NETWORK INTERDICTION AND STOCHASTIC INTEGER PROGRAMMING

edited by
DAVID L. WOODRUFF

KLUWER ACADEMIC PUBLISHERS

NETWORK INTERDICTION AND STOCHASTIC INTEGER PROGRAMMING

OPERATIONS RESEARCH/COMPUTER SCIENCE INTERFACES SERIES

Series Editors

Professor Ramesh Sharda
Oklahoma State University

Prof. Dr. Stefan Voß
Technische Universität Braunschweig

Other published titles in the series:

Greenberg, Harvey J. / *A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE*

Greenberg, Harvey J. / *Modeling by Object-Driven Linear Elemental Relations: A Users Guide for MODLER*

Brown, Donald/Scherer, William T. / *Intelligent Scheduling Systems*

Nash, Stephen G./Sofer, Ariela / *The Impact of Emerging Technologies on Computer Science & Operations Research*

Barth, Peter / *Logic-Based 0-1 Constraint Programming*

Jones, Christopher V. / *Visualization and Optimization*

Barr, Richard S./ Helgason, Richard V./ Kennington, Jeffery L. / *Interfaces in Computer Science & Operations Research: Advances in Metaheuristics, Optimization, & Stochastic Modeling Technologies*

Ellacott, Stephen W./ Mason, John C./ Anderson, Iain J. / *Mathematics of Neural Networks: Models, Algorithms & Applications*

Woodruff, David L. / *Advances in Computational & Stochastic Optimization, Logic Programming, and Heuristic Search*

Klein, Robert / *Scheduling of Resource-Constrained Projects*

Bierwirth, Christian / *Adaptive Search and the Management of Logistics Systems*

Laguna, Manuel / González-Velarde, José Luis / *Computing Tools for Modeling, Optimization and Simulation*

Stilman, Boris / *Linguistic Geometry: From Search to Construction*

Sakawa, Masatoshi / *Genetic Algorithms and Fuzzy Multiobjective Optimization*

Ribeiro, Celso C./ Hansen, Pierre / *Essays and Surveys in Metaheuristics*

Holsapple, Clyde/ Jacob, Varghese / Rao, H. R. / *BUSINESS MODELLING: Multidisciplinary Approaches — Economics, Operational and Information Systems Perspectives*

Sleezer, Catherine M./ Wentling, Tim L./ Cude, Roger L. / *HUMAN RESOURCE DEVELOPMENT AND INFORMATION TECHNOLOGY: Making Global Connections*

Voß, Stefan, Woodruff, David / *Optimization Software Class Libraries*

Upadhyaya et al/ *MOBILE COMPUTING: Implementing Pervasive Information and Communications Technologies*

Reeves, Colin & Rowe, Jonathan/ *GENETIC ALGORITHMS—Principles and Perspectives: A Guide to GA Theory*

Bhargava, Hemant K. & Ye, Nong / *COMPUTATIONAL MODELING AND PROBLEM SOLVING IN THE NETWORKED WORLD: Interfaces in Computer Science & Operations Research*

NETWORK INTERDICTION AND STOCHASTIC INTEGER PROGRAMMING

Edited by
DAVID L. WOODRUFF
University of California, Davis

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 0-306-48109-X
Print ISBN: 1-4020-7302-X

Visit Kluwer Online at: http://kluweronline.com
and Kluwer's eBookstore at: http://ebooks.kluweronline.com

Contents

Preface

On March 15, 2002 we held a workshop on network interdiction and the more general problem of stochastic mixed integer programming at the University of California, Davis. Jesús De Loera and I co-chaired the event, which included presentations of on-going research and discussion. At the workshop, we decided to produce a volume of timely work on the topics. This volume is the result.

Each chapter represents state-of-the-art research and all of them were refereed by leading investigators in the respective fields. Problems associated with protecting and attacking computer, transportation, and social networks gain importance as the world becomes more dependent on interconnected systems. Optimization models that address the stochastic nature of these problems are an important part of the research agenda. This work relies on recent efforts to provide methods for addressing stochastic mixed integer programs. The book is organized with interdiction papers first and the stochastic programming papers in the second part. A nice overview of the papers is provided in the Foreward written by Roger Wets.

We are grateful to Roger Wets not only for providing us with a Foreward to this volume, but also for his leadership at UC Davis that essentially made the workshop possible. People often speak of seminal papers, but Roger has also provided us with a seminal presence. In addition, we want to thank Kevin Wood who came to Davis before the workshop and helped introduce us to problems in interdiction and introduced us to some of the research and researchers in the field. Finally, we wish to acknowledge the Air Force Office of Sponsored Research for their support of the workshop under grant F49620-01-0327.

DAVID L. WOODRUFF

Contributing Authors

Ahmet Balcioglu Naval Postgraduate School

Carl Burch College of St Benedict and St John's University

Robert Carr Sandia National Laboratories

William S. Charlton The University of Texas at Austin

Raymond Hemmecke University of California, Davis

Julia L. Higle The University of Arizona

Sven Krumke Konrad-Zuse-Zentrum

Guglielmo Lulli University of Rome "La Sapienza"

Madhav Marathe Los Alamos National Laboratory

David P. Morton The University of Texas at Austin

Lewis Ntaimo The University of Arizona

Feng Pan The University of Texas at Austin

Cynthia Phillips Sandia National Laboratories

Rüdiger Schultz Gerhard-Mercator University, Duisburg

Suvrajeet Sen University of Arizona

R. Kevin Wood Naval Postgraduate School

David L. Woodruff University of California, Davis

Foreword

It's certainly a sign of the vitality of the field of Stochastic Programming that in the week of March 11-15, 2002, two major workshops could be organized. Both of them with the participation of some of the leaders in the field and about than 10,000 km apart. The meeting in Laxenburg (Austria) concentrated on 'Dynamic Stochastic Optimization' whereas the Davis (California) meeting was devoted to 'Network Interdiction and Stochastic Integer Programming.' This volume is the proceedings of the Davis workshop.

Under the leadership of David Woodruff (Graduate School of Management) and Jesus De Loera (Department of Mathematics), research at the University of California, Davis in Stochastic Programming is now branching out in one of the most challenging areas: Stochastic Integer Programming. In fact, as suggested by the papers in this volume, Stochastic Integer Programming has become a major research theme in the Western United States with the non-negligible help of the Duisburg (Germany) connection.

A major component of David's research had been in manufacturing and his int ital interest in stochastic programming was motivated by the need to take into account the inherent uncertainty in the manufacturing process. How to deal with certain disjunctive variables in a stochastic programming model for oil exploration came up a little later, when Tore Jonsbrøaten from Stavanger (Norway) came on a post-doc visit to Davis. By then, David had a serious commitment to designing practical procedures for solving stochastic integer programs of various types. One direction that seemed promising was to work on (stochastic) network interdiction problems that had already been investigated just a few miles south of Davis at the Naval Postgraduate School in Monterey (California).

Jesus followed a more 'theoretical' path to come to the field. His initial interests were centered around a number of problems in Algebraic Geometry, but a post-doc year at Cornell University motivated him to also include Integer Programming and Combinatorial Optimization in

his research plans. It was difficult to ignore, one of the older member of his department pointing out that stochastic integer programs were a super-abundant source of mathematically challenging problems and that progress in this area was bound to have significant impact in practice. This lead Jesus to visit Duisburg were Raymond Hemmecke was finishing his Ph.D. thesis under the leadership of Rüdiger Schultz, one of the leaders in the field. Raymond is now a Visiting Research Assistant Professor in Davis and, among others, provides a bridge between David's and Jesus' teams.

The Network Interdiction Problem has a wide variety of applications both in the transportation area, but more recently and very prominently, in the communications area. An elementary version of this problem would read: Given a network carrying flow from a source node s to a destination node t, find those 'operations' that will reduce as much as possible the maximum flow that can be carried from s to t; an operation could be destruction of certain nodes or reducing the capacity on certain arcs, possibly to zero. In view of the max-flow/min-cut theorem, the problem can roughly be viewed as one of finding the operations that will affect the topology of the network so as to minimize the min-cut. The problem is NP-hard! The analysis provides both strategies that would minimize flow between origin and destination, and could help in the design of networks that have high level reliability properties. In practice, however, network interdiction problems come with uncertainties or various types: characteristics of the network itself, actual results of interdiction actions, etc. On the other hand, because of the network structure, such problems are relatively simple (an oxymoron?) stochastic integer programs, and possibly more amenable to efficient, implementable solution procedures.

The two papers, 'Enumerating near-min $s-t$ cuts' by Ahmet Balcioglu and R. Kevin Wood, and 'A decomposition-based pseudo approximation algorithm for network flow inhibition' by Carl Burch, Robert Carr, Sven Krumke, Ladhav Marathe, Cynthia Phillips and Eric Sundberg, deal with deterministic versions of the network interdiction problem. The first one of these is devoted to finding approximating solutions that would simplify the computational complexity, The second one the introduced a new procedure to deal with non-planar networks again with the goal of exploiting approximate solutions; the model includes a budgetary constraint that renders it more directly applicable in a number of significant applications.

The two papers, 'A stochastic program for interdicting smuggled nuclear material' by Feng Pan, William Charlton and David Morton, and 'Interdicting stochastic networks with binary interdiction effort' by Ray-

mond Hemmecke, Rüdiger Schultz and David Woodruff, deal with two stochastic versions of the Network Interdiction Problem. In the first one of these, an interdictor is to install sensors on a network so as to maximize the probability of detecting smuggled nuclear material. The problem is formulated as a stochastic mixed-integer program with recourse. The second paper also deals with the interdiction of undesirable materials or information, but the configuration of the network can only be conjectured. The problem can be formulated as a stochastic programming problem but with some special features that can be exploited effectively in the design of solution procedures.

The two papers, 'Stochastic batch-sizing problems: models and algorithms' by Guglielmo Lulli and Suvrajeet Sen, and 'A summary and illustration of disjunctive decomposition with set convexification' by Suvrajeet Sen, Julia Higle and Lewis Ntaimo deal with two particular classes of Stochastic Integer Programming problems. The first one of these provides a unified treatment of the batch-sizing problem that, in particular, considers trade-offs between costs and reliability. The second paper reviews and illustrates a general, and promising, procedure for two-stage stochastic mixed-integer programs: the Disjunctive Decomposition algorithm first proposed by Higle and Sen.

Roger J-B Wets
Truckee, California
August 26, 2002

This page intentionally left blank

Chapter 1

A STOCHASTIC PROGRAM FOR INTERDICTING SMUGGLED NUCLEAR MATERIAL

Feng Pan
Graduate Program in Operations Research
Mechanical Engineering Department
The University of Texas at Austin
Austin, TX 78712, USA
pan@mail.utexas.edu

William S. Charlton
Nuclear Engineering Teaching Laboratory
Mechanical Engineering Department
The University of Texas at Austin
Austin, TX 78758, USA
charlton@mail.utexas.edu

David P. Morton
Graduate Program in Operations Research
Mechanical Engineering Department
The University of Texas at Austin
Austin, TX 78712, USA
morton@mail.utexas.edu

Abstract This paper describes a stochastic network interdiction model for identifying locations for installing detectors sensitive to nuclear material. A nuclear material smuggler selects a path through a transportation network that maximizes the probability of avoiding detection. An interdictor installs sensors to minimize that maximum probability. This problem is formulated as a bi-level stochastic mixed-integer program. The program is stochastic because the evader's origin and destination

are unknown at the time the detectors are installed. The model is reformulated as a two-stage stochastic mixed-integer program with recourse and is shown to be strongly NP-Hard. We describe an application of our model to help strengthen the overall capability of preventing the illicit trafficking of nuclear materials.

Keywords: network interdiction, stochastic programming

Introduction

Smuggling of nuclear material, equipment, and technology has become a threat to international security. One method to combat this threat entails placing radiation sensors at customs checkpoints to deter the smuggling of nuclear material. This paper describes a general stochastic network interdiction model that can be used to select sites to install sensors to minimize the probability a smuggler can travel through a transportation network undetected. We reformulate the model as a computationally tractable stochastic mixed-integer program and show that the related decision problem is strongly NP-Complete.

We model two adversaries, an interdictor and an evader, and an underlying network $G(N, A)$ on which the evader travels. In the *deterministic* version of the model, the evader starts at a specified source node $s \in N$ and wishes to reach a specified terminal node $t \in N$. The model is deterministic in that this origin-destination pair is known. If the interdictor has not installed a *sensor* on arc $(i, j) \in A$, then the probability that the evader can traverse (i, j) undetected is p_{ij}, and this probability is $q_{ij} < p_{ij}$ if the interdictor has installed a detector on (i, j). (We use the terms sensor and detector interchangeably.) The events of the evader being detected on distinct arcs are assumed to be mutually independent. The evader chooses a path from s to t so as to maximize the probability of traversing the network without being detected. With limited resources, the interdictor must select arcs on which to install detectors in order to minimize the probability the evader travels from s to t undetected.

Our *stochastic* network interdiction model differs from the above description only in that the (s, t) pair for the evader is unknown when the interdictor must select sites for installing sensors. However, the origin-destination pair (s, t) is assumed to be governed by a known probability mass function, $p^\omega = P\{(s, t) = (s^\omega, t^\omega)\}$, $\omega \in \Omega$. The interdictor's goal is to minimize the probability that the evader traverses the network undetected, i.e., the objective function is a sum of (conditional) evasion probabilities, each weighted by p^ω, over the population of possible evaders. We call this problem SNIP, for stochastic network interdiction

problem. The timing of decisions and realizations in SNIP is key: First, the interdictor installs sensors. Then, a random origin-destination pair for the evader is revealed and the evader selects an s^{ω}-t^{ω} path to maximize the probability of avoiding detection. The evader selects this path knowing the locations of the detectors and knowing the evasion probabilities p_{ij} and q_{ij} for all $(i, j) \in A$. An evader can be caught by indigenous law enforcement without detection equipment and so $p_{ij} < 1$. (To date, nuclear smuggling attempts that have been stopped have been by this means.)

In Section 1 we formulate SNIP as a bi-level stochastic mixed-integer program, using a "min-max" structure because the interdictor is minimizing the evader's maximum evasion probability. We then develop an equivalent mixed-integer program (MIP). The need for the MIP formulation is justified by the fact that SNIP or, more precisely, the related decision problem, SNIP-DECISION, is strongly NP-Complete (Section 2). In Section 3 we consider an important special case of SNIP that arises when sensors can only be installed at border crossings of a single country. In this special case, the underlying network, and the associated MIP, can be simplified. Our SNIP model requires the following parameters: the probability a smuggler can traverse a physical transportation arc undetected, the probability that sensitive nuclear material will be detected by an installed sensor, and the probability a smuggler steals material from a particular origin and wants to travel to a specific destination. A discussion of the methods by which we estimate these parameters is beyond the scope of this paper.

The study of network interdiction models in operations research began in the 1970s (see [IW01b] for a richer discussion of the history of interdiction, dating from antiquity). During the Vietnam War, [MM70] and [GMT71] developed deterministic mathematical programs to disrupt flow of enemy troops and materiel. The problem of maximizing an adversary's shortest path is considered in [FH77] and [Gol78]. A closely related problem concerns maximizing the longest path in an adversary's PERT network [Ree94]. In these linear programs, the interdictor can continuously increase the length of an arc, subject to a budget constraint. A discrete version of maximizing the shortest path removes an interdicted arc from the network, and when the budget constraint is simply a cardinality constraint, this is known as the *k*-most-vital arcs problem [CS82, MMG89]. The related decision problem is strongly NP-Complete [BGV89, BKS95]. Generalizations of the *k*-most-vital arcs problem, and associated solution procedures, are considered in [IW01a]. The interdiction problem of removing arcs to minimize flow in an adversary's maximum-flow network is considered in [Woo93], where the

(decision) problem is shown to be strongly NP-Complete, and integer programming formulations are developed. See [Wol64, WW94] for game-theoretic approaches to related network interdiction problems, [CL95] for an interdiction model on a minimum-cost-flow network, and [IW01b] for interdiction models of more general systems.

All of the interdiction models described above are deterministic in nature in the following senses: First, the arc lengths in the shortest-path and PERT problems, and the arc capacities in the maximum-flow problems, are known with certainty. Second, when increasing the length of an arc in the former problems or when removing or decreasing the capacity of an arc in the latter problem, these modifications occur in a deterministic manner, i.e., with certainty. In [CMW98] the work of [Woo93] on interdicting a maximum-flow network is generalized to allow for random arc capacities and random interdiction successes. In these models, the interdictor does not know whether an interdiction attempt will successfully remove an arc and the interdictor does not know the true capacity of some arcs. However, the adversary does know the realizations of these random variables and maximizes flow in the residual network. The interdictor's goal is to minimize the expected value of this maximum flow.

The SNIP model we develop here is analogous. The interdictor must place detectors on the network without knowing the evader's origin-destination pair. Then, an (s^ω, t^ω) pair is realized. Of course, the evader knows (s^ω, t^ω) and selects a path that maximizes the probability of avoiding detection. The evader's optimization problem is known as the maximum-reliability path problem and, even though there are probability values on the arcs (p_{ij}'s and q_{ij}'s), this problem can be solved as a deterministic shortest-path problem (e.g., exercise 4.39 of [AMO93]). The interdictor's objective is to minimize the expected value of this maximized conditional probability, i.e., to minimize the probability that a smuggler avoids detection.

1. A Stochastic Network Interdiction Model

We now more formally define the SNIP model.

Network and Sets:

$G(N, A)$	directed network with node set N and arc set A
$FS(i)$	set of arcs leaving node i
$RS(i)$	set of arcs entering node i
$AD \subset A$	set of arcs where a detector can be installed

Data:

b	total budget for installing detectors
c_{ij}	cost of installing a detector on arc $(i, j) \in AD$
p_{ij}	probability evader can traverse arc (i, j) undetected when no detector is installed
q_{ij}	probability evader can traverse arc (i, j) undetected when a detector is installed

Random Elements:

(s^ω, t^ω)	realization of random origin-destination pair
$\omega \in \Omega$	sample point and sample space
p^ω	probability mass function

Interdictor's Decision Variables:

x_{ij}	takes value 1 if a detector is installed on arc $(i, j) \in AD$ and 0 otherwise

Evader's Decision Variables:

y_{ij}	takes positive value only if evader traverses arc (i, j) and no detector is installed on that arc
z_{ij}	takes positive value only if evader traverses arc (i, j) and a detector is installed on that arc

Boundary Conditions:

$x_{ij} \equiv 0$	$(i, j) \notin AD$
$z_{ij} \equiv 0$	$(i, j) \notin AD$

Formulation:

$$\min_{x \in X} \sum_{\omega \in \Omega} p^\omega h(x, (s^\omega, t^\omega)), \tag{1.1}$$

where

$$X = \left\{ x \, : \sum_{(i,j) \in AD} c_{ij} x_{ij} \leq b, \; x_{ij} \in \{0, 1\}, \; (i, j) \in AD \right\},$$

and where

$$h(x,(s^\omega,t^\omega)) =$$

$$\max_{y,z} \quad y_{t^\omega} \tag{1.2a}$$

$$\text{s.t.} \quad \sum_{(s^\omega,j)\in FS(s^\omega)} (y_{s^\omega j} + z_{s^\omega j}) = 1 \tag{1.2b}$$

$$\sum_{(i,j)\in FS(i)} (y_{ij} + z_{ij}) - \sum_{(j,i)\in RS(i)} (p_{ji}y_{ji} + q_{ji}z_{ji}) = 0 \quad \forall\, i \in N \setminus \{s^\omega, t^\omega\} \tag{1.2c}$$

$$y_{t^\omega} - \sum_{(j,t^\omega)\in RS(t^\omega)} (p_{jt^\omega}y_{jt^\omega} + q_{jt^\omega}z_{jt^\omega}) = 0 \tag{1.2d}$$

$$0 \le y_{ij} \le 1 - x_{ij} \qquad \forall\,(i,j) \in AD \tag{1.2e}$$

$$0 \le z_{ij} \le x_{ij} \qquad \forall\,(i,j) \in AD. \tag{1.2f}$$

The conditional probability a smuggler avoids detection, given (s^ω, t^ω), is $h(x,(s^\omega,t^\omega))$ as defined in (1.2). The objective function in (1.1) is the expected value of this evasion probability, where the expectation is taken over all possible origin-destination pairs. The set of feasible detector installation locations defined through X is governed by a budget constraint and binary restrictions on x.

Each link in the network on which a detector can be placed may actually be viewed as two arcs in parallel. If a detector is installed, i.e., $x_{ij} = 1$, then flow may occur only on the "detector" arc, z_{ij}. Conversely, if no detector is installed then flow can only occur on the "no detector" arc, y_{ij}. A unit of flow on arc (i,j) is multiplied by that arc's gain (either p_{ij} or q_{ij}). So, if P_{s^ω,t^ω} is a path from s^ω to t^ω then

$$y_{t^\omega} = \prod_{(i,j)\in P_{s^\omega,t^\omega}} [p_{ij}(1 - x_{ij}) + q_{ij}x_{ij}] \tag{1.3}$$

is the probability that an evader can travel from s^ω to t^ω on P_{s^ω,t^ω} without being detected. The evader's goal is to select a path P_{s^ω,t^ω} that maximizes y_{t^ω}. The evader's subproblem (1.2) accomplishes this by forcing one unit of flow out of s^ω in (1.2b), enforcing flow conservation at all intermediate nodes in (1.2c), defining the flow that reaches t^ω as y_{t^ω} in (1.2d) and maximizing that value in (1.2a). Flow is forced on the appropriate arc, and incurs the associated gain (actually, loss), by the interdictor's decision variable x_{ij} in constraints (1.2e) and (1.2f).

The timing of decisions and realizations of the uncertainties in (1.1) and (1.2) is as follows: First, the interdictor selects sites for installing

sensors. When making this decision, the interdictor knows: (i) the network topology $G(N, A)$, (ii) the indigenous detection probability on each arc p_{ij}, $(i,j) \in A$, (iii) the value the detection probability would be if a sensor were installed q_{ij}, $(i,j) \in AD$, (iv) the budget constraint, (v) the probability distribution governing the random (s,t) pair, and (vi) the method by which the evader will select a path. Next, an (s^ω, t^ω) realization is revealed, and the evader selects an s^ω-t^ω path that maximizes the probability of not being detected. The evader selects this path knowing: (i), (ii) and (iii) as well as where detectors are installed.

Assuming the smuggler solves an optimization model to select an s^ω-t^ω path is a behavioral assumption. We note that even in cases when this assumption may not be valid, the optimal value of (1.1) still provides a potentially useful *pessimistic* prediction of the evasion probability.

The SNIP model (1.1)/(1.2) is a bi-level stochastic mixed-integer program. In bi-level programs (e.g., [BM90, BA93]) each player has an objective function, and these can differ because the players' motives differ. In our case, the objective function is the same for both players, but the interdictor is trying to minimize that function and the evader is trying to maximize it. Restated, the problem is formulated with a nested "min-max" structure, and so it is not possible to solve in this form as a single large-scale mathematical program. The natural way to attempt to circumvent this difficulty (e.g., [FH77, Woo93]) is to take the dual of the linear programming subproblem (1.2) so that the problem is expressed in a nested "min-min" form. We could then construct a single optimization model in which we simultaneously minimize over the interdictor's decision x and the dual-variable decisions of the evader under each scenario, $\omega \in \Omega$. The difficulty with this is that there are nonlinear terms involving x and the dual variables associated with constraints (1.2e) and (1.2f) and the prospects for solving realistically-sized instances of the resulting stochastic nonlinear nonconvex mixed-integer program are not good. Instead we will employ the exact-penalty result of Lemma 1, which is adapted from [MW99, Lemma 2], in order to reformulate (1.2).

Lemma 1 *Consider the following linear program*

$$\begin{array}{rlll} z_1^* = & \max\limits_{x \geq 0} & cx & \\ & \text{s.t.} & Ax = b & : \pi \\ & & x \leq u & : \gamma, \end{array} \tag{1.4}$$

where $A \in \Re^{m \times n}$, the remaining vectors are dimensioned to conform and π and γ are dual variables. Assume (1.4) has a finite optimal solution,

(π^*, γ^*) is an optimal dual vector, and consider

$$\begin{aligned} z_2^* = \max_{x \geq 0} \quad & cx - \gamma'(x-u)^+ \\ \text{s.t.} \quad & Ax = b, \end{aligned} \tag{1.5}$$

where $(x-u)^+ = \max(x-u, 0)$ and $\gamma' \in \Re^n$. If $\gamma' \geq \gamma^*$ then $z_1^* = z_2^*$.

The following theorem uses Lemma 1 to establish an equivalent expression to (1.2) for $h(x, (s^\omega, t^\omega))$.

Theorem 1 *Assume that G has an s^ω-t^ω path $\forall \omega \in \Omega$, $0 \leq p_{ij} \leq 1 \; \forall (i,j) \in A$, and $0 \leq q_{ij} \leq 1 \; \forall (i,j) \in AD$. Then, for all $x \in X$ and $\omega \in \Omega$, $h(x, (s^\omega, t^\omega))$ is the optimal value of the following linear program*

$$\begin{aligned} \min_{\pi} \quad & \pi_{s^\omega} \\ \text{s.t.} \quad & \pi_i - p_{ij}\pi_j \geq 0 && \forall (i,j) \in A \setminus AD && \text{(1.6a)} \\ & \pi_i - p_{ij}\pi_j \geq -x_{ij} && \forall (i,j) \in AD && \text{(1.6b)} \\ & \pi_i - q_{ij}\pi_j \geq x_{ij} - 1 && \forall (i,j) \in AD && \text{(1.6c)} \\ & \pi_{t^\omega} = 1. \end{aligned}$$

Proof: Let $\omega \in \Omega$ and $x \in X$. G has an s^ω-t^ω path and hence (1.2) is feasible and has a finite optimal solution. Let λ_{ij}^* and γ_{ij}^*, $(i,j) \in AD$, be optimal dual variables for constraints (1.2e) and (1.2f), respectively. These dual variables are bounded above by one because the network gains, p_{ij}, $(i,j) \in A$, and q_{ij}, $(i,j) \in AD$, are at most unity and hence an increase in the capacity of an arc by ϵ can increase the flow exiting that arc by no more than ϵ and therefore contribute at most ϵ to the flow reaching t^ω. So, employing Lemma 1 we can conclude that $h(x, (s^\omega, t^\omega))$ is the optimal value of

$$\begin{aligned} \max_{y \geq 0, z \geq 0} \quad & y_{t^\omega} - \sum_{(i,j) \in AD} \left[(y_{ij} - (1 - x_{ij}))^+ + (z_{ij} - x_{ij})^+ \right] \\ \text{s.t.} \quad & \sum_{(s^\omega, j) \in FS(s^\omega)} (y_{s^\omega j} + z_{s^\omega j}) = 1 \\ & \sum_{(i,j) \in FS(i)} (y_{ij} + z_{ij}) - \\ & \sum_{(j,i) \in RS(i)} (p_{ji} y_{ji} + q_{ji} z_{ji}) = 0 \quad \forall i \in N \setminus \{s^\omega, t^\omega\} \\ & y_{t^\omega} - \sum_{(j, t^\omega) \in RS(t^\omega)} (p_{jt^\omega} y_{jt^\omega} + q_{jt^\omega} z_{jt^\omega}) = 0. \end{aligned} \tag{1.7}$$

Because of the binary nature of $x \in X$, we have $(y_{ij} - (1 - x_{ij}))^+ = x_{ij}y_{ij}$ and $(z_{ij} - x_{ij})^+ = (1 - x_{ij})z_{ij}$ for y_{ij} and z_{ij}, $(i,j) \in AD$, satisfying the constraints of (1.7). Making these substitutions in the objective function of (1.7) and taking the dual of the resulting linear program yields (1.6). □

The expression for $h(x,(s^\omega,t^\omega))$ in (1.6) has the following interpretation: The dual variable π_i is the conditional probability of traveling from node i to destination t^ω undetected, given that the evader has reached node i undetected. Constraints (1.6a)-(1.6c), coupled with minimizing the objective function, ensure the correct computation of π_{s^ω}, i.e., the probability of traversing the network from s^ω to t^ω undetected. Constraints (1.6a)-(1.6c) are tight for (i,j) on the optimal s^ω-t^ω path. When $x_{ij} = 1$, i.e., a detector is installed on (i,j), the indigenous-arc constraint (1.6b) is vacuous and when $x_{ij} = 0$ the detector-arc constraint (1.6c) is vacuous. The unit multiplicative coefficients on the right-hand sides of (1.6b) and (1.6c) can be tightened to improve the subsequent MIP formulation. For example, (1.6b) can be rewritten $\pi_i - p_{ij}\pi_j \geq -\alpha_{ij}x_{ij}$, provided the coefficient α_{ij} satisfies $\alpha_{ij} \geq p_{ij}\pi_j$. While $\alpha_{ij} = 1$ is a valid bound, this can be decreased by bounding π_j, the conditional probability of traversing from j to t^ω undetected. This can be bounded using the sensor-free network or, better, in a network where certain necessary detector locations have been determined, either by other logical arguments or fixed within the branch-and-bound tree. (For notational simplicity we will not modify the formulation in this way.)

The value of Theorem 1 is that we can now express our original nested "min-max" formulation of SNIP, i.e., (1.1)/(1.2), as the following two-stage stochastic mixed-integer linear program

$$\begin{aligned}
\min_{x,\pi} \quad & \sum_{\omega\in\Omega} p^\omega \pi^\omega_{s^\omega} \\
\text{s.t.} \quad & x \in X \\
& \pi^\omega_i - p_{ij}\pi^\omega_j \geq 0 && \forall\,(i,j) \in A \setminus AD, \omega \in \Omega \\
& \pi^\omega_i - p_{ij}\pi^\omega_j + x_{ij} \geq 0 && \forall\,(i,j) \in AD, \omega \in \Omega \\
& \pi^\omega_i - q_{ij}\pi^\omega_j + (1 - x_{ij}) \geq 0 && \forall\,(i,j) \in AD, \omega \in \Omega \\
& \pi_{t^\omega} = 1 && \forall\,\omega \in \Omega.
\end{aligned} \tag{1.8}$$

As indicated above, the evader's optimization problem is known as the maximum-reliability path problem and for each (s^ω,t^ω) pair this can be formulated as a shortest-path problem. Instead, we used what may appear to be a less natural generalized network-flow model. The reason for

this is that using the shortest-path reformulation via, essentially, applying "$\exp(\ln(\cdot))$" to (1.3) leads to a nonlinear mixed-integer program (at least when $|\Omega| > 1$). In contrast, the generalized network-flow approach allows for the preservation of linearity in the mixed-integer program.

SNIP, as formulated in (1.8), is "simply" a mixed-integer linear program that one can attempt to solve using commercially-available optimization software, and this is what we do in Section 3. That said, we note the following with respect to the potential for using decomposition schemes to solve SNIP. Formulation (1.1) minimizes $E_\omega h(x, (s^\omega, t^\omega))$ with respect to x where $h(x, (s^\omega, t^\omega))$, defined by (1.2), is a maximizing linear program with x appearing on the right-hand side of constraints (1.2e) and (1.2f). This implies that $h(x, (s^\omega, t^\omega))$, and hence $E_\omega h(x, (s^\omega, t^\omega))$, is a concave function over the convex hull of X. This does not bode well for employing an outer-approximation cutting-plane scheme like the L-Shaped method [VW69]. Laporte and Louveaux [LL93] have developed variants of the L-Shaped method that are valid for such nonconvex forms of $h(x, (s^\omega, t^\omega))$, but they require cutting-planes that are tight at a specific (binary) value of $x = \hat{x}$ and drop to an *a priori* lower bound at all other (binary) values of $x \in X$. Theorem 1 shows that (1.2) and (1.6) are equivalent formulations in that they give the same objective value, i.e., $h(x, (s^\omega, t^\omega))$, for $x \in X$ and $\omega \in \Omega$. Interestingly, while $h(x, (s^\omega, t^\omega))$ as defined by (1.2) is concave over the convex hull of X, $h(x, (s^\omega, t^\omega))$ as defined by (1.6) is convex over the convex hull of X because it is a minimizing linear program with x in the right-hand side. This is possible because the objective function values of these two linear programs are only ensured to be equal when $x \in X$, i.e., when x takes on binary values (and satisfies the budget constraint). As a result, SNIP formulated as (1.1)/(1.6), or equivalently (1.8), is amenable to solution by the L-Shaped method in which the master program has binary restrictions on x and the subproblem separates into one generalized network-flow subproblem for each $\omega \in \Omega$ [LL93, Wol80]. When the underlying network is large and/or the number of scenarios is large, this approach may lead to significant computational savings over trying to solve (1.8) directly as a large-scale mixed-integer program.

2. Complexity

In this section we establish that SNIP is strongly NP-Hard by showing the related decision problem, SNIP-DECISION, is strongly NP-Complete. Consider the following decision problem:

PROBLEM: SNIP-DECISION

GIVEN: A directed graph $G(N, A)$, a finite set of scenarios with probability mass function p^ω, $\omega \in \Omega$, origin-destination node pairs (s^ω, t^ω), $\omega \in \Omega$, and reliability weights on each arc $0 \leq p_{ij} \leq 1, (i,j) \in A$, and $0 \leq q_{ij} \leq 1, (i,j) \in A$, nonnegative costs $c_{ij}, (i,j) \in A$ to convert each arc reliability from p_{ij} to q_{ij}, nonnegative budget b, and value v.
QUESTION: Does there exist a subset $K \subset A$ with $\sum_{(i,j)\in K} c_{ij} \leq b$ such that

$$\sum_{\omega\in\Omega} p^\omega \left(\prod_{(i,j)\in P^\omega \cap K} q_{ij} \prod_{(i,j)\in P^\omega \cap (A\setminus K)} p_{ij} \right) \leq v,$$

where P^ω is the maximum-reliability path from s^ω to t^ω on G with arc reliabilities $p_{ij}, (i,j) \in A \setminus K$ and $q_{ij}, (i,j) \in K$?

We now state the k-most-vital arcs problem (MVAP-DECISION) alluded to above, which is known to be strongly NP-Complete [BKS95, BGV89]. Here, we use the budget b in place of k.
PROBLEM: MVAP-DECISION
GIVEN: A directed graph $G(N, A)$, origin-destination node pair (s,t), unit arc lengths, nonnegative budget b, and value v'.
QUESTION: Does there exist a subset $K \subset A$ with $|K| = b$ such that removing these arcs results in a shortest path from s to t on $G(N, A\backslash K)$ whose length is at least v'?

Consider the following special case of SNIP-DECISION: Let $|\Omega| = 1$, the installation costs be unit-valued $c_{ij} = 1, (i,j) \in A$, the indigenous arcs have constant reliability $p_{ij} = p, (i,j) \in A$, $0 < p < 1$, and the sensors have perfect detection capability $q_{ij} = 0, (i,j) \in A$. This special case of SNIP-DECISION is equivalent to MVAP-DECISION because, as indicated above, computation of a maximum-reliability s-t path is equivalent to a shortest-path problem via a log-transformation and multiplying by -1. (The constant arc lengths, $-\log p$, can be factored out.) The answer to the SNIP-DECISION question is affirmative for value v if and only if the answer to the MVAP-DECISION question is affirmative for value $v' = \lceil \log v / \log p \rceil$, where the ceiling operator $\lceil \cdot \rceil$ gives the smallest integer greater than or equal to its argument. Clearly, both SNIP and MVAP are in NP. As a result, we have the following theorem.

Theorem 2 SNIP-DECISION *is strongly NP-Complete.*

Theorem 2 shows that SNIP is strongly NP-Hard and provides justification for our approach to formulating and solving SNIP by mixed-integer programming.

3. Application to Smuggling Out of a Single Country

One of the models we have developed is restricted to proposing detector locations at customs checkpoints leaving a single country. When potential sensor locations are limited in this manner, our SNIP model (1.8) can be simplified as described in this section. We also describe our computational experience with a test instance of this model.

