NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NETWORK INTERDICTION BY LAGRANGIAN RELAXATION AND BRANCH-AND-BOUND

by

Adnan Uygun

June 2002

Thesis Advisor: R. Kevin Wood
Second Reader: Kelly J. Cormican

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Network Interdiction by Lagrangian Relaxatian and Branch-and-Bound			**5. FUNDING NUMBERS**
6. AUTHOR(S) Uygun, Adnan			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The maximum-flow network-interdiction problem (MXFI) arises when an interdictor, using limited interdiction resources, wishes to restrict an adversary's use of a capacitated network.

MXFI is not easy to solve when converted to a binary integer program. Derbes (1997) uses Lagrangian relaxation to solve the problem, at least approximately, for a single value of available resource, *R*. Bingol (2001) extends this technique to solve MXFI approximately for all integer values of *R* in a specified range. But, "problematic *R*-values" with substantial optimality gaps do arise. We reduce optimality gaps in two ways. First, we find the best Lagrangian multiplier for problematic *R*-values by following the slope of the Lagrangian function. Secondly, we apply a limited-enumeration branch-and-bound algorithm.

We test our algorithms on six different test networks with up to 402 nodes and 1826 arcs. The algorithms are coded in Java 1.3 and run on a 533 MHz Pentium III computer. The first technique takes at most 39.3 seconds for any problem, and for one instance, it solves 8 of the problem's 15 problematic *R*-values. For that problem, the second technique solves four of the remaining problematic *R*-values, but run time increases by two orders of magnitude.

14. SUBJECT TERMS Network Interdiction, Mathematical Modeling, Lagrangian Relaxation, Branch-and-Bound			**15. NUMBER OF PAGES** 65
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

NETWORK INTERDICTION BY LAGRANGIAN RELAXATION AND BRANCH-AND-BOUND

Adnan Uygun
First Lieutenant, Turkish Army
B.S., Turkish Army Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author: Adnan Uygun

Approved by: R. Kevin Wood
Thesis Advisor

Kelly J. Cormican
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The maximum-flow network-interdiction problem (MXFI) arises when an interdictor, using limited interdiction resources, wishes to restrict an adversary's use of a capacitated network.

MXFI is not easy to solve when converted to a binary integer program. Derbes (1997) uses Lagrangian relaxation to solve the problem, at least approximately, for a single value of available resource, *R*. Bingol (2001) extends this technique to solve MXFI approximately for all integer values of *R* in a specified range. But, "problematic *R*-values" with substantial optimality gaps do arise. We reduce optimality gaps in two ways. First, we find the best Lagrangian multiplier for problematic *R*-values by following the slope of the Lagrangian function. Secondly, we apply a limited-enumeration branch-and-bound algorithm.

We test our algorithms on six different test networks with up to 402 nodes and 1826 arcs. The algorithms are coded in Java 1.3 and run on a 533 MHz Pentium III computer. The first technique takes at most 39.3 seconds for any problem, and for one instance, it solves 8 of the problem's 15 problematic *R*-values. For that problem, the second technique solves four of the remaining problematic *R*-values, but run time increases by two orders of magnitude.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I must express my deepest gratitude and appreciation to Professor Kevin Wood for his guidance, encouragement, and patience during this most challenging part of my education at the Naval Postgraduate School. I wish to thank LCDR Kelly Cormican for his valuable suggestions and motivational leadership. I am grateful to the Turkish Army for providing me with the privilege of studying at NPS.

My special thanks go to my parents, Resul and Saziye Uygun, for their unending support from thousands of miles away. Last but not least, I would like to thank my nephew, Dorukhan Uygun, for being an inadvertent source of inspiration through his own success.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis is concerned with solving or approximately solving a maximum-flow network-interdiction problem (MXFI). This problem arises when an interdictor, using limited interdiction resources, wishes to restrict an adversary's use of a capacitated network. In particular, the adversary, or "network user" seeks to maximize flow through the network, and the interdictor seeks to minimize that maximum flow by destroying arcs.

MXFI is not easy to solve when converted to a binary integer program. Derbes (1997) uses Lagrangian relaxation to attempt to solve this problem for a single value of interdiction resource, R. Bingol (2001) extends this technique to solve MXFI for all integer values of R in some specified range assuming that exactly one unit of resource is required to interdict an arc. However, about 25% of these values are "problematic" in tests, i.e., positive optimality gaps arise.

In our study, we first focus on decreasing the optimality gaps by finding the best Lagrangian multiplier. We simply follow the slope of the Lagrangian function to accomplish this. For most test problems, this method decreases the gap for problematic R-values by approximately 50%. This method helps find better feasible solutions and sometimes prove optimality. But, some large optimality gaps may remain.

We tackle the remaining problematic R-values with a restricted branch-and-bound technique, which has only one level of enumeration below the root. This technique creates a strict partition of the feasible region and the minimum lower bound from among these partitions is a lower bound on the optimal solution value. In our tests, we solve approximately 70% of the problematic R-values that remain after applying the first technique. However, branch-and-bound increases run times dramatically.

Both procedures are written and compiled using the Java 1.3 programming language. All tests are performed on a personal computer with a 533 MHz Pentium III processor and 192 MB of RAM, running under the Windows XP operating system. For test purposes, we have used 6 different grid networks. The smallest one has 27 nodes and 86 arcs and the largest one has 402 nodes and 1826 arcs.

Applying both techniques together, we have been able to either solve problematic *R*-values exactly or improve their optimality gaps for most cases. In fact, 85% of the problematic *R*-values are solved optimally in our tests. One problem with a large optimality gap does not improve, however.

I. INTRODUCTION

This thesis develops a method based on Lagrangian relaxation to solve a maximum-flow network-interdiction problem. This problem arises when an interdictor, using limited interdiction resources, wishes to restrict an adversary's use of a capacitated network. In particular, the adversary, or "network user" seeks to maximize flow through the network, and the interdictor seeks to minimize that maximum flow by destroying arcs.

