specified value of ϵ, and columns **CPU** state the CPU

seconds for solving BDND to optimality given the corresponding threshold ϵ. We report LIMIT on instances that could not be solved to optimality within the one-hour time limit.

Table 5-2. Number of cuts generated and CPU time required as the cut-switching threshold parameter takes values of $\epsilon = 30\%,\ 20\%$, and 10%.

		$\epsilon = 30\%$		$\epsilon = 20\%$		$\epsilon = 10\%$	
	Instance	# Cuts	CPU	# Cuts	CPU	# Cuts	CPU
Unweighted	15-s-1	12	21.14	17	16.73	26	19.62
	15-m-1	6	19.08	11	14.92	20	15.16
	15-d-1	4	2.49	5	1.34	7	1.72
	30-s-1	21	78.65	25	67.92	29	73.96
	30-m-1	15	62.72	19	59.93	25	57.25
	30-d-1	7	64.30	14	56.80	23	61.02
	45-s-1	27	379.65	32	342.85	39	344.60
	45-m-1	18	267.91	25	276.90	35	279.48
	45-d-1	10	130.99	17	114.81	26	125.82
Weighted	15-s-1	21	204.38	25	145.90	31	137.95
	15-m-1	14	72.56	17	49.72	24	52.88
	15-d-1	9	27.62	13	19.50	18	17.62
	30-s-1	32	496.03	42	537.24	> 50	541.59
	30-m-1	25	227.59	33	188.91	44	167.41
	30-d-1	19	84.42	26	69.36	37	78.24
	45-s-1	> 50	LIMIT	> 50	LIMIT	> 50	LIMIT
	45-m-1	> 50	1739.25	> 50	1416.93	> 50	1483.06
	45-d-1	35	376.96	47	334.25	> 50	295.78

In general, our approaches perform the best in terms of CPU times when $\epsilon = 20\%$. As predicted, more Benders cuts are needed to improve the gap from 20% to 10% than from 30% to 20%, verifying that the Benders approach tends to be more effective at early iterations. Also, for the same threshold value ϵ, we need to execute more Benders iterations before changing to optimality cuts if (i) the graph is sparser; (ii) the number of nodes in the graph is larger; or (iii) the graph is weighted. This behavior occurs because the LP dual relaxations are relatively weaker compared with the exact MIP formulations in these cases. The CPU times decrease as we decrease ϵ, but then increase as we continue to decrease ϵ after a certain value (e.g., 20% for most instances). This is due to the observation that when ϵ is smaller, we potentially reduce the CPU time by using a

better incumbent $\hat{q}$ for computing the optimality cuts; however, the saved time is offset by the time required to close the gap using Benders cuts when the given ϵ is too small.

Next, we use $\epsilon = 20\%$ and no more than 50 Benders iterations to test our hybrid cutting-plane algorithms on all instances. Tables 5-3 and 5-4 compare CPU times between the MIP models and the decomposition approaches for solving BDND on unweighted and weighted 15-, 30-, and 45-node instances (stated in the "2-Stage" columns in these tables). For any instances that are not solved within the time limit, we report the optimality gaps (in percentage) instead of the CPU seconds.

Table 5-3. CPU times or solution gaps for solving the unweighted BDND as a monolithic MIP or a 2-stage decomposition model.

Instance	MIP	2-Stage	Instance	MIP	2-Stage	Instance	MIP	2-Stage
15-s-1	10.46%	16.73	30-s-1	33.23%	67.92	45-s-1	51.28%	342.85
15-s-2	7.31%	12.46	30-s-2	33.63%	79.07	45-s-2	52.36%	402.16
15-s-3	9.04%	17.82	30-s-3	35.01%	89.68	45-s-3	49.57%	385.72
15-s-4	9.52%	18.34	30-s-4	32.69%	71.36	45-s-4	51.41%	321.03
15-s-5	8.77%	11.99	30-s-5	33.17%	90.52	45-s-5	50.56%	370.31
15-m-1	1534.05	14.92	30-m-1	24.32%	59.93	45-m-1	43.26%	276.90
15-m-2	1331.24	10.21	30-m-2	24.09%	62.27	45-m-2	44.71%	195.73
15-m-3	1269.41	16.71	30-m-3	24.72%	55.12	45-m-3	43.66%	178.90
15-m-4	1482.73	13.57	30-m-4	23.86%	54.39	45-m-4	45.92%	205.28
15-m-5	1400.96	10.22	30-m-5	24.15%	64.26	45-m-5	41.84%	184.25
15-d-1	303.97	1.34	30-d-1	14.38%	56.80	45-d-1	29.35%	114.81
15-d-2	386.10	1.00	30-d-2	7.19%	49.26	45-d-2	27.52%	123.55
15-d-3	345.02	1.25	30-d-3	12.97%	42.11	45-d-3	27.76%	97.46
15-d-4	372.86	0.94	30-d-4	10.62%	33.06	45-d-4	26.80%	128.72
15-d-5	329.57	1.01	30-d-5	10.85%	37.18	45-d-5	28.59%	121.44

First, note that the decomposition approach performs much better than the MIP model, especially for unweighted cases. Only the 15-m and 15-d unweighted BDND instances can be solved within an hour using the MIP models, while the decomposition approach solves all test instances within the time limit except for the five 45-s weighted ones. On instances where both algorithms terminate with an optimal solution within an hour, the decomposition approach required less than 1% of the MIP computational time on unweighted 15-m instances, and less than 0.33% of the MIP computational

Table 5-4. CPU times or solution gaps for solving the weighted BDND as a monolithic MIP or a 2-stage decomposition model.

Instance	MIP	2-Stage	Instance	MIP	2-Stage	Instance	MIP	2-Stage
15-s-1	46.13%	145.90	30-s-1	56.61%	537.24	45-s-1	64.39%	23.42%
15-s-2	42.68%	145.23	30-s-2	53.53%	492.87	45-s-2	64.28%	27.87%
15-s-3	44.22%	173.72	30-s-3	52.79%	571.82	45-s-3	60.51%	20.44%
15-s-4	43.04%	139.98	30-s-4	53.41%	605.21	45-s-4	66.74%	15.96%
15-s-5	42.93%	146.31	30-s-5	54.60%	624.92	45-s-5	63.55%	22.76%
15-m-1	30.88%	49.72	30-m-1	51.89%	188.91	45-m-1	60.95%	1416.93
15-m-2	29.87%	56.32	30-m-2	53.63%	123.74	45-m-2	58.26%	1927.19
15-m-3	29.06%	47.48	30-m-3	48.22%	193.62	45-m-3	57.03%	1409.61
15-m-4	30.11%	59.01	30-m-4	49.74%	146.97	45-m-4	56.12%	2127.68
15-m-5	31.84%	47.69	30-m-5	46.53%	158.03	45-m-5	58.94%	1596.41
15-d-1	4.86%	19.50	30-d-1	34.87%	69.36	45-d-1	48.62%	334.25
15-d-2	13.80%	21.85	30-d-2	31.44%	62.88	45-d-2	49.30%	278.72
15-d-3	10.52%	18.71	30-d-3	28.65%	79.06	45-d-3	47.36%	341.89
15-d-4	7.94%	19.27	30-d-4	32.48%	63.74	45-d-4	48.68%	332.95
15-d-5	10.45%	21.48	30-d-5	30.06%	78.15	45-d-5	45.25%	347.68

time on unweighted 15-d instances. Second, all weighted instances require more CPU times than their unweighted counterparts as predicted, because we need to solve MIP subproblems for computing the optimality cut coefficients. Third, the CPU times are longer when the graph is sparser. Note that on denser graphs, both $rad(G)$ and $|W|$ are smaller for the same node-size graph G, thus decreasing the size of the corresponding MIP formulations and reducing the effort required to solve BD subproblems.

Finally, as discussed in Remark 5.1, we can also employ the CCB strategy of Saharidis et al. (2010b) to potentially reduce the CPU time required to generate classical (LP-based) Benders cuts, before our algorithm switches to the integer-programming-based cuts that are required to optimally solve BDND. We investigate the effectiveness of this strategy on our early Benders-cut phase with $\epsilon = 20\%$. We use a 0.1- and a 0.2-CCB strategy, and vary the maximum number of cuts allowed in a bundle to be 15 and 10. Table 5-5 reports CPU time required to solve the first instance of each medium-density test set using CCB generation, along with the relative CPU time reduction (in percentage) compared to the 2-stage results given in Tables 5-3 and 5-4.

Note that instances prefixed with “u-” represent unweighted instances, and those prefixed with “w-” represent weighted instances.

Table 5-5. CPU seconds comparison between classical Benders and varied CCB generation settings for solving BDND with $\epsilon = 20\%$.

Instance	2-Stage	T = 0.1, MaxCut = 15		T = 0.1, MaxCut = 10	
		CCB Time	Decrease	CCB Time	Decrease
u-15-m-1	14.92	14.25	4.70%	14.78	0.95%
u-30-m-1	59.93	57.62	4.01%	57.83	3.63%
u-45-m-1	276.90	275.11	0.65%	273.92	1.09%
w-15-m-1	49.72	48.04	3.50%	47.73	4.17%
w-30-m-1	188.91	190.15	-0.65%	192.32	-1.77%
w-45-m-1	1416.93	1380.91	2.61%	1405.21	0.83%
Instance	2-Stage	T = 0.2, MaxCut = 15		T = 0.2, MaxCut = 10	
		CCB Time	Decrease	CCB Time	Decrease
u-15-m-1	14.92	14.56	2.47%	15.02	-0.67%
u-30-m-1	59.93	57.54	4.15%	56.74	5.62%
u-45-m-1	276.90	271.68	1.92%	279.80	-1.04%
w-15-m-1	49.72	49.01	1.45%	47.39	4.92%
w-30-m-1	188.91	185.35	1.92%	187.44	0.78%
w-45-m-1	1416.93	1409.26	0.54%	1411.19	0.41%

The CCB generation procedure indeed decreases the CPU time for solving most instances, although a few instances report some increase in computational times (i.e., negative percentages in the “Decrease” columns). However, these savings are very modest, due primarily to the facts that (i) the Benders cuts we generated are often not low-intensity cuts (each cut density is normally between 30%–40%), and (ii) the LP-based cut generation phase is very brief relative to the overall decomposition procedure for this algorithm. For instance, the 2-Stage CPU time for u-15-m-1 was 14.92s overall, with only 1.69s being consumed in the initial LP-based Benders phase.

CHAPTER 6
CONCLUSIONS

In this chapter, we comment on the research tasks that we completed for the problems we have discussed in Chapter 2 through Chapter 5. We review the methodologies and techniques employed, and suggest future research directions.

In Chapter 2, we considered a two-stage stochastic integer programming formulation for the stochastic task insurance problem (STIP). We showed that the problem is naturally decomposable for convex penalty function problems, and becomes far more difficult when the penalty is general nonconvex. Our approach was to employ the Reformulation-Linearization Technique to make STIP amenable to Benders decomposition, when the penalty functions are piecewise-linear and nonconvex. Rather than explicitly computing the duals values of the resulting formulation, we proposed an algorithm that quickly recovers all coefficients of Benders cuts based on the solution of a single critical path problem. We examined alternative dual optimal solutions to the dual problem, which yield alternative cuts, and expanded our decomposition technique to handle general lower semi-continuous penalty functions. We also cast STIP as a chance-constrained optimization problem, and provided a cutting-plane algorithm for its solution. Future research may focus on investigating sophisticated upper bounding algorithms for STIP.