Our underlying network model has four basic location entities: *facilities* from which sensitive nuclear material could be stolen, geographic *regions, destinations* where a nuclear smuggler may desire to go, and *customs checkpoints* where sensors can be installed. The nominal transportation network has a node representing each of these locations (some aggregation is possible as we describe below). These nodes are linked by arcs representing transport by surface roads, railroads, airline flights, ship transport, etc. A sample point $\omega \in \Omega$ specifies a facility-destination pair. In the SNIP model of Section 1, sensors are installed on arcs and this can be modeled by splitting each customs-checkpoint node into two nodes with an associated arc representing travel through the checkpoint.

The key to simplifying the formulation, when the customs checkpoints of a single country are under consideration, is that on each potential smuggling route (i.e., each possible s^ω-t^ω path) there is exactly one arc on which the smuggler could encounter a sensor. We formalize this in the following manner: Let $\mathcal{P}^\omega$ be the set of all paths for origin-destination pair (s^ω, t^ω). (These paths need not be enumerated.) Then, in our BiSNIP model (bipartite SNIP, for reasons soon apparent) we assume that each path in $\mathcal{P}^\omega$ contains exactly one arc in AD, i.e., each path has exactly one arc that is a candidate to receive a sensor. Let $AC^\omega = \{(i,j) : (i,j) \in AD, (i,j) \in \mathcal{P}^\omega\}$ be all such checkpoint arcs for $\omega \in \Omega$. The evader, under scenario ω, must select an s^ω-t^ω path, but this now depends on the sensor locations in a much simpler way than in the general model. For each ω we perform a preprocessing step to compute the value of the maximum-reliability path from s^ω to the tail of each checkpoint arc and the value of the maximum-reliability path from the head of each checkpoint arc to t^ω. Call the product of these two probabilities r_c^ω, $c = (i,j) \in AC^\omega$. Then, the value of the maximum-reliability path under scenario ω is

$$h(x, (s^\omega, t^\omega)) = \max_{c \in AC^\omega} \{r_c^\omega p_c (1 - x_c), r_c^\omega q_c x_c\}. \tag{1.9}$$

By linearizing (1.9), we can express BiSNIP as the following stochastic mixed-integer program with simple recourse

$$\min_{x,\theta} \quad \sum_{\omega\in\Omega} p^{\omega}\theta^{\omega} \tag{1.10a}$$

$$\text{s.t.} \quad x \in X \tag{1.10b}$$

$$\theta^{\omega} \geq r_c^{\omega} p_c (1 - x_c) \qquad \forall c \in AC^{\omega} \quad \forall \omega \in \Omega \tag{1.10c}$$

$$\theta^{\omega} \geq r_c^{\omega} q_c x_c \qquad \forall c \in AC^{\omega} \quad \forall \omega \in \Omega. \tag{1.10d}$$

BiSNIP (1.10) may be visualized on an underlying bipartite network with node sets Ω and $\cup_{\omega\in\Omega} AC^{\omega}$. Arcs (ω, c) link each facility-destination pair, $\omega \in \Omega$, with its possible intermediate checkpoints, $c \in AC^{\omega}$. Excluding the possibility of being detected at the checkpoint, r_c^{ω} is the evader's probability of traveling from ω's facility to ω's destination, via c, undetected. This reliability is multiplied by q_c or p_c depending on whether or not we install a detector at c.

The specific problem we consider has 85 facilities, 79 geographic regions, 79 customs checkpoints and 9 destinations. (The two 79s are coincidental.) There are 30 regions with checkpoints and 34 regions with facilities. Facilities within a region are aggregated. Maintaining checkpoint integrity is important, so they are not aggregated. After facility aggregation, and allowing all possible facility-destination combinations, the model has $|\Omega| = 34 \cdot 9 = 306$ scenarios. The aggregated network has $|N| = 246$ nodes (79 regions plus $2 \cdot 79$ checkpoints plus 9 destinations) and $|A| = 774$ arcs (555 region-region plus 79 region-checkpoint plus 140 checkpoint-destination). As a result, SNIP model (1.8) has 75,355 decision variables (79 binary first stage variables plus $246 \cdot 306$ second stage variables) and 261,019 structural constraints (1 budget constraint plus $(774 + 79) \cdot 306$ dual network-flow constraints). In contrast, the simplified BiSNIP formulation (1.10) has 385 decision variables, 79 of which are binary and, nominally, 9453 structural constraints. There are $2|\cup_{\omega\in\Omega} AC^{\omega}|$ structural constraints of the form (1.10c)-(1.10d), but further simplifications are possible. First, we can replace all of the constraints of form (1.10d) with simple lower bounds $\theta^{\omega} \geq \max_{c\in AC^{\omega}} r_c^{\omega} q_c$, $\omega \in \Omega$. Second, for each $\omega \in \Omega$, any constraints of the form (1.10c) with $r_c^{\omega} p_c < \max_{c\in AC^{\omega}} r_c^{\omega} q_c$ can be eliminated. This reduces the number of structural constraints from 9453 to 4720, for our data. Performing the reduction from SNIP to BiSNIP requires finding the maximum-reliability path from each facility to each customs site and from each customs site to each destination. In our model the latter is trivial as each permissible customs site-destination combination is represented by a single arc. These $34 \cdot 79 = 2686$ shortest paths are computed in about 3.5 seconds on a 1.7 GHz, Dell Xeon dual-processor machine with 2 Gb of memory. (All computations reported here are on this computer.)

One challenge we face in providing decision support is maintaining an appropriate level of consistency in our recommendations for locating sensors. For example, after solving (1.10) with budget b (denote the model BiSNIP(b)) and recommending $x^*(b)$, we may be asked to re-solve the model with budget $b' > b$ and obtain $x^*(b')$. The solutions $x^*(b')$ and $x^*(b)$ can be dramatically different, and this can be disconcerting to decision-makers. Sometimes there are compelling reasons for such differences. For example, with $b' > b$ we may be able to redirect resources to now shutdown a key part of the network that we previously had inadequate resources to control. However, more frequently, differences in solutions are unnecessary in the sense that there are multiple optimal, or near-optimal, solutions to BiSNIP(b') and by selecting an appropriate one we can avoid drastic changes. The need to obtain so-called *persistent* solutions is common in optimization and we adopt a technique for doing so described in [BDW97]. Given that $\hat{x}$ has already been announced, we modify (1.10) by adding a term to the objective function, "$+\rho \sum_{c \in AD} |x_c - \hat{x}_c|$," where $\rho \geq 0$. When ρ is small this places a mild penalty on deviations from $\hat{x}$ so that most of the weight is on the original objective function.

We assume $c_{ij} = 1$, for all $(i,j) \in AD$, and solve our test problem for all possible values of the budget b, ranging from 0 up to 79, and for four values of ρ. The computational effort to solve representative instances of (1.10) using GAMS/CPLEX [BKMR98, CPL01] are shown in Table 1.1. All MIPs are solved to within a relative tolerance of 10^{-4}. Table 1.1 shows an additional benefit of the persistence approach, namely that run times may be shortened (see also [BDW97]).

Figures 1.1 and 1.2 represent solutions from sequentially solving BiSNIP for budget levels ranging from 0 to 79 with $\rho = 0, 0.0005, 0.001$ and ∞. (We enforced the budget constraint in X with equality.) Figure 1.1 indicates that near-optimal solutions are obtained with small values of ρ (contrast this with $\rho = \infty$). Figure 1.2 shows that this is possible with solutions that maintain a high degree of persistence relative to that of the $\rho = 0$ solutions. More specifically, Figure 1.2 measures the number of "moves" in going from $x^*(b-1)$ to $x^*(b)$ via $\frac{1}{2}\left(\sum_{c \in AD} |x_c^*(b) - x_c^*(b-1)| - 1\right)$. Solution $x^*(b)$ installs one more sensor than $x^*(b-1)$ and our measure is zero if $x^*(b)$ and $x^*(b-1)$ are otherwise identical. On the other hand, the number of sensors moved in going from $x^*(2) = (1,1,0,\ldots,0)$ to $x^*(3) = (0,0,1,1,1,0,\ldots,0)$ is two.

budget b	time $\rho = 0$	time $\rho = 0.0005$	time $\rho = 0.001$
0	0.4	0.4	0.4
10	5.7	7.4	6.4
20	15.1	13.4	5.5
30	15.2	4.7	4.2
40	6.3	6.2	5.5
50	7.1	6.4	5.3
60	3.3	1.1	0.8
70	8.8	6.3	0.5
79	0.2	0.2	0.2

Table 1.1. The table shows the computational effort (in elapsed seconds) required to solve some representative instances of BiSNIP(b) under different values of the budget b.

Figure 1.1. The graph shows the evasion probability as a function of the number of installed detectors. (The numerical values on the y-axis are suppressed.) When $\rho = 0$ we solve BiSNIP(b) to optimality. When $\rho > 0$ we append a persistence term, $+\rho \sum_{c \in AD} |x_c - \hat{x}_c|$, to the objective function of (1.10), where $\hat{x} = x^*(b-1)$. When $\rho = \infty$, $x^*(b)$ and $x^*(b-1)$ are identical except for one additional detector that is installed. For small values of ρ, the primary objective is to minimize the evasion probability and the secondary objective is to minimize deviation from solution $x^*(b-1)$. The figure shows that for $\rho = 0.0005$ and $\rho = 0.001$, near-optimal solutions are obtained, but when ρ is large the solution quality degrades significantly.

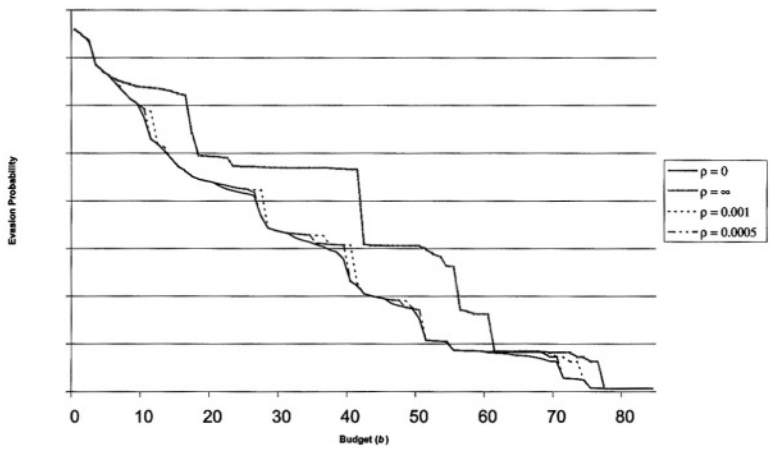
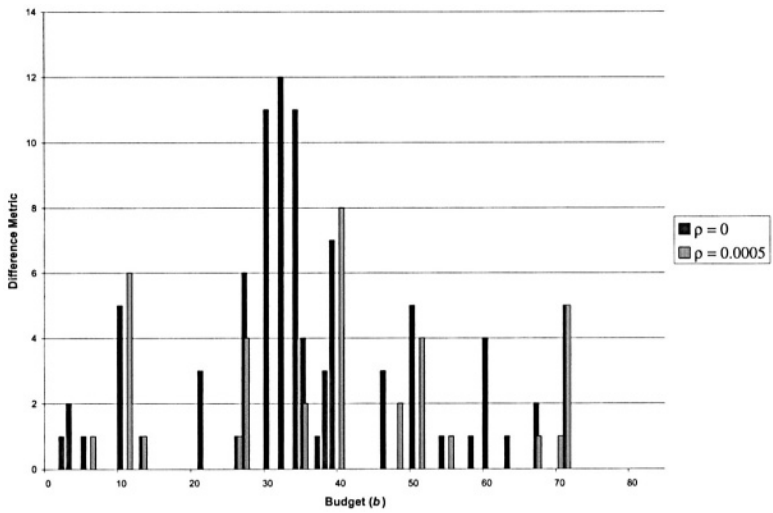


Figure 1.2. The graph shows the difference between solutions $x^*(b)$ and $x^*(b-1)$ as a function of b for $\rho = 0$ and $\rho = 0.0005$. This difference is measured by the number of sensors that have been "moved" in going from $x^*(b-1)$ to $x^*(b)$, not counting the one additional sensor installed by $x^*(b)$. The graph shows that a small, but positive, value of ρ can significantly decrease differences between adjacent solutions.



4. Summary

We have described a stochastic network interdiction model whose solution can be used to select sites to install sensors for detecting smuggled nuclear material. Our goal is to minimize the probability that an intelligent and informed smuggler can successfully travel through an underlying transportation network undetected. The smuggler is informed, knowing the detector locations and the probability of traversing each arc in the network undetected. And, the smuggler is intelligent, selecting an origin-destination path of maximum reliability. We showed the related decision problem to be strongly NP-Complete, reformulated the model as a tractable two-stage stochastic mixed-integer program, and specialized the model to the case where sensors can only be installed at border crossings of a single country. We have presented a summary of our computational experience with this problem.

While the focus of this paper has been on a stochastic network interdiction model for an informed and intelligent smuggler, other important topics need to be addressed. Different models will be needed if: A smuggler is unaware of some or all of the detector locations; more generally, if

the interdictor and evader have different perceptions of the arc reliabilities; or, the smuggler does not select a maximum-reliability path. We anticipate our ongoing work on these topics to be reported in the near future.

5. Acknowledgements

This research was supported by the National Science Foundation under grant DMI-9702217 and the State of Texas Advanced Research Program under grant #003658-0405. We thank Kevin Wood and an anonymous referee, who helped improve the paper with several suggestions.

References

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows.* Prentice Hall, Upper Saddle River, NJ, 1993.

[BA93] O. Ben-Ayed. Bi-level linear programming. *Computers and Operations Research,* 20:485–501, 1993.

[BDW97] G.G. Brown, R.F. Dell, and R.K. Wood. Optimization and persistence. *Interfaces,* 27:15–37, 1997.

[BGV89] M.O. Ball, B.L. Golden, and R.V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters,* 8:73–76, 1989.

[BKMR98] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. GAMS, A User's Guide, 1998. http://www.gams.com/.

[BKS95] A. Bar-Noy, S. Khuller, and B. Schieber. The complexity of finding most vital arcs and nodes. Technical Report CS-TR-35-39, Computer Science Department, University of Maryland, 1995.

[BM90] J. Bard and J. Moore. A branch and bound algorithm for the bi-level programming problem. *SIAM Journal on Scientific and Statistical Computing,* 11:281–292, 1990.

[CL95] M.S. Chern and K.C. Lin. Interdicting the activities of a linear program–a parametric approach. *European Journal of Operational Research,* 86:580–591, 1995.

[CMW98] K. Cormican, D.P. Morton, and R.K. Wood. Stochastic network interdiction. *Operations Research,* 46:184–197, 1998.

[CPL01] ILOG CPLEX 7.1 User's Manual, 2001.

[CS82] H.W. Corley and D.Y. Sha. Most vital links and nodes in weighted networks. *Operations Research Letters,* 1:157–160, 1982.

[FH77] D.R. Fulkerson and G.C. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming,* 13:116–118, 1977.

[GMT71] P.M. Ghare, D.C. Montgomery, and T.M. Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly,* 18:37–45, 1971.

[Gol78] B. Golden. A problem in network interdiction. *Naval Research Logistics Quarterly,* 25:711–713, 1978.

[Isr99] E. Israeli. System interdiction and defense. Operations Research Department, Naval Postgraduate School, Monterey, California, 1999. PhD Dissertation.

[IW01a] E. Israeli and R.K. Wood. Shortest-path network interdiction. *Networks,* to appear.

[IW01b] E. Israeli and R.K. Wood. System interdiction and defense. Working paper, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2001.

[LL93] G. Laporte and F.V. Louveaux. The integer *L*-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters,* 13:133–142, 1993.

[MM70] A.W. McMasters and T.M. Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly,* 17:261–268, 1970.

[MMG89] K. Malik, A.K. Mittal, and S.K. Gupta. The k-most vital arcs in the shortest path problem. *Operations Research Letters,* 8:223–227, 1989.

[MW99] D.P. Morton and R.K. Wood. Restricted-recourse bounds for stochastic linear programming. *Operations Research,* 47:943–956, 1999.

[Ree94] B.K. Reed. Models for proliferation interdiction response analysis. Operations Research Department, Naval Postgraduate School, Monterey, California, 1994. M.S. Thesis.

[VW69] R.M. Van Slyke and R.J.-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics,* 17:638–663, 1969.

[Wol64] R.D. Wollmer. Removing arcs from a network. *Journal of the Operations Research Society of America,* 12:934–940, 1964.

[Wol80] R.D. Wollmer. Two-stage linear programming under uncertainty with 0 – 1 integer first stage variables. *Mathematical Programming,* 19:279–288, 1980.

[Woo93] R.K. Wood. Deterministic network interdiction. *Mathematical and Computer Modeling,* 17:1–18, 1993.

[WW94] A.R. Washburn and R.K. Wood. Two-person zero-sum games for network interdiction. *Operations Research,* 43:243–251, 1994.

This page intentionally left blank

Chapter 2

ENUMERATING NEAR-MIN S-T CUTS

Ahmet Balcioglu
Operations Research Dept.
Naual Postgraduate School
Monterey, CA 93943

R. Kevin Wood
Operations Research Dept.
Naval Postgraduate School
Monterey, CA 93943
kwood@nps.navy.mil

Abstract We develop a factoring (partitioning) algorithm for enumerating near-minimum-weight s-t cuts in directed and undirected graphs, with application to network interdiction. "Near-minimum" means within a factor of $1+\epsilon$ of the minimum for some $\epsilon \geq 0$. The algorithm requires only polynomial work per cut enumerated provided that ϵ is sufficiently (not trivially) small, or G has special structure, e.g., G is a complete graph. Computational results demonstrate good empirical efficiency even for large values of ϵ and for general graph topologies.

Keywords: graphs, networks, cuts, enumeration, polynomial-time algorithm

Introduction

Researchers have studied various classes of cuts in graphs and devised efficient algorithms for enumerating these cuts. This paper addresses a particular class of cuts that has not received the same attention as others, specifically, near-minimum-weight minimal s-t cuts. We develop, implement and test an algorithm to enumerate these cuts in directed or undirected graphs.

We focus on *directed graphs* $G = (V,E)$, with positive integer edge weights and two special vertices, a source s and a sink t. A *minimal s-t cut* C is a minimal set of edges whose removal breaks all directed s-t paths; if removal of C breaks all paths but C is non-minimal, it is a *non-minimal s-t cut*. Non-minimal s-t cuts do not interest us in this paper, except in the way that they interfere with our identification of minimal s-t cuts. All cuts discussed are minimal s-t cuts unless otherwise specified, so we often drop "minimal" and even "s-t." Note that a minimum-weight cut must be minimal because all edge weights are positive.

The problem of finding an s-t cut of minimum weight among all possible s-t cuts in G is the *minimum s-t cut problem* (MCP). This paper studies two extensions of MCP, the problem of *enumerating all minimum-weight s-t cuts* in G (AMCP) and the problem of *enumerating all near-minimum (minimal) s-t cuts* (ANMCP) whose weight is within a factor of $1 + \epsilon$ of the minimum for some $\epsilon \geq 0$. The main contribution of this paper is an efficient procedure for the latter extension, when ϵ is small, or for certain graph topologies. Even when not provably efficient, the algorithm shows good empirical efficiency on our test problems. A cut-enumeration algorithm is "efficient" if the amount of work per cut enumerated is polynomial in the size of G.

The analogs of AMCP and ANMCP in undirected graphs G can also be solved using our techniques. An s-t cut C in an undirected graph is defined just as in a directed graph, the only difference being that the paths broken by C consist of undirected edges. However, if we make the standard transformation that replaces each undirected edge in G by two directed, anti-parallel edges, each with the weight of the original undirected edge, then each s-t cut in the resulting directed graph corresponds directly to an s-t cut in the original graph. Thus, an efficient technique to enumerate s-t cuts in directed graphs will efficiently enumerate s-t cuts in undirected graphs.

Another type of cut can be defined in an undirected graph G. A *disconnecting set* (DS) is a minimal set of edges whose deletion disconnects G. (Often called a "cut," we use "DS" to avoid confusion.) The problems of finding and enumerating certain DSs are related to our problems and will be discussed briefly, so: (a) The problem of finding a minimum-weight DS is denoted MDSP, (b) the problem of enumerating all minimum-weight DSs is denoted AMDSP, and (c) the problem of enumerating all near-minimum-weight DSs is denoted ANMDSP.

In the remainder of this paper, we denote a minimum-weight s-t cut as C_0 and a near-minimum-weight (minimal) s-t cut as C_ϵ. $\mathbf{C}_0(G)$ and $\mathbf{C}_\epsilon(G)$ denote the set of minimum and near-minimum cuts in G,

respectively. We will often substitute "max" and "min" for "maximum" and "minimum," respectively.

A military application, namely network interdiction, first brought ANMCP to our attention; see Wood (1993) and references therein, Boyle (1998) and Gibbons (2000). A "network user" attempts to communicate between vertices s and t in a directed network while an "interdictor," using limited resources (aerial sorties, cruise missiles, etc.), tries to interdict (break, destroy) all s-t paths to prevent communication between s and t. By treating the amount of resource required to interdict an edge as its weight or capacity, the interdictor can solve a max-flow problem and identify a min-weight s-t cut, i.e., min-resource s-t cut, to prevent that communication. It is clear from this application why we are only interested in minimal s-t cuts.

But, there may be secondary criteria, e.g., collateral damage, risk to attacking forces, etc., that the interdictor wishes to consider when determining the best interdiction plan. In this case, near-optimal solutions with respect to the primary criterion can be obtained by solving ANMCP; then those solutions can be evaluated against the secondary criteria for suitability. One of those near-optimal "good solutions" might produce more desirable results than an "optimal solution" obtained by solving MCP or AMCP (Boyle 1998, Gibbons 2000). Integer-programming techniques could substitute for this enumeration approach, but the empirical efficiency of our methods bodes well for enumeration. In fact, the secondary criteria could be incorporated into our recursive cut-enumeration algorithm to force peremptory backtracking, i.e., to help trim the "enumeration tree." This might result in an even more efficient interdiction algorithm.

Another application of ANMCP arises in assessing the reliability and connectivity of networks; see Provan and Ball (1983) and Colbourn (1987).

AMCP and AMDSP have been intensively studied, but ANMCP has not received the same attention. One brute-force approach for ANMCP is to enumerate all s-t cuts, i.e., solve AMCP, and then discard the cuts that do not have near-minimum weight. All s-t cuts can be enumerated efficiently (Tsukiyama et al. 1980, Abel and Bicker 1982, Karzanov and Timofeev 1986, Shier and Whited 1986, Ahmad 1990, Sung and Yoo 1992, Prasad et al. 1992, Nahman 1995, Patvardhan et al. 1995, and Fard and Lee 1999). Unfortunately, this fact cannot lead to an efficient general approach for AMCP or ANMCP because the number of minimal s-t cuts in a graph may be exponential in the size of that graph while the number of minimum and near-minimum cuts may be polynomial. For instance, if G is a complete directed graph with edge weights of 1,

the total number of minimal s-t cuts is $2^{(|V|-2)}$, the number of minimum cuts is 2 and the number of cuts of the next largest size is $2(|V|-2)$.

All minimum s-t cuts can be enumerated efficiently, that is, AMCP can be solved efficiently. Picard and Queyranne (1980) find a max flow in a weighted directed graph G, create the corresponding residual graph, and then demonstrate the one-to-one correspondence of minimum s-t cuts in G to closures in the residual graph. (A closure is a set of vertices with no edges directed out of the set.) They go on to present an algorithm, not necessarily an efficient one, to enumerate these closures and thus all min cuts. Provan and Ball (1983) use the concept of "s-directed minimum cuts" to enumerate minimum s-t cuts in both directed and undirected graphs. However, neither their algorithm nor Picard and Queyranne's may be efficient for directed graphs (Provan and Shier 1996). Gusfield and Naor (1993), Provan and Shier (1996) and Curet et al. (2002) all give efficient algorithms for AMCP based on results from Picard and Queyranne. Provan and Shier's work is related to Kanevsky (1993) who finds all minimum-cardinality "separating vertex sets" as opposed to separating edge sets.

Ramanathan and Colbourn (1987) enumerate "almost-minimum cardinality s-t cuts." They bound the number of cuts enumerated, and the complexity of their algorithm, by $O(m^k n^{k+2})$, where $n = |V|$, $m = |E|$ and where $k \geq 1$ is a constant by which the cardinality of an almost-min cut exceeds the cardinality of a min cut. This algorithm applies only to undirected graphs and has polynomial complexity only if k is fixed.

Karger and Stein (1996) introduce a randomized algorithm for solving ANMDSP by repeated applications of edge contraction: Identify an edge that is probably not part of a near-min-weight DS and merge its endpoints into a single new vertex such that the new graph still contains a near-min DS with high probability. With high probability, their algorithm enumerates all DSs whose weight is within a factor α of the minimum in $O(n^{2\alpha} \log^2 n)$ expected time. They also derive an upper bound $O(n^{2\alpha})$ on the number of these DSs. Karger (2000) later improves this upper bound to $O(n^{\lfloor 2\alpha \rfloor})$. Nagamochi et al. (1997) give a deterministic algorithm for solving ANMDSP based on Karger and Stein's techniques. They show that all near-min DSs can be enumerated in $O(m^2 n + n^{2\alpha} m)$ time. Unfortunately, it is unlikely that this approach can be extended to enumeration problems involving s-t cuts (Karger and Stein 1996).

Vazirani and Yannakakis (1992) propose an algorithm for solving ANMCP and ANMDSP. Their extended abstract claims that the algorithm has polynomial complexity, but that claim is based on this unproven assertion: "Fact: Given a partially specified cut, we can find with one max-flow computation a minimum weight s-t cut consistent with it." We

believe that their claim is false, but do not yet have a proof. In any case, since those authors provide no proof and thus no method for identifying an appropriate cut, their "algorithm" can only be viewed as conjecture.

Boyle's algorithm (Boyle 1998) for solving constrained network-interdiction problems on undirected planar graphs can be modified to enumerate near-minimum cuts, but generalization to the non-planar case seems unlikely. Gibbons (2000) describes an algorithm for solving ANMCP in directed or undirected graphs, but that algorithm may enumerate a cut more than once. Empirically, the running time and number of cuts enumerated in his algorithm grow rapidly as the size of graph and ϵ increase, so that algorithm is impractical except for small problems.

There is a connection between the problems of enumerating near-min s-t cuts in graphs and enumerating extreme points of polytopes (e.g., Avis and Fukuda 1996, Bussieck, and Lübbecke 1998), because a modified dual of the max-flow linear program can be guaranteed to possess 0-1 extreme-point solutions that identify cuts (e.g., Wood 1993). But the literature on extreme-point enumeration is silent on efficient enumeration of near-optimal extreme points which is analogous to enumerating near-min minimal and non-minimal cuts. Nor does this literature address the enumeration of near-optimal extreme points possessing special properties, which might be analogous to enumerating near-min minimal cuts. Further research on extreme-point enumeration may lead to results applicable to enumerating near-min minimal s-t cuts, but is beyond the scope of this paper.

The discussion above shows the need for additional work on ANMCP, so this paper proposes a new algorithm to solve the problem, and provides theoretical and empirical results on its efficiency. The algorithm first identifies a min-weight s-t cut C_0, and then recursively partitions ("factors") the space of possible cuts, possibly including some non-minimal ones, by forcing inclusion and/or exclusion of edges $e \in C_0$ in subsequent cuts.

1. Preliminaries

Let $G = (V, E)$ be an *edge-weighted directed graph* with a finite set of *vertices* V and a set of ordered pairs of vertices, $E \subseteq V \times V$, called *edges.* An *undirected graph* is defined similarly, except that its edges are unordered pairs from $V \times V$. We typically use e or (u, v) to denote an edge $e = (u, v)$, and we let $n = |V|$ and $m = |E|$. We distinguish two vertices s and t in V as the *source* and *sink,* respectively. Edge weights are specified by a *weight function* $w : E \rightarrow Z^+\backslash\{0\}$. We denote the

weight of edge $e = (u, v)$ as w_e or $w(u, v)$ and the vector of edge weights as $\mathbf{w} = (w_{e_1}, w_{e_2}, \ldots, w_{e_m})$.

A directed *s-t path* in G is a sequence of vertices and edges of the form $s, (s, v_1), v_1, (v_1, v_2), v_2, \ldots, v_{k-1}, (v_{k-1}, t), t$. A minimal *s-t cut* in G is a minimal set of edges C whose removal disconnects s from t in G, i.e., breaks all directed s-t paths. If C is a proper superset of some s-t cut, it is a *non-minimal s-t cut.* When no confusion will result, we use "s-t cut" and "cut" interchangeably with "minimal s-t cut." The value $w(C) = \sum_{e \in C} w_e$ is the *weight* of cut C.

A *minimum cut* C_0 is an s-t cut whose weight, $w_0 = w(C_0)$, is minimum among all s-t cuts. All minimum cuts are minimal because edge weights are positive. A *near-minimum minimal cut* C_ϵ is a minimal s-t cut whose weight is at most $\bar{w}_\epsilon = (1 + \epsilon)w_0$ for some $\epsilon \geq 0$. $\mathbf{C}_0(G)$ and $\mathbf{C}_\epsilon(G)$ denote the set of minimum and near-minimum (minimal) cuts, respectively.

An *s-t flow* $\mathbf{f}$ in a directed graph G is a function $f : E \rightarrow Z^+$ where $0 \leq f(e) \leq w_e$ for all $e \in E$ and $\sum_{(u,v) \in E} f(u, v) = \sum_{(v,u) \in E} f(v, u)$ for all $u \in V \backslash \{s, t\}$. The value of the flow from s to t is $F = \sum_{(s,u) \in E} f(s, u) - \sum_{(u,s) \in E} f(u, s)$. In the *maximum-flow problem,* we wish to find a flow $\mathbf{f}^*$ that yields a maximum value for F, denoted F^*.

As a result of the max-flow min-cut theorem and its proof (e.g., Ahuja et al. 1993, pp. 184-185), we know that $w(C_0) = F^*$. It is also well known that, given any maximum flow $\mathbf{f}^*$, we can identify a minimum cut C_0 in $O(m)$ time.

A rooted tree T is a connected, acyclic, undirected graph in which one node (vertex), called the "root" and denoted by r, is distinguished from the others. A rooted tree, called an *enumeration tree*, will describe the enumeration process used for solving AMCP and ANMCP on a graph G. To avoid confusion with "vertices" in G, we use the term "node" to mean a vertex in an enumeration tree. A node i in tree T with root r is said to be at level (depth) l if the length of the unique path from r to i is of the form $P_i = r, (r, i_1), i_1, \ldots, i_{l-1}, (i_{l-1}, i), i$. Every node along path P_i, except node i, is an *ancestor* of i, and if i is ancestor of j, then j is a *descendant* of i. For any path P_i, i_{k-1} is the *parent* of i_k, and i_k is the *child* of i_{k-1}.

2. Theoretical Results

This section develops our algorithm for ANMCP through two intermediate stages.

2.1 Basic Algorithm

Algorithm 1 below outlines an approach to solving ANMCP. Some steps may be difficult to implement, but it illustrates the general approach our final algorithm will use.

Algorithm A1
DESCRIPTION: A generic partitioning algorithm for enumerating near-min, minimal s-t cuts.
INPUT: A directed graph $G = (V, E)$, distinct source and sink vertices $s, t \in V$, edge-weight vector $\mathbf{w}$ of positive integers, and tolerance $\epsilon \geq 0$.
OUTPUT: All minimal s-t cuts C_ϵ such that $w(C_\epsilon) \leq (1+\epsilon)w(C_0)$ where C_0 is a min-weight cut of G.
begin
  Find a min-weight s-t cut C_0 in G;
  $\bar{w}_\epsilon \leftarrow \lfloor (1+\epsilon)w(C_0) \rfloor$;
  $E^+ \leftarrow \emptyset$; /* set of edges to be included */
  $E^- \leftarrow \emptyset$; /* set of edges to be excluded */
  EnumerateA1 $(G,s,t,\mathbf{w},E^+,E^-, \bar{w}_\epsilon)$;
end

Procedure *EnumerateA1* $(G,s,t,\mathbf{w},E^+,E^-,\bar{w}_\epsilon)$
begin
  Step A: Let C' be min-weight minimal cut in G such that
    $E^+ \subseteq C'$ and $E^- \cap C' = \emptyset$;
  if (no such cut exists) **return**;
  if ($w(C') > \bar{w}_\epsilon$) **return**;
  Step B: print (C');
  for (each edge $e \in C' \backslash E^+$) **begin**;
    $E^- \leftarrow E^- \cup \{e\}$;
    EnumerateA1$(G, s, t, \mathbf{w}, E^+, E^-, \bar{w}_\epsilon)$;
    $E^- \leftarrow E^- \backslash \{e\}$;
    $E^+ \leftarrow E^+ \cup \{e\}$;
  endfor;
  return;
end.

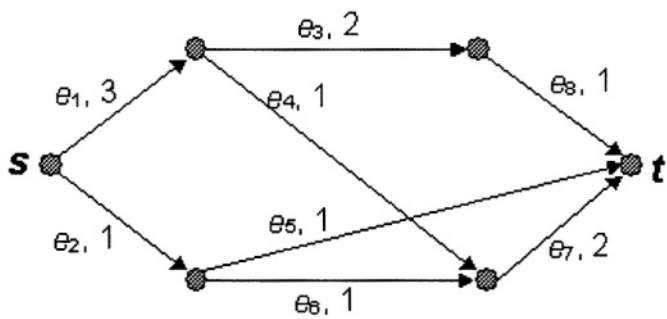
Algorithm A1 begins by finding an initial min-weight cut C_0 and its weight $w(C_0)$. The algorithm then calls the procedure *EnumerateA1* which attempts to find a new min cut by *processing* the edges of the initial cut such that the edges are forced into (*included in*) or out of (*excluded from*) any new near-min cuts. Suppose that $C_0 = \{e_1, e_2, \ldots, e_k\}$

is the initial minimum cut. Based on this cut, the set of near-min cuts, $\mathbf{C}_\epsilon(G)$, is partitioned as

$$\begin{aligned}\mathbf{C}_\epsilon(G) \;=\; & [\mathbf{C}_\epsilon(G) \cap \bar{e}_1)] \cup [\mathbf{C}_\epsilon(G) \cap e_1 \cap \bar{e}_2)] \\ & \cup [\mathbf{C}_\epsilon(G) \cap e_1 \cap e_2 \cap \bar{e}_3] \\ & \cup \cdots \cup [\mathbf{C}_\epsilon(G) \cap e_1 \cap e_2 \cap \cdots \cap e_{k-1} \cap \bar{e}_k] \\ & \cup [\mathbf{C}_\epsilon(G) \cap e_1 \cap \cdots \cap e_k], \end{aligned} \tag{2.1}$$

where $\mathbf{C}_\epsilon(G) \cap e_1 \cap e_2 \cap \cdots \cap e_{k'-1} \cap \bar{e}_{k'}$ is shorthand notation for a set that contains all near-min cuts incorporating e_1 through $e_{k'-1}$ but not $e_{k'}$. The cuts in this partition, except for the unique cut of the last term which has already been found as C_0, are identified by recursively calling *EnumerateA1* with the argument sets E^+ and E^-, where E^+ denotes included edges and E^- denotes excluded edges. The procedure calls itself recursively for every edge of the locally minimum cut that has not already been forced into that cut at higher level in the enumeration. ("Local" and "locally" refer to flows and cuts defined on graphs within the enumeration tree.) The procedure backtracks when it determines that no acceptable cuts remain below a given node.

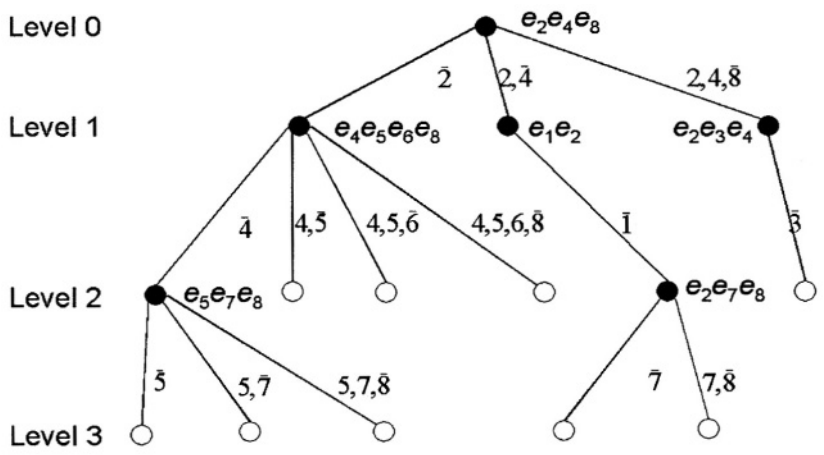
Figure 2.1. Sample graph. Numbers represent edge weights.