The maximum-flow network-interdiction problem (MXFI), converted to an integer program, is difficult to solve because of an interdiction-resource constraint. Derbes (1997) uses Lagrangian relaxation to move this constraint into the objective function, and is able to solve MXFI efficiently for certain resource levels R. However, he solves for only one value of R at a time. The goal of this thesis is to solve MXFI efficiently for all reasonable values of R in a single procedure.

The underlying reasons that motivate us to obtain solutions for all values of R are: First, R may not be specified. For instance, at an early stage of planning, we may not know how much resource will be available when the interdiction plan is to be carried out. Thus, we would like to provide the decision-maker with solutions for all potential values of R. Secondly, decision-makers often like to have some input on the amount of resource to be expended. This helps them review and maybe revise their decisions. Finally, it is not much harder to solve MXFI for all reasonable values of R than for a single value.

Derbes models the interdiction of arc k as requiring a small, integer amount of resource, $r_k > 0$. For the sake of simplicity, we assume $r_k = 1$ for all interdictable arcs k, although arcs may also be specified as non-interdictable. Derbes also considers a dynamic version of MXFI in which flows require time to move through the network and interdicted arcs can be repaired over time. The dynamic version of MXFI is beyond the scope of this thesis.

Bingol (2001) investigates our version of MXFI and also attempts to solve for all R using a Lagrangian-relaxation procedure. Essentially, he estimates the values of the Lagrangian functions for all R simultaneously. He assumes that these functions only break at points at which the Lagrangian multiplier equals the capacity of some arc. In carrying

out computations, he encounters some values of R for which his procedure cannot find an exact solution, so he develops a heuristic to handle these cases and determines associated optimality gaps. He does not, however, optimize the Lagrangian-based lower bounds.

In this thesis, we use a procedure similar to Bingol's to find solutions for all values of R but will optimize the Lagrangian lower bounds as needed. When significant optimality gaps are identified, we apply a restricted branch-and-bound procedure to improve the lower bounds and possibly identify better solutions. This procedure is restricted to create enumeration trees having limited depth.

A. BACKGROUND

There are different types of network-interdiction problems. For instance, Israeli (1999) focuses on Maximizing the Shortest Path (MXSP) to slow a network user's movement between two specified nodes in a network. The k-most-vital-arcs problem is a special case of MXSP in which the interdictor seeks to destroy exactly k arcs to maximize the shortest path length. Steinrauf (1991) designs a model to isolate a targeted demand node in a network. Morton (2001) describes a stochastic network-interdiction model to help detect the smuggling of stolen nuclear materials. However, in this thesis, we are only concerned with the deterministic maximum-flow network-interdiction problem, MXFI.

The scientific literature on network interdiction begins with Wollmer (1966) who focuses on "Removing Arcs from a Network," where the objective is to maximize the reduction of the flow between an origin and a destination node. Later, Wollmer (1970) presents two algorithms for targeting strikes in an LOC (lines-of-communication) network attempting to make the network user's costs as large as possible over time. The LOC networks are described by a directed network, with arcs representing road, rail, or waterway segments. The costs in his study include both flow costs in dollars, vehicle hours, manpower units, or any other appropriate measures; plus repair costs incurred by strikes, measured in similar units.

McMasters and Mustin (1970) devise an algorithm to determine the optimum interdiction plan for minimizing network flow capacity when the minimum capacity on an arc is positive and the cost of interdiction is linear in the amount of arc capacity reduction.

Ratiff, Sicilia and Lubore (1975) show how to find a set of arcs, called "the N most vital links," whose simultaneous removal from a single-commodity flow network results in the greatest decrease in the throughput capacity of the remaining system.

Steinrauf (1991) develops two mathematical programs to determine strategies to interdict a network using limited resources. The first model identifies a set of arcs whose interdiction minimizes the maximum flow through the network, constrained by the available resources. This is essentially MXFI. The second model identifies a set of arcs to interdict that isolates a target node and the largest possible set of contiguous nodes. The latter model can be used if the exact location of a target node is uncertain.

Wood (1993) shows that the MXFI is NP-complete even if the interdiction of an arc requires exactly one unit of resource. New, flexible integer-programming models are developed for the problem and a number of variants (partial arc interdiction, multiple sources and sinks, undirected networks, multiple resources, multiple commodities). Valid inequalities and a reformulation are derived to tighten the LP relaxations of some of these models.

Cormican (1995) solves MXFI using Bender's decomposition, which takes advantage of easy-to-solve network-flow subproblems. Cormican et al. (1998) solve stochastic versions of MXFI where arc capacities and/or interdiction success may be uncertain. Stochastic versions of MXFI are beyond the scope of this thesis, however.

Whiteman (1999) discusses techniques for planning the interdiction of a complex infrastructure network to improve single-strike effectiveness. The motivation for his work arises from the huge cost of modern weapons and delivery platforms and intolerance for casualties, and the consequential desire to minimize the number of subsequent attacks. To optimize target selection, he modifies the integer program developed by Wood (1993), specifying different flow goals.

Bingol (2001) extends the earlier work by Derbes (1997) on a Lagrangian-relaxation technique to solve MXFI for all integer values of total interdiction resource, R, in some specified range. His basic procedure solves MXFI exactly for most values of R, but "problematic values" of R arise with positive optimality gaps.

B. OUTLINE OF THESIS

This chapter has provided background on network interdiction problems and has surveyed previous studies in this field. The following chapter explains MXFI in detail, and develops the Lagrangian-relaxation technique for solving it, at least approximately. In Chapter III, we first study the algorithms used by Derbes (1997) and Bingol (2001). Subsequently, we present two new procedures to improve optimality gaps for problematic *R*-values and sometimes even solve these problems exactly. Chapter IV provides the computational results for the new procedures. Finally, Chapter V provides conclusions and points out directions for further research.

II. FUNDAMENTALS OF NETWORK INTERDICTION

This chapter first defines necessary concepts and notation for our study. The notation is similar to that used by Wood (1993) and Bingol (2001). The second part of the chapter starts with MXFI, which is a min-max model, and develops a minimizing integer-programming formulation (MXFI-IP) and a Lagrangian relaxation (MXFI-LR) of that integer program.