In Chapter 3, we developed polynomial-time dynamic programming (DP) algorithms for solving MaxNum and MinMaxC on trees and series-parallel graphs (SPGs). We considered two network connectivity metrics: maximizing the remaining number of components, and minimizing the largest component size. We also discussed the solution of these problems on k-hole-graphs, and on trees for a variant that involves general node deletion costs and general node weights.

Our approaches can readily be extended for solving the edge-deletion version of both MaxNum and MinMaxC on trees and SPGs by making the following modifications.

For each edge $(i,j) \in \mathcal{E}$, create a node v_{ij} and two undirected edges (i, v_{ij}) and (v_{ij}, j). This transformation retains the special structures of G (i.e., trees or SPGs), although it will polynomially increase the graph size. (Assuming that the original graph G has n nodes and m edges, the transformed graph has $(n + m)$ nodes and $2m$ edges.) We mark node v_{ij} as "active" for all $(i,j) \in \mathcal{E}$, and execute the DP recursions on the transformed graph, except that we only permit deletion of active nodes, and active nodes do not contribute to the component cardinality when solving MinMaxC instances. The complexity of our algorithms remains polynomial. We leave the study of edge-deletion problems on more general graphs for future research.

In Chapter 4, we formulated mixed-integer programming models for solving network disconnection problems on general graphs. In addition to the two network connectivity metrics that we considered in Chapter 3, we also considered a problem variant that maximizes the minimum graph reconnection cost. For the first two cases, we further studied bounds and valid inequalities computed based on optimal DP solutions on some k-hole subgraphs/partitions of the original graph. We reported computational results of directly solving the MIP models for all three metrics, and demonstrated the computational efficacy of using valid inequalities for solving MaxNum and MinMaxC.

A future research direction could examine methods that seek to improve the subgraph relaxations for MaxNum and MinMaxC, and investigating whether or not any tight relaxations for MaxMinLC can be utilized within a similar scheme. More computationally intensive methods can also be applied to improve the valid inequality generation scheme that we propose for this problem. One idea may utilize several copies of Inequalities 4–14 and 4–16 corresponding to different partitions of the input graph. Another may dynamically update these partitions within the branch-and-bound tree by re-solving DPs based on x-variable values that have been fixed in the branch-and-bound process. The inequalities would hence become locally valid rather than globally valid, and may improve the quality of the relaxation, leading to

quicker termination and more effective fathoming rules. Finally, while the emphasis of our research has been on exact methods, the bounding mechanisms here naturally lend themselves to heuristic approaches. The difficulty of optimally solving these problems demonstrates that heuristics will be necessary on large-scale problem instances.

Finally, in Chapter 5, we analyzed a broadcast domination network design problem. We showed that this problem is $\mathcal{NP}$-hard in the strong sense, and formulated it using a MIP model. Furthermore, we proposed a decomposition strategy, and generated valid inequalities based on optimal broadcast domination solutions for both unweighted and weighted graphs. Future research tasks include exploring the algorithmic complexity of solving BD on weighted graphs. If the weighted BD is polynomially solvable, our given decomposition algorithm for solving the weighted BDND problem would benefit substantially due to the increased efficiency of solving the subproblems.

APPENDIX A
EXPECTATION-BASED SAMPLE AVERAGE APPROXIMATION

The SAA is an approach for solving stochastic optimization problems by using Monte Carlo simulation. A set of N sample scenarios $\omega^1, \ldots, \omega^N$ is generated from Ω according to its probability distribution. We then solve a deterministic optimization problem specified by scenarios $\omega^1, \ldots, \omega^N$. We approximate the expected second-stage recourse costs by the sample average function $\frac{1}{N}\sum_{n=1}^{N} F(x, \xi(\omega^n))$, and the overall optimal objective value as:

$$f_N = \min_{x \in X} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij} + \frac{1}{N} \sum_{n=1}^{N} F(x, \xi(\omega^n)). \tag{A–1}$$

Now suppose that we generate M independent sets of samples, each of size N, and solve Formulation A–1 independently for each set of samples. We denote f_N^m and $\hat{x}_N^m$ as the optimal objective value and solution to Formulation A–1, respectively, for each set of samples, $m = 1, \ldots, M$.

The average of the M optimal objective values, $\bar{f}_{N,M} = \sum_{m=1}^{M} f_N^m / M$ provides a statistical estimate for the lower bound on the optimal objective function value.

Next, we pick any feasible first-stage solution from among optimal solutions $\hat{x} = \hat{x}_N^m$ to Formulation A–1, for some $m \in \{1, \ldots, M\}$. By fixing $x = \hat{x}$ in the second stage, we estimate the optimal objective value using a reference sample of size N'. We then compute an upper bound on the optimal objective function value as

$$\hat{f}_{N'}(\hat{x}) = \sum_{(i,j) \in \mathcal{A}} c_{ij} \hat{x}_{ij} + \frac{1}{N'} \sum_{n=1}^{N'} F(\hat{x}, \xi(\omega^n)). \tag{A–2}$$

Since computing $\hat{f}_{N'}(\hat{x})$ requires only the solution of N' subproblems, we can generate the reference sample with size N' much larger than N, independent of the samples used in Formulation A–1. To obtain the best upper bound, we thus select

$$x^* \in \arg\min_{\hat{x} \in \{\hat{x}_N^1, \ldots, \hat{x}_N^M\}} \{\hat{f}_{N'}(\hat{x})\}, \tag{A–3}$$

which yields an absolute optimality gap $\hat{f}_{N'}(x^*) - \bar{f}_{N,M}$.

APPENDIX B
AN $O(N^5)$ ALGORITHM FOR SOLVING MINMAXC ON TREES

It is again necessary to separately derive recursions based on whether or not a deletion takes place at node i (i.e., $o_i = 0$ or $o_i > 0$). For the case of $o_i = 0$, we have

$$f_i(0, m_i) = \min \quad \sum_{v \in S_i} f_v(o_v, m_v) + 1 \tag{B–1}$$

$$\text{s.t.} \quad m_i = \max_{v \in S_i} \{o_v, m_v\}, \tag{B–2}$$

where Eq. B–1 accounts for deletions in the subtrees T_v, for $v \in S_i$, plus the deletion at node i. Constraint B–2 updates m_i as the maximum size of all components and open sets at all subtrees of T_i.

The recursion for updating elements for which $o_i > 0$ is given by

$$f_i(o_i, m_i) = \min \quad \sum_{v \in S_i} f_v(o_v, m_v) \tag{B–3}$$

$$\text{s.t.} \quad o_i = \sum_{v \in S_i} o_v + 1 \tag{B–4}$$

$$m_i = \max_{v \in S_i} \{m_v\}. \tag{B–5}$$

Objective B–3 is the same as Eq. B–1, but omits a deletion at node i. Constraint B–4 updates o_i as the sum of all child open set sizes in addition to the one at node i (noting that O_i will include i and all open sets $O_v,\ v \in S_i$). Constraint B–5 updates m_i as the maximum size of all components existing at all subtrees. We again set $f_i(o_i, m_i) = \infty$ if it is infeasible or requires more than B deletions.

An overview of the DP solution scheme for solving MinMaxC on trees is identical to the algorithm given in Section 3.2.1 for MaxNum, with the following exceptions. In Step 0, we initialize $f_l(1, 0) = 0$ at every leaf node l, and set all the other f_l-values to ∞. In Step 2, we update entries $f_i(o_i, m_i)$ at T_i according to the above recursion. Also, in Step 3, we seek among all finite values of $f_r(o_r, m_r)$ that has the least value of $\max\{m_r, o_r\}$, which corresponds to the minimum objective function value in the MinMaxC case. We also update the H-functions as follows.

Step 2a (Initializing $H^1(i)$). We update the values of entries $h_i^1(o_i^1, m_i^1)$ according to whether $o_i^1 = 0$ or $o_i^1 > 0$. The respective recursive equations are

$$h_i^1(0, m_i^1) = \min\left\{f_{i_1}(o_{i_1}, m_{i_1}) + 1 : m_i^1 = \max\{m_{i_1}, o_{i_1}\}\right\}, \tag{B–6}$$

$$h_i^1(o_i^1, m_i^1) = f_{i_1}(o_i^1 - 1, m_i^1). \tag{B–7}$$

Equation B–6 describes the case where we delete node i, and $o_i^1 = 0$. The objective accounts for the number of nodes deleted at T_{i_1}, plus the deletion at node i. Observe that if $o_{i_1} > 0$, then the open set (if one exists) at node i_1 becomes one component in subtree T_i. Hence, we set m_i^1 as $\max\{m_{i_1},\ o_{i_1}\}$. In the case of $o_i^1 > 0$, Eq. B–7 notes that the open set size o_i^1 increases by 1 from o_{i_1}, and the largest component size stays unchanged.

Step 2b (Updating $H^s(i),\ \forall\ s = 2, \ldots, |S_i|$).

$$\begin{aligned} h_i^s(0, m_i^s) = \min \quad & h_i^{s-1}(0, m_i^{s-1}) + f_{i_s}(o_{i_s}, m_{i_s}) \\ \text{s.t.} \quad & m_i^s = \max\{m_i^{s-1}, o_{i_s}, m_{i_s}\} \end{aligned} \tag{B–8}$$

$$\begin{aligned} h_i^s(o_i^s, m_i^s) = \min \quad & h_i^{s-1}(o_i^{s-1}, m_i^{s-1}) + f_{i_s}(o_{i_s}, m_{i_s}) \\ \text{s.t.} \quad & o_i^s = o_i^{s-1} + o_{i_s} \\ & o_i^{s-1} > 0 \\ & m_i^s = \max\{m_i^{s-1}, m_{i_s}\}, \end{aligned} \tag{B–9}$$

We compute $h_i^s(0, m_i^s)$ in Eq. B–8 by merging $h_i^{s-1}(0, m_i^{s-1})$ with solutions to the subtree T_{i_s}. This recursion computes m_i^s as the maximum of m_i^{s-1}, o_{i_s}, and m_{i_s}, noting that a new component of size o_{i_s} (if $o_{i_s} > 0$) appears in the subtree T_i when i is deleted. Equation B–9 handles the case of $o_i^s > 0$, for which we generate $h_i^s(o_i^s, m_i^s)$ by merging $h_i^{s-1}(o_i^{s-1}, m_i^{s-1})$ having $o_i^{s-1} > 0$ with $f_{i_s}(o_{i_s}, m_{i_s})$. The objective minimizes the sum of deletions accumulated in the first $s - 1$ children of i, plus those in a deletion on T_{i_s}.

We compute the new open set size by combining o_i^{s-1} at node i with o_{i_s} at node i_s, and compute the largest non-open component size as the maximum of m_i^{s-1} and m_{i_s}.