To illustrate how this enumeration works, suppose we wish to solve ANMCP on the graph of Figure 2.1 for $\epsilon = 0.4$. The associated enumeration tree (an instance of a rooted tree) for this problem is given in Figure 2.2. The enumeration algorithm first finds a minimum cut $\{e_2, e_4, e_8\}$ at the root node (level 0), and then recursively partitions the solution space via $\{\bar{e}_2\}$, $\{e_2, \bar{e}_4\}$ and $\{e_2, e_4, \bar{e}_8\}$. Once an edge of a cut at some node k has been processed, it will never be processed again at any descendant node of k, because its status as "included" or "excluded" with

respect to the current cut has been fixed at node k. The branches with $\{\bar{e}_2\}$,$\{e_2, \bar{e}_4\}$ and $\{e_2, e_4, \bar{e}_8\}$ correspond to searches for a new min cut by processing the edges as described. If a search is successful, it defines a *productive* node where a new near-min cut is identified and that cut's unprocessed edges are recursively processed. Otherwise, the search leads to a *terminal* node, where the algorithm backtracks. The procedure is correct because it implements inclusion-exclusion (Equation 2.1) in a straightforward, recursive manner. The actual implementation of the algorithm could be difficult, and efficiency poor however, because edge inclusion may be difficult to ensure (although edge exclusion is easy). This topic is explored further in Section 2.2.

Figure 2.2. Enumeration tree for the graph of Figure 2.1 with $\epsilon = 0.4$. The enumeration scheme is represented from top to bottom and left to right. Each filled-in node corresponds to a cut whose weight is no larger than $\bar{w}_\epsilon = \lfloor (1 + 0.4) w(C_0) \rfloor = \lfloor (1 + 0.4)3 \rfloor = 4$. The edges of the cut are listed next to the node. Numbers with bars over them represent the number of the edge excluded from a cut; numbers without bars represents edges to be included. Unfilled nodes are terminal nodes where the algorithm backtracks.



A "relaxed" version of Algorithm A1, denoted "Algorithm A2," can be defined by modifying Steps A and B to:

> Step A: Let C' be min-weight minimal or non-minimal s-t cut in G such that $E^+ \subseteq C'$ and $E^- \cap C' = \emptyset$;

Step B: **if** (C' is non-minimal) print(C');

Algorithm A2 may waste time working with non-minimal cuts because it partitions the space consisting of all minimal cuts and possibly some non-minimal ones, It does solve ANMCP, however, because it prints only the minimal cuts. Our final implementable algorithm, Algorithm B, closely mimics Algorithm A2.

Note that Algorithm A2 will not necessarily identify all non-minimal cuts C' satisfying $w(C') \leq \bar{w}_\epsilon$, which is good because their identification wastes computational effort. Consider, for instance, the last term of equation (2.1), $\mathbf{C}_\epsilon(G) \cap e_1 \cap \cdots \cap e_k$ when $C_0 = \{e_1, \ldots, e_k\}$. This subset of the partition includes all non-minimal cuts containing $C_0 = \{e_1, \ldots, e_k\}$. However, Algorithm A2 does not partition $\mathbf{C}_\epsilon(G) \cap e_1 \cap \cdots \cap e_k$ further, because any cut it might contain other than C_0 is a superset of C_0 and must therefore be non-minimal.

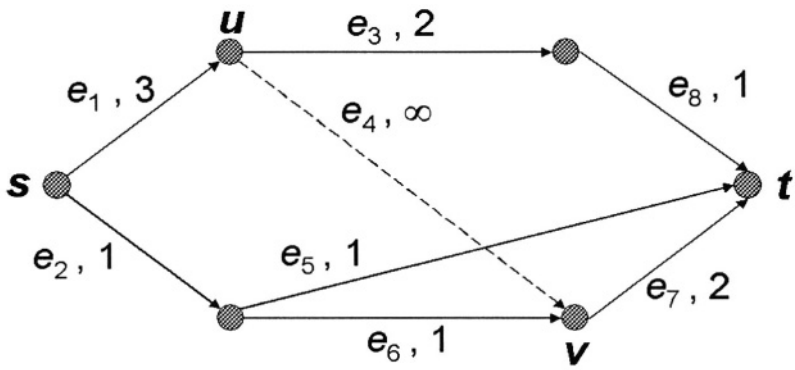
2.2 An Implementable Algorithm

Algorithm B, below, implements a variant of Algorithm A2. We *quasi-exclude* an edge e from every cut in a subtree of the enumeration tree by simply setting $w_e = \infty$, represented by a suitably large integer. Every near-min cut in G must have a finite weight, so setting $w_e = \infty$ effectively eliminates cuts containing e. This means that quasi-exclusion implements true exclusion. The graph G with edge e quasi-excluded is denoted $G - e$.

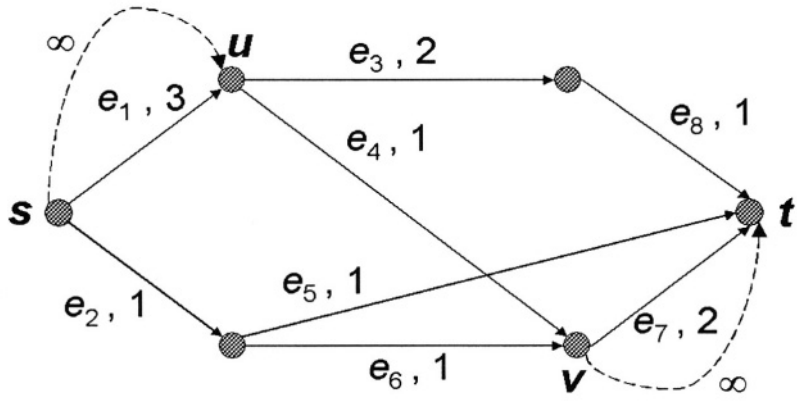
We *quasi-include* $e = (u, v)$ by effectively adding two edges to G, (s, u) and (v, t), both with infinite weights. The graph G with edge e quasi-included is denoted $G + e$. Now, any cut of a graph must contain at least one edge from every path, so any cut of $G+e$ must contain (s, u), (u, v) or (v, t), and any finite-weight cut of $G+e$ must contain $e = (u, v)$. (We can omit (s, u) if $u = s$, and omit (v, t) if $v = t$.) In reality, we implement quasi-inclusion of $e = (u, v)$ by temporarily treating u as an additional source and v as an additional sink. Unfortunately, quasi-inclusion can create modified graphs with minimal cuts that correspond to non-minimal cuts in the original graph G. (See the example below.) Thus, Algorithm B must screen for non-minimal cuts, just as Algorithm A2 does.

Figure 2.3 illustrates the quasi-inclusion and -exclusion of edges. $G + E^+ - E^-$ denotes G with E^+ quasi-included and E^- quasi-excluded. The caption describes an example of how quasi-inclusion of an edge in the depicted graph can create a modified graph in which a minimal cut is non-minimal in the original graph G.

Figure 2.3. Quasi-inclusion and -exclusion of an edge from cuts in a graph G. (a) Edge e_4 is quasi-excluded from every cut by setting $w_4 = \infty$. (b) Edge e_4 is quasi-included in every cut by, in effect, adding infinite-weight edges (s, u) and (v, t) to G. Quasi-inclusion can create difficulties (not indicated explicitly in the figures): $\{e_1, e_2, e_8\}$ forms a minimal cut of $G + e_8$ (G with edge e_8 quasi-included), but $\{e_1, e_2, e_8\}$ is not a minimal cut of G.



(a)



(b)

Note that Algorithm B calls a subroutine *MaxFlow* which is assumed to return a min cut C' in the current graph G' along with the weight of that cut $w(C')$, which equals the value of the max flow. Also note that, in order to keep the notation similar to the earlier algorithms, Algorithm B repeatedly modifies a copy of the original graph G, rather than recursively modifying and "unmodifying" a single copy of G. The actual Java implementation of Algorithm B uses the latter, more efficient approach.

Algorithm B
DESCRIPTION: An implementable version of Algorithm A2 to solve ANMCP;
INPUT:A directed graph $G = (V, E)$, s, t, $\mathbf{w}$, and ϵ.
OUTPUT: All minimal s-t cuts C_ϵ in G with
$w(C_\epsilon) \leq (1+\epsilon)w(C_0)$.
begin
$[w_0, C_0] \leftarrow MaxFlow(G, s, t, \mathbf{w})$;
$\bar{w}_\epsilon \leftarrow \lfloor (1+\epsilon)w_0 \rfloor$;
$E^+ \leftarrow \emptyset$; /* set of edges to be included */
$E^- \leftarrow \emptyset$; /* set of edges to be excluded */
EnumerateB$(G, s, t, \mathbf{w}, E^+, E^-, \bar{w}_\epsilon)$;
end.

Procedure *EnumerateB*$(G, s, t, \mathbf{w}, E^+, E^-, \bar{w}_\epsilon)$
begin
$\mathbf{w}' \leftarrow \mathbf{w}$; $G' \leftarrow G$;
for (each edge $e = (u, v) \in E^-$) $w'(u, v) \leftarrow \infty$;
for (each edge $e = (u, v) \in E^+$) **begin**
add artificial edge (s, u) to G' and let $w'(s, u) \leftarrow \infty$;
add artificial edge (v, t) to G' and let $w'(v, t) \leftarrow \infty$;
endfor;
/* G' and $\mathbf{w}'$ are now interpreted to include artificial edges */
$[w', C'] \leftarrow MaxFlow(G', s, t, \mathbf{w}')$;
if ($w' > \bar{w}_\epsilon$) **return**;
if (C' is minimal in G) print (C');
for (each edge $e \in C' \backslash E^+$) **begin**
$E^- \leftarrow E^- \cup \{e\}$;
EnumerateB $(G, s, t, \mathbf{w}, E^+, E^-, \bar{w}_\epsilon)$;
$E^- \leftarrow E^- \backslash \{e\}$;
$E^+ \leftarrow E^+ \cup \{e\}$;
endfor;
return;
end.

2.3 Correctness of Algorithm B

We assume the correctness of Algorithm A2. Algorithm B begins by finding a min cut in G and determining $\bar{w}_\epsilon$ using a max-flow algorithm, all in an obviously correct manner. That is, the main routine of Algorithm B correctly implements the main routine of Algorithm A2. Then, where Algorithm A2 finds a minimal or non-minimal, min-weight cut that includes edges in E^+ and excludes edges in E^-, Algorithm B solves a max-flow problem and finds a min cut C' in $G + E^+ - E^-$. C' may or may not be a minimal cut in the original graph G, but its deletion from G does disconnect s from t in G, because (a) $C' \subseteq E$ because artificial edges have infinite weight and thus cannot be contained in C', and (b) every s-t path in $G + E^+ - E^-$ is disconnected by deleting C', and (c) every path in G is also a path in $G + E^+ - E^-$ by construction; thus, all paths in G are disconnected by deleting C' from that graph. Algorithm B is clearly finite and will be correct as long as it correctly partitions the space of all minimal cuts in G (along with some non-minimal cuts perhaps).

The partitioning will be correct if no non-minimal cuts that Algorithm A2 might identify are lost in the calls to *EnumerateB* and no non-minimal cuts are repeated. The following two lemmas suffice to prove this.

Lemma 2 *Let C be a finite-weight set of edges in G and let E^+ and E^- be quasi-inclusion and quasi-exclusion sets, respectively, produced while running Algorithm B. Suppose that $C \cap E^+ = E^+$ and $C \cap E^- = \emptyset$. Then C is a minimal cut of G only if C is also a finite-weight minimal cut of $G + E^+ - E^-$.*

Proof: Since $C \cap E^- = \emptyset$, C has finite weight in $G + E^+ - E^-$. $G + E^+$ has the same topological structure as $G + E^+ - E^-$, so we need only be concerned with the former. Now, C is clearly a cut in $G + E^+$ because the fact that $C \cap E^+ = E^+$ means that no edges crossing from the s side of the cut in G to the t side have been added; only edges from the t side of the cut to t or from s to some vertex on the s side of C could have been added through quasi-inclusion of E^+. So, C is a cut in $G + E^+$, and it must be minimal because every path in G is also a path in $G + E^+$. ∎

Lemma 3 *Let C be a set of edges in G and let E^+ and E^- be, respectively, quasi-inclusion and quasi-exclusion sets produced while running Algorithm B. Suppose that $C \cap E^+ \neq E^+$ or $C \cap E^- \neq \emptyset$. Then, C is a not a finite-weight minimal cut of $G + E^+ - E^-$.*

Proof: We know that quasi-exclusion properly implements edge exclusion. Thus, C cannot be a finite-weight minimal cut of $G + E^+ - E^-$ if

some edge of C has been excluded, i.e., if $C \cap E^- \neq \emptyset$. From the discussion on quasi-inclusion, we know that all finite-weight minimal cuts of $G + E^+$ must contain E^+, i.e., C cannot be a finite-weight minimal cut of $G + E^+ - E^-$ if $C \cap E^+ \neq E^+$. ∎

The following theorem results.

Theorem 1 *Algorithm B solves ANMCP.* ∎

2.4 Complexity of Algorithm B

We show below that Algorithm B has polynomial complexity when G and/or **w** satisfy certain conditions. This discussion ignores minimality testing after the next lemma because it cannot add to Algorithm B's worst-case complexity for any problem, assuming that at least $O(m)$ work must arise at every node of the enumeration tree:

Lemma 4 *Testing whether or not a set of edges C in G is a minimal s-t cut can be accomplished in $O(m)$ time.*
Proof: Assume that C is a minimal or non-minimal cut and mark all vertices as "unreachable." Now, perform a breadth-first search starting at s, trying to reach as many vertices as possible without traversing any cut edges $(u, v) \in C$. Mark the vertices reached as "reachable from s." Conduct a similar search, traversing edges backward from t, marking the vertices reached as "can reach t." (If a backward search from t can reach v, then v can reach t along a directed path.) By definition, C is minimal if and only if, for all edges $(u, v) \in C$. u is reachable from s and v can reach t. The amount of work involved in the two searches and testing the edges in C is clearly $O(m)$. ∎

2.4.1 Complexity Analysis of Min-Cut Enumeration.

We first analyze the complexity of enumerating minimum cuts ($\epsilon = 0$), since this is an important special case of near-min cut enumeration. Consider the enumeration tree of Figure 2.2. Every node in that tree is either *productive* and defines a new cut (the filled-in nodes), or it is an *unproductive* terminal node from which backtracking occurs immediately. In general, the quasi-inclusion technique can result in unproductive nonterminal nodes because it can identify a non-minimal cut and be unable to backtrack immediately. Fortunately, any non-minimal cut encountered while solving AMCP must correspond to a terminal node and an efficient procedure results.

We know that the worst-case complexity of solving, "from scratch," an initial max-flow problem on $G = (V,E)$ is $O(f(n, m))$, where $f(n, m)$ is a polynomial function of $n = |V|$ and $m = |E|$. (For instance, the

first polynomial-time algorithm for max flows, a flow-augmenting path algorithm due to Edmonds and Karp (1972), has worst-case complexity $O(nm^2)$; the more modern pre-flow push algorithm due to Goldberg and Tarjan (1988) has $O(nm \log(n^2/m))$ worst-case complexity.) At each non-root node of the enumeration tree, the local max flow can be obtained by performing flow augmentations starting with the feasible flow from the parent node. (A feasible flow $\mathbf{f}$ in G must be feasible for $G - E^- + E^+$ because the latter graph is obtained from the former by increasing the capacity on certain edges, specifically $e \in E^-$, and adding some other edges, specifically $e \in E^+$. Neither of these operations reduces the capacity on any path in the original graph G.) Each flow augmentation requires $O(m)$ work using breadth-first search in a standard fashion, but the total amount of work performed at each node can be limited to $O(m)$, because (a) if the first search does not find a flow-augmenting path, a new min cut has been identified (this fact follows from the standard constructive proof of the max-flow min-cut theorem, e.g., Ahuja, et al. 1993, pp. 184-185), and (b) if a flow-augmenting path is found, the locally maximum flow is at least $w_0 + 1$ and the algorithm can backtrack immediately. (The algorithm must be modified slightly to enable this "peremptory backtracking.") Thus the number of productive nodes is $|\mathbf{C}_0(G)|$.

Now, each non-terminal node can generate at most n child nodes assuming G has no parallel edges, and thus each productive node can generate at most n unproductive (terminal) nodes. Therefore, the total number of nodes generated is bounded by $n|\mathbf{C}_0(G)|$. The amount of work to generate each node except the first is $O(m)$, and the amount of work to generate the first node is $O(f(n,m))$ so we have the following result.

Theorem 2 *Algorithm B with* $\epsilon = 0$ *finds all minimum-weight s-t cuts (solves AMCP) in* $O(f(n,m) + mn|\mathbf{C}_0(G)|)$ *time.* ∎

This shows that Algorithm B is theoretically efficient for AMCP since only a polynomial amount of work is expended for each cut enumerated. The Algorithm is admittedly less efficient for solving AMCP than are some other algorithms from the literature: For instance, the algorithm of Provan and Shier (1996) solves AMCP in $O(f(n,m)+(m+n)(|\mathbf{C}_0(G)|))$ time. Nevertheless, our algorithm has several advantages in that (a) it is easy to implement, (b) its empirical efficiency is quite good (see Section 3) and, (c) it extends to near-min cut enumeration, i.e., to solving ANMCP, by simply setting $\epsilon > 0$.

2.4.2 Complexity Analysis of Near-Min Cut Enumeration.

The argument of the previous section leads quickly to this corollary:

Corollary 1 *If* $\min_{e\in E} w_e > w(C_0)\epsilon$, *then Algorithm B solves ANMCP in* $O(nf(n,m)|\mathbf{C}_0(G)|)$ *time.*

Proof: "$f(n,m)$" is a multiplier on $|\mathbf{C}_0(G)|$ here because we are unable to bound the number of flow augmentations required in *EnumerateB* to establish a new max flow: We simply resort to the bound implied by solving each max-flow problem from scratch. As before, the multiplicative factor n will bound the number of terminal nodes emanating from a productive node.

Just as in Theorem 2, the statement of the Corollary will be true if Algorithm B can always backtrack when it finds a non-minimal cut, that is, if every non-minimal cut corresponds to a terminal node. This is true because any non-minimal cut must have weight at least $w(C_0) + \min_{e\in E} w_e > w(C_0) + w(C_0)\epsilon = w(C_0)(1+\epsilon) = \bar{w}_\epsilon$, ∎

So, Algorithm B is efficient when ϵ is sufficiently small. It is also efficient when G has special topology.

Theorem 3 *Algorithm B solves ANMCP in* $O(nf(n,m)|\mathbf{C}_0(G)|)$ *time whenever G contains an edge of the form* (s,u) *for each* $u \in V\backslash\{s,t\}$ *and an edge of the form* (v,t) *for each* $v \in V\backslash\{s,t\}$.

Proof: The statement will be true if quasi-inclusion and -exclusion never change the vertex-to-vertex connectivity of a graph, because then any minimal cut of $G + E^+ - E^-$ must be a minimal cut of G. But quasi-inclusion never changes connectivity irrespective of graph topology. Quasi-exclusion for $e = (u,v)$ always adds edges of the form (s,u) and (v,t), but as specified, G already contains such edges. ∎

Corollary 2 *Algorithm B solves ANMCP in* $O(nf(n,m)|\mathbf{C}_0(G)|)$ *time when G is a complete directed graph or complete acyclic graph with* $s < t$ *in the acyclic (topological) ordering of the vertices.* ∎

Of course, the problem is trivial if $s > t$.

By the arguments of the preceding section, the number of nodes in enumeration tree should be bounded by $n(|\mathbf{C}_\epsilon(G)| + |\mathbf{C}''|)$, where $\mathbf{C}''$ denotes the set of near-minimum, non-minimal cuts identified as nodes in Algorithm B's enumeration tree where immediate backtracking is not allowed, i.e., the set of unproductive, non-terminal nodes in that tree.

The test for non-minimality takes $O(m)$ time at each node by Lemma 4. The search for a local max flow might require multiple flow augmentations and might be as hard as solving a max-flow problem from scratch. Therefore, the work expended at every node is $O(f(n,m)+m) = O(f(n,m))$.

If we could backtrack whenever a non-minimal cut was identified, then we could state that $\mathbf{C}'' = \emptyset$ and the resulting complexity for the whole algorithm would be $O(nf(n,m)|\mathbf{C}_\epsilon(G)|)$. But, it is easy to show by example that backtracking when a non-minimal cut is encountered can result in the loss of some valid minimal cuts. Thus, Algorithm B must continue partitioning, even on non-minimal cuts, until it can backtrack based on cut weight. This results in a complexity of $O(nf(n,m)(|\mathbf{C}_\epsilon(G)| + |\mathbf{C}''|))$, which may not be polynomial if $|\mathbf{C}''|$ is exponentially larger than $\mathbf{C}_\epsilon(G)$. Therefore, the worst-case complexity of Algorithm B for arbitrary ϵ and/or arbitrary graph topology is not well determined. We leave this complexity issue as a topic for future research.

3. Computational Results

This section reports on computational experiments with Algorithm B to demonstrate its empirical efficiency for solving both AMCP and ANMCP. We test Algorithm B on both weighted and unweighted grid graphs and on several problem instances from the literature.

Algorithm B is written and compiled using the Java 1.2.2 programming language (Sun Microsystems 1998). All tests are performed on a personal computer with a 733 MHz Pentium III processor and 128 MB of RAM, running under the Windows 98 SE operating system.

3.1 Efficient Implementation of Algorithm B

We have described Algorithm B in a simple form for clarity, but there are several modifications that improve its performance in practice. As discussed in Section 2.4.1, the *MaxFlow* routine of Algorithm B is implemented to solve an "incremental" max-flow problem. Specifically, $\mathbf{f}^*$ in G is a feasible flow in $G + E^+ - E^-$, so a flow-augmenting path algorithm operating on a graph at some non-root node in the enumeration tree simply begins with the maximum flow from the parent node, rather than starting with $\mathbf{f} = \mathbf{0}$. (See Ahuja et al. 1993, pp. 180-184.)

Another issue in an efficient implementation is edge inclusion. In theory, we quasi-include an edge (u,v) by adding infinite-weight edges (s,u) and (v,t) to the graph, but in practice we simulate this by simply treating u as an additional source and v as an additional sink.

Algorithm B also incorporates "peremptory backtracking" from within the *MaxFlow* routine. In particular, that routine does not need to solve a max-flow problem to completion if it augments enough flow to learn that the local max flow exceeds $\bar{w}_\epsilon$, which implies that any locally minimum cut C' must have $w(C') > \bar{w}_\epsilon$. When this situation occurs, *MaxFlow*

halts and returns the current feasible flow value F, which causes *EnumerateB* to return immediately.

The rest of the implementation is straightforward. We use forward and reverse star representation of G as our data structure (Ahuja et al. 1993, pp. 35-38) and a variant of the shortest flow-augmenting-path algorithm of Edmonds and Karp (1972) for solving max-flow problems. (More sophisticated algorithms would speed computations somewhat, but this algorithm is more than adequate to verify the usefulness of our methodology.)

3.2 Test Problems

Our literature search has not uncovered any particular problem family designed for testing algorithms for AMCP and ANMCP, except for Grid Graph Families (GGFs) (Curet et al. 2002, Gibbons 2000), which we will explore. Additionally, we have modified some DIMACS problems (The Center for Discrete Mathematics and Theoretical Computer Science, DIMACS 1991) and several problem classes from Levine (1997) to test Algorithm B.

We have coded a GGF generator (GGFGEN) in Java to generate grid graphs. The height H of the grid measured in nodes, and its length L in nodes, determine the size of the generated graph. One other parameter is q, which indicates whether the graph is weighted $(q = 1)$ or unweighted $(q = 0)$. "Unweighted" simply means that all edge weights are 1 or ∞: For both weighted and unweighted graphs, the edges beginning at s and ending at t have infinite weights. Every vertex $u \in V\backslash\{s,t\}$ is connected to each adjacent vertex v (vertically and horizontally, assuming such adjacent vertices exist) with two directed edges, (u,v) and (v,u). Edge weights for weighted graphs are pseudo-random, uniformly distributed integer weights in the range [1,10]. GGFGEN produces a directed graph with $HL + 2$ vertices and $4HL - 2L$ edges. Figure 7 shows a graph generated by GGFGEN with inputs $H = 3$, $L = 4$, $q = 0$. Table 2.1 specifies the problems instances that are tested.

We have also chosen two other graph generators from the literature, implemented in the C programming language and available via Internet for research use. The first is the Double-Cycle Generator (DBLCYCLEGEN) (Levine 1997), which generates undirected graphs that we convert to directed graphs. The single input parameter for DBLCYCLEGEN is $n = |V|$. DBLCYCLEGEN generates two interleaved cycles on n vertices: The outer cycle includes all n vertices with edge weights of 1000 and 997, and the inner cycle connects every third vertex of the outer cycle with the edges of weights 1 or 4. Vertices s and t are chosen to

Figure 2.4. An unweighted, directed grid graph Generated by GGFGEN with inputs $H = 3$, $L = 4$, and $q = 0$. Edges incident to s and t have infinite weights. Other weights are all 1. Bi-directional edges between u and v represent two directed edges, (u,v) and (v,u).

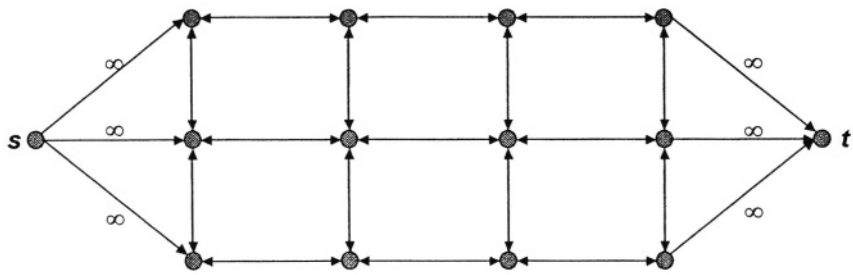


Table 2.1. Problem groups for GGF. This table shows the GGF input graphs and parameter settings with which we test Algorithm B. "$q = 0$" indicates unweighted graphs, and "$q = 1$" indicates weighted ones.

Problem Name	H	L	ϵ	q
GGF-square	5,10,15,20,25, 30,40,. . ., 90	5,10,15,20,25, 30,40,. . .,90	0.0, 0.05 .10, .15	0, 1
GGF-long	25	100,125, . . .,250	0.0	0

Table 2.2. Problem types for DBLCYCLEGEN and AD. Graphs of type DBLCYC-I are generated with DBLCYCLEGEN and have $n = 500$ vertices and 1000 undirected edges on two interleaved cycles; the undirected edges are converted to $m = 2000$ directed edges. Graphs of type AD are fully dense, directed acyclic graphs with $n = 50$ and $m = 1225$ unweighted edges.

Problem Name	n	ϵ	q
DBLCYC-I (Levine 1997)	500	0.00, 0.10, 1.25, 1.50, 1.75, 2.00	0, 1
AD (DIMACS 1991)	50	0.10, 0.20, ..., 0.70	0

be as distant from each other as possible. A minimum cut lies in the middle of the graph with a weight of 2000 and there are many near-min cuts with weight 2006.

The second generator is the Acyclic Dense (AD) graph generator from DIMACS (1991). AD takes n as its input parameter and generates a fully dense, directed acyclic graph with n vertices and $m = n(n-1)/2$ edges. We replace the pseudo-randomly generated edge weights in AD with unit weights to observe the behavior of our algorithm on the underlying topological structure. In all cases, $s = 1$ and $t = n$ in the acyclic ordering of the vertices. Table 2.2 gives the generated problem types for DBLCYCLEGEN and AD.

3.3 Experiments on Unweighted Graphs

Table 2.3 presents run times of Algorithm B on GGF instances for solving AMCP. It takes less than 1 second for Algorithm B to identify all minimum cuts in grid graphs with up to 402 vertices and 1,560 edges. The number of calls to *MaxFlow*—this corresponds to the number of nodes in the enumeration tree—increases roughly linearly with n.

Table 2.4 summarizes the results for ANMCP on GGF-square instances with $\epsilon = 0.05$, 0.10, and 0.15. Solution times are expected to increase as ϵ increases because the number of cuts in any graph might be exponential in the size of the graph. The algorithm is quite efficient for modest-size grid graphs with modest values of ϵ. Compared with Gibbons' results for ANMCP (Gibbons 2000), our results show a vast reduction in calls to *MaxFlow* and in run times.

Table 2.5 presents results for an unweighted AD graph. Corollary 2 requires that Algorithm B not generate any unproductive non-terminal nodes for the AD topology, and results displayed in the table provide empirical verification of this.

Table 2.3. Run times (in CPU seconds) for Algorithm B solving AMCP on unweighted instances of GGF-square and GGF-long graphs. GGF$H\times L$ denotes a GGF graph with an $H \times L$ grid of vertices. "Non-min cuts" denotes the number of non-minimal cuts encountered at non-terminal nodes of the enumeration tree. As predicted by Theorem 2, this number is 0 for AMCP. "Calls to MF" indicates the number of calls to *MaxFlow* and thus the total number of nodes in the enumeration tree.

Problem Name	n	m	$\lvert C_0\rvert$	$\lvert \mathbf{C}_0(G)\rvert$	Non-min Cuts	Calls to MF	Run Time (sec.)
GGF10×10	102	380	10	9	0	92	0.1
GGF20×20	402	1560	20	19	0	382	0.1
GGF30×30	902	3540	30	29	0	872	0.1
GGF40×40	1602	6320	40	39	0	1562	0.2
GGF50×50	2502	9900	50	49	0	2452	0.3
GGF60×60	3602	14280	60	59	0	3542	0.5
GGF70×70	4902	19460	70	69	0	4832	2.1
GGF80×80	6402	25440	80	79	0	6322	5.8
GGF25×100	2502	9800	25	99	0	2477	0.2
GGF25×125	3127	12250	25	124	0	3102	1.2
GGF25×150	3752	14700	25	149	0	3727	3.0
GGF25×175	4377	17150	25	174	0	4352	5.7
GGF25×200	5002	19600	25	199	0	4977	9.0
GGF25×225	5627	22050	25	224	0	5602	13.0
GGF25×250	6252	24500	25	249	0	6227	17.6

Table 2.4. Computational results for Algorithm B solving ANMCP on instances of unweighted GGF-square graphs with $\epsilon = 0.05, 0.10, 0.15$. $|C^+|$ denotes the cardinality of the largest acceptable cut.

Problem Name	n	m	$\lvert C_0\rvert$	$\lvert C^+\rvert$	$\lvert \mathbf{C}_\epsilon(G)\rvert$	Non-min Cuts	Calls to MF	Run Time (sec.)
$\epsilon = 0.05$								
GGF5×5	27	90	5	5	4	0	22	0.1
GGF10×10	102	380	10	10	9	0	92	0.1
GGF15×15	227	870	15	15	14	0	212	0.1
GGF20×20	402	1560	20	21	703	0	7906	1.5
GGF25×25	627	2450	25	26	1128	0	15506	3.9
GGF30×30	902	3540	30	31	1653	0	26856	12.0
$\epsilon = 0.10$								
GGF5×5	27	90	5	5	4	0	22	0.1
GGF10×10	102	380	10	11	153	0	956	0.1
GGF15×15	227	870	15	16	378	0	3306	0.5
GGF20×20	402	1560	20	22	13319	0	113090	20.1
GGF25×25	627	2450	25	27	27014	0	274550	74.4
GGF30×30	902	3540	30	33	924723	378	8911698	4535.1
$\epsilon = 0.15$								
GGF5×5	27	90	5	5	4	0	22	0.1
GGF10×10	102	380	10	11	153	0	956	0.2
GGF15×15	227	870	15	17	5264	0	35905	4.9
GGF20×20	402	1560	20	23	168283	153	1202033	215.3
GGF25×25	627	2450	25	28	431728	253	3621978	973.3
GGF30×30	902	3540	30	34	13465371	21843	113463496	46395.8

Table 2.5. Computational results on an unweighted, acyclic dense graph(AD) with various threshold levels ϵ. This is a fully dense graph with 50 vertices and 1225 edges. As predicted by Corollary 2, no non-minimal cuts are generated at non-terminal nodes, irrespective of ϵ.

ϵ	$\lvert C_0\rvert$	$\lvert \mathbf{C}_\epsilon(G)\rvert$	Non-min Cuts	Calls to MF	Run Time (sec.)
0.00	49	49	0	1275	0.3
0.10	49	544	0	13650	3.1
0.20	49	4063	0	101625	25.7
0.30	49	19798	0	495000	134.9
0.40	49	75893	0	1897375	542.9
0.50	49	249270	0	6231800	1861.6
0.60	49	730603	0	18265125	5544.8
0.70	49	1962849	0	49071275	15126.6

Table 2.6. Computational results for min-cut enumeration (AMCP) on weighted, GGF-square problems. As in the unweighted case, no non-minimal cuts are encountered. All min cuts are identified in less than one second for these instances.

Problem Name	n	m	w_0	$\lvert \mathbf{C}_0(G)\rvert$	Non-min Cuts	Calls to MF	Run Time (sec.)
GGF5×5w	27	90	17	1	0	7	0.1
GGF10×10w	102	380	42	2	0	92	0.1
GGF15×15w	227	870	45	1	0	19	0.1
GGF20×20w	402	1560	69	1	0	32	0.1
GGF25×25w	627	2450	87	1	0	33	0.1
GGF30×30w	902	3540	108	6	0	217	0.3

3.4 Experiments on Weighted Graphs

Here we use the GGF-square problems with edge weights that are pseudo-randomly generated integers in the range [1,10]. Results for minimum and near-minimum cut enumeration are summarized in Tables 2.6 and Table 2.7, respectively.

Finally, we test Algorithm B on the DBLCYC-I problems with ϵ ranging from 0.0 to 2.0. These are the only problems where Algorithm B encounters substantial numbers of non-minimal cuts from which immediate backtracking would be incorrect. At $\epsilon = 1.25$, the ratio of the number of

Table 2.7. Computational results for near-minimum cut enumeration (ANMCP) on weighted GGF-square problems. w_0 is the minimum cut weight and $\bar{w}_\epsilon$ is the weight of the largest acceptable cut.

Problem Name	$\|V\|$	$\|E\|$	w_0	$\bar{w}_\epsilon$	$\|\mathbf{C}_\epsilon(G)\|$	Non-min Cuts	Calls to MF	Run Time (sec.)
$\epsilon = 0.05$								
GGF5×5	27	90	5	5	4	0	22	0.1
GGF10×10	102	380	10	10	9	0	92	0.1
GGF15×15	227	870	15	15	14	0	212	0.1
GGF20×20	402	1560	20	21	703	0	7906	1.5
GGF25×25	627	2450	25	26	1128	0	15506	3.9
GGF30×30	902	3540	30	31	1653	0	26856	12.0
$\epsilon = 0.10$								
GGF5×5	27	90	5	5	22	0	22	0.1
GGF10×10	102	380	10	11	956	0	956	0.1
GGF15×15	227	870	15	16	3306	0	3306	0.5
GGF20×20	402	1560	20	22	113090	0	113090	20.1
GGF25×25	627	2450	25	27	274550	0	274550	74.4
GGF30×30	902	3540	30	33	8911698	378	8911698	~4.5k
$\epsilon = 0.15$								
GGF5×5	27	90	5	5	22	0	22	0.1
GGF10×10	102	380	10	11	956	0	956	0.2
GGF15×15	227	870	15	17	35905	0	35905	4.9
GGF20×20	402	1560	20	23	1202033	153	1202033	215.3
GGF25×25	627	2450	25	28	3621978	253	3621978	973.3
GGF30×30	902	3540	30	34	13465371	21843	113463496	~46k

Table 2.8. Computational results for near-min cut enumeration (ANMCP) on weighted DBLCYC-I problems with $n = 500$ (and $m = 2000$). The number of non-minimal cuts encountered at non-terminal nodes increases substantially when ϵ becomes sufficiently large.