A. DEFINITIONS AND NOTATION

$G = (N, A)$ denotes a directed network with node set N and arc set A. An arc $k = (i, j)$ starts from "tail node" i and ends at "head node" j. Each arc k has a capacity $u_k > 0$. The network has a source node s and a sink node t. For mathematical purposes, we add to A the artificial arc $a = (t, s)$ with $u_a = \infty$. The *forward star* of node i, $FS(i)$, includes all arcs of the form (i, j). The *reverse star* of node i, $RS(i)$, represents all arcs of the form (j, i).

In this study, only arc interdiction is allowed. In the test networks, r_k is the amount of *interdiction resource* necessary to destroy arc k. For simplicity, we assume $r_k = 1$ or $r_k = \infty$; the latter case simply means the arc cannot be interdicted.

A *cut* (N_s, N_t) partitions the node set N into two subsets, N_s and N_t where $s \in N_s$ and $t \in N_t$. An arc (i, j) is a *forward arc* of cut (N_s, N_t) if $i \in N_s$ and $j \in N_t$.

A_C represents the set of forward arcs for cut (N_s, N_t). The capacity of a cut, $u(N_s, N_t)$, is the sum of the capacities of the forward arcs in that cut:

$$u(N_s, N_t) = \sum_{k \in A_C} u_k .$$

If $u(N_s, N_t)$ is the minimum of all cut capacities, then (N_s, N_t) is a *minimum cut*. We know by the maximum-flow minimum-cut theorem that the capacity of the minimum cut equals the *maximum s-t flow* in that network; the maximum flow model will be described shortly.

B. NETWORK INTERDICTION MODEL

Here, we define the basic MXFI, which is then converted into MXFI-IP. Lagrangian relaxation for MXFI-IP is presented subsequently.

1. Maximum Flow Network Interdiction Problem (MXFI)

In MXFI, an interdictor, using limited interdiction resources, destroys arcs of the enemy's network $G= (N, A)$ and makes them useless. The objective is to minimize the maximum amount of flow that the enemy, i.e., the network user, can transport through the network. We define MXFI as a function of total interdiction resource R:

MXFI(R):

Indices:

$i, j \in N$	nodes of network $G=(N,A)$
$s \in N$	source node
$t \in N$	sink node
$k = (i, j) \in A$	network arcs
$a = (t, s) \in A$	artificial return arc included in A

Data:

u_k	nominal, uninterdicted capacity of arc k
r_k	amount of resource necessary to interdict arc k ($r_a = \infty$)
R	total amount of available resource

Decision Variables:

y_k	amount of flow on arc k (network user)
x_k	1 if interdiction of arc k is attempted (interdictor)
	0 otherwise

Formulation:

$$z(R) = \min_{\mathbf{x} \in X} \max_{\mathbf{y}} y_a$$

$$\text{s.t.} \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = 0 \qquad \forall i \in N \tag{1}$$

$$0 \le y_k \le u_k (1 - x_k) \qquad \forall k \in A \tag{2}$$

$$\text{where } X = \left\{ \mathbf{x} \in \{0,1\}^{|A|} \,\middle|\, \sum_{k \in A} r_k x_k \le R,\ x_a \equiv 0 \right\}. \tag{3}$$

In this model, constraints (2) force capacities down to zero for all interdicted arcs. For fixed **x**, the inner maximizing problem is a standard maximum-flow model, which maximizes the flow from a source node to a sink node subject to flow-balance constraints (1) and arc capacities (2). MXFI(R) can also be reformulated as MXFIR(R) (Cormican et al. 1998):

MXFIR(R):

$$z(R) = \min_{\mathbf{x} \in X} \max_{\mathbf{y}} y_a - \sum_{k \in A} x_k y_k$$

$$\text{s.t.} \quad \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = 0 \qquad \forall i \in N$$

$$0 \le y_k \le u_k \qquad \forall k \in A$$

$$\text{where } X = \left\{ \mathbf{x} \in \{0,1\}^{|A|} \,\middle|\, \sum_{k \in A} r_k x_k \le R,\ x_a \equiv 0 \right\}.$$

2. An Integer Program for MXFI (MXFI-IP)

Taking the dual of the inner maximization of MXFIR gives us a linear minimization model (which is derived by Wood 1993 directly from MXFI):

MXFI-IP(*R*)

Indices: As in MXFI

Data: As in MXFI

Decision Variables:

α_i	1 if $i \in N_t$ and 0 if $i \in N_s$, for a model-identified cut (N_s,N_t)
β_k	if k is a forward arc of cut (N_s,N_t) and not interdicted, 0 otherwise
x_k	1 if interdiction of arc k is interdicted, 0 otherwise (interdictor)

Formulation:

$$z(R) = \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{x}} \sum_{k \in A} u_k \beta_k$$

$$\text{s.t.} \quad \alpha_i - \alpha_j + x_k + \beta_k \geq 0 \qquad \forall k = (i,j) \in A - \{a\} \tag{4}$$

$$\alpha_t - \alpha_s + x_a + \beta_a \geq 1 \qquad \text{for } a = (t, s) \tag{5}$$

$$\sum_{k \in A} r_k x_k \leq R \tag{6}$$

$$\alpha_i \in \{0,1\} \quad \forall i \in N$$

$$x_k , \beta_k \in \{0,1\} \quad \forall k \in A$$

$$\alpha_s \equiv 0, \ \alpha_t \equiv 1$$

$$\beta_a \equiv x_a \equiv 0$$

3. Lagrangian Relaxation for MXFI (MXFI-LR)

Lagrangian relaxation is a general solution strategy for solving mathematical programs that permits us to decompose problems to exploit their structure. This solution approach leads to bounds on the optimal objective function value and, frequently, to good, though not necessarily optimal solutions (Ahuja, Magnanti and Orlin 1993, p. 601).