APPENDIX C
AN $O(N^7)$ ALGORITHM FOR SOLVING MINMAXC ON SPGS

Recall that o_i is the open set size at node i. In particular, for an SPG $G(s,t)$ in which s and t are connected (i.e., $c_{st}=1$), we have that $O_s = O_t$ (and $o_s = o_t$). Given an SPG $G(s,t)$, we define a function $f_G(o_s, o_t, c_{st}, m_G)$ as the fewest number of deletions in G to achieve state variable values o_s, o_t, and c_{st}, such that the largest component size, excluding O_s and O_t, equals m_G. We give the initialization of $f_G(o_s, o_t, c_{st}, m_G)$ for $G = K_2$ as

$$f_{K_2}(2,2,1,0)=0,\ f_{K_2}(0,1,0,0)=1,\ f_{K_2}(1,0,0,0)=1,\ f_{K_2}(0,0,0,0)=2. \tag{C–1}$$

C.1 The Series Operation

When $G = S(G_1, G_2)$, we provide the following recursions to update f_G-elements. The first three recursions take the minimum of two optimization subproblems: One in which node $t_1 = s_2$, common to G_1 and G_2 is deleted, and the other in which it is not deleted. Equations C–2 through C–4 describe the cases of both terminals being deleted, only source s being deleted, and only sink t being deleted, respectively.

$$f_G(0,0,0,m_G) \;=\; \text{min:} \begin{cases} \min & f_{G_1}(0,0,0,m_{G_1}) + f_{G_2}(0,0,0,m_{G_2}) - 1 \\ \text{s.t.} & m_G = \max\{m_{G_1}, m_{G_2}\}; \\ \min & f_{G_1}(0,o_{t_1},0,m_{G_1}) + f_{G_2}(o_{s_2},0,0,m_{G_2}) \\ \text{s.t.} & m_G = \max\{m_{G_1}, m_{G_2}, o_{t_1}+o_{s_2}-1\} \\ & o_{t_1}>0,\ o_{s_2}>0 \end{cases} \tag{C–2}$$

$$f_G(0,o_t,0,m_G) \;=\; \min_{o_t>0}: \begin{cases} \min & f_{G_1}(0,0,0,m_{G_1}) + f_{G_2}(0,o_{t_2},0,m_{G_2}) - 1 \\ \text{s.t.} & o_t = o_{t_2},\ m_G = \max\{m_{G_1}, m_{G_2}\}; \\ \min & f_{G_1}(0,o_{t_1},0,m_{G_1}) + f_{G_2}(o_{s_2},o_{t_2},c_{s_2t_2},m_{G_2}) \\ \text{s.t.} & o_t = o_{t_2} + (o_{t_1}-1)c_{s_2t_2} \\ & m_G = \max\{m_{G_1}, m_{G_2}, (o_{t_1}+o_{s_2}-1)(1-c_{s_2t_2})\} \\ & o_{t_1}>0,\ o_{s_2}>0,\ o_{t_2}>0 \end{cases} \tag{C–3}$$

$$f_G(o_s, 0, 0, m_G) = \min_{o_s>0} : \begin{cases} \min & f_{G_1}(o_{s_1}, 0, 0, m_{G_1}) + f_{G_2}(0, 0, 0, m_{G_2}) - 1 \\ \text{s.t.} & o_s = o_{s_1},\ m_G = \max\{m_{G_1}, m_{G_2}\}; \\ \min & f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1t_1}, m_{G_1}) + f_{G_2}(o_{s_2}, 0, 0, m_{G_2}) \\ \text{s.t.} & o_s = o_{s_1} + (o_{s_2} - 1)c_{s_1t_1} \\ & m_G = \max\{m_{G_1}, m_{G_2}, (o_{t_1} + o_{s_2} - 1)(1 - c_{s_1t_1})\} \\ & o_{s_1} > 0,\ o_{t_1} > 0,\ o_{s_2} > 0 \end{cases} \tag{C–4}$$

For the case in which there exists a nonzero-size open set at the source $s = s_1$ and destination $t = t_2$ of the merged graph, we explore the case in which s_1 will be disconnected from t_2 separately from the case in which the two will be connected. First, if s_1 will be disconnected from t_2, we have

$$f_G(o_s, o_t, 0, m_G) = \min_{o_s,\ o_t>0} : \begin{cases} \min & f_{G_1}(o_{s_1}, 0, 0, m_{G_1}) + f_{G_2}(0, o_{t_2}, 0, m_{G_2}) - 1 \\ \text{s.t.} & o_s = o_{s_1},\ o_t = o_{t_2},\ m_G = \max\{m_{G_1}, m_{G_2}\}; \\ \min & f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1t_1}, m_{G_1}) + f_{G_2}(o_{s_2}, o_{t_2}, c_{s_2t_2}, m_{G_2}) \\ \text{s.t.} & c_{s_1t_1} + c_{s_2t_2} \le 1 \\ & o_s = o_{s_1} + (o_{s_2} - 1)c_{s_1t_1} \\ & o_t = o_{t_2} + (o_{t_1} - 1)c_{s_2t_2} \\ & m_G = \max\{m_{G_1}, m_{G_2}, (o_{t_1} + o_{s_2} - 1)(1 - c_{s_1t_1})(1 - c_{s_2t_2})\} \\ & o_{s_1} > 0,\ o_{t_1} > 0,\ o_{s_2} > 0,\ o_{t_2} > 0. \end{cases} \tag{C–5}$$

Constraint C–5 describes a case that can happen in multiple characteristics of G_1 and G_2. The first optimization subproblem in Formulation C–5 handles the case in which $t_1 = s_2$ is deleted, and thus $c_{s_1t_1} = c_{s_2t_2} = 0$. Note that the number of deletions at node $t_1 = s_2$ is counted in both f_{G_1} and f_{G_2}, so we subtract the objective value by one. We also do not count the generation of additional components after merging, and hence $o_s = o_{s_1},\ o_t = o_{t_2},\ m_G = \max\{m_{G_1}, m_{G_2}\}$. The second optimization subproblem considers the case in which neither t_1 nor s_2 is deleted. In this case, a path does not exist from $s = s_1$ to $t = t_2$ if and only if either the path from s_1 to t_1 or the path from s_2 to t_2 does not exist, and thus we constrain the sum of $c_{s_1t_1}$ and $c_{s_2t_2}$ to be no more than 1. The open set size at $s = s_1$ will now be o_{s_1} (o_{t_1}), plus $(o_{s_2} - 1)$ if $c_{s_1t_1} = 1$, i.e., we sum up

the open sets at t_1 and s_2, but subtract it by one due to the merging operation. Similarly, the open set size at $t = t_2$ will now be o_{t_2} (o_{s_2}), plus $(o_{t_1} - 1)$ if $c_{s_2 t_2} = 1$. The maximum component size in G is considered to be the maximum value of m_{G_1}, m_{G_2}, together with $(o_{t_1} + o_{s_2} - 1)$ as the new formed component size at node $t_1 = s_2$, if $c_{s_1 t_1} = c_{s_2 t_2} = 0$.

If s_1 will be connected to t_2, then

$$
\begin{aligned}
f_G(o_s, o_t, 1, m_G) = \min \quad & f_{G_1}(o_{s_1}, o_{t_1}, 1, m_{G_1}) + f_{G_2}(o_{s_2}, o_{t_2}, 1, m_{G_2}) \\
\text{s.t.} \quad & o_s = o_{s_1} + o_{s_2} - 1 \\
& o_t = o_{t_2} + o_{t_1} - 1 \\
& m_G = \max\{m_{G_1}, m_{G_2}\} \\
& o_{s_1} > 0,\ o_{t_1} > 0,\ o_{s_2} > 0,\ o_{t_2} > 0.
\end{aligned}
\tag{C–6}
$$

Here we require $c_{s_1 t_1} = c_{s_2 t_2} = 1$ to impose the case of $c_{st} = 1$. The open set size o_s is updated by using the summation of o_{s_1} and o_{s_2} minus one due to merging at $t_1 = s_2$. Also, we update the open set size o_t by summing up o_{t_2} and o_{t_1} minus one. Finally, recall that all open sets at s_1, t_1, s_2, and t_2 are merged into a new open set, which is not counted in as a new component, and so we update m_G as the maximum of m_{G_1} and m_{G_2}.

C.2 The Parallel Operation

We give the following recursions for updating $f_G(o_s, o_t, c_{st}, m_G)$ when $G = P(G_1, G_2)$. Note that in parallel synthesis operations, we must have the same deletion status at the pair of s_1 and s_2, and at the pair of t_1 and t_2 for the merge to be valid.

$$f_G(0, 0, 0, \max\{m_{G_1},\ m_{G_2}\}) = \min f_{G_1}(0, 0, 0, m_{G_1}) + f_{G_2}(0, 0, 0, m_{G_2}) - 2 \tag{C–7}$$

$$f_G(o_{s_1} + o_{s_2} - 1, 0, 0, \max\{m_{G_1},\ m_{G_2}\}) = \min f_{G_1}(o_{s_1}, 0, 0, m_{G_1}) + f_{G_2}(o_{s_2}, 0, 0, m_{G_2}) - 1 \tag{C–8}$$

$$f_G(0, o_{t_1} + o_{t_2} - 1, 0, \max\{m_{G_1},\ m_{G_2}\}) = \min f_{G_1}(0, o_{t_1}, 0, m_{G_1}) + f_{G_2}(0, o_{t_2}, 0, m_{G_2}) - 1 \tag{C–9}$$

Recursions C–7 through C–9 handle the three cases of both $s = s_1 = s_2$ and $t = t_1 = t_2$, only $t = t_1 = t_2$, and only $s = s_1 = s_2$ are deleted. In each case, node s is not connected to node t, and thus $c_{st} = c_{s_1 t_1} = c_{s_2 t_2} = 0$. We also subtract the objective value by one each time we merge two deleted terminals. The open set size o_s is updated by the

sum of o_{s_1} and o_{s_2} minus one due to the merge operation if neither s_1 nor s_2 is deleted. Similarly, the open set size o_t is updated by the sum of o_{t_1} and o_{t_2} minus one if neither t_1 nor t_2 is deleted. The maximum component size m_G in G is the maximum of m_{G_1} and m_{G_2}.

When all terminals $s_1,\ t_1,\ s_2$, and t_2 are not deleted, we consider the following two cases. If s is not connected with t, we have

$$f_G(o_{s_1}+o_{s_2}-1, o_{t_1}+o_{t_2}-1, 0, \max\{m_{G_1},\ m_{G_2}\}) = \min f_{G_1}(o_{s_1}, o_{t_1}, 0, m_{G_1}) + f_{G_2}(o_{s_2}, o_{t_2}, 0, m_{G_2}). \tag{C–10}$$

Note that in parallel operations, $c_{st}=0$ if and only if $c_{s_1t_1}=c_{s_2t_2}=0$, and we obtain the new open set at s by merging O_{s_1} with O_{s_2}, and the new open set at t by merging O_{t_1} with O_{t_2}. The maximum component size m_G in G is the maximum of m_{G_1} and m_{G_2}.