ϵ	w_0	$\bar{w}_\epsilon$	$\lvert\mathbf{C}_0(G)\rvert$	Non-min Cuts	Calls to MF	Run Time (sec.)
0.00	1000	1000	2	0	10	0.1
0.10	1000	1100	499	0	1976	0.5
1.00	1000	2000	511	8	2032	0.6
1.25	1000	2250	2479	237411	957178	207.8
1.50	1000	2500	2479	237411	957178	208.7
1.75	1000	2750	2479	237411	957178	207.4
2.00	1000	3000	2509	238041	959683	213.4

near-min non-minimal cuts (encountered at non-terminal nodes) to the number of near-min minimal cuts jumps dramatically; see Table 2.8.

In summary, computational results above show that Algorithm B performs quite well on a variety of graph types. However, non-minimal cuts defining unproductive, non-terminal nodes in the enumeration tree can slow computations when the threshold parameter ϵ becomes large, at least for certain graph topologies. For dense acyclic graphs, the behavior of Algorithm B verifies Corollary 2: No non-minimal cuts are encountered at non-terminal nodes. However, for the double-cycle graphs DBLCYC-I, the number of non-minimal cuts generated can outnumber the minimal cuts by a large margin, at least when ϵ becomes large.

4. Conclusions and Recommendations

In this paper, we have developed an algorithm for ANMCP, defined as the problem of enumerating *all near-minimum-weight, minimal s-t cuts* C_ϵ in a directed graph $G = (V, E)$ with positive integer edge weights $w_e \ \forall \ e \in E$. The users specifies a value $\epsilon \geq 0$, and the algorithm finds all minimal s-t cuts C_ϵ such that $w(C_\epsilon) \leq (1 + \epsilon)w(C_0)$, where $w(C)$ denotes the weight of cut C, and C_0 is a min-weight cut. The algorithm first finds a min-weight cut C_0 in the input graph via a maximum-flow algorithm, and then recursively partitions the space of near-min cuts. Given a cut C, this partitioning is carried out by forcing inclusion and exclusion of edges from subsequent cuts. An edge (u, v) is *quasi-excluded* by simply setting its weight to infinity and *quasi-included* by implicitly introducing two infinite-weight edges in G, one extending form s to u

and the other from v to t. The algorithm solves a max-flow min-cut problem for each modified graph that is obtained in the enumeration tree.

We have implemented our algorithm using the following enhancements to improve computational speed: (a) The algorithm solves a complete max-flow problem at the root node of enumeration tree but solves only "incremental" max-flow problems at the all other nodes (the max flow at a parent node is feasible for all child nodes and thus provides an advanced start for maximizing flows at those child nodes), and (b) quasi-inclusion of an edge (u, v) is simulated by treating u as an additional source and v as an additional sink, and (c) the algorithm backtracks directly from the max-flow subroutine, without identifying a locally minimum cut, if a feasible flow is found that exceeds the backtrack threshold. (That flow is a lower bound on the weight of the min cut.)

Unfortunately, the quasi-inclusion technique can lead to the enumeration of non-minimal cuts at non-terminal nodes of the enumeration tree. Non-minimal cuts are easily identified (and ignored), but they can increase the computational workload and stop us from deriving a polynomial-time bound for the worst-case complexity of the general algorithm: The algorithm cannot always backtrack when it finds a non-minimal cut. We do obtain, however, a polynomial bound of $O(f(n,m)+nm|\mathbf{C}_\epsilon(G)|)$ when $\min_{e\in E} w_e > w(C_0)\epsilon$; here $\mathbf{C}_\epsilon(G)$ denotes the set of near-min cuts and $O(f(n,m))$ is the worst-case complexity of the max-flow algorithm being used. Thus, the algorithm has polynomial complexity, per cut enumerated, for the important special case of ANMCP when $\epsilon = 0$, i.e., AMCP: Enumerate all min-weight s-t cuts in G. We also determine the polynomial bound of $O(nf(n,m)|\mathbf{C}_\epsilon(G)|)$ for certain graph topologies such as complete graphs and complete acyclic graphs.

Computational results for $\epsilon > 0$ show that Algorithm B has good empirical efficiency as long as ϵ is not too large. Unfortunately, large ϵ can lead to the identification of many non-minimal cuts where the algorithm cannot immediately backtrack. Thus, many "unproductive" non-terminal nodes can be encountered in the enumeration tree, and it is only these nodes that stop us from proving polynomial complexity.

To improve the algorithm, one might try to create a better quasi-inclusion technique or develop a completely different technique for edge inclusion. For instance, we have not tried simply setting to 0 the capacity of an arc to be included. If "true edge inclusion" (as opposed to quasi-inclusion) can be efficiently implemented, this should yield a provably polynomial-time algorithm for near-min cut enumeration. However, it

can be proven that the problem of finding a min cut that includes a specific set of edges is actually NP-complete.

If the current quasi-inclusion technique is retained, another approach might be used to avoid enumerating non-minimal cuts. In particular, edges that cannot occur in any minimal cut given those that are already included might be identified and marked as "forbidden for inclusion." These edges would be excluded, as usual, by setting their weights to infinity. An edge (u, v) can be forbidden from inclusion if (a) every s-u or v-t path contains at least one included edge, or (b) some included edge (u', v') must contain (u, v) in every s-u' path or in every v'-t path. This list is not all-inclusive, however.

Another practical improvement might result from this: The algorithm can backtrack whenever the set of quasi-included edges forms a cut in the original graph.

Computation times on some large graphs would be improved by solving the initial maximum-flow problem using a more efficient algorithm, e.g., Goldberg and Rao (1998). It may also be possible to show that the worst-case complexity of the algorithm is actually better than reported by amortizing the work involved in augmenting flows over the course of running the algorithm.

We have shown that Algorithm B will not enumerate any non-minimal cuts if every vertex $v \in V \backslash \{s, t\}$ has incident edges (s, v) and (v, t). It would be interesting to determine if the algorithm will enumerate only minimal cuts for other graph topologies, too. For instance, using the dual of a planar graph and shortest-path techniques, it is possible to enumerate near-min cuts in an undirected s-t planar graph in polynomial time per cut. Thus, it is natural to wonder if Algorithm B can too.

Acknowledgments

The authors thank Matthew Carlyle, David Morton, Alexandra Newman and Craig Rasmussen for their helpful suggestions regarding this paper. The authors also thank the Office of Naval Research and the Air Force Office of Scientific Research for their support of this research.

References

Abel, V., and Bicker, R., (1982), "Determination of All Minimal Cut-Sets between a Vertex Pair in an Undirected Graph," *IEEE Transactions on Reliability,* Vol. R-31, pp. 167-171.

Ahmad, S.H., (1990), "Enumeration of Minimal Cutsets of an Undirected Graph," *Microelectronics Reliability,* Vol. 30, pp. 23-26.

Avis, D., and Fukuda, K., (1996) "Reverse Search for Enumeration," *Discrete Applied Mathematics,* Vol. 65, pp. 21-46.

Boyle, M.R., (1998), "Partial-Enumeration For Planar Network Interdiction Problems," Master's Thesis, Operations Research Department, Naval Postgraduate School, Monterey, California, March.

Bussieck, M.R., and Lübbecke, M.E., (1998), "The Vertex Set of a 0/1-Polytope is Strongly P-enumerable," *Computational Geometry,* Vol. 11, pp. 103-109.

Colbourn, C.J., 1987, *The Combinatorics of Network Reliability,* Oxford University Press.

Curet, N.D., DeVinney, J., Gaston, M.E. 2002, "An Efficient Network Flow Code for Finding All Minimum Cost s-t Cutsets," *Computers and Operations Research*, Vol. 29, pp. 205-219.

DIMACS, (1991), *The First DIMACS International Algorithm Implementation Challenge,* Rutgers University, New Brunswick, New Jersey. (Available via anonymous ftp from dimacs.rutgers.edu.)

Edmonds, J. and Karp, R.M., (1972), "Theoretical Improvements in Algorithm Efficiency for Network Flow Problems," *Journal of the ACM*, Vol. 19, pp. 248-264.

Fard, N.S., and Lee, T.H., (1999), "Cutset Enumeration of Network Systems with Link and Node Failures," *Reliability Engineering and System Safety*, Vol. 65, pp. 141-146.

Gibbons, M., (2000), "Enumerating Near-Minimum Cuts in a Network," Master's Thesis, Operations Research Department, Naval Postgraduate School, Monterey, California, June.

Goldberg, A.V., and Rao, S., (1998), "Beyond the Flow Decomposition Barrier," *Journal of the ACM*, Vol. 45, pp. 783-797.

Goldberg, A.V., and Tarjan, R.E., (1988), "A New Approach to the Maximum Flow Problem," *Journal of the ACM*, Vol. 35, pp. 921-940.

Gusfield, D., and Naor, D., (1993), "Extracting Maximal Information About Sets of Minimum Cuts," *Algorithmica*, Vol. 10, pp. 64-89.

Kanevsky, A., (1993), "Finding All Minimum-Size Separating Vertex Sets in a Graph," *Networks*, Vol. 23, pp. 533-541.

Karger, D.R., (2000), "Minimum Cuts in Near-Linear Time," *Journal of the ACM,* Vol. 47, pp. 46-76.

Karger, D.R., and Stein, C., (1996), "A New Approach to the Minimum Cut Problem," *Journal of the ACM,* Vol. 43, pp. 601-640.

Lawler, E.L., Lenstra, J.K., Rinooy Kan, A.H.G., and Shmoys, D.B., (1985), *The Traveling Salesman Problem,* John Wiley & Sons, Chichester, England.

Levine, M., (1997), "Experimental Study of Minimum Cut Algorithms," Master's Thesis, Department of Electrical Engineering and Computer

Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, May. (http://theory.lcs.mit.edu/~mslevine)

Nagamochi, H., Ono, T., and Ibaraki, T., (1994), "Implementing an Efficient Minimum Capacity Cut Algorithm," *Mathematical Programming*, Vol. 67, pp. 297-324.

Nagamochi, H., Nishimura, K., and Ibaraki, T., (1997), "Computing All Small Cuts in an Undirected Network," *SIAM Journal on Discrete Mathematics*, Vol. 10, pp. 469-481.

Nahman, J.M., (1997), "Enumeration of Minimal Cuts of Modified Networks," *Microelectronics Reliability,* Vol. 37, pp. 483-485.

Patvardhan, C., Prasad, V.C. and Pyara, V.P., (1995), "Vertex Cutsets of Undirected Graphs," *IEEE Transactions on Reliability,* Vol. 44, pp. 347-353.

Picard, J.C., and Queyranne, M., (1980), "On The Structure of All Minimum Cuts in a Network and Applications," *Mathematical Programming Study*, Vol. 13, pp. 8-16.

Prasad, V.C., Sankar, V., and Rao, P., (1992), "Generation of Vertex and Edge Cutsets," *Microelectronics Reliability,* Vol. 32, pp. 1291-1310.

Provan, J.S., and Ball, M.O., (1983), "Calculating Bounds on Reachability and Connectedness in Stochastic Networks," *Networks,* Vol. 13, pp. 253-278.

Provan, J.S., and Shier, D.R., (1996), "A Paradigm for Listing (s,t)-Cuts in Graphs," *Algorithmica,* Vol. 15, pp. 351-372.

Ramanathan, A., and Colbourn, C.J., (1987), "Counting Almost Minimum Cutsets with Reliability Applications," *Mathematical Programming*, Vol. 39, pp. 253-261.

Shier, D.R., and Whited, D.E., (1986), "Iterative Algorithms for Generating Minimal Cutsets in Directed Graphs," *Networks,* Vol. 16, pp. 133-147.

Sung, C.S., and Yoo, B.K., (1992), "Simple Enumeration of Minimal Cutsets Separating 2 Vertices in a Class of Undirected Planar Graphs," *IEEE Transactions on Reliability,* Vol. 41, pp. 63-71.

Sun Microsystems Inc., (1998), Java Platform Version 1.2.2.

Tsukiyama, S., Shirakawa, I., Ozaki, H., and Ariyoshi, H., 1980, "An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset," *Journal of the ACM*, Vol. 27, pp. 619-632.

Vazirani, V.V., and Yannakakis, M., (1992), "Suboptimal Cuts: Their Enumeration, Weight and Number," *Automata, Languages and Programming. 19th International Colloquium Proceedings,* Vol. 623 of Lecture Notes in Computer Science, Springer-Verlag, pp. 366-377.

Wood, R.K., (1993), "Deterministic Network Interdiction," *Mathematical and Computer Modeling,* Vol. 17, pp. 1-18.

This page intentionally left blank

Chapter 3

A DECOMPOSITION-BASED PSEUDOAPPROXIMATION ALGORITHM FOR NETWORK FLOW INHIBITION

Carl Burch
College of St Benedict and St John's University
cburch@csbsju.edu

Robert Carr
Sandia National Laboratories
rdcarr@sandia.gov

Sven Krumke
Konrad-Zuse-Zentrum
krumke@zib.de

Madhav Marathe
Los Alamos National Laboratory
madhav@lanl.gov

Cynthia Phillips
Sandia National Laboratories
caphill@sandia.gov

Eric Sundberg
Rutgers University
sundberg@math.rutgers.edu

Abstract In the network inhibition problem, we wish to expend a limited budget attacking a given edge-capacitated graph by "paying" to remove edge capacity from some subset of the edges. We wish to minimize the resulting maximum flow between two designated vertices s and t. The problem is strongly $\mathcal{NP}$-hard. Previous approximation algorithms applied only to planar graphs. In this chapter, we give a polynomial-time algorithm, based on a linear-programming relaxation of an integer program, that finds an attack with cost B_a and residual network capacity (max flow) C_a such that

$$\frac{B_a}{B} + \epsilon \frac{C_a}{C^*} \leq 1 + \epsilon,$$

where $\epsilon > 0$ is a given error parameter, B is the given budget (the amount of resources to expend damaging the network), and C^* is the minimum (optimal) residual capacity for any attack with budget B. For example, our algorithm returns a $(1, 1 + 1/\epsilon)$-approximation or a $(1 + \epsilon, 1)$-pseudoapproximation, but we do not know which *a priori*. The parameter ϵ biases the nature of the solution, but does not affect the running time.

We generalize the pseudoapproximation algorithm to multiple attack methods/budgets and give a polynomial-time algorithm to compute the most cost-effective attack.

Keywords: network interdiction, multicriteria optimization, integer programming, linear programming, minimum cut

1. Introduction

The *Network Inhibition or Network Interdiction Problem* models the computation of a strategy to attack the transportation capacity of a transportation network. One may wish to compute such a strategy to inhibit the flow of dangerous material through a network. Alternatively, if one wishes to protect transportation capacity, optimal and near-optimal attack strategies can indicate vulnerabilities in a network, areas that need reinforcement.

Our transportation-capacity metric is the classic *maximum s-t flow*. Given an undirected graph $G = (V, E)$ and two designated vertices $s, t \in V$, material "flows" through the network from the *source* vertex s to the *sink* vertex t. Each edge $e = (u, v)$ has an initial capacity $c_e \equiv c_{uv} \geq 0$. A feasible flow assigns a flow value f_{uv}, f_{vu} to each direction for edge e obeying capacity constraints on each edge and flow conservation constraints on each vertex other than s and t. More formally, for each edge (u, v), we have $f_{uv} + f_{vu} \leq c_{uv}$ (generally at least one of f_{uv}, f_{vu} will be zero) and for each vertex $v \neq s, t$, we have $\sum_{u \in V - \{v\}} f_{uv} = \sum_{w \in V - \{v\}} f_{vw}$. The maximum flow is the maximum total flow into the

sink (equivalently out of the source) for any feasible flow. This represents the maximum possible steady-state flow of material through the network.

If the nodes of a flow network are partitioned into two sets, S and T with $s \in S$ and $t \in T$, the set of edges with one endpoint in S and the other endpoint T is called a *cut.* The *capacity* of a cut is the sum of the capacities of the edges in the cut. By Ford and Fulkerson's classic max flow/min cut theorem [Ford and Fulkerson, 1962], the value of a maximum flow equals the minimum capacity of any s-t cut.

In the *Network Inhibition Problem*, we are given a capacitated graph as described above. In addition each edge e has removal cost r_e representing the cost to remove the edge from the graph. This removal cost may be infinite. We assume removal is linear for all edges with finite removal cost, so that paying αr_{uv} for $0 \leq \alpha \leq 1$ removes αc_{uv} units of capacity from edge (u, v). We wish to expend at most a fixed budget B removing capacity from edges of G to minimize the resulting maximum s-t flow. By the above discussion, this is equivalent to minimizing the resulting minimum cut. We call the post-attack mininum cut capacity (maximum flow), the *residual capacity* of the network.

Phillips provided the first full characterization of the complexity of this problem [Phillips, 1993]. She proved the problem is weakly NP-complete for planar graphs and restrictive subsets of planar graphs and gave a fully-polynomial-time approximation scheme for planar graphs. She also showed that the problem is strongly NP-complete for general graphs, but gave no approximation algorithms for the general case. These results also hold for the case where no edge can be partially cut.

Wood [Wood, 1993] independently showed that the network-inhibition problem is strongly NP-complete. He describes methods for effectively solving integer-programming formulations of variants on the network inhibition problem. For example, he defines valid inequalities that can tighten the gap between the linear-programming relaxation and the integer polytope. Since network inhibition is strongly NP-complete, these methods require exponential time in the worst case. Earlier work from the 70's include a branch-and-bound strategy for general graphs [Ghare et al., 1971], and methods of varying quality for inhibition of s-t-planar graphs (planar graphs with both the source and the sink on the outer face) [McMasters and Mustin, 1970, Helmbold, 1971].

In this chapter we consider approximation methods for general instances of the network inhibition problem. Let C^* be the optimal (minimum) residual capacity for any attack of cost at most B on graph G. A β-approximation algorithm for the network inhibition computes an attack strategy for network G of cost at most B such that the residual capacity is no more than βC^*. A (ρ, β)-approximation algorithm for

the network inhibition problem relaxes the budget constraint. It finds an attack strategy using budget at most ρB with residual capacity at most βC^*. If the budget is negotiable, this is a *multicriteria approximation* algorithm. If the budget is a tight constraint, this is a *pseudo-approximation* algorithm, since the solution is not technically feasible.

In this chapter, we give a simple algorithm that, given an $\epsilon > 0$ returns either a $(1, 1+1/\epsilon)$ approximation, (a true $(1+1/\epsilon)$ approximation), or a $(1+\epsilon, 1)$ pseudoapproximation. If the solution exceeds the budget, then it is actually superoptimal. We do not know which type of solution we will compute *a priori.* The parameter ϵ biases the nature of the solution, but does not affect the running time.

The algorithm uses the *decomposition method.* A number of recent papers, for example [Bar-Noy et al., 2001, Naor and Schieber, 1997, Phillips et al., 2002, Srinivasan, 1997] have used the decomposition method to find true approximation algorithms for combiniatorial optimization problems. Others such as [Boyd and Carr, 1999] have used decomposition to prove structural results about the set of feasible solutions for a combinatorial optimization problem. In the decomposition method, we first model a problem with an integer (or mixed-integer) linear program: minimize $c^T x$ such that $Ax \leq b$, $x \geq 0$, and some subset of the variables must take on integer values. Here x is an n-vector of unknowns and A is an $n \times m$ constraint matrix. We assume $c \geq 0$. Let $\mathcal{S}_{IP}$ be the set of feasible solutions.

Solving a general integer program (IP) is *NP* hard. However, we can relax the integrality constraints to obtain an efficiently-solvable linear program (LP). We compute x^*, an optimal (extreme-point) solution to this LP relaxation. Because we have only relaxed constraints, $C^* = c^T x^*$ is a lower bound on the optimal solution to the integer program.

A *(convex) decomposition* of an IP is a set $\{x^{(1)}, \ldots, x^{(N)}\} \subseteq \mathcal{S}_{IP}$ of feasible solutions to the IP and corresponding weights $\lambda_i \geq 0$, $i = 1, \ldots, N$ such that $\sum_{i=1}^{N} \lambda_i = 1$. A decomposition β-approximates the original integer-programming problem if $\sum_{i=1}^{N} \lambda_i c^T x^{(i)} \leq \beta C^*$. The solutions β-approximate the problem on average. Because of this averaging, we conclude that one of the $x^{(i)}$ is a β-approximate solution.

A *β-approximate decomposition* for a solution x^* of an LP relaxation of an integer program is a convex decomposition of (IP) feasible solutions such that $\sum_{i=1}^{N} \lambda_i x^{(i)} \leq \beta x^*$. If there is a β-approximate decomposition of x^* and the cost vector c is nonnegative, then this decomposition also β-approximates the IP.

In this chapter, we model the network inhibition problem as a mixed-integer program (MIP). We find an exact (1-approximate) decomposition $\{x^{(1)}, \ldots, x^{(N)}\}$ for the linear-programming relaxation, except we do not

require each $x^{(i)}$ to satisfy the budget constraint. Because we have a 1-approximate decomposition and x^* satisfies the budget constraint, the $x^{(i)}$ satisfy the budget constraint on average. Thus we will show that one of these solutions has cost B_a and residual capacity C_a such that

$$\frac{B_a}{B} + \epsilon \frac{C_a}{C^*} \leq 1 + \epsilon.$$

We show that this implies our stated approximation bounds. This is an application of the general methods for multicriteria approximation through decomposition studied rigorously in [Burch et al., 2001]. Their methods allow us to generalize to cases where there are multiple types of budget (say money and time).

Our decomposition algorithm is extremely simple because of the structure of the network-inhibition polytope. In fact, we can always find a decomposition with $N = 2$.

One may use parametric-search techniques to achieve similar approximation bounds (for the single-budget case). Parametric search has been applied to multicriteria problems where all criteria are "similar" [Krumke et al., 1998, Marathe et al., 1998]. For example, one can find a spanning tree of approximately minimum weight subject to a budget constraint on the cost of the tree (as specified by a second edge-weight function). A multicriteria optimization problem is reduced to a single-criteria problem using a weighted combination of the cost functions. The method requires only an approximation algorithm for solving the single-criterion problem.

Rao, Shmoys, and Tardos have independently achieved results similar to ours for the single-budget network inhibition problem using parametric search [Shmoys, 1997]. In fact, using the parametric-search approach, we can obtain a theoretically-faster algorithm with running time $O(m \log m + T_c \log^2 m)$, where m is the number of edges in the graph and T_c is the time to compute an *s-t* minimum cut. The advantage of our approach over parametric search is that it's extremely simple and non-iterative. LPs are usually solved much faster than the worst-case bound in practice, so our algorithm may be competitive or even superior in practice.

The remainder of this chapter is organized as follows. In section 2 we give a mixed-integer program for network inhibition. In section 3, we describe the pseudo-approximation algorithm in more detail and prove (pseudo)approximation bounds. In section 4 we show how to decompose the solution to the linear-programming relaxation of the mixed-integer program. In section 5 we give a geometric interpretation of the decomposition and the algorithm. Finally, in section 6 we discuss an extension

to the multiple-budget case and show how to efficiently find a most cost-effective attack.

2. A Mixed-Integer Program for Network Inhibition

Phillips [Phillips, 1993] observes that for a particular (fixed) cut the greedy attack strategy is optimal: One removes edges in decreasing order of c_e/r_e until the budget is exhausted. Thus a solution to the network inhibition problem can be expressed as an s-t cut, which is then attacked in this manner. We base our mixed-integer program upon this observation.

The classic minimum-cut integer program [Schrijver, 1986] specifies a cut by specifying the vertex partition S and T used to determine the cut. The IP has a binary decision variable d_v for each vertex $v \in V$, where $d_t = 1$ and $d_s = 0$. Thus, $d_v = 1$ if vertex v is on the t side of the partition and $d_v = 0$ otherwise. An edge is in the cut if it has exactly one endpoint on the s side. Each such edge contributes to the capacity of the cut. We model this with a binary variable y_{uv} for each edge (u, v). We have $y_{uv} = 1$ if (u, v) is in the cut represented by the d_v variables (that is, the values of d_u and d_v differ), and $y_{uv} = 0$ otherwise. The min-cut integer program is:

$$\text{(MC-IP)} \quad \text{minimize} \quad \sum_{(u,v)\in E} c_{uv} y_{uv}$$

$$\text{where} \quad \begin{cases} y_{uv} \geq d_u - d_v & \forall (u,v) \in E \\ y_{uv} \geq d_v - d_u & \forall (u,v) \in E \\ d_s = 0,\ d_t = 1 & \\ d_v \in \{0,1\} & \forall v \in V \end{cases}$$

The first two sets of constraints set the y_{uv} variables as described above. Because the constraint matrix is totally unimodular, all extreme points are naturally integer, and therefore the integrality constraints are redundant.

We formulate the network inhibition problem from the min-cut formulation. If an edge (u, v) is in the cut designated by the vertex variables (d_u and d_v differ) then we must pay to remove the edge, or pay for the remaining capacity in the objective function. We introduce a continuous variable z_{uv} for each edge (u, v), which represents the fraction of edge (u, v) removed through payment. Rational variable y_{uv} represents the remaining fraction of edge (u, v), which contributes to the cut's residual capacity. Thus for any edge (u, v) in the cut, we have $y_{uv} + z_{uv} = 1$.

The network-inhibition mixed-integer program is:

$$\text{(NI-MIP)} \quad \text{minimize} \quad \sum_{(u,v)\in E} c_{uv} y_{uv}$$

$$\text{where} \quad \begin{cases} \sum_{(u,v)\in E} r_{uv} z_{uv} \leq B & \\ y_{uv} + z_{uv} \geq d_u - d_v & \forall (u,v) \in E \\ y_{uv} + z_{uv} \geq d_v - d_u & \forall (u,v) \in E \\ d_s = 0,\ d_t = 1 & \\ d_v \in \{0,1\} & \forall v \in V \\ y_{uv}, z_{uv} \geq 0 & \forall (u,v) \in E \end{cases}$$

The first constraint enforces the budget. We could also add the constraints $y_{uv} + z_{uv} \leq 1$ for all $(u,v) \in E$. However these constraints will hold in any optimal solution for the MIP as given above, so we do not explicitly add them to the formulation.

3. The Pseudo-approximation Algorithm

In this section we describe the decomposition-based approximation algorithm and prove its approximation bounds given a correct decomposition.

Let $\mathcal{S}$ be the set of feasible solutions to the problem NI-MIP for the network-inhibition problem given in section 2 without the single (first) budget constraint. Then the algorithm to (pseudo)approximate network inhibition is:

1 Compute x^*, an optimal extreme point solution to the LP relaxation for NI-MIP with residual capacity C^*.

2 Compute $x^{(1)}, x^{(2)} \in \mathcal{S}$ and $0 \leq \lambda_1, \lambda_2$ such that $\lambda_1 + \lambda_2 = 1$ and $\lambda_1 x^{(1)} + \lambda_2 x^{(2)} = x^*$. [1]

3 For $x^{(i)}$, $i \in \{1,2\}$, apply the greedy attack strategy to the cut C_i determined by the vertex variables. Compute the cost $B_a^{(i)}$ (the sum of the removal costs for all edges removed) and residual capacity $C_a^{(i)}$ (the sum of the capacities of all remaining edges in the cut). At most one edge will be partially cut, contributing fractionally to both the cost and the residual capacity.

4 For the given (prespecified) value of ϵ, return the strategy for the solution $x^{(i)}$ such that

$$\frac{B_a^{(i)}}{B} + \epsilon \frac{C_a^{(i)}}{C^*} \leq 1 + \epsilon.$$

[1] One can show = instead of $\geq$ due to a strictly-positive cost vector c. We can assume c is strictly positive without loss of generality because we can preprocess the graph to eliminate all zero-cost edges. The greedy solution is also optimal for the fractional case.

We now argue that one of the two solutions computed in step 3 has the property required in step 4.

Theorem 1 *Given an $\epsilon > 0$, for one of the cuts C_i computed in the algorithm, the greedy attack strategy on C_i has cost $B_a^{(i)}$ and residual capacity $C_a^{(i)}$ such that:*

$$\frac{B_a^{(i)}}{B} + \epsilon \frac{C_a^{(i)}}{C^*} \leq 1 + \epsilon.$$

Proof. By construction, x^* is a convex combination of $x^{(1)}$ and $x^{(2)}$. Therefore, any linear function (with nonnegative coefficients) of x^* is a convex combination of the same linear function applied to $x^{(1)}$ and $x^{(2)}$. In particular, this applies to the budget and residual capacity functions. For example, we have

$$\begin{aligned} \lambda_1 x^{(1)} + \lambda_2 x^{(2)} &= x^* \\ c^T(\lambda_1 x^{(1)} + \lambda_2 x^{(2)}) &= c^T x^* \\ \lambda_1 c^T x^{(1)} + \lambda_2 c^T x^{(2)} &= C^* \\ \lambda_1 C_a^{(1)} + \lambda_2 C_a^{(2)} &= C^* \end{aligned}$$

Suppose that no member of the decomposition has the property required in step 4. That is,

$$\frac{B_a^{(i)}}{B} + \epsilon \frac{C_a^{(i)}}{C^*} > 1 + \epsilon$$

for $i = 1, 2$. Then we have:

$$\begin{aligned} 1 + \epsilon &\geq \frac{r^T x^*}{B} + \epsilon \frac{C^*}{C^*} \\ &= \frac{\sum_{i=1}^2 \lambda_i B_a^{(i)}}{B} + \epsilon \frac{\sum_{i=1}^2 \lambda_i C_a^{(i)}}{C^*} \\ &= \sum_{i=1}^2 \lambda_i \left(\frac{B_a^{(i)}}{B} + \epsilon \frac{C_a^{(i)}}{C^*} \right) \\ &> \sum_{i=1}^2 \lambda_i (1 + \epsilon) \\ &= 1 + \epsilon, \end{aligned}$$

which is a contradiction.

The first step follows because x^* is a feasible solution to the linear-programming relaxation of NI-MIP, and therefore it satisfies the budget constraint: $r^T x^* \leq B$, so $\frac{r^T x^*}{B} \leq 1$. The step with the strict inequality applies the assumption that none of the attack strategies satisfies the required condition. The last step follows from the definition of a convex combination: $\lambda_1 + \lambda_2 = 1$. □

Suppose we now have an $x \in \mathcal{S}$ representing an attack with budget B_a and residual capacity C_a that satisfies the conditions in step 4 for some *epsilon* > 0. Recall that $\mathcal{S}$ is the set of integer feasible solutions except that the budget constraint may be violated. We now show this is either a $(1 + 1/\epsilon)$-approximation or that residual capacity is at least optimal and the budget is violated by no more than a factor of $1 + \epsilon$.

Theorem 2 *Suppose we have a solution with cost B_a and residual capacity C_a that satisfies*

$$\frac{B_a}{B} + \epsilon \frac{C_a}{C^*} \leq 1 + \epsilon,$$

where $\epsilon > 0$, B is the budget, and C^* is the objective value of the linear-programming relaxation of NI-MIP. Then either $B_a \leq B$ and $C_a \leq (1 + 1/\epsilon)C^*$, or $C_a \leq C^*$ and $B_a \leq (1 + \epsilon)B$.

Proof: Suppose $C_a > C^*$. Since $C_a/C^* > 1$, we have

$$\frac{B_a}{B} + \epsilon < \frac{B_a}{B} + \epsilon \frac{C_a}{C^*} \leq 1 + \epsilon.$$

Thus $B_a/B < 1$. The solution satsifies the budget constraint and therefore is feasible. Furthermore, because $B_a/B \geq 0$, we have $\epsilon C_a/C^* \leq 1+\epsilon$ and therefore $C_a \leq (1 + 1/\epsilon)C^*$. An analogous argument applies to the case where $B_a/B > 1$. □

4. Decomposition

In this section, we show how to decompose an extreme-point solution for the LP relaxation of the network-inhibition IP. That is, given x^*, an extreme-point optimal solution to the LP relaxation of NI-MIP, we compute $x^{(1)}, x^{(2)} \in \mathcal{S}$ and $0 \leq \lambda_1, \lambda_2 \leq 1$ such that $\lambda_1 + \lambda_2 = 1$ and $\lambda_1 x^{(1)} + \lambda_2 x^{(2)} = x^*$. The structure of the solution x^* makes this computation easy.

Theorem 4 proves that for each vertex-variable component d_v of x^*, either $d_v = 0$, or $d_v = 1$, or $d_v = \alpha$ for some $0 < \alpha < 1$. Assuming such structure for an extreme-point optimal LP solution, we now describe how to decompose the solutuion. Let $V_0 \subset V$ be the set of vertices v such

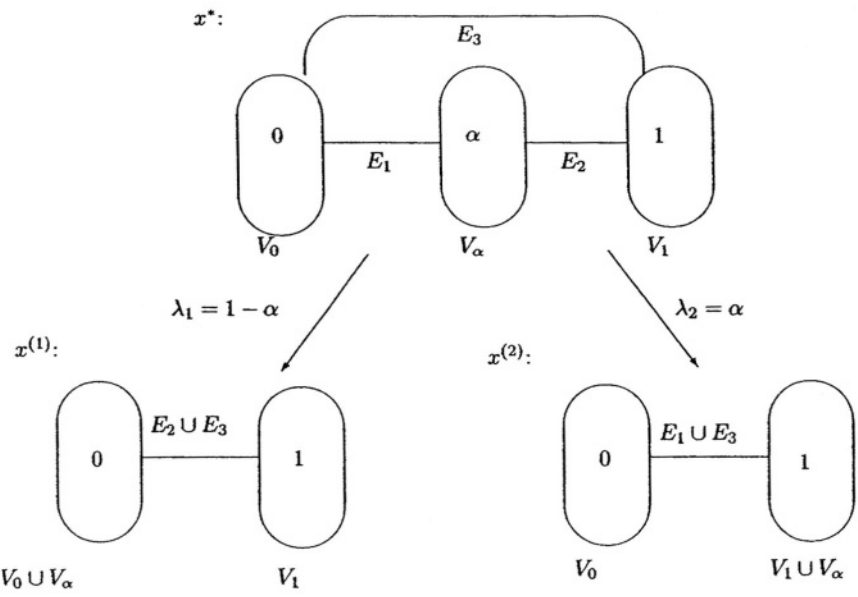


Figure 3.1. The LP optimal solution is a convex combination of two feasible solutions that differ only in the partition assignment for one set of vertices. This set is an α fraction on the t side in the LP solution. Values of d_v are shown inside the ovals. For example, $d_v = 0$ for all vertices in V_0. Edges sets E_j for $j = 1, 2, 3$ are defined by the d_v values in the LP solution. For example, E_1 is the set of edges with one endpoint in V_0 and the other in V_α. In either the LP solution or decomposed solutions, the value of $y_{uv} + z_{uv}$ for an edge (u, v) is the difference in d_v values for its two endpoints. For example, in the LP solution (at the top), $y_{uv} + z_{uv} = \alpha$ for all $(u, v) \in E_1$. In solution $x^{(1)}$, this quantity is 0 and in solution $x^{(2)}$ it is 1.

that $d_v = 0$, let $V_1 \subset V$ be the set of vertices v such that $d_v = 1$, and let $V_\alpha = V - V_0 - V_1$ be the set of vertices such that $d_v = \alpha$. We construct $x^{(1)}$ and $x^{(2)}$ as follows. If $v \in V_0$, then $d_v^{(1)} = d_v^{(2)} = 0$. Similarly, If $v \in V_1$, then $d_v^{(1)} = d_v^{(2)} = 1$. There are no other options since all d_v are bounded between zero and one. If $v \in V_\alpha$, then $d_v^{(1)} = 0$ and $d_v^{(2)} = 1$. This decomposition is illustrated in Figure 3.1. We set $\lambda_1 = 1 - \alpha$ and $\lambda_2 = \alpha$.