MXFI-IP can be difficult to solve because of constraint (6), so we transfer this constraint into the objection function following Derbes (1997). After this modification, the new model, MXFI-LR, can be easily solved, although it may not always yield an optimal solution. The model is:

MXFI-LR (λ , R):

$$z(\lambda, R) = \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{x}} \sum_{k \in A} u_k \beta_k + \lambda(\sum_{k \in A} r_k x_k - R) \tag{7}$$

$$\text{s.t.} \quad \alpha_i - \alpha_j + x_k + \beta_k \geq 0 \qquad \forall k = (i,j) \in A - \{a\}$$

$$\alpha_t - \alpha_s + x_a + \beta_a \geq 1 \qquad \text{for } a = (t, s)$$

$$\alpha_i \in \{0,1\} \quad \forall i \in N$$

$$x_k \,,\ \beta_k \in \{0,1\} \quad \forall k \in A$$

$$\alpha_s \equiv 0,\ \alpha_t \equiv 1$$

$$\beta_a \equiv x_a \equiv 0$$

The term $\lambda(\sum_{k \in A} r_k x_k - R)$ represents a penalty for exceeding or a reward for not consuming the resource budget R. The parameter λ can be interpreted as the dual cost of interdicting an arc.

The linear-programming relaxation of MXFIR has integer extreme points, so we can take the dual of that relaxation to obtain MXFI-LRD after some simplifications (Derbes 1997):

MXFI-LRD (λ , R):

$$z(\lambda, R) = \max_{\mathbf{y}} y_a - \lambda R$$

$$\text{s.t.} \quad \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = 0 \qquad \forall i \in N$$

$$0 \le y_k \le \min\{u_k, \lambda r_k\} \qquad \forall k \in A$$

Of course, $z(\lambda,R)$ is only a lower bound on $z(R)$, but we can use this lower bound, and interdiction plans obtained during its solution to solve MXFI(R), at least approximately. Procedures to do this are presented in the next chapter.

III. NETWORK INTERDICTION BY LAGRANGIAN RELAXATION AND BRANCH-AND-BOUND

This chapter explains our algorithm for solving MXFI. The algorithm uses MXFI-LRD as Bingol (2001) does, but adds two new procedures to decrease optimality gaps. The first procedure finds optimal Lagrangian multipliers, which Bingol's approach does not do. The second procedure implements a restricted version of branch-and-bound. "LRD" and "MXFI-LRD" are used interchangeably below.

A. PRELIMINARIES

Derbes (1997) shows that Lagrangian relaxation can fail to solve MXFI(R) when capacities are not unique, but may solve the problem correctly if the capacities are perturbed slightly so as to be unique. We use the procedure suggested by Bingol (2001) to guarantee unique capacities:

Procedure Perturb(**u**):

{

Order u_k so that $u_1 \leq u_2 \leq \cdots \leq u_{|A|}$;

For (k and m such that $u_{k-1} < u_k = u_{k+1} = \cdots = u_{k+m} < u_{k+m+1}$){

$u_k \leftarrow 10^4 u_k$;

$u_{k+1} \leftarrow 10^4 u_k + 1$;

.

.

.

$u_{k+m} \leftarrow 10^4 u_k + m - 1$;

}

Return **u**;

}

So, suppose that **u** = (1, 1, 3, 3, 3). After running this procedure, the network has unique arc capacities, **u** = (10000, 10001, 30000, 30001, 30002). These small perturbations are unlikely to change the optimal solutions to MXFI. (A scaling factor larger than 10^4 would have to be used if the network were very large and there were many arcs with

identical original capacities.) The use of this procedure is assumed and not shown explicitly below.

B. THE LAGRANGIAN-RELAXATION MODEL

The function $z(\lambda, R)$, from MXFI-LRD(λ, R), is a piecewise-linear concave function in λ for fixed R. Since LRD(λ, R) relaxes MXFI-IP(R), $z(\lambda, R) \le z(R)$ for all λ, where $z(R)$ is the optimal solution for MXFI-IP(R). Naturally, we wish to maximize $z(\lambda, R)$ over λ.

Given a fixed λ, we can solve, or attempt to solve, MXFI(R) and obtain the interdiction set $A_I(\lambda)$, for an unspecified R, through LRD(λ, R) as follows (Bingol 2001):

1. Modify $G = (N, A)$ such that arc capacities become $u'_k = \min\{u_k, \lambda r_k\} \ \ \forall k \in A$, and use a max-flow algorithm to find a minimum-capacity cut A_C,
2. Define the interdiction set $A_I(\lambda) = \{k \in A_C \mid u_k \ge \lambda r_k\}$.

The solution found through LRD(λ, R) identifies a minimum cut (N_s, N_t) that corresponds to a feasible or infeasible solution $\hat{\mathbf{x}}$ to MXFI(R), where $\hat{x}_k = 1 \ \ \forall k \in A_I(\lambda)$, and $\hat{x}_k = 0$ otherwise. If $\hat{\mathbf{x}}$ satisifies the budget constraint $\sum_{k \in A} r_k \hat{x}_k \le R$, then it is a feasible solution. By solving LRD(λ, R) for several values of λ, we may be able to find $\hat{\mathbf{x}}$, with $\sum_{k \in A} r_k \hat{x}_k = R$. (Intuitively, this seems more likely to occur at $\lambda = u_k$ which is why Bingol only evaluates such values of λ.) In this case, MXFI(R) is solved exactly. If not, we will establish a non-zero optimality gap.

To determine optimality gaps, we need upper and lower bounds on $z(R)$. For a feasible interdiction set $A_I(\lambda)$, the corresponding maximum flow is an upper bound (*UB*); $z(\lambda, R)$ from LRD is a lower bound (*LB*) for $z(R)$ for any λ; the best LB from the Lagrangian relaxation is found by solving $\max_{\lambda} z(\lambda, R)$.

Derbes (1997) uses a binary search on the interval of uncertainty for λ to maximize $z(\lambda, R)$. His algorithm, when specialized for our version of MXFI(R), is:

Algorithm D(R):

Input: $G=(N, A)$ directed network with source $s \in N$ and sink $t \in N$; arcs indexed by k,

u_k unique, scaled-up integer arc capacities $u_k > 0$, $\forall k \in A$,

r_k $r_k = 1$ for all interdictable arcs $k \in A$; $r_k = \infty$ otherwise,

R total interdiction resource available.