If node s is connected with t, then

$$\begin{aligned} f_G(o_s, o_t, 1, \max\{m_{G_1},\ m_{G_2}\}) = & \\ & \min f_{G_1}(o_{s_1}, o_{t_1}, c_{s_1t_1}, m_{G_1}) + f_{G_2}(o_{s_2}, o_{t_2}, c_{s_2t_2}, m_{G_2}) \\ & \text{s.t. } c_{s_1t_1}+c_{s_2t_2}\geq 1 \\ & (o_s, o_t) = \begin{cases} (o_{s_2}+o_{s_1}+o_{t_1}-2, o_{t_2}+o_{s_1}+o_{t_1}-2) & \text{if } c_{s_1t_1}=0,\ c_{s_2t_2}=1; \\ (o_{s_1}+o_{s_2}+o_{t_2}-2, o_{t_1}+o_{s_2}+o_{t_2}-2) & \text{if } c_{s_1t_1}=1,\ c_{s_2t_2}=0; \\ (o_{s_1}+o_{s_2}-2, o_{t_1}+o_{t_2}-2) & \text{if } c_{s_1t_1}=c_{s_2t_2}=1 \end{cases} \\ & o_{s_1}>0,\ o_{t_1}>0,\ o_{s_2}>0,\ o_{t_2}>0. \end{aligned} \tag{C–11}$$

To create an SPG with s and t connected, we require either $c_{s_1t_1}=1$ or $c_{s_2t_2}=1$, and thus present the sum of those values are no less than one. In the case of $c_{s_1t_1}=0,\ c_{s_2t_2}=1$, note that $O_{s_2}=O_{t_2}$ in this case, and the new open sets at s and t are formed by merging o_{s_2} (o_{t_2}) with o_{s_1} and o_{t_1} minus two, due to the two merges at $s=s_1=s_2$ and $t=t_1=t_2$. Similarly, if $c_{s_1t_1}=1,\ c_{s_2t_2}=0$, we have $O_{s_1}=O_{t_1}$, and the new open sets at s and t are formed by merging o_{s_1} (o_{t_1}) with o_{s_2} and o_{t_2} minus two. Finally, in the case when $c_{s_1t_1}=c_{s_2t_2}=1$, we have $O_{s_1}=O_{t_1}$ and $O_{s_2}=O_{t_2}$, and the new

open sets at s (t) are formed by merging o_{s_1} (o_{t_1}) and o_{s_2} (o_{t_1}) minus two. The maximum component size m_G in G is again the maximum of m_{G_1} and m_{G_2}.

Given a graph $\overline{G}$, we first construct a decomposition tree $T(\overline{G})$, and compute $f_G(o_s, o_t, c_{st}, m_G)$ at every node $G \in T(\overline{G})$. Note here that if we record objectives for $f_G(o_s, o_t, c_{st}, m_G)$, we would store at most $O(n^3)$ values at each SPG-graph node G (for the $O(n)$ possible values of o_s, o_t, and m_G). Then the merging for either series-operations or parallel-operations will require $O(n^6)$ steps. Because there are $O(n)$ nodes in $T(\overline{G})$, the overall complexity of solving MinMaxC on SPGs by the procedure above would be $O(n^7)$.

Considering the space capacity, we need $O(n^3)$ to store the values of $f_G(o_s, o_t, c_{st}, m_G)$ at each G, and the total space complexity is $O(n^4)$.

APPENDIX D
SMALL-WORLD NETWORKS

A class of "small-world" networks are first introduced by Watts & Strogatz (1998) in 1998, which have immediately received substantial attention in the past decade due to their ability to model many real-world scenarios. Relative research and popular result include the famous "six degrees of separation", which refers to the idea that, if a person is one step away from each person they know and two steps away from each person who is known by one of the people they know, then everyone is at most six steps away from any other person on Earth (Watts, 2004). The small-world model is firstly proposed to demonstrate the "small-world" phenomenon appearing in collaboration networks and the World Wide Web, and then has been shown to accommodate a large amount of complex systems in psychology, sociology, political science, economics, epidemiology, biology, medicine, environmental, and geological science (Newman et al., 2002; Petreska et al., 2010). Jeong et al. (2000) characterize the metabolic networks of 43 organisms representing all three domains of life as scale-free networks. From a microscope view, Bork et al. (2004) describe the interactions between proteins as heterogeneous networks, which assist in constructing reliable protein networks. Other studies have shown that small-world topology and related network robustness analysis can also be applied to the information-processing systems in brain networks (Bassett & Bullmore, 2006), dynamics in biochemical networks (Fox & Hill, 2001), and power grid networks (Holmgren, 2006). We refer readers to Cami & Deo (2007); Barabási (2009) for a comprehensive list of questions, challenges, and achievements accomplished in the filed of small-world networks research. In particular, Alderson (2008) proposes opportunities in analyzing complex networks for operations researchers from an optimization perspective.

Besides WS small world (Watts & Strogatz, 1998) models, the other two generic models that have mostly been investigated in the analysis of complex networks are

ER random graph (Erdös & Rényi, 1959) and BA scale-free (Barabási & Albert, 1999). ER random graph is usually denoted as $G(n, p)$, in which every possible edge occurs independently with probability p. On the other hand, BA scale-free networks origin from the observation that "many large random networks share the common feature that the distribution of their local connectivity is free of scale, following a power law" (Barabási & Albert, 1999), and are proposed to explain the ubiquity of power laws in a network structure. Watts & Strogatz (1998) reproduce WS small-world networks by using simple graph models that interpolate between regular and random graph structures. The graph is characterized by a few nodes (relative to the size of the network) called "hubs" that have high degrees, and many other nodes with low degrees. We refer the interested readers to Albert & Barabási (2002); Lü et al. (2004); Newman (2003) for detailed descriptions and mathematical models of topology, dynamics, and synchronization of complex networks. Studies show that BA scale-free networks are a subclass of WS small-world networks (Amaral et al., 2000). Due to many common statistical properties and real-world applications shared by BA scale-free and WS small-world networks, we do not distinguish between the two in the remainder of our discussion.

The analysis of random-error and targeted-attack tolerance is one common interest arising in the study of complex networks. Both theoretical and empirical results show that ER random graphs are functionally and topologically robust to malicious attacks but easily get disconnected under random failures. On the other hand, small-world networks are often error-tolerant, yet extremely vulnerable to targeted attacks (i.e., to the selected removal of a few nodes that play vital roles in maintaining the network connectivity) (e.g., Albert et al. (2000); Crucitti et al. (2004b); Nazarova (2006); Wang et al. (2006); He et al. (2009); Zhao & Xu (2009)). Crucitti et al. (2004a) provide an example to show that it is possible to collapse a complex network by breaking down a single node that has the largest load. Although connecting nodes with low degrees may enhance the

attack survivability of scale-free networks, it has little impact on the improvement of error tolerance (Zhao & Xu, 2009).

Due to small-world networks' vulnerability to targeted attacks, previous researchers suggested to solve the node deletion problem by using a greedy algorithm, i.e., we first delete a node with the largest degree in a current graph (if more than one nodes have the same largest degrees, break the tie arbitrarily), and then update the graph connectivity status by eliminating the deleted node and all its adjacent edges. We repeat this procedure for a certain amount of times according to a pre-given budget representing the largest number of nodes allowed to be deleted (Tu, 2000). However, there are two main weaknesses associated with the use of this greedy algorithm. First, the algorithm does not necessarily yield an optimal solution. Demonstrated by applications in communication and transportation, Latora & Marchiori (2005) develop a method based on *improvement analysis* to identify the crucial functioning points in a complex system modeled as a small-world network, which appear not to be always the highest-degree nodes.

Moreover, optimal solutions given by the greedy algorithm may not be consistent due to the fact that degrees are sometimes identical among different nodes, and thus results vary depending on the sequence of nodes deleted in each stage of the attack. To illustrate, we run the greedy algorithm for five times (i.e., Trail 1 to Trail 5) on the same scale-free network, allowing for different sequences of node deletion. We show the results in Figure D-1 in terms of the remaining largest component size. Although minor, there are noticeable differences between solutions to each trail.

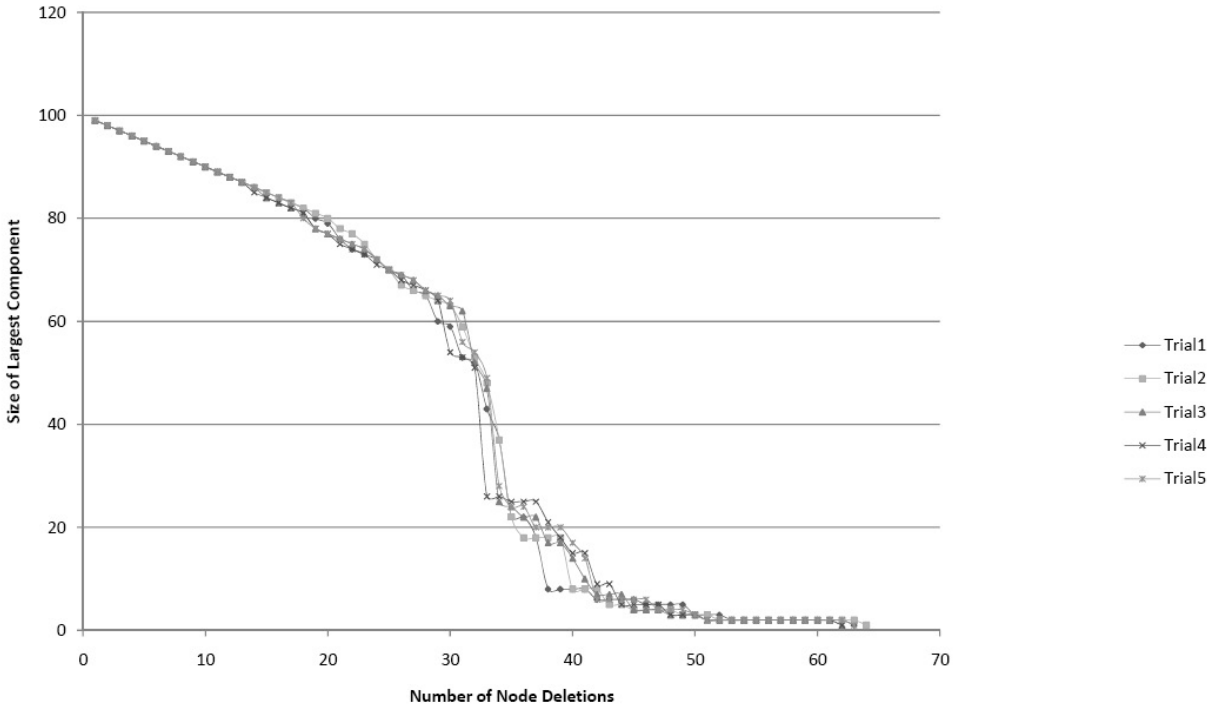


Figure D-1. Inconsistency of greedy algorithm.