We now show how to set the edge variables $y_{uv}^{(i)}$ and $z_{uv}^{(i)}$ for $i = 1, 2$. For ease of exposition let us define $h_{uv} = y_{uv} + z_{uv}$ for any solution to NI-MIP (integral or fractional). For cost vectors $c > 0$, in an optimal LP solution x^* we have $h_{uv}^* = |d_v^* - d_u^*|$, and this value is assigned to z_{uv}^* and

y^*_{uv} according to a greedy cut attack. Define $h^{(i)}_{uv} = |d^{(i)}_v - d^{(i)}_u|$ for $i = 1, 2$. Intuitively, we now follow the decisions made in the fractional solution. If an edge was cut in the fractional solution (usually totally cut), then treat it the same way (proportionally) in any integer solution where it is part of the cut. Formally we set $z^{(i)}_{uv} = (\frac{z^*_{uv}}{h^*_{uv}})h^{(i)}_{uv}$ and $y^{(i)}_{uv} = (\frac{y^*_{uv}}{h^*_{uv}})h^{(i)}_{uv}$ for $i = 1, 2$.

Solutions $x^{(1)}$ and $x^{(2)}$ are feasible solutions for the network inhibition problem without the budget constraint. Any setting for the d_v is acceptable and we have set $h^{(i)}_{uv} = |d^{(i)}_v - d^{(i)}_u|$ for $i = 1, 2$ to satisfy the other constraints.

Theorem 3 *The two solutions $x^{(1)}$ and $x^{(2)}$ are an exact decomposition of the LP optimal x^*: $\sum_{i=1}^{2} \lambda_i x^{(i)} = x^*$*

Proof: We have $d^*_v = \lambda_1 d^{(1)}_v + \lambda_2 d^{(2)}_v$ by construction.

We must now show $z^*_{uv} = \lambda_1 z^{(1)}_{uv} + \lambda_2 z^{(2)}_{uv}$ and $y^*_{uv} = \lambda_1 y^{(1)}_{uv} + \lambda_2 y^{(2)}_{uv}$. First we argue that $\sum_{i=1}^{2} \lambda_i h^{(i)}_{uv} = h^*_{uv}$. We prove this by case analysis. Figure 3.1 illustrates the 4 cases for edge (u, v): it can be in set E_j for $j = 1, 2$, or 3, or it can be contained within one of the vertex sets V_0, V_1, or V_α. If edge (u, v) is in set E_1, then $h^*_{uv} = \alpha$, $h^{(1)}_{uv} = 0$, and $h^{(2)}_{uv} = 1$. Thus $\sum_{i=1}^{2} \lambda_i h^{(i)}_{uv} = (1 - \alpha) * 0 + \alpha * 1 = h^*_{uv}$. We omit the other cases because checking them is straighforward.

We can now complete the proof:

$$\begin{aligned} \sum_{i=1}^{2} \lambda_i z^{(i)}_{uv} &= \sum_{i=1}^{2} \lambda_i (\frac{z^*_{uv}}{h^*_{uv}}) h^{(i)}_{uv} \\ &= \frac{z^*_{uv}}{h^*_{uv}} \sum_{i=1}^{2} \lambda_i h^{(i)}_{uv} \\ &= z^*_{uv}. \end{aligned}$$

The argument for the y_{uv} is completely analogous. □

The $y^{(i)}_{uv}$ and $z^{(i)}_{uv}$ variables for each solution are set to implement the greedy attack strategy on the cut designated by the $d^{(i)}_v$. One could show this algebraically. However, it also follows directly from the optimality of the greedy strategy. The values of the objective functions on the $x^{(i)}$ solutions must equal that of the greedy strategy. Otherwise, there would be a way to improve these solutions, use the improved solutions in a new convex combination, and achieve a better LP solution. There may be multiple optimal greedy solutions if edges tie on the ratio r_{uv}/c_{uv}, but no non-greedy strategy can be optimal.

We need only show that the solution x^* has the required structure.

Theorem 4 *Let x^* be an extreme-point solution to the LP relaxation of NI-MIP. Then there is a $0 < \alpha < 1$ such that for each vertex variable d_v of x^*, we have $d_v \in \{0, \alpha, 1\}$.*

Proof: Let $\hat{\mathcal{S}}$ be the polytope of feasible solutions to NI-MIP after all integrality constraints and the single budget constraint are removed. We first argue that $\hat{\mathcal{S}}$ is naturally integer. We then argue that adding the budget constraint to $\hat{\mathcal{S}}$, creates new vertices only in the middle of edges of $\hat{\mathcal{S}}$. This suffices to prove that x^* can have at most four values represented by its components: $0, \alpha, 1-\alpha$, and 1. We then argue that extreme points of the new polytope have only three values. That is, any vertex containing all four values is necessarily in the interior of the new polytope.

We now argue that $\hat{\mathcal{S}}$ is naturally integer. Consider the minimum-cut polytope formed by the system $Ax \leq b, x \geq 0$. That is, consider MC-IP without the integrality constraints. The matrix A is totally unimodular and therefore all the vertices have integer coordinates. The system for $\hat{\mathcal{S}}$ is $A'x \leq b, x \geq 0$, where we derive A' from A by replacing y_{uv} with $y_{uv} + z_{uv}$. Thus matrix A' is simply matrix A with some duplicated columns.

Matrix A' is totally unimodular (TU). To prove this, it suffices to show that the matrix A^+ derived from a TU matrix A by replicating a single column a^i is also TU. We can build A' from A by a series of such operations. By definition, a matrix is totally unimodular if the determinant of each square submatrix is 0, 1 or -1 [Nemhauser and Wolsey, 1988]. Consider a square submatrix of A^+. If both a^i and its duplicate are in the submatrix, then the determinant is 0. Otherwise, the submatrix is also a submatrix of A and therefore its determinant is 0, 1, or -1. Thus $\hat{\mathcal{S}}$ is totally unimodular and all of its vertices have integer coefficients.

We now add the budget constraint to polytope $\hat{\mathcal{S}}$ to create polytope $\tilde{\mathcal{S}}$. Figure 3.2 shows this geometrically. The vertices of $\tilde{\mathcal{S}}$ could be original vertices of $\hat{\mathcal{S}}$, and therefore integer. All new vertices are created by the intersection of the new hyperplane with an edge of $\hat{\mathcal{S}}$. Therefore, they are points on an external edge of $\hat{\mathcal{S}}$ and are the convex combination of two integer points. Any convex combination of two vectors having only binary values can have at most four values: 0, α, $1-\alpha$ and 1 for some $0 < \alpha < 1$.

To show that vertices of $\tilde{\mathcal{S}}$ in fact have only 3 values for the d_v variables, we show that any convex combination of two extreme points in $\hat{\mathcal{S}}$ that uses all four values over d_v must be strictly inside $\tilde{\mathcal{S}}$. Consider two

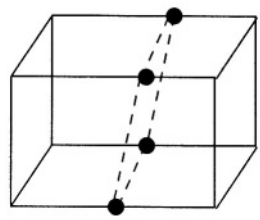


Figure 3.2. When adding a new constraint (cutting plane) to a linear program, new vertices are created only on the edges of the original polytope.

extreme points of $\hat{\mathcal{S}}$ denoted $x^{(1)}$ and $x^{(2)}$. The line segment between $x^{(1)}$ and $x^{(2)}$ is an external edge of $\hat{\mathcal{S}}$ if and only if all points on that line segment have a unique representation as a convex combination of extreme points, those extreme points of course being $x^{(1)}$ and $x^{(2)}$. Suppose there is a point on that line segment that uses exactly four values for d_v. This is only possible if there exists vertex sets $V_i, V_j \subset V$ such that $d_v^{(1)} = 1$ and $d_v^{(2)} = 0$ for all $v \in V_i$ and $d_v^{(1)} = 0$ and $d_v^{(2)} = 1$ for all $v \in V_j$. That is, there is a set of vertices that switches from 0 to 1 as we move from $x^{(1)}$ to $x^{(2)}$ and another set of vertices that switches from 1 to 0. The two solutions agree on the settings for all other vertex variables. Now consider x', the midpoint of segment $(x^{(1)}, x^{(2)})$. We have $x'_v = 1/2$ for all $v \in V_i \cup V_j$. For all other components x'_v, the components $x_v^{(1)}$ and $x_v^{(2)}$ agree. Component x'_v takes this common value. For the vertex variables we have $x' = \frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)}$. However, for the vertex variables we also have $x' = \frac{1}{2}x^{(3)} + \frac{1}{2}x^{(4)}$, where $x^{(3)}$ has $d_v = 0$ for all $v \in V_i \cup V_j$ and $x^{(4)}$ has $d_v = 1$ for all $v \in V_i \cup V_j$.

We now show how to set the edge variables of $x^{(3)}$ and $x^{(4)}$ such that $x' = \frac{1}{2}x^{(3)} + \frac{1}{2}x^{(4)}$ for the edge variables as well. First we describe the structure of an extreme point of $\hat{\mathcal{S}}$. The vertex variables can take on binary values in any combination. Once the vertex variables are set, edges (u, v) going between vertices with different settings must have either z_{uv} or y_{uv} equal to 1, with the other set to 0. Edges (u, v) going between vertices with the same value can have both $z_{uv} = 0$ and $y_{uv} = 0$. However, either one of these variables can also be 1. This will never be optimal for strictly positive capacities. However these are still extreme points of the polytope, and could be optimal solutions for cases where some of the capacities are negative.

There is some flexibility in the choice of edge-variable values once the vertex variables d_v are set. Fix the choice of edge variables for $x^{(1)}$ and $x^{(2)}$ to any values consistent with the extreme-point constraints. We will

set the variables for each edge in $x^{(3)}$ and $x^{(4)}$ so that one will have the value from $x^{(1)}$ and the other will have the value from $x^{(2)}$. There are four classes of vertices: V_0 (those that are zero in both $x^{(1)}$ and $x^{(2)}$), V_1 (one for both extreme points), and V_i and V_j as defined. For ease of exposition, let (V_w, V_q) represent the set of all edges going between sets V_w and V_q.

We describe how to set the z_{uv} variables. Set the y_{uv} in the same manner. For any edge (u, v) that is internal to one of the four vertex sets, and those in $(V_0, V_1), (V_i, V_j), (V_0, V_i)$, and (V_i, V_1) set $z_{uv}^{(3)} = z_{uv}^{(1)}$ and $z_{uv}^{(4)} = z_{uv}^{(2)}$. For edges in the remaining sets (V_0, V_j) and (V_j, V_1), set the variables the other way: $z_{uv}^{(3)} = z_{uv}^{(2)}$ and $z_{uv}^{(4)} = z_{uv}^{(1)}$. These settings obey the constraints on extreme point edge-variable settings and guarantee that $x' = \frac{1}{2}(x^{(1)} + x^{(2)}) = \frac{1}{2}(x^{(3)} + x^{(4)})$.

Therefore, x'does not have a unique representation as a convex combination of extreme points and the entire segment $(x^{(1)}, x^{(2)})$ is in the interior of $\hat{\mathcal{S}}$. Therefore any extreme point of $\hat{\mathcal{S}}$ uses only three values for the vertex variables d_v. □

5. Geometry

In this section, we give a geometric interpretation of the network-inhibition decomposition algorithm. We look at the two-dimensional structure of the set of (integer) feasible (budget, optimal residual capacity) pairs for a particular instance of the network inhibition problem.

Phillips [Phillips, 1993] defined the *cut function* for a particular *s-t* cut. This is the residual capacity as a function of budget, computed by greedily attacking the cut. It is a piecewise-linear convex function, where each segment corresponds to a budget region where some particular edge is partially cut. If we were to plot all exponentially-many cut functions on the same set of axes, the lower envelope of this set of functions determines the optimal attack strategy for each budget (perform the greedy attack for the cut which has the minimum cut-function value at budget B). For a given budget B_a, the set of points where any cut function intersects the line $B = B_a$ corresponds exactly to the set of feasible solutions to the integer program NI-MIP that fully consume budget B_a.

Let F be the set $\{(B_a, C_a)\}$ corresponding to all possible integer solutions to NI-MIP for all interesting budgets B_a. We can compute the maximum useful budget $B_{\max}$ with a single minimum-cut computation using removal values as capacities. Any budget beyond this cannot cause any further damage to the network. Let H be the convex hull of F (in

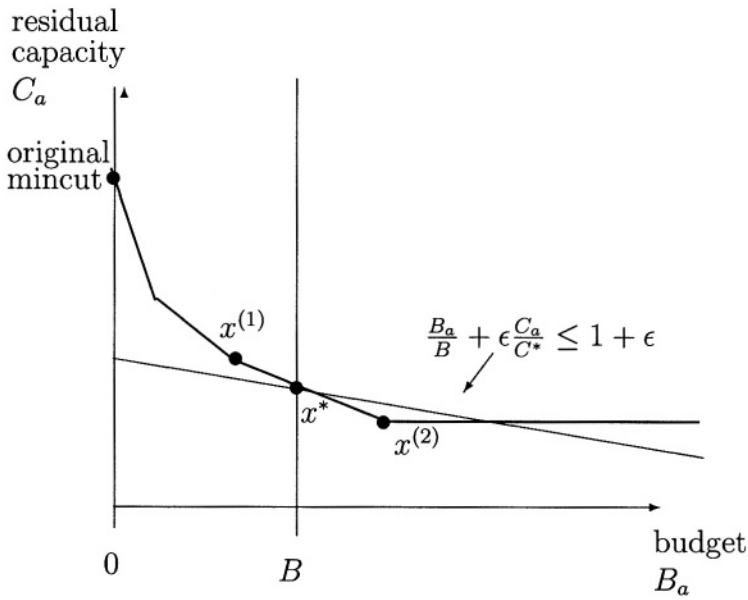


Figure 3.3. The optimal residual capacity in the linear-programming relaxation as a function of budget. The LP optimal value is always on the lower envelope of the set of cut functions. The decomposition finds the endpoints of this convex-hull segment.

the two-dimensional budget-capacity space). The following is a corollary of a more general result proved by Burch et. al.:

Corollary [Burch et al., 2001] *All optimal solutions of the LP relaxation of NI-MIP map to H, the convex hull of the set of integer solutions.*

Figure 3.3 illustrates this geometry. The optimal residual capacity for budget 0 is the capacity of the minimum cut in the original (unharmed) graph. The piecewise-linear convex function is the lower envelope of the set of cut functions (F). The points where the convex-hull segments meet correspond to valid integer solutions, normally each corresponding to a different cut. Points on the segments do not usually correspond to valid integer solutions, but they are convex combinations of integer solutions (the two endpoints), and valid solutions for the linear program. The optimal LP-relaxation for a given budget B corresponds to the point where this convex hull meets the line $B_a = B$.

The following is also a corollary of a more general result proved by Burch et. al.:

Corollary [Burch et al., 2001] *Let (p_1, p_2) be the convex hull segment upon which the image of x^* lies. The two feasible solutions $x^{(1)}$ and $x^{(2)}$ computed by the decomposition procedure correspond to p_1 and p_2.*

Figure 3.3 shows the line $\frac{B_a}{B} + \epsilon\frac{C_a}{C^*} \leq 1 + \epsilon$. Point (B, C^*) lies on this line, where C^* is the objective value of the optimal LP solution for budget B. The line has slope $-\frac{C^*}{\epsilon B}B_a + (\frac{1+\epsilon}{\epsilon})C^*$, which is always negative. At least one of the points p_1, p_2 (corresponding to $x^{(1)}$ and $x^{(2)}$) will lie below line. The algorithm will return the corresponding solution. The choice of ϵ biases the choice between a budget-feasible suboptimal solution and a budget-infeasible superoptimal solution. This is clear from the nature of the bounds we prove. Here it is reflected in the slope of the line used for selecting one of the two candidate solutions.

6. Extensions

In this section we consider two related problems: network inhibition with multiple budgeted properties, and computing a most cost-effective attack.

Suppose there are a number of criteria one wishes to bound when attacking a network, such as time, money, and/or "effort." We can apply the multicriteria-approximation results of [Burch et al., 2001] to extend the results discussed in this chapter.

Suppose we have k criteria, with the ith having budget B_i. Select $\gamma_1, \ldots, \gamma_k, \gamma_{k+1} > 1$ such that

$$\sum_{j=1}^{k+1} \frac{1}{\gamma_j} = 1.$$

These represent our priorities for each criterion, where the $k+1$st criterion is the objective function (residual capacity). In the one-budget case described in this chapter, we have $\gamma_1 = 1 + \epsilon$ and $\gamma_2 = 1 + 1/\epsilon$. Consider a feasible solution to this multicriteria problem with budget B_i' for $i = 1, \ldots, k+1$. Let $\rho_i = B_i'/B_i$ be the amount the budget for the ith criteria is violated. We can always find a solution such that $\sum_{i=1}^{k+1} \frac{\rho_i}{\gamma_i} \leq 1$. As with the two-criteria case we studied in detail, if any of the criteria violates its budget, at least one other criterion is guaranteed under budget.

We now show that we can compute a most cost-effective attack efficiently. Let C_{orig} be the minimum cut in the original network. We wish to find an attack using budget B_a and achieving capacity C_a such that $\frac{C_{\text{orig}} - C_a}{B_a}$ is minimized. We could compute this strategy using parametric search or other binary search methods. However, we can also use a single invocation of our simple, noniterative algorithm.

Consider once again the cost-benefit graph in Figure 3.3. For any feasible attack strategy with cost B_a and residual capacity C_a, the value

we wish to minimize is the slope from the point $(0, C_{\text{orig}})$ to the point (B_a, C_a). Because the lower envelope of the feasible set is a convex function, the optimal point (one with the most negative slope) is the endpoint of the first convex-hull segment. We can compute this by performing the algorithm given in Section 3 using a tiny budget $\delta > 0$.

More generally, we can compute the entire convex hull by "walking" it. Once we have computed the endpoint of the first segment, which uses budget B_a, we can get the next segment by running the algorithm with budget $B_a + \delta$, and so on. This requires polynomial time for each convex-hull segment. We do not yet know the complexity of the convex hull, so it may be more efficient to explore forward and backward from budget points of particular interest.

Acknowledgments

Carl Burch was Supported in part by a National Science Foundation Graduate Fellowship and completed some of this work while at Sandia National Laboratory. Sven Krumke was supported by the German Science Foundation (DFG, grant Gr 883/5-3). Madhav Marathe was supported by the Department of Energy under Contract W-7405-ENG-36. Sandia National Labs is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

[Bar-Noy et al., 2001] Bar-Noy, A., Guha, S., Naor, J., and Schieber, B. (2001). Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing,* 31(2):331–352.

[Boyd and Carr, 1999] Boyd, S. and Carr, R. (1999). A new bound for the ratio between the 2-matching problem and its linear programming relaxation. *Mathematical Programming, Ser A,* 86:499–514.

[Burch et al., 2001] Burch, C., Carr, R. D., Krumke, S. O., Marathe, M., Phillips, C., and Sundberg, E. (2001). Multicriteria approximation through decomposition. (unpublished manuscript).

[Ford and Fulkerson, 1962] Ford, L. R. and Fulkerson, D. R. (1962). *Flows in Networks.* Princeton University Press, Princeton, NJ.

[Ghare et al., 1971] Ghare, P. M., Montgomery, D. C., and Turner, W. C. (1971). Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly,* 18:37–45.

[Helmbold, 1971] Helmbold, R. L. (1971). A counter capacity network interdiction model. Technical Report R-611-PR, Rand Corporation, Santa Monica, CA.

[Krumke et al., 1998] Krumke, S. O., Noltemeier, H., Ravi, S. S., Marathe, M. V., and Drangmeister, K. U. (1998). Modifying networks to obtain low cost subgraphs. *Theoretical Computer Science,* 203(1):91–121. A preliminary version appeared in the Proceedings of the 22nd International Workshop on Graph-Theoretic Concepts in Computer Science, 1996, vol. 1197 of Lecture Notes in Computer Science.

[Marathe et al., 1998] Marathe, M. V., Ravi, R., Sundaram, R., Ravi, S. S., Rosenkrantz, D. J., and Hunt III, H. B. (1998). Bicriteria network design problems. *Journal of Algorithms,* 28(1):142–171.

[McMasters and Mustin, 1970] McMasters, A. W. and Mustin, T. M. (1970). Optimal interdiction of a supply network. *Naval Research Logistics Quarterly,* 17(3):261–268.

[Naor and Schieber, 1997] Naor, J. and Schieber, B. (1997). Improved approximations for shallow-light spanning trees. In *Proceedings of the 38th Annual IEEE Symposium on the Foundations of Computer Science,* pages 536–541.

[Nemhauser and Wolsey, 1988] Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* John Wiley & Sons.

[Phillips, 1993] Phillips, C. (1993). The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing,* pages 776–785.

[Phillips et al., 2002] Phillips, C. A., Uma, R. N., and Wein, J. (2002+). Off-line admission control for general scheduling problems. *Journal of Scheduling,* to appear.

[Schrijver, 1986] Schrijver, A. (1986). *Theory of Linear and Integer Programming.* John Wiley & Sons.

[Shmoys, 1997] Shmoys, D. (1997). Personal communication.

[Srinivasan, 1997] Srinivasan, A. (1997). A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing,* pages 636–643.

[Wood, 1993] Wood, K. (1993). Deterministic network interdiction. *Mathmatical and Computer Modeling,* 17(2):1–18.

Chapter 4

INTERDICTING STOCHASTIC NETWORKS WITH BINARY INTERDICTION EFFORT

Raymond Hemmecke
Mathematics Department
University of California, Davis
ramon@math.ucdavis.edu

Rüdiger Schultz
Institute of Mathematics
Gerhard-Mercator University, Duisburg
schultz@math.uni-duisburg.de

David L. Woodruff
Graduate School of Management
University of California, Davis
dlwoodruff@ucdavis.edu

Abstract We provide formulations, test instances and benchmark results for a new class of network interdiction problems. The formulations are appropriate for computer, terrorist or drug transportation networks where the characteristics of the network cannot be known completely in advance but rather interdiction must be planned based on conjectured configurations. The models support maximization of the expected minimum path length between two nodes, s and t. We also model maximizing the probability of causing the minimum path length to be above a specified threshold. Examples make our formulations concrete and benchmarks establish the computational requirements for solution. Our benchmarks also help quantify the importance of using a special formulation provided for instances when a cut between s and t is the goal.

Keywords: Stochastic Programming, Network Interdiction

1. Introduction

Consider the problem of interdicting the flow of information or goods in a network whose characteristics are not certain, but instead there are a number of possible network configurations. The goal is to maximize the minimum distance between a node s and a node t. We concern ourselves here with the situation where the decision maker can associate a weight, or probability estimate, with each possible configuration of the network.

Since the network is uncertain, the objective must be stated in probabilistic terms such as maximization of the expected minimum distance. We provide a formulation and computational experience for this problem. In addition, we note that for applications in some computer, drug transport, or terrorist networks, expectations are not the appropriate objective. A more appropriate objective is to maximize the probability that the minimum path exceeds a certain length, presumably selected so as to be long enough to render the network essentially unusable.

1.1 Deterministic Formulation

To formalize these notions we begin with models based on those used in previous interdiction work [1, 2, 3, 4, 5]. Consider a directed graph given by (N, A), where an interdictor intends to maximize the minimum distance from node s to node t. We are given as data the node-arc incidence matrix of the network, G, which includes an artificial arc (t, s). The arc distances are given in a vector c. Decision variables include a vector, x, that gives interdiction effort for each arc, which lengthens the arc at a rate given as data by the vector d. For the moment we summarize the constraints on x using the set X, although our intention is to mainly consider cases where the interdiction is binary and subject to a system of linear budget constraints.

We set $c_{ts} = d_{ts} = 0$. The model makes use of the flow vector y as a decision variable for the network operator. The interdictor's problem is:

$$\max_{x \in X} \min_{y} \sum_{k \in A} (c_k + d_k x_k)\, y_k \quad (F)$$

subject to:

$$\begin{aligned} Gy &= 0, \\ y_{ts} &= 1, \\ y_k &\geq 0,\ k \in A. \end{aligned}$$

Using the dual of the inner problem, we obtain:

$$\max_{x \in X} \max_{\pi} \pi_t - \pi_s \quad (F')$$

subject to:

$$\begin{aligned} -\pi_i + \pi_j &\leq c_{ij} + d_{ij} x_{ij}, \ (i,j) \in A \\ \pi_s &= 0. \end{aligned}$$

Notice that in formulation (F), we used a single index for the arcs: $k \in A$. In (F') we used the node pair that defines the arc: $(i,j) \in A$. They are equivalent, but the node pairs are more appropriate for the dual and the direct arc indices are more appropriate for the primal formulations.

Our intention is to provide budget constraints for interdiction decision variables, which we will consider to be binary. Hence, we envision something like $X = \{x : Qx \leq B, x \in \{0,1\}^n\}$ with a matrix Q and a vector B provided as data, where n is the number of arcs.

Formulation (F) considers interdiction only on arcs, but in many computer networks, drug transport, or terrorist networks it is the nodes that can be most easily interdicted. To model interdiction on nodes, each interdictable node must be replaced by two artificial nodes: one for all incoming arcs to the original node and another for outgoing. An artifical, zero-length, interdictable arc must connect the two nodes. The budget impacts of interdiction on the original node then become the budget impact of interdicting the artificial arc between the nodes. The effect of interdicting the node becomes the effect on the artificial arc as captured by d_k. In the sequel, when we refer to a "node" it is understood that we may also mean the artificial arc representing the node in formulation (F).

1.2 Stochastic Formulations

Case where the interdiction efforts are stochastic are treated in [1] and [4]; our primary interest here is in the case where the network is uncertain. We have a set of scenarios Ω and for every scenario $\omega \in \Omega$ a probability $Pr(\omega)$. Associated with each scenario is a network given by $(N(\omega), A(\omega))$.

For many interdiction problems, it makes sense to plan for interdiction of nodes that may not be connected to the network in every scenario (especially given that the true network may never be revealed). Consequently, we use the notation (N, A) to refer to

$$N = \bigcup_{\omega \in \Omega} N(\omega) \text{ and } A = \bigcup_{\omega \in \Omega} A(\omega)$$

and G to refer to the corresponding incidence matrix. We take n to be the number of arcs in (N, A) with the data for our formulations defined over these arcs. Thus, we can refer to a budget constraint set $X = \{x : Qx \leq B, x \in \{0,1\}^n\}$ that does not depend on the scenario. The length of the arcs will, of course, depend on the scenario as will the flow decisions. At least conceptually, we can set $c_k(\omega) = \infty$ and $d_k(\omega) = 0$ for arcs $k \notin A(\omega)$.

1.2.1 Maximizing Expected Length. Our first stochastic model addresses the problem of maximizing the expected minimum path length between nodes s and t. Since we have a discrete set of scenarios, we can write the expectation explicity as a sum:

$$\max_{x \in X} \sum_{\omega \in \Omega} Pr(\omega) \left[\min_{y} \sum_{k \in A} \left(c_k(\omega) + d_k(\omega) x_k\right) y_k(\omega) \right] \quad (E)$$

subject to:

$$\begin{aligned} Gy(\omega) &= 0, \ \omega \in \Omega, \\ y_{ts}(\omega) &= 1, \ \omega \in \Omega, \\ y_k(\omega) &\geq 0, \ k \in A, \ \omega \in \Omega. \end{aligned}$$

Formulation (E), which is based on formulation (F), provides a clear statement of our problem. However, the problem can be solved in straightforward fashion if we rely on a reformulation based on (F'), which is

$$\max_{x \in X} \sum_{\omega \in \Omega} Pr(\omega) \left[\max_{\pi} \pi_t(\omega) - \pi_s(\omega) \right] \quad (E')$$

subject to:

$$\begin{aligned} -\pi_i(\omega) + \pi_j(\omega) &\leq c_{ij}(\omega) + d_{ij}(\omega) x_{ij}, \ (i,j) \in A(\omega), \ \omega \in \Omega, \\ \pi_s(\omega) &= 0, \ \omega \in \Omega. \end{aligned}$$

1.2.2 Maximize the Probability of Sufficient Disruption. The goal is to maximize the probability that the minimum length path from s to t exceeds φ, where φ is given as data. This results in the following optimization problem:

$$\max_{x \in X} \left[Pr \left(\min_{y} \sum_{k \in A} \left(c_k(\omega) + d_k(\omega) x_k\right) y_k(\omega) \right) \geq \varphi \right] \quad (P)$$

subject to:

$$\begin{aligned} Gy(\omega) &= 0,\ \omega \in \Omega, \\ y_{ts}(\omega) &= 1,\ \omega \in \Omega, \\ y_k(\omega) &\geq 0,\ k \in A,\ \omega \in \Omega. \end{aligned}$$

Formulation (P), which is based on formulation (F), provides a clear statement of our problem. However, exposition of our solution method is much more straightforward if we rely on an equivalent formulation based on (F'), which is

$$\max_{x \in X} Pr\left(\left[\max_{\pi} \pi_t(\omega) - \pi_s(\omega)\right] \geq \varphi\right) \quad (P')$$

subject to:

$$\begin{aligned} -\pi_i(\omega) + \pi_j(\omega) &\leq c_{ij}(\omega) + d_{ij}(\omega)x_{ij},\ (i,j) \in A(\omega),\ \omega \in \Omega, \\ \pi_s(\omega) &= 0,\ \omega \in \Omega. \end{aligned}$$

The deterministic equivalent of this problem can be written by making use of an additional vector of variables $\theta_\omega,\ \omega \in \Omega$, which indicate for each scenario if the threshold path length is exceeded. Making use of the notion that probabilities are expectations of indicator functions, we write:

$$\max_{x \in X} \sum_{\omega \in \Omega} Pr(\omega)\theta_\omega \quad (D')$$

subject to:

$$\begin{aligned} -\pi_i(\omega) + \pi_j(\omega) &\leq c_{ij}(\omega) + d_{ij}(\omega)x_{ij},\ (i,j) \in A(\omega),\ \omega \in \Omega, \\ \pi_s(\omega) &= 0,\ \omega \in \Omega, \\ (\varphi - \pi_t(\omega))/M &\leq 1 - \theta_\omega,\ \omega \in \Omega, \\ \theta_\omega &\in \{0,1\},\ \omega \in \Omega. \end{aligned}$$

where M is the ubiquitous "big M," which is provided as data and is large enough to exceed all possible values of the absolute value of $\pi_t(\omega) - \varphi$.

2. Example

Figure 4.1 shows a small example of the sort of problem our models are intended to address. There are three scenarios. The first scenario contains all of the nodes and arcs in the network between nodes $s = 1$ and $t = 6$. In many settings, particularly involving drugs or terrorists, it might be possible to know the nodes and arcs in a network, but not know if all of them are involved in the flow of interest. This is captured

by the other two scenarios where the even numbered or odd numbered nodes (other than s and t) respectively, are not part of the traffic of interest. As an aside, we note that the nodes and arcs are labeled with numbers only as a matter of convenience in making statements such as "odd numbered nodes" and in connecting arc labels with vector indexes. It is not our intention to put an order or value on the nodes or arcs; any type of label would suffice for our models.

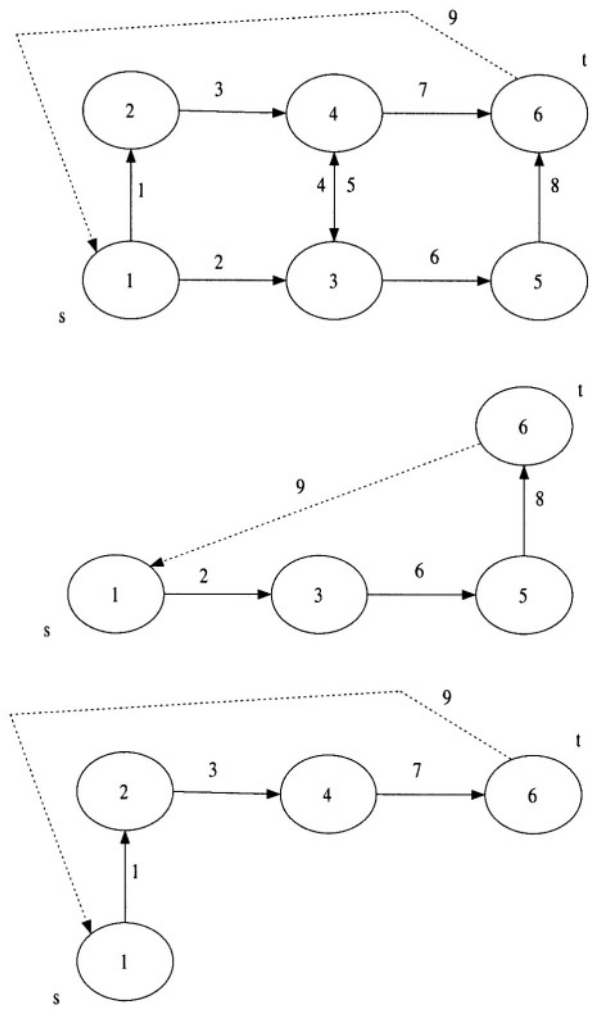


Figure 4.1. The Three Scenarios for the Example

As already noted, when node interdiction is possible the problem must be transformed in order to make use of the models that we have pre-

sented. Note that nodes s and t are not interdictable. If they were interdictable, then the appropriate problem statements would be quite different. Our models are appropriate when s and t are special nodes that cannot be directly interdicted such as when they are proxies for an organization or people in a geographic area.

To specify a problem instance we need to supply values for the data elements such as the following data:

- Arc lengths: $c_k = 1$ for arcs k in the original arc set, and $c_k = 0$ for arcs added to allow the interdiction of an original node to be represented by arc interdiction. This suggests that the network operator's goal is to minimize the number of hops. Of course, for other examples the c_k values might vary, although they would often be zero for the arcs introduced to stand in for nodes.

- Penalties for interdiction: $d_k = 0$ for arcs k in the original arc set and $d_k = 20$ for arcs added to allow the interdiction of an original node to be represented by arc interdiction. This suggests that interdiction of the nodes in the original network (i.e., the nodes shown in Figure 4.1) will dramatically increase the path length while interdiction of the arcs in the original network is not possible for this example.

- Scenario probabilities: Suppose that the full graph (i.e., scenario one) is least likely and the other two graphs are equally likely. We might model this using probabilities of 0.2, 0.4 and 0.4 for the respective scenarios.

- Threshold: A threshold such as $\varphi = 4$ for problem (P) would result in maximizing the probability that the flow was disrupted by the interdiction. If the goal is to maximize the probability that only an interdicted route will be available then the non-zero d_k values must be sufficiently large (presumably all the same value, d, for all interdictable arcs, k) so that $\varphi = d$ will provide the desired effect. The meaning of "sufficiently large" depends on the data, but a value equal to the longest path length through the uninhibited network clearly suffices.

- Interdiction budget constraints: The data to operationalize the expression $x \in X$ must supplied. If we assume binary interdiction subject to a general budget constraint, we have $X = \{x : Qx \leq B, x \in \{0,1\}^n\}$. So for a specific, simple example we use a single row of 13 ones for the matrix Q and right hand side value of $B = 1$ so that we would be required to select a single arc for interdiction. One could imagine a situation where there were multiple

resource classes each with a row in Q to indicate the quantities of the resources that would need to be combined to interdict each arc and the right hand side would indicate the budget for each type of resource.

Figure 4.2 shows the instantiation of (D') for this example. In order to model a situation where the nodes shown in Figure 4.1 can be interdicted, they must be converted to arcs in the formulation used (for simplicity, arcs are added for nodes s and t as well; however, these particular arcs cannot be interdicted). This is done by replacing node i as shown in Figure 4.1 with nodes $2i-1$ and $2i$ in the problem formulation. These nodes are connected by an arc labeled as $(2i-1, 2i)$ in Figure 4.2 because this is a dual formulation where we have chosen to label arcs using their terminal nodes (while Figure 4.1 used primal labels).

With data as suggested (i.e., $B = 1$, $\varphi = 4$, probabilities 0.2, 0.4, and 0.4, $c = 1$ and $d = 0$ for the original arcs and $c = 0$ and $d = 20$ for the arcs added to represent original nodes), arc (9,10) is selected for interdiction. I.e., node 5 from the original graph is selected for interdiction. This results in a probability of 0.4 of exceeding the threshold, which makes sense. With a budget for interdiction of only one arc, the path can be lengthened for only one scenario. If the budget is increased to 2, then flow can be impeded in all scenarios. The threshold value $\varphi = 4$ is exceeded for this particular problem whenever the shortest path has been altered by the interdiction.