Output: LB lower bound for the network interdiction problem given R;

{

Initialize: $LB \leftarrow -\infty$; $\lambda_{\max} \leftarrow u_{\max} + 1$; $\lambda_{\min} \leftarrow u_{\min} - 1$; $\lambda \leftarrow \lambda_{\max}$;

while ($\lambda_{\max} - \lambda_{\min} \geq 1$) {

$u'_k \leftarrow \min\{u_k, \lambda r_k\}$ for all $k \in A$; /* adjust arc capacities */

/* maxFlow( ) below returns a min cut A_C and max flow value f */

$(A_C, f) \leftarrow$ maxFlow (G, s, t, $\mathbf{u}'$);

$A_I \leftarrow \{k \in A_C \mid u'_k = \lambda\}$;

$LB \leftarrow \max\{LB, f - \lambda R\}$;

if ($|A_I| = R$) { break; } /* from while loop */

if ($|A_I| < R$) { $\lambda_{\max} \leftarrow \lambda$; }

if ($|A_I| > R$) { $\lambda_{\min} \leftarrow \lambda$; }

$\lambda = (\lambda_{\min} + \lambda_{\max})/2$;

}

print (" LB for R =", R, "is", LB);

}

In Algorithm D(R), the interval of uncertainty initially has left endpoint $\lambda_{\min} = u_{\min} - 1$ and right endpoint $\lambda_{\max} = u_{\max} + 1$. For $\lambda = (\lambda_{\max} + \lambda_{\min})/2$, a max-flow problem is solved in $G = (N, A)$ with $u'_k = \max\{u_k, \lambda r_k\}$ and A_I is determined. The endpoints of the interval are adjusted by determining the sign of the slope of $z(\lambda, R)$ which

is $\text{sgn}\left(\sum_k r_k \hat{x}_k - R\right)$: If $|A_I| < R$, λ becomes the right endpoint, $\lambda_{\max}$, of the interval. If $|A_I| > R$, then λ becomes the left endpoint, $\lambda_{\min}$, of the interval. If $|A_I| = R$, then MXFI(R) has been solved optimally. If an optimal solution to MXFI(R) is not found, this process goes on until $\hat{\lambda}$ is found such that $\lambda_{\min} \le \hat{\lambda} \le \lambda_{\max}$ and $\lambda_{\max} - \lambda_{\min} < 1$. This $\hat{\lambda}$ is close enough to λ^*, because the arc capacities have been scaled up by a large multiplicative factor. (See Section A of this chapter.)

With this procedure we can obtain a solution for only a single value of R. But, we are interested in solving MXFI(R) for all "reasonable" values of R, which means $R \in [0, |A_{\min}|]$, where $A_{\min}$ is a minimum-cardinality cut consisting of only interdictable arcs. (We assume such a cut exists.) In other words, $|A_{\min}|$ is the minimum resource level that guarantees a feasible interdiction plan can force the network flow to zero.

Performing a binary search for each $R \in [0, |A_{\min}|]$ seems inefficient. This is especially true because $z(\lambda, R)$ and $z(\lambda, R')$ are closely related:

$$z(\lambda, R) = z(\lambda, R') + \lambda(R' - R).$$

Furthermore, $z(\lambda, R) = z(\lambda, 0) - \lambda R$ where $z(\lambda, 0)$ is simply the maximum flow $f(\lambda)$ in $G = (N, A)$ with arc capacities $u'_k = \max\{u_k, \lambda r_k\}\ \forall k \in A$.

Bingol (2001) attempts to solve MXFI(R) for all (reasonable) values of R by approximately solving LRD(λ, R). He considers λ in the range $[0, u_{\max}]$ and assumes the slope of $z(\lambda, R) = f(\lambda) - \lambda R$ can change only at $\lambda = u_k$. That is, he assumes that $A_I(\lambda)$ changes only in going from $\lambda = u_k - \varepsilon$ to $\lambda = u_k + \varepsilon$ for certain u_k and some small $\varepsilon > 0$. With this assumption, he can solve many instances of MXFI(R) with acceptable accuracy by only evaluating $z(\lambda, R)$ at such points. Bingol's algorithm is given next. The routine ProblematicRn() initially represents Bingol's procedure ProblematicR1() for handling problematic R-values. That will be replaced subsequently with our routine, ProblematicR2(), to improve results.

Algorithm A: Attempts to solve MXFI(R) for each value of $R \in [0,|A_{\min}|]$ by computing $z(\lambda, R)$ for $\lambda = u_1, u_2, \ldots, u_{|A|}$ and determining a feasible interdiction set $A_I^{(R)}$

Input: $G = (N, A)$ directed network with source $s \in N$ and sink $t \in N$; arcs indexed by k

u_k unique, scaled up integer arc capacities $u_k > 0$, $\forall k \in A$,

r_k $r_k = 1$ for all interdictable arcs $k \in A$; $r_k = \infty$ otherwise.

Output: $\hat{\mathbf{x}}(R)$ optimal or suboptimal solution for MXFI(R),

$A_I^{(R)}$ interdiction set $\forall R \in [0, |A_{\min}|]$ that corresponds to $\hat{\mathbf{x}}(R)$,

$z(R)$ for non-problematic R: optimal objective function value to MXFI(R),

$UB(R)$ for problematic R: feasible objective value,

$LB(R)$ for problematic R: lower bound on optimal objective value,

$absGap(R)$ for problematic R: absolute optimality gap $UB(R) - LB(R)$,

$relGap(R)$ for problematic R: relative optimality gap $100 \times absGap(R)/LB(R)$.