APPENDIX E
FORMULATING MINMAXC BY USING A SS-RLT-BASED APPROACH

Similar to procedures in Section 4.2.1, we formulate the MinMaxC subproblem as a network flow problem on an auxiliary network. We first create a source node 0, a corresponding node in $\mathcal{N}$ for each node $i \in \mathcal{V}$, and a set of arcs $(0, k)$ for all $k \in \mathcal{N}$. Also, include directed arcs (i, j) and (j, i) corresponding to every $(i, j) \in \mathcal{A}$. We will disable the use of any arc (i, j) in this network either node i or j has been deleted. Our model will have a supply of n units at node 0 and a unit demand at every node $i \in \mathcal{N}$. These demands can be satisfied either by primary or secondary flows from node 0. Primary flows can only traverse a single arc from node 0, and can hence reach only a single graph component. Secondary flows are unit flows direct from node 0 to nodes in $\mathcal{N}$, and are not limited.

The primary flow exiting node 0 is equal to the number of node demands it serves, and thus represents a lower bound of maximum component size. Thus, to maximize component size, we maximize primary flow on the arc existing node 0.

Let f_{ij} represent the primary flow decision variables, for all arcs (i, j) in our network. Let $\bar{f}_{0j}$ represent the secondary flow decision variables, $\forall j \in \mathcal{N}$. Moreover, we associate binary variables z_k with each arc $(0, k),\ \forall k \in \mathcal{N}$, such that $f_{0k} > 0$ if $z_k = 1$, and $z_k = 0$ otherwise. We formulate the subproblem as

$$\eta'(y) = \max \quad \sum_{k \in \mathcal{N}} f_{0k} \tag{E–1a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{N}} (f_{0k} + \bar{f}_{0k}) = n \tag{E–1b}$$

$$\sum_{i \in FS(k)} f_{ki} - \left(f_{0k} + \bar{f}_{0k} + \sum_{j \in RS(k)} f_{jk} \right) = -1 \quad \forall k \in \mathcal{N} \tag{E–1c}$$

$$f_{ij} \leq (n-1) y_{ij} \quad \forall (i, j) \in \mathcal{E} \tag{E–1d}$$

$$f_{0k} - n z_k \leq 0 \quad \forall k \in \mathcal{N} \tag{E–1e}$$

$$\sum_{k \in \mathcal{N}} z_k = 1 \tag{E–1f}$$

$$z_k \in \{0,1\},\ \forall k \in \mathcal{N}, \tag{E–1g}$$

$$f_{0k},\ \bar{f}_{0k} \geq 0,\ \forall k \in \mathcal{N},\ f_{ij} \geq 0,\ \forall (i,j) \in \mathcal{E}, \tag{E–1h}$$

where Constraints E–1b and E–1c enforce flow balance at 0 and $k,\ \forall k \in \mathcal{N}$, respectively. (Note that Eq. E–1b is in fact unnecessary, since it is implied by the summation of Eq. E–1c.) Constraints E–1d allow no flow to use (i,j) if it is disconnected (i.e., $y_{ij} = 0$), and constraints E–1e prohibit flow f_{0k} on $(0,k)$ if $z_k = 0$. Lastly, Eq. E–1f allows exactly one arc of $(0,k),\ k \in \mathcal{N}$ to have a positive value of f_{0k}.

Figure E-1 illustrates the graph transformation and an optimal solution to the above formulation. Suppose that given first-stage inputs $(\hat{x}, \hat{y})$, the node and arc sets are $\{1,2,3,4,5,6\}$ and $\{(1,2),(2,3),(4,5)\}$ in the remaining graph, respectively. An optimal solution is $f_{01} = 3$, $\bar{f}_{0j} = 1,\ \forall j = 4,5,6$, and $f_{12} = 2,\ f_{23} = 1$, all remaining variable values equal 0. The maximum component is $\{1,2,3\}$ with size of $f_{01} = 3$.

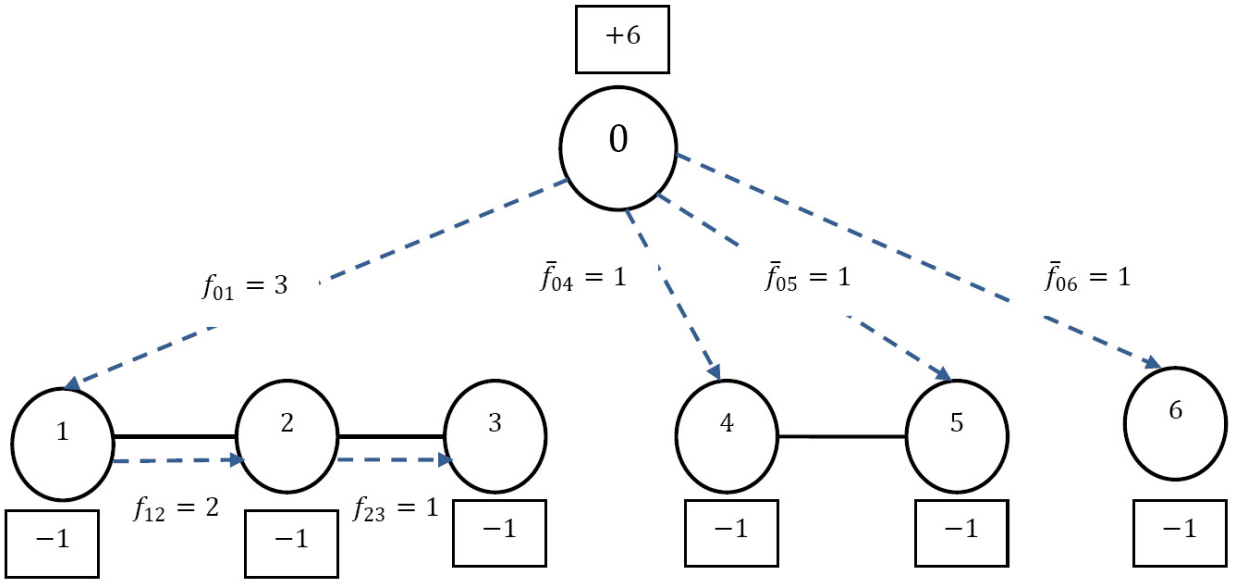


Figure E-1. Illustration of the auxiliary network for solving MinMaxC subproblem.

The LP relaxation of the above formulation does not provide the convex hull of the original MIP formulation. However, because all binary-valued variables appear in

the partitioning Constraints E–1f, we can recover the convex hull by using the Special Structures Reformulation-Linearization Technique (SSRLT) (Sherali et al., 1998).

E.1 Problem Reformulation

The key SSRLT step multiplies Constraints E–1c through E–1f, as well as the bounds $y_{ij} \leq 1$ and $y_{ij} \geq 0$ by $z_k,\ \forall k = 1, \ldots, n$, and $y \geq 0$ by $(1 - \sum_{k \in \mathcal{N}} z_k)$ (recall that Constraint E–1c is redundant). After linearizing all quadratic terms induced by this reformulation, we obtain the convex hull of solutions for which $z_1, \ldots, z_n$ are binary-valued.

Observe that $f_{0k} z_k = f_{0k}$, and $f_{0k} z_l = 0,\ \forall l \neq k$. Also note that $f_{0k} \bar{f}_{0k} = 0,\ \forall k \in \mathcal{N}$ since both f_{0k} and $\bar{f}_{0k}$ cannot be positive in an optimal solution. Thus, we have $\bar{f}_{0k} z_k = 0,\ \forall k \in \mathcal{N}$. We complete the reformulation by substituting $(z_k)^2 = z_k,\ \forall k,\ z_l z_k = 0,\ \forall l \neq k$; and by defining $\bar{u}_{0lk} \equiv \bar{f}_{0l} z_k,\ \forall l,\ k \in \mathcal{N},\ l \neq k$, $u_{ijk} \equiv f_{ij} z_k,\ \forall (i,j) \in \mathcal{E},\ k \in \mathcal{N}$, and $v_{ijk} \equiv y_{ij} z_k,\ \forall (i,j) \in \mathcal{A},\ k \in \mathcal{N}$. The following reformulation represents the convex hull.

$$\max \quad \sum_{k \in \mathcal{N}} f_{0k} \tag{E–2a}$$

$$\text{s.t.} \quad \sum_{i \in FS(k)} u_{kik} - \sum_{j \in RS(k)} u_{jkk} - f_{0k} + z_k = 0 \quad \forall k \in \mathcal{N} \tag{E–2b}$$

$$\sum_{i \in FS(l)} u_{lik} - \sum_{j \in RS(l)} u_{jlk} - \bar{u}_{0lk} + z_k = 0 \quad \forall l,\ k \in \mathcal{N},\ l \neq k \tag{E–2c}$$

$$f_{0k} - n z_k \leq 0 \quad \forall k \in \mathcal{N} \tag{E–2d}$$

$$u_{ijk} - (n-1) v_{ijk} \leq 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{N} \tag{E–2e}$$

$$\sum_{k \in \mathcal{N}} v_{ijk} \leq y_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{E–2f}$$

$$v_{ijk} - z_k \leq 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{N} \tag{E–2g}$$

$$\sum_{k \in \mathcal{N}} z_k = 1 \tag{E–2h}$$

$$\bar{u}_{0lk} \geq 0,\ \forall l,\ k \in \mathcal{N},\ l \neq k,\ u_{ijk},\ v_{ijk} \geq 0,\ \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{N}$$

$$f_{0k},\ \bar{f}_{0k},\ z_k \geq 0,\ \forall k \in \mathcal{N},\ f_{ij} \geq 0,\ \forall (i,j) \in \mathcal{A}. \tag{E–2i}$$

Next, we streamline the formulation by performing the following simplifications. Note that for a given $k \in \mathcal{N}$, summing Constraints E–2b and E–2c, $\forall l \in \mathcal{N},\ l \neq k$, we get $f_{0k} + \sum_{l \in \mathcal{N},\ l \neq k} \bar{u}_{0lk} = nz_k$, which implies Constraints E–2d since $\bar{u}_{0lk} \geq 0,\ \forall l \neq k$. Furthermore, in Constraints E–2b, we have $u_{jkk} = 0,\ \forall k = 1, \ldots, n$, since we never push flow back to node k if $f_{sk} > 0$ (i.e., $f_{jk} = 0$ if $z_k = 1,\ \forall j \in RS(k),\ j \neq s$), and $u_{jkk} = f_{jk} z_k = 0$ when $z_k = 0$.