For this example, we can set $\varphi = 2d = 40$ to test for the event that the shortest path has been interdicted in two places. In general, of course, the value might have to be higher than $2d$, but in this case d is very high compared to the length of the paths. If the budget is $B = 2$ then the objective value is 0.4 again, as is the case for a budget of 3. A budget of 4 yields 1 as an objective function value. An important special case is when the goal is disconnect the network. For the example, this can be accomplished with a threshold $\varphi > 20$. This special case is treated more generally in the next section.

3. A Special Case: Disconnection as the Threshold

Suppose that our objective is maximize the probability of disconnecting the network, which we take to mean that all paths are interdicted at least once. We can model the problem by introducing an additional arc θ from s to t, assign to it a cost of 1 and assign to all other arcs zero costs.

Minimization of this special cost function leads to an optimal objective value of 0 if there is a path from s to t that does not use θ, and of 1 if the original graph (without θ) is disconnected (no directed path from s to t).

This problem can be stated as:

$$\min_y y_{st}$$

subject to:

$$\begin{aligned} Gy &= 0, \\ y_{ts} &= 1, \\ y_k &\geq 0,\ k \in A, \end{aligned}$$

As the dual of this problem we obtain:

$$\max_\pi \pi_t - \pi_s$$

subject to:

$$\begin{aligned} -\pi_i + \pi_j &\leq 0,\ (i,j) \in A \setminus \{\theta\}, \\ -\pi_s + \pi_t &\leq 1,\ \theta = (s,t), \\ \pi_s &= 0. \end{aligned}$$

Note that $\pi_s = 0$ and $-\pi_i + \pi_j \leq 0, k = (i,j) \in A \setminus \{\theta\}$ imply $\pi_j \leq 0$ for all nodes j such that there is a directed path from s to j with arcs from $A \setminus \{\theta\}$. Thus, if s is connected to t with arcs from $A \setminus \{\theta\}$, an optimal solution is given by putting π identically to 0. Otherwise, if s is not connected to t with arcs from $A \setminus \{\theta\}$, an optimal solution is given by $\pi_j = 0$, if s is connected to j, and $\pi_i = 1$, if i is connected to t with arcs from $A \setminus \{\theta\}$. Therefore, we may assume without loss of generality that $\pi_i \leq 1$ for all nodes i.

Now let us turn back to interdiction of arcs. Assume that arc (i,j) is interdicted. Thus, the inequality $-\pi_i + \pi_j \leq 0$ should be removed from the constraint system. As $\pi_i \leq 1$ for all nodes i, this can be achieved by replacing $-\pi_i + \pi_j \leq 0$ with $-\pi_i + \pi_j \leq x_{ij}$. If $x_{ij} = 1$, this inequality becomes redundant.

Therefore, we may model the interdiction problem as follows:

$$\max_{x \in X} \max_\pi \pi_t - \pi_s$$

subject to:

$$\begin{aligned} -\pi_i + \pi_j &\leq x_{ij},\ (i,j) \in A \setminus \{\theta\}, \\ -\pi_s + \pi_t &\leq 1,\ \theta = (s,t), \\ \pi_s &= 0. \end{aligned}$$

The optimal objective function value is 1 if and only if the interdiction led to a disconnected (original) network, and 0 otherwise. Hence, the stochastic program that maximizes the probability of successful interdiction can be modeled as:

$$\max_{x \in X, \pi(\omega)} \sum_{\omega} \Pr(\omega)\pi_t(\omega) \quad (D'')$$

subject to:

$$\begin{aligned} -\pi_i(\omega) + \pi_j(\omega) &\leq x_{ij},\ (i,j) \in A \setminus \{\theta\}, \\ -\pi_s(\omega) + \pi_t(\omega) &\leq 1,\ \theta = (s,t), \\ \pi_s(\omega) &= 0. \end{aligned}$$

4. Benchmarks

In order to establish some benchmarks concerning the computational requirements for solving instances of the formulations given in this paper, we conducted a number of experiments using simulated data. The networks have nodes s and t with three layers of nodes in between. The first layer has n_1 nodes that are known to be connected to node s, the third layer has n_3 nodes that are known to be connected to node t. The second, or middle, layer of n_2 nodes has connections with the first and third layers that are uncertain. For each scenario, the arcs connecting the first with the second layer and those connecting the second with the third layer are specified.

Figure 4.3 shows a particular scenario where the realization is for 7 arcs between the middle layer and the outer two layers. These arcs are shown with broken lines to make it easy to identify them. Note that it is possible for nodes to be disconnected in a scenario if there are no arcs connecting the nodes. In the Figure, this has happened to the top node in the first layer and the bottom node in the third layer. In other scenarios, presumably, these nodes are connected.

4.1 Expected Value Formulation

Table 4.1 shows the results of maximization of the expected value of the minimum path from s to t using formulation (E'). For these problems, the value of c is set to one for every arc and the value of d is ten for every node (which is represented by an artificial arc in formulation (E')). The first column of the table gives the size of the instance in the form $n_1 \times n_2 \times n_3$. The next column indicates the expected number of arcs that are randomly generated between the first and second and second and third layers of nodes in each scenario (for Figure 4.3 the realization

is seven so the expected value would be presumably near this value). The next column, labeled $|\Omega|$, gives the number of scenarios. The budget for interdiction (the value of B, which is the right hand side of the budget constraint) is the column labeled "Budget." The final column gives the CPU time for XPress-MP, version 13 running on a 1GHz Pentium III processor with default parameter settings.

For these instances, interdiction is allowed only on the nodes, so the number of binary variables is equal to the total number of nodes that can be interdicted (e.g., 30 for a problem of size $10 \times 10 \times 10$). This highlights the fact that although the number of binary variables is modest, these problems require considerable effort in order to prove optimality. For the last example, there are only 45 binary variables, but after 175,000 seconds there is still a significant gap: the best solution found has objective function value 23 but the upper bound was 133.72 when the run was terminated.

Table 4.1. Results for a Few Expected Value Instances

Size	arcs/sc.	$\|\Omega\|$	Budget	CPU Time
$10 \times 10 \times 10$	50	100	7	2641s
$10 \times 10 \times 10$	100	100	7	3181s
$10 \times 10 \times 10$	150	100	7	$\sim$ 6h
$15 \times 15 \times 15$	80	100	10	$>$ 48h

For further experimentation, we created a set of benchmark instances that are appropriate for problems (E'), (D'), and (D"). In all instances, d was set to 20 for all arcs that were added to allow for interdiction of nodes in the three layers between s and t. We used a budget of $B = 9$. For each network configuration, five instances were generated. For each instance, 100 scenarios were generated.

Table 4.2 shows the results for instances of problem (E'). The first column of the table gives the size of the instance in the form $n_1 \times n_2 \times n_3$. The next column indicates the expected number of arcs that are randomly generated between the first and second and second and third layers of nodes in each scenario. The optimum objective function value for each replicate is given in the column labeled "Obj. Val." The final two columns give the CPU time for XPress-MP, version 13 running on a 1GHz Pentium III processor and the number of branch and bound nodes explored.

Table 4.2. Results Expected Value Problems for the Benchmark Set

Size	arcs/sc.	Obj. Val.	CPU Sec	BB Nodes
10 × 10 × 10	25	12.4	106	260
		13.8	68	178
		12.6	92	228
		13.2	89	229
		13.2	71	181
10 × 20 × 10	25	15	10	57
		14.8	11	57
		14.2	13	63
		14.6	11	61
		14.6	12	63
10 × 20 × 10	50	7.2	995	1,296
		7.4	811	1,065
		7	2,140	3,449
		6.4	2,203	3,202
		7.2	845	940

4.2 Maximum Probability Formulation

Table 4.3 shows the results of using the benchmark instances for problem (D'). These are the same instances that were used for the experiments shown in Table 4.2, but we used the probability formulations. In all instances, d was set to 20 for all arcs that were added to allow for interdiction of nodes in the three layers between s and t. Consequently, by using $\varphi = 23$, we able to solve identical instances using formulations (D') and (D"), which gives us some idea of the relative importance of using (D") when that is appropriate. We used a budget of $B = 9$. For each network configuration, five instances were generated. For each instance, 100 scenarios were generated. The first column of the table gives the size of the instance in the form $n_1 \times n_2 \times n_3$. The next column indicates the expected number of arcs that are randomly generated between the first and second and second and third layers of nodes in each scenario.

The number of scenarios that are disconnected in the optimal solution for each replicate is given in the column labeled "Scen. Disc." Since there are 100 scenarios and the objective is to maximumize the probability of disconnection, this corresponds to 100 times the optimal objective value. The next two columns give the CPU time in seconds for (D') and (D") respectively for XPress-MP, version 13 running on a 1GHz Pentium III processor with default parameter settings. The final two columns give the number of branch and bound nodes explored for (D') and (D") respectively.

Table 4.3. Benchmark Results for the Maximum Probability Problem

Size	arcs/sc.	Scen. Disc.	CPU	CPU"	BB Nodes	BB Nodes"
10 × 10 × 10	25	47	354	198	1,724	451
		54	180	160	839	379
		48	291	184	1,068	412
		51	247	153	1,494	372
		51	228	118	1,228	269
10 × 20 × 10	25	60	24	17	308	80
		59	29	20	445	107
		56	29	25	441	103
		58	48	23	652	103
		58	33	22	606	99
10 × 20 × 10	50	21	21,815	1,671	79,275	1,721
		22	18,293	1,424	58,459	1,772
		20	42,238	1,909	235,497	2,790
		17	136,486	3,441	873,843	4,371
		21	20,204	1,506	89,094	1,680

As anticipated, formulation (D") is much more tractable. For all replicates, lower effort was required for solution. The importance of using (D") when appropriate, increases dramatically with the problem size.

5. Conclusions

We have given formulations, test instances and benchmark computational results for a new class of network interdiction problems. The formulations are appropriate for computer, terrorist or drug transportation networks where the characteristics of the network cannot be known completely in advance but rather interdiction must be planned based on conjectured configurations. We have presented formulations that support minimization of the expected path length between two arcs and formulations that maximize the probability of causing the minimum path length to be above a specified threshold.

The law of large numbers suggests that minimizing the expected value is appropriate for problems that are solved, and whose solutions are implemented, many times. Many problems in the interdiction of stochastic networks are not subject to repeated solution and so the appropriate objective is maximizing the probability that the network will not function properly. This is particularly true when unimpeded operation of the network is disastrous. In these cases, parametric variation of the budget to get the desired threshold with adequate probability or to generate tradeoff curves would be valuable. For the case where the appropriate

threshold is disconnection of all paths between nodes s and t, we have provided a reformulation that leads to a great reduction in the computational effort required for solution.

We have provided examples to make our formulations concrete and benchmarks to establish the computational requirements for solution. Our benchmarks also help quantify the importance of using the appropriate formulation when a cut between s and t is the goal. We hope that these benchmarks will provide a starting point for additional work on this class of problems. For example, decomposition as suggested in [6] should be investigated.

The broad class of problems associated with network interdiction has received increased attention due to changing conditions in the world. This paper introduced a class of problems where the network is uncertain and the objective is to maximize the probability of successful interdiction of the network, which is intended to contribute to ongoing research in network interdiction.

References

[1] Cormican, K.J., Morton, D.P. and Wood, R.K., "Stochastic Network Interdiction," *Operations Research,* 46 (1998) 184-197.

[2] Fulkerson, D.R. and Harding, G.C., "Maximizing the Minimum Source-Sink Path subject to a Budget Constraint", *Mathematical Programming,* 13 (1977) 116-118.

[3] Golden, B.L., "A Problem in Network Interdiction," *Naval Research Logistics Quarterly,* 25 (1978), 711-713.

[4] Israeli, E. and R.K. Wood, "Shortest-path Network Interdiction," *Networks,* to appear.

[5] Phillips, C., "The Network Inhibition Problem," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing,* 1993, pp.776-785.

[6] Schultz, R., "Probability Objectives in Stochastic Programs with Recourse," Technical Report, Mathematics Institute, Gerhard-Mercator University, Duisburg, Germany, 2002.

$$
\begin{array}{rcl}
\max\limits_{x\in X,\pi(1),\pi(2)\pi(3)} & \Pr(1)\theta_1 + \Pr(2)\theta_2 + \Pr(3)\theta_3 & \\
\text{subject to: } x_{1,2} & = & 0, \\
x_{11,12} & = & 0, \\
x_{12,1} & = & 0, \\
-\pi_1(1) + \pi_2(1) & \leq & c_{1,2}(1) + d_{1,2}(1)x_{1,2}, \\
-\pi_3(1) + \pi_4(1) & \leq & c_{3,4}(1) + d_{3,4}(1)x_{3,4}, \\
-\pi_5(1) + \pi_6(1) & \leq & c_{5,6}(1) + d_{5,6}(1)x_{5,6}, \\
-\pi_7(1) + \pi_8(1) & \leq & c_{7,8}(1) + d_{7,8}(1)x_{7,8}, \\
-\pi_9(1) + \pi_{10}(1) & \leq & c_{9,10}(1) + d_{9,10}(1)x_{9,10}, \\
-\pi_{11}(1) + \pi_{12}(1) & \leq & c_{11,12}(1) + d_{11,12}(1)x_{11,12}, \\
-\pi_{13}(1) + \pi_{14}(1) & \leq & c_{13,14}(1) + d_{13,14}(1)x_{13,14}, \\
-\pi_2(1) + \pi_3(1) & \leq & c_{2,3}(1) + d_{2,3}(1)x_{2,3}, \\
-\pi_2(1) + \pi_5(1) & \leq & c_{2,5}(1) + d_{25}(1)x_{2,5}, \\
-\pi_4(1) + \pi_7(1) & \leq & c_{4,7}(1) + d_{47}(1)x_{4,7}, \\
-\pi_6(1) + \pi_7(1) & \leq & c_{6,7}(1) + d_{67}(1)x_{6,7}, \\
-\pi_8(1) + \pi_5(1) & \leq & c_{8,5}(1) + d_{85}(1)x_{8,5}, \\
-\pi_6(1) + \pi_9(1) & \leq & c_{6,9}(1) + d_{69}(1)x_{6,9}, \\
-\pi_8(1) + \pi_{11}(1) & \leq & c_{8,11}(1) + d_{8,11}(1)x_{8,11}, \\
-\pi_{10}(1) + \pi_{11}(1) & \leq & c_{10,11}(1) + d_{10,11}(1)x_{10,11}, \\
-\pi_{12}(1) + \pi_1(1) & \leq & c_{12,1}(1) + d_{12,1}(1)x_{12,1}, \\
\pi_1(1) & = & 0, \\
(\phi - \pi_{12}(1))/M & \leq & 1 - \theta_1, \\
-\pi_1(2) + \pi_2(2) & \leq & c_{1,2}(2) + d_{1,2}(2)x_{1,2}, \\
-\pi_3(2) + \pi_4(2) & \leq & c_{3,4}(2) + d_{3,4}(2)x_{3,4}, \\
-\pi_5(2) + \pi_6(2) & \leq & c_{5,6}(2) + d_{5,6}(2)x_{5,6}, \\
-\pi_7(2) + \pi_8(2) & \leq & c_{7,8}(2) + d_{7,8}(2)x_{7,8}, \\
-\pi_9(2) + \pi_{10}(2) & \leq & c_{9,10}(2) + d_{9,10}(2)x_{9,10}, \\
-\pi_{11}(2) + \pi_{12}(2) & \leq & c_{11,12}(2) + d_{11,12}(2)x_{11,12}, \\
-\pi_{13}(2) + \pi_{14}(2) & \leq & c_{13,14}(2) + d_{13,14}(2)x_{13,14}, \\
-\pi_2(2) + \pi_5(2) & \leq & c_{2,5}(2) + d_{2,5}(2)x_{2,5}, \\
-\pi_6(2) + \pi_9(2) & \leq & c_{6,9}(2) + d_{6,9}(2)x_{6,9}, \\
-\pi_{10}(2) + \pi_{11}(2) & \leq & c_{10,11}(2) + d_{10,11}(2)x_{10,11}, \\
-\pi_{12}(2) + \pi_1(2) & \leq & c_{12,1}(2) + d_{12,1}(2)x_{12,1}, \\
\pi_1(2) & = & 0, \\
(\phi - \pi_{12}(2))/M & \leq & 1 - \theta_2, \\
-\pi_1(3) + \pi_2(3) & \leq & c_{1,2}(3) + d_{1,2}(3)x_{1,2}, \\
-\pi_3(3) + \pi_4(3) & \leq & c_{3,4}(3) + d_{3,4}(3)x_{3,4}, \\
-\pi_5(3) + \pi_6(3) & \leq & c_{5,6}(3) + d_{5,6}(3)x_{5,6}, \\
-\pi_7(3) + \pi_8(3) & \leq & c_{7,8}(3) + d_{7,8}(3)x_{7,8}, \\
-\pi_9(3) + \pi_{10}(3) & \leq & c_{9,10}(3) + d_{9,10}(3)x_{9,10}, \\
-\pi_{11}(3) + \pi_{12}(3) & \leq & c_{11,12}(3) + d_{11,12}(3)x_{11,12}, \\
-\pi_{13}(3) + \pi_{14}(3) & \leq & c_{13,14}(3) + d_{13,14}(3)x_{13,14}, \\
-\pi_2(3) + \pi_3(3) & \leq & c_{2,3}(3) + d_{2,3}(3)x_{2,3}, \\
-\pi_4(3) + \pi_7(3) & \leq & c_{4,7}(3) + d_{4,7}(3)x_{4,7}, \\
-\pi_8(3) + \pi_{11}(3) & \leq & c_{8,11}(3) + d_{8,11}(3)x_{8,11}, \\
-\pi_{12}(3) + \pi_1(3) & \leq & c_{12,1}(3) + d_{12,1}(3)x_{12,1}, \\
\pi_1(3) & = & 0, \\
(\phi - \pi_{12}(3))/M & \leq & 1 - \theta_3, \\
\theta_1, \theta_2, \theta_3 & \in & \{0, 1\}.
\end{array}
$$

Figure 4.2. Formulation (D') for the Example

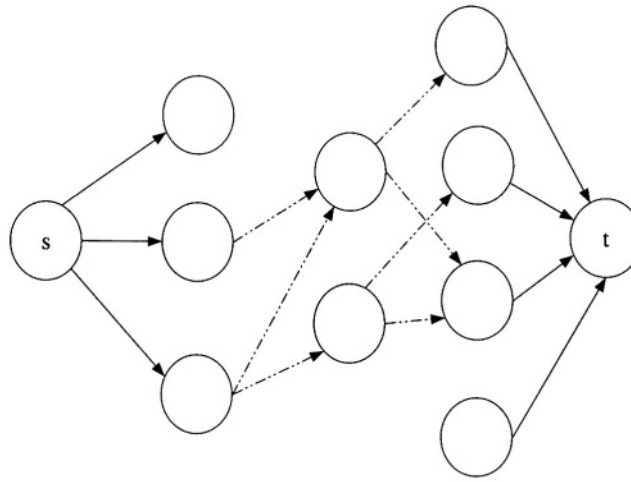


Figure 4.3. An Example of a Benchmark Scenario with $n_1 = 3$, $n_2 = 2$, and $n_3 = 4$

Chapter 5

STOCHASTIC BATCH-SIZING PROBLEMS: MODELS AND ALGORITHMS.

Guglielmo Lulli
Dept. of Statistics, Probability and Applied Statistics
University of Rome "La Sapienza" - P.le A. Moro 5, I-00185 Rome, Italy
guglielmo.lulli@uniroma1.it

Suvrajeet Sen
Dept. of Systems and Industial Engineering
University of Arizona - Tucson AZ85721, USA.
sen@sie.arizona.edu

Abstract In this paper we study the stochastic batch sizing problems. We provide a unifying treatment of the problem, in which we formulate a multi-stage recourse problem as well as a probabilistically constrained problem. The solution approach that we adopt for these problems may be classified as a branch and price (B&P) method. Through our computational experiments turns out that the proposed B&P methodology is quite effective for the recourse constrained model. We also demonstrate how trade-offs between cost and reliability can be investigated for the stochastic batch-sizing problem.

Keywords: Stochastic Batch-Sizing Problem, Probabilistic Constraints, Branch-and-Price Algorithm.

Introduction

Many practical decisions problems can be modeled as mathematical programs. Typical applications may be found in the areas of industrial production, transportation, agriculture, engineering, and many others.

In many of these modeling situations, it is unreasonable to assume that the problem parameters are deterministically known. Operations problems often involve parameters (e.g. demand, lead-time etc.) that are unknown at the time of planning, and their values are unveiled over time and are modeled as random variables. For instance, future productivities in production planning, inflows into a reservoir for a hydro power plant, demands at various nodes in a transportation network, are all subject to variation. The need to explicitly model uncertainty leads to the so-called stochastic programming (SP) problems. SP problems are aimed at determining non-anticipative (here-and-now) decisions that must be taken prior to knowing the realization of the random variables. These decisions are required to be made in such a way that total expected costs or revenues (from here-and-now decisions and possible recourse actions) are optimized.

In this paper, we discuss alternative models of the stochastic batch-sizing problem. The deterministic version of these problems belongs to the class of economic lot-sizing (ELS) models, and may be stated as follows: given a demand and a cost structure for T time periods, the object of production planning is to minimize the total production and inventory costs. The papers of Manne [15] and Wagner and Whitin [21] are the seminal contributions in this area of research. Manne formulated the multiple item capacitated version of the problem as a mixed integer linear program and proposed solving a linear programming approximation of it. Wagner and Whitin [21] studied the uncapacitated model with fixed set-up cost and linear inventory and production costs. Their main contribution was in demonstrating that an optimal replenishment policy is one in which production is undertaken when inventory is zero. Furthermore, they proposed an efficient forward dynamic programming algorithm to solve the problem. In[10], Krarup and Bilde provided a formulation of the economic lot-sizing problem which describes the convex hull of the corresponding polyhedron. Starting with the seminal works of Manne and of Wagner and Whitin, a broad variety of ELS models have been studied in the literature. These models include the ones in which backlogging may be allowed, production and inventory capacities may be finite, start-up costs may be non-zero, etc. On this subject we can mention the work of Leung, Magnanti and Vachani [12] and of Hsu [8]. Leung, Magnanti and Vachani [12] studied the single-item capacitated lot-sizing problem. In their paper, the authors studied the polyhedral structure of the corresponding integer programming formulation. Moreover, they introduced a set of valid inequalities for the problem and showed that they define facets of the underlying integer programming polyhedron. Such inequalities can be used effectively to

develop an efficient branch-and-cut procedure. Hsu [8] discussed situations where the traditional ELS models are not applicable, and proposed a new model with concave production and inventory functions. The author explored the structural properties of the optimal solution, which were used to develop a polynomial time dynamic programming algorithm. This summary provides some insights on the results achieved in this area of research. For a more complete description of the state of the art on deterministic ELS models, and several extensions the reader may refer to Martin [16], Kuik et al. [9] and Aggarwal and Park [1].

When data for such models is uncertain, deterministic techniques such as those mentioned above must be extended to accommodate uncertainty. A simple approach is to apply multi-period models in a rolling horizon environment. When this technique is applied, the only first period's decisions are implemented and the model is rerun after one period with an updated data set. Baker [3] showed that simple lot-sizing heuristic, like Silver/Meal or Groff's heuristic, may outperform solutions obtained applying exact algorithms, such as Wagner and Whitin, in a rolling horizon environment. Moreover Baker observed that the performance of rolling schedules depends on the length of the planning horizon, cost structures and demand pattern. In particular, the length of the planning horizon is a crucial parameter in such models. Stadtler [20] presented modified rolling horizon models in order to obtain solutions that are at least as good as the heuristics mentioned above, and fairly insensitive to the length of the planning horizon. Only recently has there been an explicit attempt to state the stochastic version of ELS models as a stochastic programming problem. In a couple of papers, Lokketangen and Woodruff [13] and Haugen, Løkketangen and Woodruff [7] have combined the progressive hedging algorithm with Tabu search to design an effective heuristic to solve the problem. The stochastic lot-sizing problem has also been studied by Miller and Ahmed [18] who have developed valid inequalities that define the convex hull of structured relaxations. Furthermore, Miller [17] developed a polynomial-time dynamic programming algorithm for the multi-stage stochastic uncapacitated lot-sizing problem. The stochastic lot-sizing problem also appears prominently in a recent paper by Ahmed, King and Parija [2] who transform a stochastic capacity planning problem into a stochastic lot-sizing problem, and use the Krarup-Bilde formulation (of the lot-sizing problem) to generate good bounds within a branch and bound algorithm. They also devise an upper bounding heuristic which is incorporated within the branch and bound method. As an interesting by-product of their work, Ahmed, King and Parija [2] showed that the deterministic optimality condition (i.e. production is undertaken only if inventory is zero) does

not apply to the stochastic case. As shown by these authors, an optimal solution of the stochastic ELS may have non-zero production levels even in those periods in which the inventory levels are non-zero. This is the manner in which a stochastic model helps hedge against future uncertainty.

In this paper we study the stochastic batch-sizing problem, which is a slight generalization of the stochastic lot sizing problem. As in the stochastic lot sizing problem, demand, production, inventory and set-up costs are uncertain problem parameters, but production is undertaken in multiples of a given batch size. The main contribution of our work is in providing a unifying treatment in which we formulate a multi-stage recourse problem as well as a probabilistically constrained problem. In addition, we discuss a solution procedure that is applicable to both. From a managerial perspective, this unifying treatment allows us to study trade-offs between production/inventory cost and the probability of stock-outs. From an algorithmic viewpoint, our contribution lies in solving this multi-stage stochastic integer programming problem (with probabilistic constraints) using a branch and price algorithm. Our computational results also demonstrate the viability of such decomposition approaches over methods that solve a deterministic equivalent problem. These computations extend the work reported in [14] in a number of ways. We demonstrate that while the LP relaxation from the Krarup-Bilde reformulation does improve lower bounds (as in the deterministic case), it does not improve the overall performance of the branch and price scheme, in general. We also provide some comparisons between solutions to the recourse model with those obtained from the probabilistically constrained model.

The paper is organized as follows. In § 1, we provide both formulations of the multi-stage stochastic batch-sizing problem. The algorithm used to solve these problems are discussed in § 2. Computational results are given in § 3 while a comparison between probabilistically constrained solutions and recourse solutions are given in § 4.Finally, § 5 contains conclusions and future research.

1. Stochastic Batch-Sizing Formulations

We begin this section by first stating the recourse formulation. In keeping with much of the literature, we deal with discrete random variables with finite support (Birge and Lauveaux [5]), i.e. if ω is the random vector then $\Omega = (\omega^1, \ldots, \omega^r)$ with probabilities $p^1, \ldots, p^r$. This hypothesis allows us to represent uncertainty by means of scenarios, which represent a realization of the random variable corresponding to an elementary

atom $\omega \in \Omega$. The relationship between scenarios is represented via a scenario tree $\Im$, which captures the evolution of all information trajectories over time. At any node of the tree, there are several branches to indicate possible outcomes of the future which is modeled by random variables (associated with each node of the tree). Such a construction allows us to specify the events and probabilities in a natural way by conditioning on the events leading up to the current stage. Because a scenario includes one node at each stage exactly once, it is represented by a path from the root node (at stage 1) to a leaf node (at stage T) of the scenario tree. Note that with the exception of leaf nodes, all other nodes of the scenario tree may belong to more than one scenario. Given the set of scenarios $\mathcal{S} = \{1, \ldots, r\}$ and the decision horizon $\mathcal{T} = \{1, \ldots, T\}$, the correspondence between nodes of the scenario tree and 2-tuples $(t, s) \in \mathcal{T} \times \mathcal{S}$ is given by the surjective map $\mathcal{H} : \mathcal{T} \times \mathcal{S} \rightarrow \Im$.

We begin the formulation by providing summary of the notation used in the model.

$\mathcal{S} = \{1, \ldots, S\}$ is the set of scenarios,
$\mathcal{T} = \{1, \ldots, T\}$ is the decision time horizon,
$b \equiv$ batch size,
$C_t \equiv$ production capacity at time period t, specified in terms of the number of batches,
$I_t \equiv$ inventory capacity at time period t, specified in terms of the number of batches,
$d_{t,s} \equiv$ demand at time period t in scenario s,
$c_{t,s} \equiv$ production cost at time period t in scenario s,
$h_{t,s} \equiv$ holding (or inventory) cost at time period t in scenario s,
$f_{t,s} \equiv$ fixed (or set-up) cost at time period t in scenario s,
$p_s \equiv$ probability of scenario s.

The decision variables are:

$x_{t,s}$ production batch level at time t in scenario s,
$i_{t,s}$ inventory level at time t in scenario s,

$$y_{t,s} = \begin{cases} 1 & \text{if production is set-up at time } t \text{ in scenario } s, \\ 0 & \text{otherwise.} \end{cases}$$

$z_{\mathcal{H}(t,s)}$ production quantity at node $\mathcal{H}(t, s)$ of the scenario tree.

The stochastic batch-sizing problem for minimizing total expected cost is given by

$$Min \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}} p_s \cdot (c_{t,s} \cdot x_{t,s} + h_{t,s} \cdot i_{t,s} + f_{t,s} \cdot y_{t,s})$$

subject to

$$\begin{aligned}
i_{t-1,s} + b \cdot x_{t,s} &= d_{t,s} + i_{t,s} && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.1)\\
x_{t,s} - z_{\mathcal{H}(t,s)} &= 0 && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.2)\\
x_{t,s} &\leq C_t \cdot y_{t,s} && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.3)\\
i_{t,s} &\leq b \cdot I_t && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.4)\\
y_{t,s} \in \{0,1\},\ \ x_{t,s} \in \mathbb{Z}^+,\ \ i_{t,s} &\in \mathbb{R}^+ && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.5)
\end{aligned}$$

Constraints (5.1), are the collection of inventory balance constraints. They define the relation between demand, production level and inventory level for each time period t of any scenario s. Constraints (5.2) are the non-anticipativity constraints on production level decisions. As in all SP problems, the non-anticipativity constraints state that decisions depend only on information revealed in the past and not in the future, i.e. all scenarios with same history until the t-th stage should result in the same decisions until this stage. Therefore, we make decisions before realizing the random outcomes of demand. Note that non-anticipativity constraints are enforced only on the production decision variables, since the non-anticipativity constraints on the set-up variables are automatically satisfied once they are satisfied by the production level variables. In this case, non-anticipativity constraints are represented by associating single decision $z_{\mathcal{H}(s,t)}$ for each node $\mathcal{H}(s,t)$ of the scenario tree $\mathcal{T}$. Constraints (5.3) and (5.4) are capacity constraints on production and inventory levels respectively. Constraints (5.3) are also set-up forcing constraints. The basic idea is to force variable y_t to be one if production takes place.

The formulation given above is the extension of the Manne formulation [15] to the stochastic batch-sizing problem. Before going further, we also provide a formulation of the problem using the Krarup-Bilde variables [10]. They formulated the economic lot-sizing problem by defining a variable $\delta_{t,\tau}$ as the quantity produced in period t to satisfy the demand in period $\tau = t, \ldots, T$. Using those variables, the inventory balance, non-negative and set-up forcing constraints describe the convex hull of the economic lot-sizing problem polyhedron. To extend the Krarup-Bilde reformulation to the stochastic batch sizing problem let us introduce the

following decision variables:

$\delta^s_{t,\tau}$ production batch level at time t for period $\tau(\geq t)$ in scenario s.

Using the notation introduced so far, the problem formulation with the Krarup-Bilde decision variables here follows.

$$Min \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}} p_s \cdot (c_{t,s} \cdot \sum_{\tau=t}^{T} \delta^s_{t,\tau} + h_{t,s} \cdot \sum_{\iota=1}^{t} \sum_{\tau=t+1}^{T} \delta^s_{\iota,\tau} + f_{t,s} \cdot y_{t,s})$$

subject to

$$b \cdot \sum_{\tau=1}^{t} \delta^s_{\tau,t} \geq d_{t,s} \qquad \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. \tag{5.6}$$

$$\delta^s_{t,\tau} - \delta^{s'}_{t,\tau} = 0 \qquad \forall t,\ \tau(\geq t) \in \mathcal{T},\ \forall s,\ s' \in \mathcal{S} : \mathcal{H}(t,s) = \mathcal{H}(t,s'). \tag{5.7}$$

$$\sum_{\tau=t}^{T} \delta^s_{t,\tau} \leq C_t \cdot y_{t,s} \qquad \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. \tag{5.8}$$

$$\sum_{\iota=1}^{t} \sum_{\tau=t+1}^{T} \delta^s_{\iota,\tau} \leq I_t \qquad \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. \tag{5.9}$$

$$y_{t,s} \in \{0,1\},\ \ \delta^s_{t,\tau} \in \mathbb{Z}^+ \qquad \forall t,\ \tau(\geq t) \in \mathcal{T},\ \forall s \in \mathcal{S}. \tag{5.10}$$

In this formulation, the inventory balance constraints are substituted by the demand constraints (5.6). They state that demand have to be satisfied in any scenario, since no backlogging is allowed. Constraints (5.7) are the non-anticipativity constraints on production level decisions. Also in this formulation, non-anticipativity constraints are enforced only on the production decision variables, since the non-anticipativity constraints on the set-up variables are automatically satisfied once they are satisfied by the production level variables. Constraints (5.8) and (5.9) are capacity constraints on production and inventory levels respectively. Again, constraints (5.8) are also set-up forcing constraints.

In the stochastic batch sizing model we formulated above, we require demand to be satisfied at any time period t, and no backlogging is allowed. According to the demand balance constraints (constraints (5.1) and (5.6)), at any stage of the system the production level should be large enough to cover all the possible demand outcomes in the next stage. Operation managers often consider such a policy to be uneconomical. To overcome this potential drawback it may be appropriate to include a probabilistic constraint which enforces the condition that the

probability of meeting demand exceed an acceptable service level q. Thus a production plan is deemed feasible if the total probability of the scenarios accommodated by an optimal solution exceeds q. Let μ_s denote a binary variable, which assumes a value one if scenario s is included in the solution and zero otherwise. Mathematically, the probabilistic constraint may be modeled by the following multiple choice constraint.

$$\sum_{s \in \mathcal{S}} p_s \cdot \mu_s \geq q$$

When dealing with a probabilistic constrained formulation, we have to modify both the inventory balance and the non-anticipativity constraints. Using notation similar to that given for the recourse formulation, the formulation of the probabilistically constrained *SBSP* is as follows:

$$Min \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} p_s \cdot (c_{t,s} \cdot x_{t,s} + h_{t,s} \cdot i_{t,s} + f_{t,s} \cdot y_{t,s})$$

subject to

$$\begin{aligned}
i_{t-1,s} + b \cdot x_{t,s} &= d_{t,s} \cdot \mu_s + i_{t,s} && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.11)\\
x_{t,s} - z_{\mathcal{H}(t,s)} &\geq -M \cdot (1 - \mu_s) && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.12)\\
x_{t,s} - z_{\mathcal{H}(t,s)} &\leq M \cdot (1 - \mu_s) && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}. && (5.13)\\
x_{t,s} &\leq C_t \cdot y_{t,s} && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}.\\
i_{t,s} &\leq b \cdot I_t && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}.\\
\sum_{s \in \mathcal{S}} p_s \cdot \mu_s &\geq q\\
y_{t,s} \in \{0,1\},\ \ x_{t,s} \in \mathbb{Z}^+,\ \ i_{t,s} &\in \mathbb{R}^+ && \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}.\\
\mu_s &\in \{0,1\} && \forall s \in \mathcal{S}.
\end{aligned}$$

Constraints (5.12) and (5.13) are the non-anticipativity constraints. They are effective only if the corresponding scenario is accommodated by the solution. Note that so long as the cost coefficients are positive, $x_{t,s} = 0$ for all t associated with scenario s not accommodated by the solution.

The above formulation is somewhat unique in that it uses both continuous and discrete decision variables to enforce non-anticipativity constraints within a probabilistically constrained model.