{

/* define and order the candidate set of λ values */

Reorder arcs so that $u_1 < u_2 < \cdots < u_{|A|}$;

Initialize: $R \leftarrow |A|+1$; $R' \leftarrow |A|+1$; $f \leftarrow 0$; $f' \leftarrow 0$; $\lambda' \leftarrow 0$; $l \leftarrow 1$;

while ($l \le |A|$ and $R \neq 0$){

$\lambda \leftarrow u_l$;

$u'_k \leftarrow \min\{u_k, \lambda r_k\}$ for all $k \in A$; /* adjust arc capacities */

$(A_C, f) \leftarrow$ maxFlow (G, s, t, $\mathbf{u}'$);

$A_I \leftarrow \{k \in A_C \mid u'_k = \lambda\}$;

$\hat{x}_k \leftarrow 1 \; \forall k \in A_I$; $\hat{x}_k \leftarrow 0 \; \forall k \in A \setminus A_I$;

$R \leftarrow |A_I|$;

$z \leftarrow f - \lambda R$;

/* ProblematicR1() is called by Bingol; we will call ProblematicR2() */

if ($R' - R > 1$){ ProblematicRn($R', R, f', f, \lambda', \lambda, A'_I$); }

if ($R \neq R'$) {

print (“ The optimal solution to MXFI(R) for R = ”, R, “is ”);

print ($\hat{\mathbf{x}}$, A_I, z) ;

}

/* In the following case, we know that MXFI(R–1) will be solved by evaluating $z(\lambda + \varepsilon, R-1)$ for a small $\varepsilon > 0$ */

if ($u_{k'} = \lambda$ for some $k' \in A_I$){

$A_I \leftarrow A_I - \{k'\}$;

$R \leftarrow R - 1$;

$z \leftarrow z + u_{k'}$;

$\hat{x}_{k'} \leftarrow 0$;

print (“ The optimal solution to MXFI(R) for R = ”, R, “is ”);

print ($\hat{\mathbf{x}}$, A_I, z) ;

}

if ($R' \neq R$) {

$R' \leftarrow R$; $\lambda' \leftarrow \lambda$; $A'_I \leftarrow A_I$; $f' \leftarrow f$; $z' \leftarrow z$;

}

$l \leftarrow l + 1$;

}

}

Algorithm A orders arcs by increasing capacity $u_1 < u_2 < \cdots < u_{|A|}$ and starts with $\lambda = u_1$. Because all arc capacities are λ, $A_I \equiv A_C$, which corresponds to a post-interdiction flow of zero. This effectively defines the solution $\hat{\mathbf{x}}(r_1)$ to MXFI(r_1), where r_1 is the number of arcs in a minimum-cardinality cut consisting of only interdictable arcs. If $u_{k'} = \lambda$ for some $k' \in A_I$, we also find a solution to MXFI$(r_1 - 1)$. If not, the next iteration sets λ to u_2 and attempts to find a solution $\hat{\mathbf{x}}(r_2)$ to MXFI(r_2) such that $r_2 < r_1$. If $r_2 = r_1 - 1$, the algorithm continues without need of ProblematicRn(), and repeats for increasing values of $\lambda = u_k$.

When we compute $R = |A_I|$, we know that we solve $\max_{\lambda} \; z(\lambda, |A_I|) = z(\lambda_R^*, R)$, i.e., we maximize the Lagrangian function $z(\lambda, R)$ for $R = |A_I|$. Furthermore, since $\sum_{k \in A} r_k \hat{x}_k = R$, $z(\lambda_R^*, R) = z(R)$; therefore we have solved MXFI(*R*). However, difficulties arise when the algorithm finds $z(\lambda_R^*, R) = z(R)$ and $z(\lambda_{R'}^*, R') = z(R')$ for $R' > R+1$, but not for $R+1, R+2, ..., R'-1$. We call these "missing" or "problematic *R*-values." For problematic *R*-values, Bingol (2001) derives a heuristic, ProblematicR1() to find a feasible solution and lower bound *LB* on $z(\lambda_R^*, R)$. The feasible solution gives an upper bound *UB* for $z(R)$, so he can compute an absolute, or relative optimality gap which are defined as *absGap=UB−LB* and *relGap= 100* × *absGap/LB*, respectively.

Procedure ProblematicR1 ($R', R, f', f, \lambda', \lambda, A_I'$)

Input:

R	current *R*-value correctly solved,
R'	previous *R*-value correctly solved ($R' > R + 1$),
λ	value of parameter that maximized $z(\lambda,R)$ to solve MXFI(*R*),
λ'	value of parameter that maximized $z(\lambda,R')$ to solve MXFI(R'),
f	maximum flow $f(\lambda)$,
f'	maximum flow $f(\lambda')$,
A_I'	interdiction set for MXFI(R').

Output:

$\hat{\mathbf{x}}(R)$	feasible (possibly optimal) solution for MXFI(*R*),
$A_I^{(R)}$	interdiction set that corresponds to $\hat{\mathbf{x}}(R)$,
$UB(R)$	feasible objective value for $\hat{\mathbf{x}}(R)$,
$LB(R)$	lower bound on optimal objective value $z(R)$,
$absGap(R)$	absolute optimality gap $UB(R) - LB(R)$ for $\hat{\mathbf{x}}(R)$,
$relGap(R)$	relative optimality gap $100 \times absGap(R)/LB(R)$) for $\hat{\mathbf{x}}(R)$.

{

Initialize: $UB \leftarrow f' - \lambda R'$; $A_I \leftarrow A_I'$;

for ($r = R' - 1$ down to $R + 1$){

$k_{\min} \leftarrow \underset{k \in A_I'}{\operatorname{argmin}} u_k$;

$LB \leftarrow \max\{ f - \lambda r, f' - \lambda' r \}$;

$UB \leftarrow UB + u_{k_{\min}}$;

$absGap \leftarrow UB - LB$;

$relGap \leftarrow 100 \times absGap/LB$;
$A_I \leftarrow A_I - k_{\min}$;
$\hat{x}_k \leftarrow 1 \; \forall k \in A_I$; $\hat{x}_k \leftarrow 0 \;\; \forall k \in A \setminus A_I$;
print (" For problematic R = ", r, "the approximate soln. to MXFI(R) is: ");
print ($\hat{\mathbf{x}}$, A_I ,UB, LB, $absGap$, $relGap$);
}
}

Algorithm A requires the solution of a sequence of maximum-flow problems. We use the shortest augmenting path max-flow procedure, maxFlow(), (Edmonds and Karp 1972), which finds the maximum flow f, and a minimum-capacity cut A_C. Since the flow on each arc is non-decreasing, this procedure uses the previous flow level from the last iteration, instead of starting from scratch. This is important for fast run-times.