We now show how to project out the v-variables to obtain a smaller equivalent formulation. Note that the role of Constraints E–2e through E–2g is to guarantee that $u_{ijk} = 0$ if $v_{ijk} = 0$, (i.e., if either $y_{ij} = 0$ or $z_k = 0$). In the case of $z_k = 0$, then $f_{0k} = 0$ as described above. Without Constraints E–2e through E–2g, we still have that $\bar{u}_{0lk} = 0,\ \forall l \neq k$ and $u_{kik} = 0,\ \forall i \in FS(k)$. Keep Constraints E–2b, although other values of u_{ijk} may take on positive values in cyclic flows. However, the values that v_{ijk} take do not affect the objective function value or any other constraints in this case, and can be ignored. To equivalently enforce the condition that $u_{ijk} = 0$ whenever $y_{ij} = 0$, we simply replace v_{ijk} in Constraints E–2e with y_{ij}. After multiplying both sides of Constraints E–2b and E–2c by -1, we get the simplified reformulation:

$$\max \quad \sum_{k \in \mathcal{N}} f_{0k} \tag{E–3a}$$

$$\text{s.t.} \quad -\sum_{i \in FS(k)} u_{kik} + f_{0k} - z_k = 0 \quad \forall k \in \mathcal{N} \tag{E–3b}$$

$$-\sum_{i \in FS(l)} u_{lik} + \sum_{j \in RS(l)} u_{jlk} + \bar{u}_{0lk} - z_k = 0 \quad \forall l,\ k \in \mathcal{N},\ l \neq k \tag{E–3c}$$

$$\sum_{k \in \mathcal{N}} u_{ijk} \leq (n-1) y_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{E–3d}$$

$$\sum_{k \in \mathcal{N}} z_k = 1 \tag{E–3e}$$

$$\bar{u}_{0lk} \geq 0,\ \forall l,\ k \in \mathcal{N},\ l \neq k,\ u_{ijk} \geq 0,\ \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{N}$$

$$f_{0k},\ \bar{f}_{0k},\ z_k \geq 0,\ \forall k \in \mathcal{N},\ f_{ij} \geq 0,\ \forall (i,j) \in \mathcal{A}. \tag{E–3f}$$

Let σ_{kk}, σ_{lk}, τ_{ij}, and λ be the duals associated with Constraints E–3b, E–3c, E–3d, and E–3e, respectively. We formulate the dual of the above formulation as

$$\begin{aligned}
\max \quad & \sum_{(i,j)\in\mathcal{A}} (n-1) y_{ij}\tau_{ij} + \lambda && \text{(E–4a)}\\
\text{s.t.} \quad & \sigma_{kk} \geq 1 \quad \forall k \in \mathcal{N} && \text{(E–4b)}\\
& \sigma_{lk} \geq 0 \quad \forall l,\ k \in \mathcal{N},\ l \neq k && \text{(E–4c)}\\
& \sigma_{jk} - \sigma_{ik} + \tau_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{N} && \text{(E–4d)}\\
& -\sum_{l\in\mathcal{N}} \sigma_{lk} + \lambda \geq 0 \quad \forall k \in \mathcal{N} && \text{(E–4e)}\\
& \tau_{ij} \geq 0,\ \forall (i,j) \in \mathcal{A}, && \text{(E–4f)}
\end{aligned}$$

where Constraints E–4b, E–4c, E–4d, and E–4e are associated with primal variables f_{0k}, $\bar{u}_{0lk}$ $(l \neq k)$, u_{ijk}, and z_k, respectively. Again, incorporating Constraints E–4 into the framework of MinMaxC leads to a "min-min" formulation. We then replace the bilinear terms with a series of linear inequalities by defining

$$\psi_{ij} \equiv y_{ij}\tau_{ij}, \quad \forall (i,j) \in \mathcal{E}. \tag{E–5}$$

Also as in our prior analysis, we see that τ_{ij} takes a value of 0 or 1, and we linearize Constraints E–5 as

$$\psi_{ij} \geq y_{ij} + \tau_{ij} - 1,\ \psi_{ij} \leq y_{ij},\ \psi_{ij} \leq \tau_{ij},\ \psi_{ij} \geq 0. \tag{E–6}$$

The reformulation of MinMaxC is then given by:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} (n-1)\psi_{ij} + \lambda + \frac{1}{n}\sum_{i=1}^{n}(1-x_i) && \text{(E–7)}\\
\text{s.t.} \quad & \text{Constraints 4–1b through 4–1e, E–4b through E–4f, E–6.}
\end{aligned}$$

E.2 Simplified Reformulation

Note that given $G(\mathcal{V},\mathcal{E})$ and a first-stage solution $(\hat{x},\hat{y})$, we can algorithmically find the optimal dual values to Formulation E–4 as follows. Let C denote the set of all nodes

that belongs to the largest component (break ties arbitrarily if there exists more than one largest component). Arbitrarily pick any node $k' \in C$, and let $f_{0k'} = |C|,\ z_{k'} = 1$ and $f_{0k} = 0,\ z_k = 0,\ \forall k \neq k'$. Define $\widehat{Y}^b = \{(i,j) \in \mathcal{A} :\ \hat{y}_{ij} = b\}$, for $b = 0$ and 1. For every node $k \in C$, define C_k as the component that contains node k.

Proposition E.1. *Given* $(\hat{x}, \hat{y})$*, an optimal dual solution to Formulation E–4 is*

- $\lambda = |C|$.
- $\tau_{ij} = 1,\ \forall (i,j) \in \widehat{Y}^0$*, and* $\tau_{ij} = 0$ *otherwise.*
- $\sigma_{ik} = 1$ *if* $C_i = C_k$*, and* $\sigma_{ik} = 0$ *otherwise.*

Proof. We first show that the solution is dual feasible. It is trivial to see that Constraints E–4b and E–4c are both satisfied by the assigned values of σ. Regarding Constraints E–4d, the only scenario in which $\sigma_{jk} - \sigma_{ik} = -1$ is possible arises when $\sigma_{ik} = 1$ and $\sigma_{jk} = 0$, and thus nodes i and j do not belong to the same component. This implies that $(i,j) \in \widehat{Y}^0$, and thus $\tau_{ij} = 1$, so that we satisfy Constraints E–4d as an equality. Furthermore, Constraints E–4e holds since for all $k = 1, \ldots, n$,

$$-\sum_{l=1}^{n} \sigma_{lk} + \lambda = -|C_k| + |C| \geq 0,$$

due to the fact that C is the largest component.

Next, we verify that all complementary slackness conditions are satisfied. With respect to the primal constraints, we must show that $\tau_{ij}\left[(n-1)y_{ij} - \sum_{k \in \mathcal{N}} u_{ijk}\right] = 0$ is satisfied for all arcs $(i,j) \in \mathcal{A}$. This is clearly the case when $\tau_{ij} = 0$; when $\tau_{ij} = 1$, then $y_{ij} = 0$ and $u_{ijk} = 0$ as well, thus satisfying the complementary slackness conditions. With respect to dual constraints, we verify that all positive primal variables are associated with dual constraints having zero slack. Note that $f_{0k'} > 0$, and Ineq. E–4b holds as an equality since $\sigma_{k'k'} = 1$. We have $\bar{u}_{0lk} = \bar{f}_{sl} z_k > 0$ when $\bar{f}_{0l} > 0$ and $z_k = z_{k'} = 1$. In this case, Ineq. E–4c holds as an equality, since l is not connected to k' and $\sigma_{lk} = 0$. Considering Constraints E–4d, $u_{ijk} = f_{ij} z_k > 0$ when $f_{ij} > 0$ and $k = k'$.

Since the flow from i to j is positive, $(i,j) \in \widehat{Y}^1$, which indicates that $\sigma_{ik} = \sigma_{jk} = 1$ and $\tau_{ij} = 0$, and so Ineq. E–4d holds as an equality. Finally, when $z_k = z_{k'} = 1$, we have that Ineq. E–4e holds as an equality such that

$$-\sum_{l=1}^{n} \sigma_{lk'} + \lambda = -|C| + |C| = 0.$$

The dual solution is feasible and complementary slack to the primal, and is therefore optimal. □

According to Proposition E.1, τ- and y-variables satisfy complementary slackness, and thus the following property follows.

Corollary 1. *The following constraint is valid to Formulation E–7.*

$$y_{ij} + \tau_{ij} = 1, \ \forall (i,j) \in \mathcal{A} \tag{E–8}$$

We therefore have that $\psi_{ij} = 0$, and can substitute τ_{ij} by $(1 - y_{ij})$, for all $(i,j) \in \mathcal{A}$. It leads to the following representation of the convex hull of MinMaxC, which is equivalent to Formulation 4–7 in Chapter 4.

$$\begin{aligned}
\min \quad & \lambda + \frac{1}{n}\sum_{i=1}^{n}(1 - x_i) \\
\text{s.t.} \quad & \text{Constraints 4–1b through 4–1e} \\
& \sigma_{kk} \geq 1 \quad \forall k \in \mathcal{N} \\
& \sigma_{jk} - \sigma_{ik} \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{A}, \ k \in \mathcal{N} \\
& \lambda \geq \sum_{i \in \mathcal{N}} \sigma_{ik} \quad \forall k \in \mathcal{N} \\
& \sigma_{ik} \geq 0 \quad \forall i, \ k \in \mathcal{N}, \ i \neq k.
\end{aligned}$$

REFERENCES

Ahmed, S., Tawarmalani, M., & Sahinidis, N. V. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, *100*(2), 355–377.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice Hall.

Akgun, I. (2000). *The k-group maximum-flow network-interdiction problem*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*, 47–97.

Albert, R., Jeong, H., & Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, *406*(6794), 378–382.

Alderson, D. L. (2008). Catching the "network science" bug: Insight and opportunity for the operations researcher. *Operations Research*, *56*(5), 1047–1065.

Alevras, D., Grötschel, M., & Wessäly, R. (1997). Capacity and survivability models for telecommunications networks. Tech. rep., Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Amaral, L. A. N., Scala, A., Barthelemy, M., & Stanley, H. E. (2000). Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*, *97*, 11149–11152.

Arora, S., Hazan, E., & Kale, S. (2010). $O(\sqrt{\log n})$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time. *SIAM Journal on Computing*, *39*(5), 1748–1771.

Arora, S., Lee, J. R., & Naor, A. (2005). Euclidean distortion and the sparsest cut. In *Proceedings of the* 37^{th} *Annual ACM Symposium on Theory of Computing*, (pp. 553–562). Baltimore, Maryland: ACM Press.

Arroyo, J. M. (2010). Bilevel programming applied to power system vulnerability analysis under multiple contingencies. *IET Generation, Transmission, and Distribution*, *4*(2), 178–190.

Arulselvan, A., Commander, C. W., Elefteriadou, L., & Pardalos, P. M. (2009). Detecting critical nodes in sparse graphs. *Computers and Operations Research*, *36*(7), 2193–2200.

Barabási, A.-L. (2009). Scale-free networks: A decade and beyond. *Science*, *325*, 412–413.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*, 509–512.

Bassett, D. S., & Bullmore, E. (2006). Small-world brain networks. *Neuroscientist*, *12*(6), 512–523.

Benders, J. F. (1962). Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, *4*(1), 238–252.

Berge, C. (1962). *Theory of Graphs and its Applications*. New York, NY: Wiley.

Bertsimas, D., Natarajan, K., & Teo, C.-P. (2006). Persistence in discrete optimization under data uncertainty. *Mathematical Programming, Series B*, *108*(2–3), 251–274.

Birge, J. R., & Louveaux, F. V. (1997). *Introduction to Stochastic Programming*. New York: Springer.