2. Algorithmic approaches for Stochastic Batch-Sizing

In this section we propose some algorithmic approaches to the solution of stochastic batch-sizing problems. It is clear from the models proposed in the previous section that these problems involve discrete decision variables. This feature, which is realistic in several practical applications,

has a significant impact on its tractability. Our approach to solving the stochastic batch-sizing problem involves decomposing the multi-stage SMIP problem using a branch and price (B&P) methodology. One of the main advantages of this approach is that it allows us to take advantage of the structure underlying the deterministic batch-sizing problem. To give the reader a preview of the main algorithmic ideas, note that the stochastic formulation essentially uses the non-anticipativity constraints to bind decisions from individual (deterministic) scenario formulations. In fact, what separates stochastic programming from deterministic optimization is the presence of non-anticipativity constraints. These constraints couple the decisions associated with different scenarios, thus making the problem "harder" to solve. If we relax the problem, discarding all the non-anticipativity constraints from the formulation, we obtain a fully decoupled block-angular mixed-integer programming problem. B&P methods are particularly attractive for problems with this feature as they allow us to split the mixed-integer programming problem into more manageable pieces corresponding to single scenario subproblems.

The B&P algorithm is motivated by the Mixed Integer Finite Basis Theorem (Theorem 4.30 in [16]) which allows the representation of a bounded integer polyhedron (e.g. Γ_s defined by constraints (5.1), (5.3), (5.4) and (5.5) of *SBSP*) by a convex hull of integer points. This method applies the Dantzig-Wolfe decomposition principle to a master problem which enforces non-anticipativity as well as integer requirements on the decision variables. As with traditional Dantzig-Wolfe decomposition, the number of variables in the master problem is exponential in terms of the original problem size. Hence the algorithm proceeds by generating columns as needed. Thus, only a subset of variables is handled in a restricted master problem (*SBS-RMP*) which is formulated as follows:

$$Min \sum_{s\in\mathcal{S},\ i\in Q_s} \tilde{c}_{i,s}\cdot\alpha_s^i$$

$$s.t.$$

$$[SBS-RMP] \qquad \sum_{i\in Q_s} x_{t,s}^i\cdot\alpha_s^i = z_{\mathcal{H}(t,s)} \qquad \forall t\in\mathcal{T},\ \forall s\in\mathcal{S}. \quad (5.14)$$

$$\sum_{i\in Q_s}\alpha_s^i = 1 \qquad \forall s\in\mathcal{S}. \quad (5.15)$$

$$\alpha_s^i \geq 0 \qquad \forall i\in Q_s,\ \forall s\in\mathcal{S}.$$

$$z_{\mathcal{H}(t,s)}\in\mathbb{Z}^+ \qquad \forall t\in\mathcal{T},\ \forall s\in\mathcal{S}.$$

where $\tilde{c}_{i,s}$ is the total cost associated with the i^{th} column of scenario s, and Q_s is the index set of columns in the *SBS-RMP* which be-

long the finite set of points of Γ_s. Constraints (5.14) and (5.15) are the non-anticipativity constraints and the convexity constraints respectively. The integer requirements are imposed directly on the convex combinations that arise in the master formulation.

To verify the optimality of the solution provided by the restricted master problem (*SBS-RMP*), we have to verify that columns that are not listed in the restricted master do not generate an improvement of the objective function, if added to the restriction *SBS-RMP*. This task of verifying the *SBS-RMP* optimality is accomplished by solving a pricing program for each scenario $s \in \mathcal{S}$. For each scenario, the pricing problem will be a deterministic batch-sizing problem. In our implementation, we solve the deterministic batch-sizing problem using a customized dynamic programming algorithm, where the state variable of the system is the inventory level and the control actions are given by the possible production levels. The transition function of the system from one stage to the next is represented by constraint (5.1) of the formulation given in § 1. If a column has negative reduced cost, thus improving problem objective function, it is added to the *SBS-RMP*.

This process is continued until no more columns price out negative and the LP relaxation of *SBS-RMP* is solved. If the integrality conditions on the decision variables are not satisfied, the branching phase takes place. A straightforward branching scheme for the *SBS-RMP* is based on the partition of the solution space using the original problem variables for which we impose the integer requirements. Therefore, if a component of the $\tilde{z}$ vector of the master LP solution $(\tilde{\alpha}, \tilde{z})$ is fractional, say the i-th component, then the branching takes the form

$$z_i \leq \lfloor \tilde{z}_i \rfloor \qquad \text{or} \qquad z_i \geq \lceil \tilde{z}_i \rceil.$$

The branch-and-price algorithm is summarized as follows.

Let π_s dual variable associated with the convexity constraint (5.15),
$u_{t,s}$ dual variables associated with the non-anticipativity constraint (5.14)

Initialization set

$\pi_s = \infty$ and $Q_s = 0 \ \forall s \in \mathcal{S}$,
$u_{t,s} = 0 \ \forall s \in \mathcal{S}, \ \forall t \in \mathcal{T}$,
$UB = \infty$ or equal to the objective function of some computed feasible solution.

Step 1 $\forall s \in \mathcal{S}$ solve the pricing problem:

$$\begin{aligned} min \ \ & \Omega_s = p^s (C_s)^T (x^i_{1,s}, \ldots, x^i_{T,s}, i^i_{1,s}, \ldots, i^i_{T,s}, y^i_{1,s}, \ldots, y^i_{T,s}) \\ s.t. \ \ & (x^i_{1,s}, \ldots, x^i_{T,s}, i^i_{1,s}, \ldots, i^i_{T,s}, y^i_{1,s}, \ldots, y^i_{T,s}) \in \Gamma_s \end{aligned}$$

where $C_s = (c_1^s - u_{1,s}, \ldots, c_T^s - u_{T,s}, h_{1,s}, \ldots, h_{T,s}, f_{1,s}, \ldots, f_{T,s})$. If $\Omega_s < \pi_s$ add the column to the current *SBS-RMP* $(i \rightarrow Q_s)$ with total cost computed using the original cost vector. Otherwise go to Step 5.

Step 2 Solve the linear relaxation of the restricted master problem *RMP,* and update dual variables $u_{t,s}$ and π_s.

Step 3 If $z_i \in \mathbb{Z}^+ \; \forall i \in \Im$ go to Step 4 b, otherwise go to Step 4 a.

Step 4 a Select $i \in \Im : \; z_i \notin \mathbb{Z}^+$ and define two subproblems adding the constraints

$$x_{t,s} \leq \lfloor z_i \rfloor \quad \text{or} \quad x_{t,s} \geq \lceil z_i \rceil \qquad \forall t, s : \mathcal{H}_{t,s} = i.$$

Go to Step 5.

Step 4 b If

$$\exists \; s' \in \mathfrak{B}_t^s \; : \; x_{t,s} \cdot x_{t,s'} = 0 \quad \text{and} \quad x_{t,s} + x_{t,s'} > 0$$

then branch on $y_{\mathcal{H}(t,s)}$. For each newly generated node of the search tree define an appropriate *SBS-RMP,* consistent with values assumed by the branching variables, i.e. eliminate all the columns whose entries are not consistent with the branching variable.

Otherwise update the incumbent solution.

Step 5 Select an active node of the branching tree and go to Step 1, otherwise the current solution is optimal, STOP.

For a more comprehensive description of branch and price applied to Stochastic Integer Programming problems the reader may refer to our related paper [14]. Moreover, a discussion on practical issues concerning efficient implementation is given in Barnhart et al. [4], Johnson et al. [11] and Martin [16] .

3. Computational Results

In this section we provide some computational results on the stochastic batch-sizing problem. We will first study whether the quality of the LP relaxation has an impact on the computational time for the B&P algorithm. Towards this end, we study differences due to the Krarup-Bilde formulation, and the formulation provided in §2. Following this experiment, we study the viability of the B&P algorithm suggested in the previous section.

We begin by describing the test instances used in our study. Our experiments involve ten randomly generated batch-sizing problems, each involving six decision stages. For each instance, the scenario tree has the structure of a binary tree. For node n of such a tree, the conditional probability associated with one branch is p_n, and that for the other branch $1 - p_n$. Here p_n is chosen from the uniform [0,1] distribution. By choosing alternative values of p_n, as well as demands we can generate different problem instances. The characteristics of the scenario tree for the test instances used in our computational analyses are given in Table 5.1.

Table 5.1. Dimension Parameters for Test Instances

Class of Instances	Number of Stages	Number of Scenarios	Scenario Tree Nodes
6-32	6	32	63

Our experiments were conducted on a workstation SUN Ultra 80 with two processors and 1 GB RAM. For the CPLEX Branch-and-Bound method (CPLEX-MIP), we implemented the formulations using AMPL as modelling language, while the branch-and-price method has been implemented using BCP (a framework for Branch, Cut and Price algorithms), which is part of COIN-OR, the Common Optimization INterface for Operations Research [6]. As the LP engine to solve the linear relaxation of RMP, we used CPLEX 7.0, while the subproblems were solved using a customized dynamic programming algorithm.

We first evaluate how well the Krarup-Bilde formulation performs in the context of the stochastic batch-sizing problem (KB-SBSP). On this subject, we compared the relative gap of the Krarup-Bilde formulation (5.6-5.10) with the one shown by the original formulation (5.1-5.5). The relative gap is the percentage deviation between the optimal value of the problem and the optimal value of its linear relaxation. As reported in Table 5.2, we observe that the Krarup-Bilde formulation provides slightly better relative gaps than those provided by the original SBSP formulation. However, the difference between the two formulations is not as significant as reported for the stochastic lot-sizing problems (see Ahmed, Parija and King). Moreover, the SBSP formulation seems to dominate the Krarup-Bilde formulation in terms of solution times (CPU secs.) as well as the number of iterations. In view of these considerations, we continue the remainder of our experiments using the SBSP formulation.

Table 5.2. SBSP formulation vs. Krarup-Bilde formulation

SBSP					KB-SBSP			
IP	LP	Gap(%)	CPU	Iter.s	LP	Gap(%)	CPU	Iter.s
15973	14699	8.67	0.02	64	14784	8.04	0.1	470
14556	13126	10.89	0.02	63	13211	10.18	0.11	529
16175	14839	9.00	0.02	63	14923	8.39	0.08	477
14255	12929	10.26	0.02	63	13015	9.53	0.11	519
16389	15098	8.55	0.03	64	15183	7.94	0.15	532

We evaluate the effectiveness of the B&P algorithm by comparing its performance with that of CPLEX branch-and-bound method (CPLEX-MIP). A time limit of 100 sec. was imposed for both the solvers. Solution statistics for B&P and CPLEX-MIP for all the instances solved are reported in Table 5.3. First, we have to highlight that CPLEX-MIP was not able to solve any of the instances, with the exception of the ninth instance, within the imposed time limit. With reference to the same statistic, the branch-and-price algorithm ran at most for 2.09 sec. Differences between the two methods are also much more evident when

Table 5.3. Branch-and-price versus Branch-and-Bound

	B&P solution		SBSP-Cplex solution		
Instance	Total running time	Optimal O.F. value	Best O.F. value	MIP Iter.ns	B&B nodes
1	0.85	15973	16011	188214	97504
2	1.00	14556	14759	170542	81958
3	1.04	16175	18702	202202	98684
4	0.89	14255	15033	207672	89913
5	0.90	16389	17525	207134	84072
6	1.02	14793	16503	206209	97108
7	0.74	15404	15474	181633	75219
8	1.09	15315	15315	162929	80350
9	2.09	14632	14632$^{(\star)}$	107315	52648
10	0.80	14721	15404	204604	103068

$^{(\star)}$ optimal solution, CPU time: 72 sec.

comparing the number of nodes of the search tree. Indeed, the branch and price procedure did not branch at all, while CPLEX-MIP visited thousands of nodes within the time limit. These results demonstrate the quality of our method and the viability of using branch-and-price as a methodology for special structured multi-stage SMIP problems.

Analogous computational results have been obtained for the probabilistic version of the problem, see Table 5.4.

Because of the combinatorial structure of the non-anticipativity constraints, the probabilistically-constrained version of the stochastic batch-sizing problem requires a greater number of search nodes.

The results shown in Table 5.4, highlight one of the intriguing points of our branch and price implementation. The computational time required for the entire process, given by the sum of the time to solve the linear relaxation of the restricted master problems (Time for LPs), time spent to generate columns (Time in VG) and time for the execution of strong branching procedures (Time in SB), is dominated by the amount

Table 5.4. B&P results for the Probabilistic formulation with q=.90

Instance	Total running time	B&B nodes	Time for LPs	Time in VG	Time in SB
1	5658.85	209	1007.88	52.85	18.61
2	3667.38	213	658.79	36.32	20.31
3	2270.67	149	473.04	15.57	8.55
4	789.72	81	131.94	5.25	4.65
5	1531.73	127	269.71	17.25	7.39
6	351.30	49	61.63	4.89	2.11
7	3134.64	185	692.89	17.83	10.13
8	1365.17	121	169.82	14.47	7.75
9	585.23	79	94.35	5.3	2.9
10	386.85	67	62.42	4.17	2.4

of time required in traversing the search tree. In fact, an inordinate amount of time is spent in traversing the search tree. The inefficiencies of the tree management routines affect irremediably the solution times of the probabilistically constrained version of the stochastic batch-sizing problem. While the number of nodes still remains manageable, the increase in running times limits the scalability of the proposed method for the formulation with probabilistic constraints.

4. Solutions from Alternative Models

In this section, we highlight the differences among solutions obtained by solving both the full recourse model and the probabilistic one. In particular, we compare full recourse solutions with probabilistic constrained solutions obtained by solving the probabilistic constrained model using two service levels, $q = 0.90$ and $q = 0.75$.

Due to computational difficulties with the probabilistic constrained problem, in this section we restrict our study and analysis to a five-stage problem. Again we use a scenario tree with a binary tree structure, as described in § 3.

For each scenario accommodated in the solution, the value of inventory level across the stages of the problem defines an inventory level trajectory or path. Figure 1.1 reports an example of a five-stage scenario inventory trajectory. More precisely the inventory trajectories for the recourse and probabilistic solutions are given. On the abscissa we report the stage of the problem. In the histogram the full black, the reticulate and the dotted bars refer to the full recourse and probabilistic constrained solutions with $q = 0.90$ and $q = 0.75$ respectively. As we can see, the probabilistic constrained solutions exhibit a lower level of

inventory than that provided by recourse solution. Analogous trends are exhibited in the other single-scenario inventory paths. To provide an overview on such trend, in Figure 1.2, we plot the period-by-period average value of inventory (averaged across all scenarios).

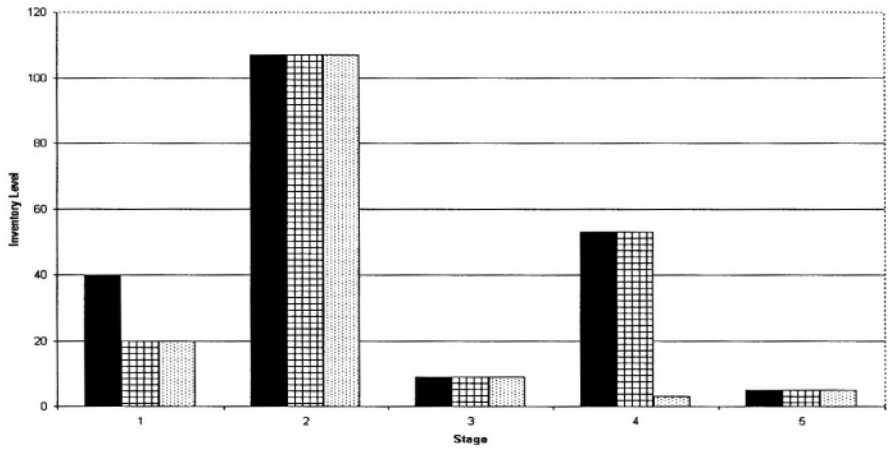


Figure 5.1. Scenario inventory level.

Again, the full, dashed and dotted lines refer to recourse, probabilistic with $q = 0.90$ and probabilistic with $q = 0.75$ solutions respectively. The line representing the recourse solution inventory level is above the other two lines.

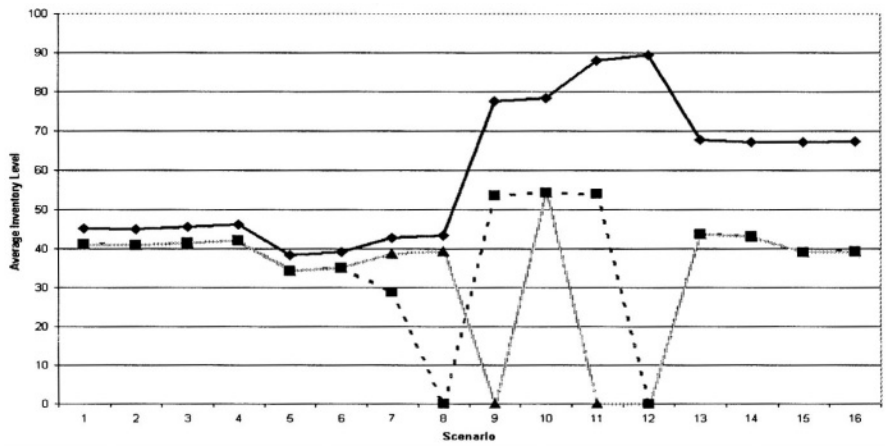


Figure 5.2. Average scenario inventory level.

In most of the instances solved, we note this system behavior. Unfortunately, it is not possible to give a description of inventory that is concise and complete. Table 5.5 reports the average values of inventory (across all scenarios) and standard deviation associated with these inventory paths. Furthermore, we also report the maximum value of inventory level in the solution.

Table 5.5. Inventory level statistics

	q = 0.75			q = 0.90			q = 1.0		
Ist	avg(μ_i)	avg(σ_i)	max	avg(μ_i)	avg(σ_i)	max	avg(μ_i)	avg(σ_i)	max
1	6.42	7.58	29	5.86	6.32	29	6.56	7.62	29
2	5.04	4.42	9	5.00	4.30	9	9.31	9.44	34
3	4.25	4.35	18	4.68	5.18	18	5.69	6.70	22
4	6.13	7.14	31	5.82	6.78	31	7.86	9.06	33
5	2.43	4.04	20	6.64	9.85	31	7.84	10.69	31
6	5.00	6.22	36	6.07	7.62	36	5.88	7.15	36
7	2.91	2.53	9	4.11	4.94	20	3.78	4.50	20
8	3.33	3.20	10	3.79	3.97	19	4.16	4.54	19
9	7.45	6.67	29	7.54	9.49	30	7.75	9.04	30
10	4.04	3.72	9	3.93	3.60	9	5.09	5.49	25

In seven out of ten instances, probabilistic solutions with $q = 0.75$ have lower mean and standard deviation of inventory. Moreover, probabilistic solutions with $q = 0.75$ show lower maximum inventory in all the instances solved.

It is important to note, that the statistics reported Table 5.5 do not provide a rigorous description of inventory, since there is a correlation between single scenario inventory paths and there are averaging effects due to the different number of scenarios accommodated in each solution (those with $q = 0.75$, 0.90 and 1.0).

The difference in inventory level between the recourse and probabilistic solutions is more and more significant as the cost structure of the stochastic batch-sizing problem has higher and higher fixed costs and the demand shows higher and higher volatility. The motivations of such behavior may be brought back to the following considerations. First, due to high fixed cost, the decision policy is to satisfy the demand of several periods with large productions whenever they occur. Second, if demand is highly volatile, productions should be fairly large to satisfy all the possible scenario demand outcomes. On the other hand, if fixed costs are negligible with respect to production and inventory costs (as in some flexible manufacturing systems and just-in-time operations), then

the differences between solutions from a probabilistically constrained model and a recourse model may not be significant.

Solutions with higher q represent fewer stock-outs and hence more reliable. Obviously, higher reliability comes at a cost. A trade-off between reliability and costs may be depicted as shown in Figure 1.3. Reliability

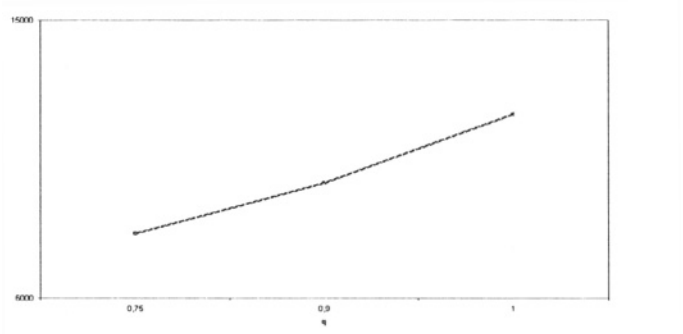


Figure 5.3. Reliability vs. Cost

cost (backlogging) should be equal to the objective function difference between the full recourse problem and the probabilistic problem

From an operational point of view, we may choose the level of service corresponding an acceptable cost structure. Such insights are made possible by the models and algorithm studied in this paper.

5. Conclusions

In this paper, we have studied the stochastic bath sizing problems in which demand, production, inventory and set-up costs are uncertain problem parameters, all of which evolve as discrete random variables. This hypothesis allows us to represent uncertainty be means of a scenario tree. In addition to the recourse formulation, we have presented a probabilistically constrained version, whose solutions accommodate a restricted number of scenarios. The latter approach may be viable in situations where the cost of meeting demand for all scenarios may be exorbitant. The solution approach that we adopt for these problems may be classified as a branch and price method. One of the main advantages of our approach is that it allows us to build on the wealth of knowledge that has been developed in connection with the deterministic version of these problems. Moreover, we handle the recourse formulation, and the probabilistically constrained formulation within the same framework. Through our computational experiments, we have also come to a variety of conclusions. First, we note that the advantage of the Krarup-Bilde reformulation does not seem to translate to lower computational times

for the batch-sizing problem. As regards the B&P methodology, it turns out that it is quite effective for the recourse constrained model. However, the multiple-choice constraints in the probabilistically constrained formulation make the model significantly more difficult. Through our study, we have also demonstrated how trade-offs between cost and reliability can be investigated for the stochastic batch-sizing problem.

Acknowledgment

We thank the anonymous referee for their useful comments that improved the content and presentation of the paper.

References

[1] Aggarwal A. and Park J.K. Improved Algorithms for Economic Lot Size Problems. Operations Research 1993; **41**:549-571.

[2] Ahmed S., King A.J. and Parija G. A Multi-Stage Integr Programming Approach for Capacity Expansion under Uncertainty. E-SPSS print series 2001.

[3] Baker K.R. An experimental study of the effectiveness of rolling schedule in production planning. Decisions Science 1977; **8**:19-27.

[4] Barnhart C., Johnson E.L., Nemhauser G.L., Savelsberg M.W.P. and Vance P.H. Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 1998; **46**:316-329.

[5] Birge J.R. and Louveaux F. "*Introduction to Stochastic Programming",* Springer, 1997.

[6] Common Optimization INterface for Operations Research, *http//www. coin-or. org*

[7] Haugen K.K., Løkketangen A. and Woodruff D.L. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. European Journal of Operational Research 2001; **132**:116-122.

[8] Hsu V.N. Dynamic Economic Lot Size Model with Perishable Inventory. Management Science 2000; **46**:1159-1169.

[9] Kuik R., Solomon M. and van Wassenhove L.N. Batching decisions: Structure and models. European Journal of Operational Research 1994; **75**:243-263.

[10] Krarup J. and Bilde O. "Plant location, set covering and economic lot size: An o(mn) algorithm for structured problems." In *Optimierung bei Graphentheoretishen und gauzzahligen Problemen,* Collatz et al , eds. Birkhauser-Verlag, 1977.

[11] Johnson E.L., Nemhauser G.L. and Savelsbergh M.W.P. Progress in Linear Programming Based Branch-and-Bound Algorithms: An Exposition. INFORMS Journal on Computing 1997; **75**:243-263

[12] Leung J.M.Y., Magnanti T.L. and Vachani R. Facets and Algorithms Capacitated Lot Sizing. Mathematical Programming 1989; **45**:331-359.

[13] Løkketangen A. and Woodruff D.L. Progressive Hedging and Tabu Search Applied to Mixed Integer (0,1) Multi-stage Stochastic Programming. Journal of Heuristics 1996; **2**:111-128.

[14] Lulli G. and Sen S. A Branch-and-Price Algorithm for Multi-stage Stochastic Integer Programming with Application to Stochastic Batch-Sizing Problems, submitted for publication.

[15] Manne A.S. Programming of Lot Sizes. Management Science 1958; **4**:115-135.

[16] Martin R.K. *"Large Scale Linear and Integer Optimization"*. Kluwer Academic Publishers, 1999.

[17] Miller A.J. A polynomial-Time Dynamic Algorithm for the Multi-Stage Stochastic Uncapacitated Lot-Sizing Problem. INFORMS Annual Conference; November 2001; Miami.

[18] Miller A.J. and Ahmed S. Uncapacitated Stochastic Lot-Sizing Problem: Algorithms and Polyhedral Structure. INFORMS Annual Conference; November 2001; Miami.

[19] Sen S. A Branch and Price Algorithm for Multi-stage Stochastic Integer Problems. INFORMS Annual Conference; November 2000; San Antonio.

[20] Stadtler H. Improved Rolling Schedules for the Dynamic Single-Level Lot-Sizing Problem. Management Science 2000; **46**:318-326.

[21] Wagner H.M. and Whitin T.M. Dynamic Version of the Lot Size Model. Management Science 1958; **5**:89-96.

[22] Zangwill W.I. A deterministic multi-period production scheduling model with backlogging. Management Science 1966; **13**:105-119.

This page intentionally left blank

Chapter 6

A SUMMARY AND ILLUSTRATION OF DISJUNCTIVE DECOMPOSITION WITH SET CONVEXIFICATION

Suvrajeet Sen
sen@sie.arizona.edu

Julia L. Higle
julie@sie.arizona.edu

Lewis Ntaimo
nlewis@sie.arizona.edu

Dept. of Systems and Industrial Engineering
The University of Arizona
Tucson, Arizona 85721

Abstract In this paper we review the Disjunctive Decomposition (D^2) algorithm for two-stage Stochastic Mixed Integer Programs (SMIP). This novel method uses principles of disjunctive programming to develop cutting-plane-based approximations of the feasible set of the second stage problem. At the core of this approach is the Common Cut Coefficient Theorem, which provides a mechanism for transforming cuts derived for one outcome of the second stage problem into cuts that are valid for other outcomes. An illustrative application of the D^2 method to the solution of a small SMIP illustrative example is provided.

Keywords: Set convexification, Disjunctive Decomposition

Introduction

Stochastic Mixed Integer Programs (SMIP) comprise one of the more difficult classes of mathematical programming problems. Indeed, this class of problems combines the extremely large scale nature of stochastic programs and the inherent computational difficulties of combinatorial optimization. The main difficulty in solving two-stage stochastic mixed-integer programs is that the recourse costs are represented as the expected value of a mixed-integer program whose value function is far more complicated than the value function of a linear program. In general, the expected recourse function is non-convex and possibly discontinuous. In this paper we illustrate the Disjunctive Decomposition (D^2) algorithm with set convexification for two stage SMIP, proposed by Sen and Higle [2000].

The method uses the principles of disjunctive programming to develop a cutting-plane-based approximation of the feasible set of the second stage problem. This task is streamlined via the Common Cut Coefficients (C^3) Theorem (Sen and Higle [2000]), which provides a simple mechanism for transforming cuts derived for one instance of the second stage problem into cuts that are valid for another instance. This significantly reduces the effort required to approximate the convexification of the feasible set, a task that must be undertaken for each possible outcome of the random variables involved. In this paper, we illustrate the D^2 algorithm and the manner in which the C^3 Theorem is used to reduce the computational effort. Because the methodology is related to, but distinctly different from, the work of Carøe[1998], we also use this forum to highlight the relationship between the two approaches.

This paper is organized as follows. In §1 we summarize the results of Sen and Higle [2000], and identify connections between their work and that of Carøe[1998]. In §2 we illustrate the application of the D^2 Algorithm with a simple numerical example with both first and second-stage binary variables. Finally, a discussion and our conclusions are found in §3.

1. Background

In this section we summarize the main results from Sen and Higle [2000] that are critical to our illustration of the D^2 algorithm. In particular, we review the C^3 theorem and discuss the details of its application. For a more thorough explanation of disjunctive decomposition concepts, proofs, and the derivation of the D^2 algorithm, we refer the reader to Sen and Higle [2000]. Throughout this paper we consider the following

stochastic mixed integer program (SMIP):

$$\underset{x \in X \bigcap \mathcal{B}}{\text{Min}} \; c^\top x + E[f(x, \tilde{\omega})], \tag{6.1}$$

where $X \subseteq \mathfrak{R}^{n_1}$ is a set of feasible first stage decisions, $\mathcal{B} \subset \mathfrak{R}^{n_1}$ is the set of binary vectors, $\tilde{\omega}$ is a random variable defined on a probability space $(\Omega, \mathcal{A}, \mathcal{P})$, and for any $\omega \in \Omega$

$$f(x, \omega) = \text{Min} \; g_u^\top u + g_z^\top z, \tag{6.2a}$$

$$s.t. \quad W_u u + W_z z = r(\omega) - T(\omega)x, \tag{6.2b}$$

$$u \in \mathfrak{R}_+^{n_u}, z \in \mathcal{B}^{n_z}. \tag{6.2c}$$

It is assumed that X is a convex polyhedron, Ω is a finite set, and that $f(x, \omega) < \infty$ for all $(x, \omega) \in X \times \Omega$. Moreover, we assume that by using appropriately penalized continuous variables, the subproblem (6.2) remains feasible for any restriction of the integer variables z. Note that the inclusion of integer variables in the second stage problem, (6.2), is the primary source of the computational and algorithmic challenges associated with (6.1). In particular, in order to evaluate the SMIP objective $cx + E[f(x, \tilde{\omega})]$, it is necessary to solve (implicitly or approximately) the MIP (6.2) for each $\omega \in \Omega$. Moreover, the structural difficulties associated with MIP objective functions are well documented (see, e.g., Blair and Jeroslow [1982] and Blair [1995]). These difficulties are compounded by the fact that the expected value operations associated with the SMIP objective function amounts to a convex combination of the complicated individual MIP objective functions. The C^3 Theorem exploits the specific structure of (6.2), thereby permitting a computationally streamlined development of SMIP objective function approximations.

1.1 Common Cut Coefficients

In an effort to develop approximations of the SMIP objective, we begin with an approximation of the convex hull of feasible integer points for (6.2). This set can be expressed as a disjunction,

$$\mathcal{S} = \bigcup_{h \in H} \mathcal{S}_h,$$

where H is a finite index set, and the sets $\{S_h\}_{h \in H}$ are polyhedral sets represented as

$$\mathcal{S}_h = \{y \mid W_h y \geq r_h, y \geq 0\}$$

Within our setting, we have $y = (u, z)$ as in (6.2) and r_h includes $r(w) - T(w)x$. More formally, we note that the constraints in (6.2),

$$W_u u + W_z z \geq r(\omega) - T(\omega)x$$

vary with the first stage decision, x, and the scenario ω. Consequently, the disjunctive representation of the set depends on $(x, \omega) \in X \times \Omega$,

$$\mathcal{S}(x, \omega) = \bigcup_{h \in H} \mathcal{S}_h(x, \omega), \tag{6.3}$$

where

$$\mathcal{S}_h(x, \omega) = \{y \mid W_{hu}u + W_{hz}z \geq r_h(x, \omega), u, z \geq 0\}.$$

A convex relaxation of the nonconvex set (6.3) can be represented by a collection of valid inequalities of the form

$$\pi_u^\top u + \pi_z^\top z \geq \pi_0(x, \omega).$$

While the disjunctive representation depends on (x, ω), the C^3 Theorem, which we state below, ensures that as the argument changes, cut validity can be maintained by a shift in the right-hand-side element without altering the gradient of the cut. In the following we use $n_2 = n_u + n_z$.

Theorem 1 *(The C^3 Theorem). Consider the stochastic program with fixed recourse as stated in (6.1), (6.2). For $(x, \omega) \in X \times \Omega$, let $Y(x, \omega) = \{y = (u, z) \mid Wy \geq r(\omega) - T(\omega)x, u \in \mathfrak{R}_+^{n_u}, z \in \mathcal{B}^{n_z}\}$, the set of mixed-integer feasible solutions for the second stage mixed-integer linear program. Suppose that $\{C_h, d_h\}_{h \in H}$, is a finite collection of appropriately dimensioned matrices and vectors such that for all $(x, \omega) \in X \times \Omega$*

$$Y(x, \omega) \subseteq \bigcup_{h \in H} \{y \in \mathfrak{R}_+^{n_2} \mid C_h y \geq d_h\}.$$

Let

$$\mathcal{S}_h(x, \omega) = \{y \in \mathfrak{R}_+^{n_2} \mid Wy \geq r(\omega) - T(\omega)x, C_h y \geq d_h\},$$

and let

$$\mathcal{S} = \bigcup_{h \in H} \mathcal{S}_h(x, \omega).$$

Let $(\bar{x}, \bar{\omega})$ be given, and suppose that $\mathcal{S}_h(\bar{x}, \bar{\omega})$ is nonempty for all $h \in H$ and $\pi^T y \geq \pi_0(\bar{x}, \bar{\omega})$ is a valid inequality for $\mathcal{S}(\bar{x}, \bar{\omega})$. There exists a function, $\pi_0 : X \times \Omega \to \mathfrak{R}$ such that for all $(x, \omega) \in X \times \Omega$, $\pi^T y \geq \pi_0(x, \omega)$ is a valid inequality for $\mathcal{S}(x, \omega)$.

Proof. See Sen and Higle [2000].

The C^3 Theorem ensures that a valid inequality for the set $S(\bar{x}, \bar{\omega})$ of the form $\pi^\top y \geq \pi_0(\bar{x}, \bar{\omega})$ can be translated to an inequality, $\pi^\top y \geq \pi_0(x, \omega)$ that is valid for the set $S(x, \omega)$. The cut coefficients, π, are

common to both sets. Thus, one may derive the left hand side coefficients, π, which may be applied to all scenario problems. The right hand side $\pi_0(x,\omega)$ is derived as necessary for each pair (x,ω) using a strategy from reverse convex programming in which disjunctive programming is used to provide facets of the convex hull of reverse convex sets (Sen and Sherali [1987]). Given the valid inequalities, $\pi^\top y \geq \pi_0(x,\omega)$, a lower bound approximation for the scenario subproblem objective function is given by:

$$f(x,\omega) \geq f_0(x,\omega) \equiv \text{Min } g^\top y \tag{6.4a}$$
$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x \tag{6.4b}$$
$$\pi^T y \geq \pi_0(x,\omega) \tag{6.4c}$$
$$y \geq 0 \tag{6.4d}$$
$$\tag{6.4e}$$

We note that a version of Theorem 1 appears in Carøe[1998], although there are some critical distinctions between Sen and Higle [2000] and Carøe[1998]. Specifically, while Sen and Higle [2000] work within the context of the temporal decomposition indicated in (6.1,6.2), Carøe[1998] works within the context of the "deterministic equivalent problem",

$$\begin{aligned} \text{Min} \quad & c^\top x + \sum_{\omega\in\Omega} p_\omega (g_u^\top u_\omega + g_z^\top z_\omega) \\ s.t. \quad & T(\omega)x + W_u u_\omega + W_z z_\omega = r(\omega) \qquad \forall \omega \in \Omega \\ & x \in \Re_+^{n_1},\ u_\omega \in \Re_+^{n_u},\ z_\omega \in \mathcal{B}^{n_z} \qquad \forall \omega \in \Omega. \end{aligned}$$

Accordingly, Carøe's cuts may be translated from one scenario to another, while being restricted to the higher dimension of (x, u_ω, z_ω) as compared to the (u_ω, z_ω) dimension restriction of the Sen and Higle cuts. It follows that the Sen and Higle cuts permit both a temporal and scenario decomposition (i.e., wrt to x and ω), while Carøe's are restricted to only scenario decomposition (i.e., wrt ω). Another recent paper, Sherali and Fraticelli [2000] also uses cuts in this higher dimensional space. Their approach uses the formulation-linearization techniques (Sherali and Adams [1990]) to construct their approximation.