C. FINDING AN OPTIMAL λ FOR PROBLEMATIC R

The procedure ProblematicR1() above does not find the best LB, i.e., it does not maximize $z(\lambda,R)$ for problematic R-values and therefore does not establish the best achievable optimality gap. We can attempt to find the best lower bound by exploring the region $[\lambda_{R'}, \lambda_R]$ by defining a finer grid for λ. But this is computationally expensive and requires solving many max-flow problems. At this point, Proposition 1 expedites the process.

Proposition 1 (Derbes 1997): The Lagrangian function $z(\lambda, R)$ has a slope of $|A_I(\lambda)| - R$, where $|A_I(\lambda)|$ is the number of arcs interdicted using λ.

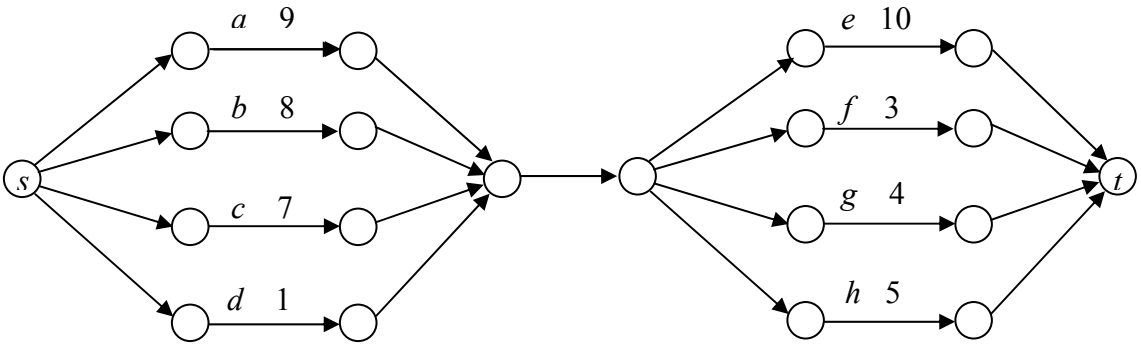

Figure 1. NETSample. Letters and numbers represent arc labels and capacities, respectively. Unlabeled arcs cannot be interdicted. All labeled arcs have $r_k = 1$.

Figure 1 shows a sample network, NETSample, which we will use to explain Proposition 1. We will compute $z(\lambda,1)$ and look at the function in detail:

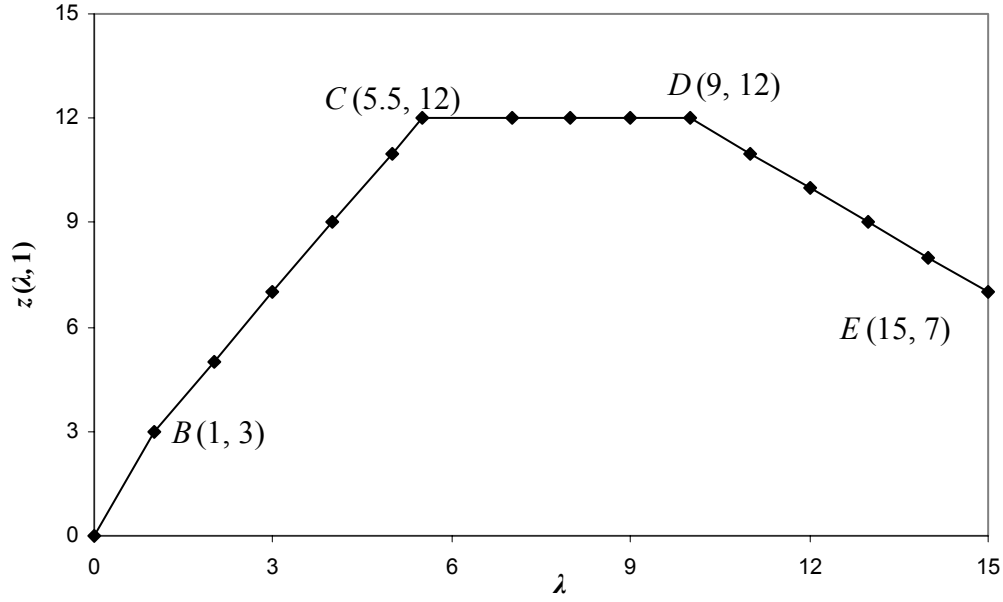

Figure 2. Lagrangian function for $z(\lambda,1)$ for NETSample. The first number in parentheses is the value of parameter λ and the second number represents the value of the function.

If we compute the slope m_{BC} of line BC in Figure 2, we see that $m_{BC} = |A_I(\lambda)| - R = 3 - 1 = 2$. Likewise, $m_{CD} = |A_I(\lambda)| - R = 1 - 1 = 0$, which also gives us the optimal

solution for $R = 1$. (When the slope is zero, $|A_I(\lambda)| = R = 1$, and we obtain an exact solution for MXFI(1).) These results are in accordance with Proposition 1.

For NETSample, if we run Algorithm A, we run into a missing value of *R*. We find a solution for $R = 3$ at $\lambda_3 = 3$ and $R = 1$ at $\lambda_1 = 7$, but not for $R = 2$. The solutions for $R = 3$ and $R = 1$ are $A_I^{(3)} = \{a, b, c\}$ and $A_I^{(1)} = \{e\}$. Bingol's heuristic finds the *UB* and *LB* for $R = 2$, with the following steps:

i. $A_I^{(2)} = A_I^{(3)} - \{k_{min}\} = \{a,b,c\} - \{c\} = \{a,b\}$,

ii. $UB = z(3) + u_{min} = 1 + 7 = 8$,

iii. $LB = \max\{f(\lambda_1) - 2\lambda_1, f(\lambda_3) - 2\lambda_3\} = \max\{19 - 14,\ 10 - 6\} = 5$,

iv. $absGap = 8 - 5 = 3$,

v. $relGap = 100 \times 3 / 5 = 60\%$.