Blair, J. R. S., Heggernes, P., Horton, S., & Manne., F. (2004). Broadcast domination algorithms for interval graphs, series-parallel graphs, and trees. *Congressus Numerantium*, *169*, 55–77.

Blair, J. R. S., & Horton, S. B. (2005). Broadcast covers in graphs. *Congressus Numerantium*, *173*, 109–115.

Bonsma, P. (2004). Sparsest cuts and concurrent flows in product graphs. *Discrete Applied Mathematics*, *136*(2–3), 173–182.

Borgatti, S. P. (2006). Identifying sets of key players in a social network. *Computational and Mathematical Organization*, *12*(1), 21–34.

Borgatti, S. P., & Everett, M. G. (2006). A graph-theoretic perspective on centrality. *Social Networks*, *28*(4), 466–484.

Bork, P., Jensen, L. J., von Mering, C., Ramani, A. K., Lee, I., & Marcotte, E. M. (2004). Protein ineraction networks from yeast to human. *Structural Biology*, *14*, 292–299.

Bowman, R. A. (1995). Efficient estimation of arc criticalities in stochastic activity networks. *Management Science*, *41*(1), 58–67.

Bowman, R. A., & Muckstadt, J. A. (1993). Stochastic analysis of cyclic schedules. *Operations Research*, *41*(5), 947–958.

Brown, G., Carlyle, M., Salmeron, J., & Wood, K. (2006). Defending critical infrastructure. *Interfaces*, *36*(6), 530–544.

Brucker, P., Drexl, A., Moehring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*(1), 3–41.

Burt, J. M., & Garman, M. B. (1971). Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science*, *18*(3), 207–217.

Cambazard, H., Hladik, P.-E., Déplanche, A.-M., Jussien, N., & Trinquet, Y. (2004). Decomposition and learning for a hard real time task allocation problem. In M. Wallace (Ed.) *Lecture Notes in Computer Science*, vol. 3258 of *Principles and Practice of Constraint Programming (CP 2004)*, (pp. 153–167). Springer.

Cambazard, H., & Jussien, N. (2005). Identifying and exploiting problem structures using explanation-based constraint programming. In R. Barták, & M. Milano (Eds.) *Lecture Notes in Computer Science*, vol. 3524 of *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2005)*, (pp. 94–109). Springer.

Cami, A., & Deo, N. (2007). Techniques for analyzing dynamic random graph models of web-like networks: An overview. *Networks*, *51*(4), 211–255.

Carnal, D. D. (2005). *An enhanced implementation of models for electric power grid interdiction*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Carøe, C. C., & Tind, J. (1997). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, *83*(1–3), 451–464.

Chawla, S., Krauthgamer, R., Kumar, R., Rabani, Y., & Sivakumar, D. (2006). On the hardness of approximating multicut and sparsest-cut. *Computational Complexity*, *15*(2), 94–114.

Chen, X., Sim, M., Sun, P., & Zhang, J. (2008). A linear-decision based approximation approach to stochastic programming. *Operations Research*, *56*(2), 344–357.

Chien, A.-C. (2006). *Optimized recovery of damaged electrical power grids*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Chtourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers and Industrial Engineering*, *55*(1), 183–194.

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, *54*(4), 756–766.

Cohen, R., Ben-Avraham, D., & Havlin, S. (2003). Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, *91*(24), 247901–247905.

Cormican, K. J., Morton, D. P., & Wood, R. K. (1998). Stochastic network interdiction. *Operations Research*, *46*(2), 184–197.

Crucitti, P., Latora, V., & Marchiori, M. (2004a). Model for cascading failures in complex networks. *Physical Review E*, *69*(4), 045104.

Crucitti, P., Latora, V., Marchiori, M., & Rapisarda, A. (2004b). Error and attack tolerance of complex networks. *Physica A*, *340*, 388–394.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, *1*(3–4), 197–206.

de Jaenisch, C. F. (1862). Applications de l'analyse mathematique au jue des echecs. *Petrograd*.

Demeulemeester, E. L., & Herroelen, W. S. (2002). *Project Scheduling: A Research Handbook*. Norwell, MA: Springer.

Di Summa, M., Grosso, A., & Locatelli, M. (2010). Complexity of the critical node problem over trees. http://www.optimization-online.org/DB_HTML/2010/02/2540.html.

Dinh, T. N., Xuan, Y., Thai, M. T., Park, E. K., & Znati, T. (2010). On approximation of new optimization methods for assessing network vulnerability. In *Proceedings of the* 29^{th} *IEEE Conference on Computer Communications (INFOCOM)*. San Diego, CA.

Dunbar, J. E., Erwin, D. J., Haynes, T. W., Hedetniemi, S. M., & Hedetniemi., S. T. (2006). Broadcasts in graphs. *Discrete Applied Mathematics*, *154*, 59–75.

Duque-Antón, M., Bruyaux, F., & Semal, P. (2009). Measuring the survivability of a network: Connectivity and rest-connectivity. *Communication Networks*, *11*(2), 149–159.

Elmaghraby, S. E., Ferreira, A. A., & Tavares, L. V. (2000). Optimal start times under stochastic activity durations. *International Journal of Production Economics*, *64*(1–3), 153–164.

Erdös, P., & Rényi, A. (1959). On random graphs. *Publicationes Mathematicae*, *6*, 290–297.

Erwin, D. J. (2004). Dominating broadcasts in graphs. *Bulletin of the Institute of Combinatorics and its Applications*, *42*, 89–105.

Flippo, O. E., & Rinnooy Kan, A. H. G. (1993). Decomposition in general mathematical programming. *Mathematical Programming*, *60*, 361–382.

Fox, J. J., & Hill, C. C. (2001). From topology to dynamics in biochemical networks. *Chaos*, *11*(4), 809–815.

Freeman, L. C. (1978–1979). Centrality in social networks I: Conceptual clarification. *Social Networks*, *1*(3), 215–239.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company.

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, *10*(4), 237–260.

Goldratt, E. M. (1997). *Critical Chain*. Great Barrington, MA: North River Press.

Goldschmidt, O., & Hochbaum, D. S. (1994). A polynomial algorithm for the k-cut problem for a fixed k. *Mathematics of Operations Research*, *19*(1), 24–37.

Golenko-Ginzburg, D., & Gonik, A. (1999). A heuristic for network project scheduling with random activity durations depending on the resource allocation. *International Journal of Production Economics*, *55*, 149–162.

Golenko-Ginzburg, D., Gonik, A., & Sitniakovski, S. (2000). Resource supportability model for stochastic network projects under a chance constraint. *Communications in Dependability and Quality Management*, *3*(1), 89–102.

Gross, J. L., & Yellen, J. (2003). *Handbook of Graph Theory*. Boca Raton, FL: CRC Press.

Grötschel, M., Monma, C. L., & Stoer, M. (1995). Design of survivable networks. In *Handbooks on Operations Research and Management Science*, vol. 7, chap. 10. North-Holland, Amsterdam: Elsevier Science.

Grubesic, T. H., Matisziw, T. C., Murray, A. T., & Snediker, D. (2008). Comparative approaches for assessing network vulnerability. *International Regional Science Review*, *31*(1), 88–112.

Grubesic, T. H., & Murray, A. T. (2006). Vital nodes, interconnected infrastructures, and the geographies of network survivability. *Annals of the Association of American Geographers*, *96*(1), 64–83.

Gutjahr, W. J., Strauss, C., & Wagner, E. (2000). A stochastic branch-and-bound approach to activity crashing in project management. *INFORMS Journal on Computing*, *12*(2), 125–135.

Hagstrom, J. N. (1990). Computing the probability distribution of project duration in a PERT network. *Networks*, *20*(2), 231–244.

Hajiaghayi, M. T., & Räcke, H. (2006). An $O(\sqrt{n})$-approximation algorithm for directed sparsest cut. *Information Processing Letters*, *97*(4), 156–160.

Haneveld, W. K. K., & Van der Vlerk, M. H. (1999). Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, *85*, 39–57.

Hartman, J. C., Büyüktahtakin, I. E., & Smith, J. C. (2010). Dynamic-programming-based inequalities for the capacitated lot-sizing problem. *IIE Transactions*, *42*(12), 915–930.

Haynes, T. W., Hedetniemi, S. T., & Slater, P. J. (Eds.) (1998a). *Domination in Graphs: Advanced Topics*. New York: Marcel Dekker, Inc.

Haynes, T. W., Hedetniemi, S. T., & Slater., P. J. (1998b). *Fundamentals of Domination in Graphs*. New York: Marcel Dekker, Inc.

He, S., Li, S., & Ma, H. (2009). Effect of edge removal on topological and functional robustness of complex networks. *Physica A*, *388*, 2243–2253.

Heggernes, P., & Lokshtanov, D. (2006). Optimal broadcast domination in polynomial time. *Discrete Mathematics*, *306*(24), 3267–3280.

Herroelen, W., & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, *19*(5), 559–577.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*(2), 289–306.

Hindelang, T. J., & Muth, J. F. (1979). A dynamic programming algorithm for decision CPM networks. *Operations Research*, *27*, 225–241.

Hochbaum, D. S., & Shmoys, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, *10*(2), 180–184.

Holmgren, A. J. (2006). Using graph models to analyze the vulnerability of electric power networks. *Risk Analysis*, *26*, 955–969.

Hooker, J. N. (2000). *Logic-based methods for optimization: Combining optimization and constraint satisfaction*. New York: John Wiley & Sons.

Hooker, J. N. (2005a). A hybrid method for the planning and scheduling. *Constraints*, *10*(4), 385–401.

Hooker, J. N. (2005b). Planning and scheduling to minimize tardiness. In *Lecture Notes in Computer Science*, vol. 3709 of *Principles and Practice of Constraint Programming (CP 2005)*, (pp. 314–327). Springer.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations research*, *55*(3), 588.

Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, *96*(1), 33–60.

Houck, D. J., Kim, E., O'Reilly, G. P., Picklesimer, D. D., & Uzunalioglu, H. (2004). A network survivability model for critical national infrastructures. *Bell Labs Technical Journal*, *8*(4), 153–172.

Iida, T. (2000). Computing bounds on project duration distributions for stochastic PERT networks. *Naval Research Logistics*, *47*(7), 559–580.

ILOG (2008). *CPLEX 11.0 User's Manual*. ILOG, Armonk, NY.

Janjarassuk, U., & Linderoth, J. T. (2008). Reformulation and sampling to solve a stochastic network interdiction problem. *Networks*, *52*(3), 120–132.

Jenelius, E., Petersen, T., & Mattson, L.-G. (2006). Importance and exposure in road network vulnerability analysis. *Transportation Research Part A*, *40*(7), 537–560.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., & Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, *407*, 651–654.

Kall, P., & Wallace, S. W. (1994). *Stochastic Programming*. Chichester, England: Wiley.

Kelley, J. E. (1961). Critical path planning and scheduling: Mathematical basis. *Operations Research*, *9*(3), 296–320.

Kelley, J. E. (1963). The critical path method: Resource planning and scheduling. *Industrial Scheduling*, *2*(1), 347–365.