1.2 Convexification of the Right-Hand-Side Function

As discussed in Sen and Higle [2000], the function $\pi_0(x,\omega)$ is piecewise linear and concave in the first argument. That is,

$$\pi_0(x,\omega) = \underset{h\in H}{\text{Min}}\{\bar{\nu}_h(\omega) - \bar{\gamma}_h(\omega)^\top x\}$$

for a specified collection $\{\bar{\nu}_h(\omega), \bar{\gamma}_h(\omega)\}_{h\in H}$. Consequently, it is necessary to develop a convexification of the function in order to facilitate the solution of the lower bounding approximation (6.4). This is accomplished using reverse convex programming techniques, in which disjunctive programming concepts are used to obtain the convex hull of reverse convex sets (Sen and Sherali [1987]).

To begin, let the epigraph of $\pi_0(\cdot, \omega)$, restricted to $x \in X$ be defined as

$$\Pi_X(\omega) = \{(\theta, x) \mid x \in X, \theta \geq \pi_0(x, \omega)\},$$

where X is a polyhedral set such that

$$X = \{x \in \mathfrak{R}_+^{n_1} \mid Ax \geq b\},$$

where $A \in \mathfrak{R}^{m_1 \times n_1}$ and $b \in \mathfrak{R}^{m_1}$.

Also let

$$E_h(\omega) = \{(\theta, x) \mid \theta \geq \bar{\nu}_h(\omega) - \bar{\gamma}_h(w)^\top x, Ax \geq b, x \geq 0\}. \tag{6.5}$$

Then $\Pi_X(\omega)$ can be defined in disjunctive normal form as

$$\Pi_X(\omega) = \bigcup_{h\in H} E_h(\omega).$$

Thus the epigraph of function π_0 can be represented as the union of half-spaces, which is a disjunctive set. In order to convexify this set, we apply the notion of reverse polars from the theory of disjunctive programming (Balas [1979]). These sets (reverse polars) characterize the set of all valid inequalities of a disjunctive set, with the extreme points providing facets of the (closure of the) convex hull of the disjunctive set. The specific construction that we adopt is provided below, and will be referred to as the epi-reverse polar because it represents the reverse polar of the epigraph of π_0.

In the following, we assume that for all $x \in X$, $\theta \geq 0$ in (6.5). As long as X is bounded, there is no loss of generality with this assumption, because the epigraph can be translated to ensure that $\theta \geq 0$. The epi-

reverse polar of this set, $\Pi_X^\dagger(\omega)$, is defined as

$$\begin{aligned}\Pi_X^\dagger(\omega) = \{&\sigma_0(\omega) \in \Re, \sigma(\omega) \in \Re^{n_1}, \delta(\omega) \in \Re \text{ such that} \\ &\forall h \in H,\ \exists\ \tau_h \in \Re^{m_1}, \tau_{0h} \in \Re \\ &\sigma_0(\omega) \geq \tau_{0h} \quad \forall h \in H \\ &\sum_h \tau_{0h} = 1 \\ &\sigma_j(\omega) \geq \tau_h^\top A_j + \tau_{0h}\bar{\gamma}_{hj}(\omega) \quad \forall h \in H, j = 1, ..., n_1 \\ &\delta(\omega) \leq \tau_h^\top b + \tau_{0h}\bar{\nu}_h(\omega) \quad \forall h \in H \\ &\tau_h \geq 0, \tau_{0h} \geq 0, \quad \forall h \in H\}.\end{aligned} \tag{6.6}$$

Note that the epi-reverse polar only allows those facets of the convex hull of $\Pi_X(\omega)$ that have positive coefficient for the variable θ. If $\{\bar{\nu}, \bar{\gamma}\}$ are given then

$$\begin{aligned}\text{conv}\big(\Pi_X(\omega)\big) = \big\{(\theta, x) \ \mid\ &\forall (\sigma_0(\omega), \sigma(\omega), \delta(\omega)) \in \Pi_X^\dagger(\omega) \\ &x \in X, \theta \geq (\frac{\delta(\omega)}{\sigma_0(\omega)}) - (\frac{\sigma^\top(\omega)}{\sigma_0(\omega)})x\big\}.\end{aligned}$$

Let $(\sigma_0^i(\omega), \sigma^i(\omega), \delta^i(\omega))_{i \in \mathcal{I}}$ denote the set of extreme points of the epi-reverse polar, and let $\nu^i(\omega) = \frac{\delta^i(\omega)}{\sigma_0^i(\omega)}$ and $\gamma^i(\omega) = \frac{\sigma^i(\omega)}{\sigma_0^i(\omega)}$. For each $(x, \omega) \in X \times \Omega$, let $\pi_c(x, \omega) = \text{Max}_{i \in \mathcal{I}}\{\nu_i(\omega) - \gamma_i(\omega)^\top x\}$. Then for each $\omega \in \Omega$, $\pi_c(\cdot, \omega)$ is a convex function. Moreover, the epigraph of $\pi_c(x, \omega)$ restricted to X is the closure of the convex hull of $\Pi_X(\omega)$. We refer to $\pi_c(\cdot, \omega)$ as the convex hull approximation of $\pi_0(\cdot, \omega)$, and note that $\pi_0(x, \omega) = \pi_c(x, \omega)$ whenever x is an extreme point of X.

1.3 An Algorithmic Context for the C^3 Theorem

As a preview of the D^2 algorithm, let us consider the scenario subproblems in a temporal decomposition of the SMIP, (6.1). If we let x^k denote the first stage solution associated with the k^{th} algorithmic iteration, the subproblems are of the form:

$$\begin{aligned}f^k(x, \omega) = \text{Min} \quad & g^\top y \\ s.t. \quad & W^k y \geq r^k(\omega) - T^k(\omega)x \\ & y \in \Re_+^{n_2}\end{aligned} \tag{6.7}$$

Of course, in the first iteration we have

$$\begin{aligned} f^1(x,\omega) = \quad & \text{Min} \quad g^\top y \\ & s.t. \quad Wy \geq r(\omega) - T(\omega)x \\ & \quad\; y \in \Re_+^{n_2}, \end{aligned}$$

the LP relaxation of (6.2). Thus, the problem is initialized with $W^1 = W$, $r^1(\omega) = r(\omega)$, and $T^1(\omega) = T(\omega)$ as in (6.2). As iterations progress, cutting planes of the form

$$\pi^k y \geq \pi_c(x^k,\omega) = \text{Max}_{i\in I}\{\nu_i(\omega) - \gamma_i(\omega)^\top x^k\}$$

are added to the subproblem, thereby refining the approximation of the convex hull of integer solutions. As such, the vector π^k is appended to the matrix W^{k-1}, and the element identified $(\nu_i(\omega), \gamma_i(\omega))$ is appended to $\left(r^{k-1}(\omega), T^{k-1}(\omega)\right)$. Let $y^k(\omega) \in \text{argmin}\{g^\top y \mid W^k y \geq r^k(\omega) - T^k(\omega)x^k,\ y \in \Re_+^{n_2}\}$. If $z^k(\omega)$, the value assigned to integer variables in $y^k(\omega)$ is integer for all ω, then no update is necessary, and $W^{k+1} = W^k, r^{k+1}(\omega) = r^k(\omega)$, and $T^{k+1}(\omega) = T^k(\omega)$.

On the other hand, suppose that the subproblems do not yield integer optimal solutions. Let $j(k)$ denote an index j, for which $z_j^k(\omega)$ is non-integer for some $\omega \in \Omega$, and let $\bar{z}_{j(k)}$ denote one of the non-integer values $\{z_j^k(\omega)\}_{\omega\in\Omega}$. To eliminate this non-integer solution, a disjunction of the following form may be used:

$$\mathcal{S}_k(x^k,\omega) = \mathcal{S}_{0,j(k)}(x^k,\omega) \bigcup \mathcal{S}_{1,j(k)}(x^k,\omega)$$

where

$$\begin{aligned} \mathcal{S}_{0,j(k)}(x^k,\omega) = \{y \in \Re_+^{n_2} \mid\ & W^k y \geq r^k(\omega) - T^k(\omega)x^k && \text{(6.8a)} \\ & -z_{j(k)} \geq -\lfloor \bar{z}_{j(k)} \rfloor\} && \text{(6.8b)} \end{aligned}$$

and

$$\begin{aligned} \mathcal{S}_{1,j(k)}(x^k,\omega) = \{y \in \Re_+^{n_2} \mid\ & W^k y \geq r^k(\omega) - T^k(\omega)x^k && \text{(6.9a)} \\ & z_{j(k)} \geq \lceil \bar{z}_{j(k)} \rceil && \text{(6.9b)} \end{aligned}$$

The index $j(k)$ is referred to as the "disjunction variable" for iteration k. Our assumptions ensure that the subproblems remain feasible for any restriction of the integer variables, and thus both (6.8) and (6.9) are non-empty. Also, since the disjunction is based on an either-or-condition, $H = \{0,1\}$ is used. It should be noted that when the integer restrictions are binary, the right hand side of (6.8b) is zero, and the right hand side

of (6.9b) is one. This is precisely the disjunction used in lift-and-project cuts of Balas, Ceria and Cornuéjols [1993].

Let $\lambda_{0,1}$ denote the vector of multipliers associated with (6.8a), and $\lambda_{0,2}$ denote the scalar multiplier associated with (6.8b). Let $\lambda_{1,1}$ and $\lambda_{1,2}$ be similarly defined for (6.9a) and (6.9b), respectively. Assuming that the sets defined in (6.8) and (6.9) are non-empty for all $\omega \in \Omega$, the following problem may be used to generate the common cut coefficients, π^k, in iteration k:

$$
\begin{aligned}
\text{Max} \quad & E[\pi_0(\tilde{\omega})] - E[y^k(\tilde{\omega})]\pi \\
s.t. \quad & \pi_j \geq \lambda_{0,1}^\top W_j^k - I_j^k \lambda_{0,2} \ \forall j \\
& \pi_j \geq \lambda_{1,1}^\top W_j^k + I_j^k \lambda_{1,2} \ \forall j \\
& \pi_0(\omega) \leq \lambda_{0,1}^\top \big(r^k(\omega) - T^k(\omega)x^k\big) - \lambda_{0,2}^\top \lfloor \bar{z}_{j(k)} \rfloor \ \forall \omega \in \Omega \\
& \pi_0(\omega) \leq \lambda_{1,1}^\top \big(r^k(\omega) - T^k(\omega)x^k\big) + \lambda_{1,2}^\top \lceil \bar{z}_{j(k)} \rceil \ \forall \omega \in \Omega \\
& -1 \leq \pi_j \leq 1, \ \forall j, \ \ -1 \leq \pi_0(\omega) \leq 1, \ \forall \omega \in \Omega \\
& \lambda_{0,1}, \ \lambda_{0,2}, \ \lambda_{1,1}, \ \lambda_{1,2} \geq 0
\end{aligned}
\tag{6.10}
$$

where $I_j^k = \begin{cases} 0, & if \ j \neq j(k) \\ 1, & otherwise. \end{cases}$

The validity of the cut coefficients generated above follows from the disjunctive cut principle (Balas [1979]) which requires the multipliers (λ) to be chosen in such a way that the cut coefficients are greater than the aggregated columns as specified above. Since the coefficients π are independent of ω, the above LP generates common cut coefficients. This LP/SLP is formulated following the standard approach of generating valid inequalities in disjunctive programming (Sherali and Shetty [1980]), and it optimizes some measure of distance of the current solution $y^k(\omega)$ from the cut. It is interesting to note that this problem is a simple recourse problem, and may be interpreted as a stochastic version of the linear program used to generate the lift-and-project cuts.

Since the disjunction used for cut formation has $H = \{0, 1\}$, the epigraph $\pi_0(x, \omega)$ is a union of two polyhedral sets. Therefore, for each $\omega \in \Omega$, the following parameters are derived from an optimal solution of (6.10),

$$\bar{\nu}_0^k(\omega) = \lambda_{0,1}^\top r^k(\omega) - \lambda_{0,2} \lfloor \bar{z}_{j(k)} \rfloor,$$

$$\bar{\nu}_1^k(\omega) = \lambda_{1,1}^\top r^k(\omega) + \lambda_{1,2} \lceil \bar{z}_{j(k)} \rceil,$$

and

$$[\bar{\gamma}_h(\omega)]^\top = \lambda_{h,1}^\top T^k(\omega), \ \forall h \in H$$

are used to update the approximation of the polyhedron defined via (6.6), which we denote as $\Pi_X^{\dagger}(\omega)^k$. This polyhedron represents the epi-reverse polar, which provides access to the convexification of π_0. Correspondingly, for each $\omega \in \Omega$, the following LP is used to approximate $\pi_0(x, \omega)$:

$$\begin{array}{ll} \text{Max} & \delta(\omega) - \sigma_0(\omega) - (x^k)^\top \sigma(\omega) \\ \text{s.t.} & (\sigma_0(\omega), \sigma(\omega), \delta(\omega)) \in (\Pi_X^{\dagger}(\omega))^k \end{array} \tag{6.11}$$

With an optimal solution to (6.11), $(\sigma_0^k(\omega), \sigma^k(\omega), \delta^k(\omega))$, we obtain $\nu^k(\omega) = \frac{\delta^k(\omega)}{\sigma_0^k(\omega)}$ and $\gamma^k(\omega) = \frac{\sigma^k(\omega)}{\sigma_0^k(\omega)}$. For each $\omega \in \Omega$, these coefficients are used to update the right-hand-side functions $r^{k+1}(\omega) = [r^k(\omega), \nu^k(\omega)]$, and $T^{k+1}(\omega) = [T^k(\omega)^\top; \gamma^k(\omega)]^\top$. Similarly, the solution to (6.10) is used to update the constraint matrix, $W^{k+1} = [(W^k)^\top; \pi^k]^\top$.

The master program is defined as:

$$\begin{array}{ll} \text{Min} & c^\top x + F^k(x) \\ s.t. & Ax \geq b \\ & x \in X \bigcap \mathcal{B} \end{array} \tag{6.12}$$

where $F^k(\cdot)$ is a piecewise linear approximation of the subproblem objective function, $E[f(x, \tilde{\omega})]$ in the k^{th} iteration.

1.4 Disjunctive Decomposition with Set Convexification

The Basic D^2 Algorithm (Sen and Higle [2000]) can be stated as follows:

Basic D^2 Algorithm

0. Initialize. $V_1 \leftarrow \infty$. $\epsilon > 0$ and $x^1 \in X$ are given. $k \leftarrow 1$, $W^1 \leftarrow W$, $T^1(\omega) \leftarrow T(\omega)$, and $r^1(\omega) = r(\omega)$.

1. Solve one LP Subproblem for each $\omega \in \Omega$ For each $\omega \in \Omega$, use the matrix W^k as well as the right hand side vector $r^k(\omega) - T^k(\omega)x^k$ to solve (6.7). If $\{y^k(\omega)\}_{\omega \in \Omega}$ satisfy the integer restrictions, $V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \tilde{\omega})], V_k\}$, and go to step 4.

2. Solve Multiplier/Cut Generation LP/SLP and Perform Updates Choose a disjunction variable $j(k)$.

(i) Formulate and solve (6.10) to obtain π^k and define $W^{k+1} = [(W^k)^\top; \pi^k]^\top$.

(ii) Using the multipliers λ_0^k, λ_1^k and the value $\bar{z}_{j(k)}$ obtained in (i) solve (6.11) for each outcome ω. The solution defines $\nu^k(\omega)$ and $\gamma^k(\omega)$ which

are used to update the right hand side functions: $r^{k+1}(\omega) = [r^k(\omega), \nu^k(\omega)]$ and $T^{k+1}(\omega) = [T^k(\omega)^\top; \gamma^k(\omega)]^\top$.

3. Update and Solve one LP Subproblem for each $\omega \in \Omega$ For each $\omega \in \Omega$ solve (6.7) using W^{k+1} and $r^{k+1}(\omega) - T^{k+1}(\omega)x^k$. If $y^k(\omega)$ satisfies the integer restrictions for all $\omega \in \Omega$, $V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \omega)], V_k\}$. Otherwise, $V_{k+1} \leftarrow V_k$.

4. Update and Solve the Master Problem Using the dual multipliers from the most recently solved subproblem for each $\omega \in \Omega$ (either step 1 or step 3), update the approximation F^k by adopting a standard decomposition method (e.g Benders [1962]). Let $x^{k+1} \in \text{argmin}\{c^\top x + F^k(x) \mid x \in X\}$, and let v_k denote the optimal value of the master problem. If $V_k - v_k \leq \epsilon$, stop. Otherwise, $k \leftarrow k + 1$ and repeat from step 1.

2. An Illustration of the D^2 Algorithm

Consider the following two-stage SIP example problem with two scenarios:

$$\begin{array}{ll} \text{Min} & -1.5x_1 - 4x_2 + E[f(x_1, x_2, \omega)] \\ s.t. & x_1, x_2 \in \{0, 1\} \end{array}$$

where,

$$\begin{array}{lll} f(x_1, x_2, \omega) = & \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 \\ & s.t. & -2y_1 - 3y_2 - 4y_3 - 5y_4 \geq -\omega^1 + x_1 \\ & & -6y_1 - 1y_2 - 3y_3 - 2y_4 \geq -\omega^2 + x_2 \\ & & y_1, y_2, y_3, y_4 \in \{0, 1\} \end{array}$$

The first stage variables are $x = [x_1, x_2]^\top$, while the second stage variables are $y = [y_1, y_2, y_3, y_4]^\top$. There are two scenarios are $\omega_1 = [5, 2]^\top$ and $\omega_2 = [10, 3]^\top$, each occurring with probability 0.5. This instance is motivated by the example in Schultz, Stougie, and van der Vlerk [1998], where the second stage involves general integer variables. To ensure that the subproblems remain feasible for any restriction on the integer variables, we include an artificial variable, denoted R, which is penalized in the objective at a rate of 100. Thus, we recast the problems as

$$\begin{array}{ll} \text{Min} & -1.5x_1 - 4x_2 + 0.5f_1(x_1, x_2, \omega_1) + 0.5f_2(x_1, x_2, \omega_2) \\ s.t. & x_1, x_2 \in \{0, 1\} \end{array}$$

where

$$
\begin{array}{rll}
f_1(x_1, x_2, \omega_1) = & \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -5 + x_1 \\
& & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -2 + x_2 \\
& & y_1, y_2, y_3, y_4 \in \{0, 1\}, \ R \geq 0
\end{array}
$$

and

$$
\begin{array}{rll}
f_2(x_1, x_2, \omega_2) = & \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -10 + x_1 \\
& & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -3 + x_2 \\
& & y_1, y_2, y_3, y_4 \in \{0, 1\}, \ R \geq 0
\end{array}
$$

In this problem we have the following input data:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$b = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

$$W = \begin{bmatrix} -2 & -3 & -4 & -5 & 1 \\ -6 & -1 & -3 & -2 & 1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},$$

$$T(\omega) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ for both scenarios,}$$

$$r(\omega_1) = \begin{bmatrix} -5 \\ -2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$$\text{and } r(\omega_2) = \begin{bmatrix} -10 \\ -3 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

Note that with A and b as defined above, binary solutions are extreme points of $X = \{x : Ax \geq b, x \geq 0\}$, as required. It is easily seen that for binary values of the second stage variables a lower bound on the objective $-16y_1 - 19y_2 - 23y_3 - 28y_4$ is -86. In order to be consistent with the requirement that the lower bound on the second stage objective value must be zero, we translate the second stage objective function by adding 86, thereby ensuring nonnegativity after the translation. We can now start the D^2 algorithm.

Iteration 1 (k = 1)

Step 0
The D^2 algorithm is initialized with the following master program:

$$\begin{array}{ll} \text{Min} & -1.5x_1 - 4x_2 + \theta \\ s.t. & -x_1 \geq -1 \\ & -x_2 \geq -1 \\ & x_1, x_2 \in \{0,1\}, \theta \geq 0. \end{array}$$

The initial master program yields $x^1 = [1,1]$, and $\theta = 0$. The upper and lower bounds are initialized as $V_0 = \infty$ and $v_0 = -5.5$, respectively. For the first iteration of the algorithm we set $V_1 = V_0$, $W^1 = W$, $T^1(\omega) = T(\omega)$, and $r^1(\omega) = r(\omega)$.

Step 1

For step 1 of the algorithm we use $x_1 = 1, x_2 = 1$ and solve the linear relaxation of the second stage subproblem for ω_1 and ω_2, which we call

LP_1 and LP_2, respectively:

$$
\begin{array}{lll}
LP_1 : f_1(1,1,\omega_1) = & \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -4 \\
& & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -1 \\
& & -y_1 \geq -1 \\
& & -y_2 \geq -1 \\
& & -y_3 \geq -1 \\
& & -y_4 \geq -1 \\
& & y_1, y_2, y_3, y_4, R \geq 0.
\end{array}
$$

and

$$
\begin{array}{lll}
LP_2 : f_2(1,1,\omega_2) = & \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -9 \\
& & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& & -y_1 \geq -1 \\
& & -y_2 \geq -1 \\
& & -y_3 \geq -1 \\
& & -y_4 \geq -1 \\
& & y_1, y_2, y_3, y_4, R \geq 0.
\end{array}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0, 1, 0, 0]$, $R(\omega_1) = 0$. and for LP_2 is $y(\omega_2) = [0, 1, 0, 0.5]$, $R(\omega_2) = 0$.

Step 2

Since $y(\omega_2)$ does not satisfy the integer restrictions, we choose y_4 as the "disjunction variable" and create the disjunction $y_4 \leq 0$ or $y_4 \geq 1$ for LP_2. We formulate (6.10), which yields the vector π^1 for updating W^1 and the data for (6.11) whose optimal solution is used to update the right-hand side of the second-stage constraints. An optimal solution for (6.10) is $\pi^1 = [1, -1, 1, -1, 1]$, $\lambda_{0,1} = [0, 0, 0, 1, 0, 0]$, $\lambda_{0,2} = 1$, $\lambda_{1,1} = [0, 1, 0, 0, 0, 0]$, and $\lambda_{1,2} = 1$. We obtain W^2 by appending π^1 to W^1: $W^2 = \begin{bmatrix} & [W^1] & \\ 1 & -1 \quad 1 \quad -1 & 1 \end{bmatrix}$. Using the solution from (6.10), we formulate and solve (6.11) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = -0.5$, $\sigma_0(\omega_1) = 0.5$, and $\sigma(\omega_1) = [0, 0]$. Based on this solution we update $r^1(\omega_1)$ and $T^1(\omega_1)$ as follows: $r^2(\omega_1) = \begin{bmatrix} [r^1(\omega_1)] \\ -1 \end{bmatrix}$, $T^2(\omega_1) = \begin{bmatrix} [T^1(\omega_1)] & \\ 0 & 0 \end{bmatrix}$. For ω_2 the optimal solution of LP (6.11)

is $\delta(\omega_2) = -1$, $\sigma_0(\omega_2) = 0.5$, and $\sigma(\omega_2) = [0, -0.5]$. Similarly, we update $r^1(\omega_2)$ and $T^1(\omega_2)$ as follows: $r^2(\omega_2) = \begin{bmatrix} [r^1(\omega_2)] \\ -2 \end{bmatrix}$, $T^2(\omega_2) = \begin{bmatrix} [T^1(\omega_2)] & \\ 0 & -1 \end{bmatrix}$. This completes Step 2 of the algorithm.

Step 3
Solving (6.7), we obtain $y(\omega_1) = [0, 1, 0, 0]$, with $R(\omega_1) = 0$, and $y(\omega_2) = [0, 1, 0.2, 0.2]$, with $R(\omega_2) = 0$. The dual solutions are $d(\omega_1) = [0, 14, 0, 0, 5, 0, 0]$ and $d(\omega_2) = [0, 10.2, 7.6, 0, 1.2, 0, 0]$. $V_2 \leftarrow V_1$, because the integer restrictions are not satisfied.

Step 4
Using the dual solution for each subproblem from Step 3, we formulate the "optimality cuts" as in Benders' decomposition. The resulting cuts are $0x_1 - 14x_2 + \eta_1 \geq -33$ for ω_1 and $0x_1 - 17.8x_2 + \eta_2 \geq -47$ for ω_2. Since the two scenarios are equally likely, the expected values associated with the cut coefficients yield $0x_1 - 15.9x_2 + \eta \geq -40$. Applying the translation $\theta = \eta + 86$ we get $0x_1 - 15.9x_2 + \theta \geq 46$ as the optimality cut to add to the master program:

$$
\begin{aligned}
\text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\
s.t. \quad & -x_1 \geq -1 \\
& -x_2 \geq -1 \\
& 0x_1 - 15.9x_2 + \theta \geq 46 \\
& x_1, x_2 \in \{0, 1\}, \theta \geq 0.
\end{aligned}
$$

Solving the master program we get $x^2 = [1, 0]$, $\theta = 46$ and an objective value of 44.5. Therefore, the lower bound becomes $v_2 = 44.5$. The upper bound remains the same, $V_2 = V_1 = \infty$, as before. This completes the first iteration of the algorithm. Since $V_2 > v_2$ $k \leftarrow 2$, and we begin the next iteration.

Iteration 2

Step 1
We start the second iteration by solving the following updated subprob-

lems with $x^2 = [1, 0]$:

$$
\begin{aligned}
LP_1 : f_1(1, 0, \omega_1) = \quad & \text{Min} \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. \quad -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -4 \\
& \quad -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& \quad y_1 - y_2 - y_3 - y_4 + R \geq -1 \\
& \quad -y_1 \geq -1 \\
& \quad -y_2 \geq -1 \\
& \quad -y_3 \geq -1 \\
& \quad -y_4 \geq -1 \\
& \quad y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

and

$$
\begin{aligned}
LP_2 : f_2(1, 0, \omega_2) = \quad & \text{Min} \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
& s.t. \quad -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -9 \\
& \quad -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -3 \\
& \quad y_1 - y_2 - y_3 - y_4 + R \geq -2 \\
& \quad -y_1 \geq -1 \\
& \quad -y_2 \geq -1 \\
& \quad -y_3 \geq -1 \\
& \quad -y_4 \geq -1 \\
& \quad y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0.108108, 1, 0.027027, 0.135135]$ and for LP_2 it is $y(\omega_2) = [0, 1, 0, 1]$.

Step 2

Since $y(\omega_1)$ does not satisfy the integer restrictions, we choose y_4 as the "disjunction variable" and create the disjunction $y_4 \leq 0$ or $y_4 \geq 1$ for LP_1. We formulate and solve (6.10), which yields the data used to update W^1 and to formulate (6.11) whose optimal solution is used to update the right-hand side of the second-stage constraints. Solving LP (6.10) we obtain $\pi^2 = [0, -0.5, 0, -0.5, 1]$, $\lambda_{0,1} = [0, 0, 0, 0, 0.5, 0, 0]$, $\lambda_{0,2} = 0.5$, $\lambda_{1,1} = [0, 0.125, 0, 0.375, 0, 0, 0]$, and $\lambda_{1,2} = 0.125$. We obtain W^3 by appending π^2 to W^2: $W^3 = \begin{bmatrix} & [W^2] & \\ 0 & -0.5 \quad 0 \quad -0.5 & 1 \end{bmatrix}$. Using the solution of (6.10) we formulate and solve (6.11) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = -0.25$, $\sigma_0(\omega_1) = 0.5$, and $\sigma(\omega_1) = [0, 0]$. Based on this solution we update $r^2(\omega_1)$ and $T^2(\omega_1)$ as follows: $r^3(\omega_1) = \begin{bmatrix} [r^2(\omega_1)] \\ -0.5 \end{bmatrix}$, $T^3(\omega_1) = \begin{bmatrix} [T^2(\omega_1)] & \\ 0 & 0 \end{bmatrix}$.

For ω_2 the optimal solution of LP (6.11) is $\delta(\omega_2) = -0.5$, $\sigma_0(\omega_2) = 0.5$,

and $\sigma(\omega_2) = [0, 0]$. Similarly, we update $r^2(\omega_2)$ and $T^2(\omega_2)$ as follows: $r^3(\omega_2) = \begin{bmatrix} [r^2(\omega_2)] \\ -1 \end{bmatrix}$, $T^3(\omega_2) = \begin{bmatrix} [T^2(\omega_2)] & \\ 0 & 0 \end{bmatrix}$. This completes Step 2 of the algorithm.

Step 3

Solving (6.7) we obtain $y(\omega_1) = [0.055556, 1, 0.22222, 0]$ with $R = 0$, and $y(\omega_2) = [0, 1, 0, 1]$ with $R = 0$. The dual solutions are $d(\omega_1) = [5, 1, 0, 2, 0, 2, 0, 0]$ and $d(\omega_2) = [0, 7.667, 0, 0, 0, 11.333, 0, 12.667]$. $V_3 \leftarrow V_2$ because the integer restrictions have not been met.

Step 4

Using the dual solution for each subproblem from step 3, we formulate the "optimality cuts" as in Benders' decomposition. The resulting cuts are $-5x_1 - 1x_2 + \eta_1 \geq -30$ for ω_1 and $0x_1 - 7.667x_2 + \eta_2 \geq -47$ for ω_2. The expected value associated with the cut coefficients yields $-2.5x_1 - 4.334x_2 + \eta \geq -38.5$. Applying the translation $\theta = \eta + 86$ we get $-2.5x_1 - 4.334x_2 + \theta \geq 47.5$ as the optimality cut to add to the master program:

$$
\begin{aligned}
\text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\
s.t. \quad & -x_1 \geq -1 \\
& -x_2 \geq -1 \\
& 0x_1 - 15.9x_2 + \theta \geq 46 \\
& -2.5x_1 - 4.334x_2 + \theta \geq 47.5 \\
& x_1, x_2 \in \{0, 1\}, \theta \geq 0.
\end{aligned}
$$

Solving the master program we get $x^3 = [0, 0]$, $\theta = 47.5$ and an objective value of 47.5. Therefore, the lower bound becomes $v_2 = 47.5$. $k \leftarrow 3$, and we begin the next iteration.

Iteration 3

Step 1
We start the third iteration by solving the following updated subproblems with $x^3 = [0, 0]$:

$$
\begin{aligned}
LP_1 : f_1(0,0,\omega_1) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
s.t. \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -5 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -1 \\
& 0y_1 - 0.5y_2 - 0y_3 - 0.5y_4 + R \geq -0.5 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

and

$$
\begin{aligned}
LP_2 : f_2(0,0,\omega_2) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
s.t. \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -10 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -3 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -2 \\
& 0y_1 - 0.5y_2 - 1y_3 - 0.5y_4 + R \geq -1 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0, 0, 0, 1]$, $R(\omega_1) = 0$ and for LP_2 the optimal solution is $y(\omega_2) = [0, 1, 0, 1]$, $R(\omega_2) = 0$. The dual solutions are $d(\omega_1) = [0, 7.6667, 0, 25.333, 0, 0, 0, 0]$ and $d(\omega_2) = [0, 7.667, 0, 0, 0, 11.333, 0, 12.667]$. We now have an incumbent integer solution $x = [0, 0]$, $y(\omega_1) = [0, 0, 0, 1]$, $y(\omega_2) = [0, 1, 0, 1]$, $\theta = 86 + 0.5(-28) + 0.5(-47) = 48.5$.
$V_4 \leftarrow \text{Min}\{48.5, V_3\} = 48.5$. We go to step 4 of the algorithm.

Step 4

Using the dual solution for each subproblem from Step 3, we formulate the "optimality cuts" as in Benders' decomposition. The resulting cuts are $0x_1 - 7.667x_2 + \eta_1 \geq -28$ for ω_1 and $0x_1 - 7.667x_2 + \eta_2 \geq -47$ for ω_2, and the expected values yield $0x_1 - 7.667x_2 + \eta \geq -37.5$. Applying the translation $\theta = \eta + 86$ we get $0x_1 - 7.667x_2 + \theta \geq 48.5$ as the optimality cut to add to the master program:

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\ s.t. \quad & -x_1 \geq -1 \\ & -x_2 \geq -1 \\ & 0x_1 - 15.9x_2 + \theta \geq 46 \\ & -2.5x_1 - 4.334x_2 + \theta \geq 47.5 \\ & 0x_1 - 7.667x_2 + \theta \geq 48.5 \\ & x_1, x_2 \in \{0,1\}, \theta \geq 0. \end{aligned}$$

Solving the master program we get $x^4 = [1,0]$, $\theta = 50$ and an objective value of 48.5. Therefore, the lower bound becomes $v_3 = 48.5$. Since the upper bound ($V_3 = 48.5$) and the lower bound are now equal, the algorithm terminates and we have an optimal solution: $x = [0,0], y(\omega_1) = [0,0,0,1], y(\omega_2) = [0,1,0,1]$ and objective value –37.5. It so happens that both [0,0] and [1,0] are optimal for the master problem, but optimality can only be concluded for the point [0,0], since that is the incumbent.

It is interesting to note that while the cuts used in LP_1 and LP_2 in iteration 3 were obtained at $x^1 = [1,1]$ and $x^2 = [1,0]$, integer solutions ($y(\omega_1) = [0,0,0,1]$ and $y(\omega_2) = [0,1,0,1]$) were obtained with the first relaxations solved at $x^3 = [0,0]$. The credit for this should go to the C^3 Theorem.

3. Conclusions

This paper has presented the main results on set convexifications for large scale Stochastic Integer Programming and has given an illustration of the new decomposition method called the D^2 algorithm. At the heart of this novel method is the C^3 Theorem, which allows both a temporal and scenario decomposition of the SMIP. We have used a simple example to illustrate the application of the D^2 algorithm. In this example the D^2 algorithm converges to an optimal solution in three iterations. The example clearly illustrates how the second-stage convexifications are sequentially carried out and how they impact the first stage objective function.

Our primary focus in this paper is the generation of cutting planes within a temporal decomposition of two-stage SMIPs. We note, however, that cutting planes alone are typically inadequate for solving large mixed-integer programs. Thus, our ultimate goal is to use cuts such as those discussed in this paper within a Branch-and-Cut (BAC) setting, where a careful generation of cuts is necessary to further enhance the success of BAC-type algorithms for solving SMIP problems. Therefore, our future work is to incorporate the D^2 algorithm in a branch-and-cut setting. Moreover, the computational demands of this class of problems calls for the use of high performance computing platforms.

Acknowledgements: This research was supported by a grant from the National Science Foundation.

References

Balas, E. [1979], "Disjunctive Programming," *Annals of Discrete Mathematics*, **5**, pp. 3-51.

Balas, E.S. Ceria, and G. Cornuéjols [1993], "A lift-and-project cutting plane algorithm for mixed 0-1 integer programs," *Math. Programming*, **58**, pp. 295-324.

Benders,J.F. [1962], "Partitioning procedures for solving mixed-variable programming problems," *Numerische Mathematic* **4**, pp. 238-252.

Blair,C. [1995], "A closed-form representation of mixed-integer program value functions," *Math. Programming*, **71**, pp. 127-136.

Blair,C., and R. Jeroslow [1982], "The value function of an integer program," *Math. Programming*, **23**, pp. 237-273.

Carøe, C.C. [1998], *Decomposition in Stochastic Integer Programming*, Ph.D. thesis, Institute of Mathematical Sciences, Dept. of Operations Research, University of Copenhagen, Denmark.

Schultz, R., L. Stougie, and M.H. van der Vlerk [1998], "Solving stochastic programs with integer recourse by enumeration; a framework using Gröbner basis reduction," *Math. Programming*, **83**, pp. 71-94.

Sen, S. and J.L. Higle [2000], The C^3 Theorem and a D^2 Algorithm for Large Scale Stochastic Integer Programming: Set Convexification, Working Paper, Dept. of Systems and Industrial Engineering, The University of Arizona, Tucson AZ 85721. (submitted to *Math. Programming*

Sen, S. and H.D. Sherali [1987], "Nondifferentiable reverse convex programs and facetial cuts via a disjunctive characterization," *Math. Programming,* **37**, pp. 169-183.

Sherali, H.D. and W.P. Adams [1990], "A hierarchy of relaxations between the continuous and convex hull representations for 0-1 programming problems," *SIAM J. on Discrete Mathematics*, **3**, pp. 411-430.

Sherali, H.D. and B.M.P. Fraticelli [2002], "A modification of Benders' decomposition algorithm for discrete subproblems: an approach for stochastic programs with integer recourse," *Journal of Global Optimization*, **22**, pp. 319-342.

Sherali, H.D. and C.M. Shetty [1980], *Optimization with Disjunctive Constraints*, Lecture Notes in Economics and Math. Systems, Vol. 181, Springer-Verlag, Berlin.