Kerivin, H., & Mahjoub, A. R. (2005). Design of survivable networks: A survey. *Networks*, *46*(1), 1–21.

Kleywegt, A. J., Shapiro, A., & Homem-de-Mello, T. (2001). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, *12*(2), 479–502.

Krebs, V. E. (2001). Mapping networks of terrorist cells. *Connections*, *21*(3), 43–52.

Kulkarni, V. G., & Adlakha, V. G. (1986). Markov and Markov-regenerative PERT networks. *Operations Research*, *34*(5), 769–781.

Laporte, G., & Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, *13*(3), 133–142.

Laporte, G., Louveaux, F. V., & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science*, *26*(3), 161–170.

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

Laslo, Z. (2003). Activity time-cost tradeoffs under time and cost chance constraints. *Computers and Industrial Engineering*, *44*(3), 365–384.

Latora, V., & Marchiori, M. (2005). Vulnerability and protection of infrastructure networks. *Physical Review E*, *71*, 015103.

Lim, C., & Smith, J. C. (2007). Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, *39*(1), 15–26.

Lokshtanov, D. (2007). *Broadcast Domination*. Master's thesis, Department of Informatics, University of Bergen, Bergen, Norway.

Lü, J., Leung, H., & Chen, G. (2004). Complex dynamical networks: Modelling, synchronization and control. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, *11a*, 70–77.

Luedtke, J., & Ahmed, S. (2008). A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, *19*(2), 674–699.

Mak, W. K., Morton, D. P., & Wood, R. K. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, *24*(1), 47–56.

Mann, C. F., Matula, D. W., & Olinick, E. V. (2008). The use of sparsest cuts to reveal the hierarchical community structure of social networks. *Social Networks*, *30*(3), 223–234.

Matisziw, T. C., & Murray, A. T. (2009). Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Computers and Operations Research*, *36*(1), 16–26.

Matisziw, T. C., Murray, A. T., & Grubesic, T. H. (2009). Exploring the vulnerability of network infrastructure to disruption. *Annals of Regional Science*, *43*(2), 307–321.

Matula, D. W., & Shahrokhi, F. (1990). Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, *27*(1–2), 113–123.

Mitchell, G., & Klastorin, T. (2007). An effective methodology for the stochastic project compression problem. *IIE Transactions*, *39*(10), 957–969.

Moehring, R. H. (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, *32*, 89–120.

Murray, A. T., Matisziw, T. C., & Grubesic, T. H. (2007). Critical network infrastructure analysis: Interdiction and system flow. *Journal of Geographical Systems*, *9*(2), 103–117.

Murray, A. T., Matisziw, T. C., & Grubesic, T. H. (2008). A methodological overview of network vulnerability analysis. *Growth and Change*, *39*(4), 573–592.

Myung, Y.-S., & Kim, H.-J. (2004). A cutting plane algorithm for computing k-edge survivability of a network. *European Journal of Operational Research*, *156*(3), 579–589.

Nazarova, I. A. (2006). Models and methods for solving the problem of network vulnerability. *Systems Analysis and Operations Research*, *45*(4), 567–578.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, *45*(2), 167–256.

Newman, M. E. J., Watts, D. J., & Strogatz, S. H. (2002). Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(Supplement 1), 2566–2572.

Norkin, V. I., Pflug, G. C., & Ruszczyński, A. (1998). A branch and bound method for stochastic global optimization. *Mathematical Programming*, *83*(1–3), 425–450.

Ntaimo, L., & Sen, S. (2008). A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, *40*(3), 299–319.

Ore, O. (1967). *Theory of Graphs*, vol. 38. Providence, RI: American Mathematical Society, Third ed.

Ozdamar, L., & Ulusoy, G. (1995). A survey on the resource constrained project scheduling problem. *IIE Transactions*, *27*(5), 574–586.

Patterson, J. H. (1984). A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. *Management Science*, *30*, 854–867.

Penuel, J., Smith, J. C., & Yuan, Y. (2010). An integer decomposition algorithm for solving a two-stage facility location problem with second-stage activation costs. *Naval Research Logistics*, *57*(5), 391–402.

Petreska, I., Tomovski, I., Gutierrez, E., Kocarev, L., Bono, F., & Poljansek, K. (2010). Application of modal analysis in assessing attack vulnerability of complex networks. *Communications in Nonlinear Science and Numerical*, *15*(4), 1008–1018.

Resende, M. G. C., & Pardalos, P. M. (Eds.) (2006). *Handbook of Optimization in Telecommunications*. New York: Springer.

Saharidis, G. K. D., Boile, M., & Theofanis, S. (2010a). Initialization of the Benders master problem using valid inequalities applied to fixed-charge network problems. *Expert Systems With Applications*, *38*(6), 6627–6636.

Saharidis, G. K. D., & Ierapetritou, M. G. (2010). Improving Benders decomposition using maximum feasible subsystem (MFS) cut generation strategy. *Computers & Chemical Engineering*, *34*(8), 1237–1245.

Saharidis, G. K. D., Minoux, M., & Ierapetritou, M. G. (2010b). Accelerating Benders method using covering cut bundle generation. *International Transactions in Operational Research*, *17*(2), 221–237.

Salmeron, J., Wood, K., & Baldick, R. (2004). Analysis of electric grid security under terrorist threat. *IEEE Transactions on Power Systems*, *19*(2), 905–912.

San Martin, P. A. (2007). *Tri-level optimization models to defend critical infrastructure*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Scholl, A. (2001). *Robuste Planung und Optimierung: Grundlagen, Konzepte, und Methoden*. Physica-verlag, Experimentelle Untersuchungen, Heidelberg.

Schultz, R. (2003). Stochastic programming with integer variables. *Mathematical Programming*, *97*(1–2), 285–309.

Sen, S., & Higle, J. L. (2005). The C^3 theorem and a D^2 algorithm for large-scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, *104*(1), 1–20.

Sen, S., & Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, *106*(2), 203–223.

Shapiro, A., & Homem-de-Mello, T. (2000). On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *SIAM Journal on Optimization*, *11*(1), 70–86.

Shen, S. (2011). Domination problems. In J. J. Cochran (Ed.) *Encyclopedia of Operations Research and Management Science*. Hoboken, NJ: Wiley.

Shen, S., Smith, J. C., & Goli, R. (2011). Exact interdiction models for disconnecting networks via node deletions. *Working paper*, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL.

Sherali, H. D. (2001). On mixed-integer zero-one representations for separable lower semi-continuous piecewise-linear functions. *Operations Research Letters*, *28*(4), 155–160.

Sherali, H. D., & Adams, W. P. (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, *3*(3), 411–430.

Sherali, H. D., & Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, *52*(1), 83–106.

Sherali, H. D., Adams, W. P., & Driscoll, P. J. (1998). Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. *Operations Research*, *46*(3), 396–405.

Sherali, H. D., & Fraticelli, B. M. P. (2002). A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, *22*(1–4), 319–342.

Sherali, H. D., & Smith, J. C. (2009). Two-stage stochastic risk threshold and hierarchical multiple risk problems: Models and algorithms. *Mathematical Programming, Series A*, *120*(2), 403–427.

Sherali, H. D., & Zhu, X. (2007). On solving discrete two-stage stochastic programs having mixed-integer first and second-stage variables. *Mathematical Programming*, *108*(2–3), 597–616.

Shier, D. T. (1991). *Network Reliability and Algebraic Structures*. Oxford, NY, USA: Oxford University Press.

Smith, J. C., & Lim, C. (2008). Algorithms for network interdiction and fortification games. In A. Migdalas, P. M. Pardalos, L. Pitsoulis, & A. Chinchuluun (Eds.) *Pareto Optimality, Game Theory and Equilibria*, Nonconvex Optimization and its Applications. New York: Springer.

Smith, J. C., Lim, C., & Sudargho, F. (2007). Survivable network design under optimal and heuristic interdiction scenarios. *Journal of Global Optimization*, *38*(2), 181–199.

Takamizawa, K., Nishizeki, T., & Saito, N. (1982). Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the Association for Computing Machinery*, *29*(3), 623–641.

Tu, Y. (2000). How robust is the internet? *Nature*, *406*(6794), 353–354.

Valdes, J., Tarjan, R. E., & Lawler, E. L. (1982). The recognition of series-parallel digraphs. *SIAM Journal on Computing*, *11*(2), 298–313.

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

Walkup, D. W., & Wets, R. J.-B. (1967). Stochastic programs with recourse. *SIAM Journal on Applied Mathematics*, *15*(5), 1299–1314.

Wang, B., Tang, H., Guo, C., Xiu, Z., & Zhou, T. (2006). Optimization of network structure to random failures. *Physica A*, *368*(2), 607–614.

Watts, D. J. (2004). *Six Degrees: The Science of a Connected Age*. W. W. Norton and Company.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440–442.

Wollmer, R. (1964). Removing arcs from a network. *Operations Research*, *12*(6), 934–940.

Wollmer, R. (1980). Two-stage linear programming under uncertainty with 0-1 first stage variables. *Mathematical Programming*, *19*(1), 279–288.

Wood, R. K. (1993). Deterministic network interdiction. *Mathematical and Computer Modelling*, *17*(2), 1–18.

Yannakakis, M. (1978). Node-and-edge-deletion NP-complete problems. In *Proceedings of the* 10^{th} *Annual ACM Symposium on Theory of Computing*, (pp. 253–264). San Diego, CA.

Yuan, Y., & Sen, S. (2009). Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, *21*(3), 480–487.

Zhao, J., & Xu, K. (2009). Enhancing the robustness of scale-free networks. *Journal of Physics A: Mathematical and Theoretical*, *42*, 195003.

Zhou, T., Fu, Z.-Q., & Wang, B.-H. (2006). Epidemic dynamics on complex networks. *Progress in Natural Science*, *16*(5), 452–457.

BIOGRAPHICAL SKETCH

Siqian Shen was born in Sanming, Fujian in China in 1985. She is the only child of Shen, Jiankang and Cao, Ying. Siqian earned her Bachelor degree in Industrial Engineering from Tsinghua University, Beijing in 2007. In her sixth semester (spring 2006) in Tsinghua, she was an exchange student in the Department of Industrial Engineering and Management at Aalto University in Finland. Siqian joined the Department of Industrial and Systems Engineering at the University of Florida (UF) in August 2007, starting her doctoral study under the guidance of Dr. J. Cole Smith. She earned her Doctor of Philosophy in Industrial and Systems Engineering in August 2011. Following graduation, she joins the Department of Industrial & Operations Engineering at the University of Michigan as a faculty member.

Siqian was named one of the two runners-up of the 2010 INFORMS Computing Society Student Prize for the best paper on computing and Operations Research by a student author. She was also one of the recipients of the 2008, 2009, 2010, 2011 UF Outstanding Academic Achievements Award, the Graduate Student Council Travel Grants, the 2010 UF I-cubed Graduate Student Interdisciplinary Research Grant, and the 2010 Mixed Integer Programming Workshop Student Travel Award. She was awarded the 2010 Chinese Government Award for Outstanding Self-financed Students Abroad, as well as the 2011 UF Industrial and Systems Engineering Graduate Student Research Award.