The relative gap, *relGap*, is quite high. To reduce this optimality gap, we exploit Proposition 1 and trace the function $z(\lambda, R)$ more closely.

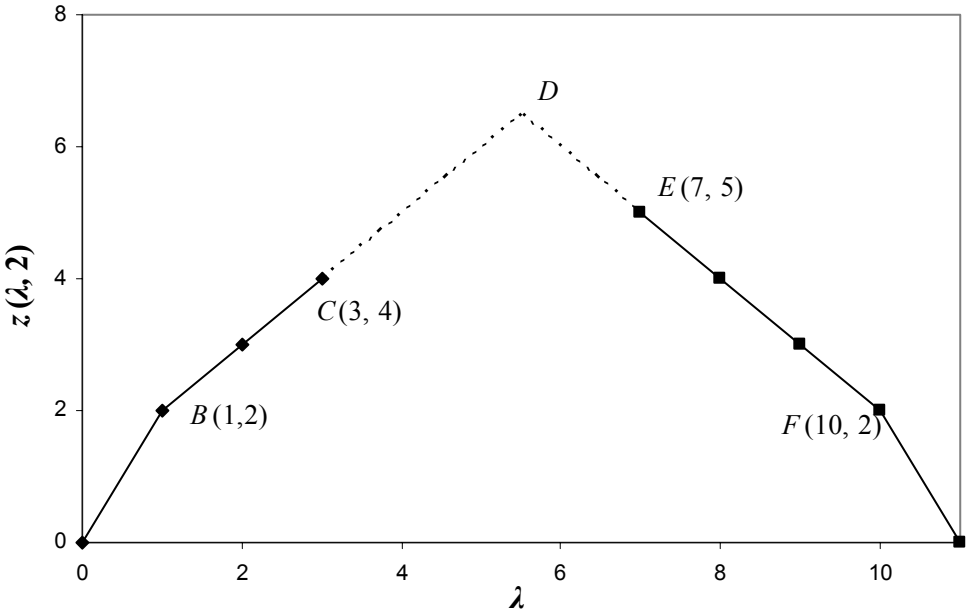

Figure 3. Lagrangian Function of $R = 2$ for NETSample. The solid line denotes that the value of the function is computed for those λ values. The dotted line shows how procedure solveExact() computes the optimal λ.

ProblematicR1() finds $LB = 5$ for $R = 2$, which is the value of the function at point E. But with Proposition 1, we can look for λ_2^* over the region [3, 7] to improve LB and possibly solve MXFI. If $A_I(\lambda_D) = 2$ (it is not in this case), then MXFI(2) is solved exactly. Otherwise, we obtain an improved LB. To accomplish this, we do the following:

i. /* Find the slopes of associated lines */
$m_{CD} = |A_I(3)| - R = 3 - 2 = 1,\ m_{DE} = |A_I(7)| - R = 1 - 2 = -1,$

ii. /* λ_2^* is the optimal parameter value for $R = 2$ */
$z(\lambda_2^*, 2) = 4 + (1)(\lambda_2^* - 3),\ z(\lambda_2^*, 2) = 5 + (-1)(\lambda_2^* - 7),$

iii. /* Compute λ_2^* from the equations above */
$\lambda_2^* = 5.5$ and $z(\lambda_2^*, 2) = 6.5,$

iv. /* Check for optimality */
$A_I(5.5) \neq 2$, /* MXFI(2) is not solved exactly */

v. /* Compute the optimality gap with the new *LB* */
$absGap = UB - z(\lambda_2^*, 2) = 8 - 6.5 = 1.5$, and
$relGap = 100 \times absGap / z(\lambda_2^*, 2) = 100 \times 1.5 / 6.5 = 23.1\%$.

The relative gap has decreased from 60% to 23.1%. The best Lagrangian multiplier is λ_2^*=5.5. Because Algorithm A computes $z(\lambda, R)$ at $\lambda = u_k \ \forall k \in A$, it would never discover this *LB*. This example has only one missing *R*-value. However, there can be contiguous problematic *R*-values. The following procedure will also handle these cases.

Procedure ProblematicR2($R', R, f', f, \lambda', \lambda, A_I'$):

Input: R current *R*-value correctly solved ,
R' previous *R*-value correctly solved ($R' > R + 1$),
λ value of parameter at which R is solved,
λ' value of parameter at which R' is solved,
f maximum flow $f(\lambda)$,
f' maximum flow $f(\lambda')$.

Output: $\hat{\mathbf{x}}(R)$ optimal or suboptimal solution for MXFI(*R*),
$A_I^{(R)}$ interdiction set $\forall\ R \in [0, |A_{min}|]$ that corresponds to $\hat{\mathbf{x}}(R)$,
$z(R)$ for non-problematic *R*: optimal objective function value to MXFI(*R*),
$UB(R)$ for problematic *R*: feasible objective value,
$LB(R)$ for problematic *R*: lower bound on optimal objective value,
$absGap(R)$ for problematic *R*: absolute optimality gap $UB(R) - LB(R)$,
$relGap(R)$ for problematic *R*: relative optimality gap $100 \times absGap(R)/LB(R)$.

{

For ($r = R' - 1$ to $R+1$) {

$point \leftarrow \lambda$; $prevPoint \leftarrow \lambda'$; $R_{prev} \leftarrow R'$; $R_{cur} \leftarrow R'$;

if ($r = R+1$){ $R_{cur} \leftarrow R$;}

while ($point \neq prevPoint$) {

$prevPoint \leftarrow point$;

$point \leftarrow (f - f' - \lambda R + \lambda' R') / (R' - R)$; /* see the explanation below */

$u_k' \leftarrow \min\{u_k, (point)\, r_k\}$ for all $k \in A$;

$(A_C, flow) \leftarrow$ maxFlow(G, s, t, $\mathbf{u}'$);

$A_I \leftarrow \{ k \in A_C \mid u'_k = \lambda \}$;

$\hat{x}_k \leftarrow 1 \; \forall k \in A_I$; $\hat{x}_k \leftarrow 0 \;\; \forall k \in A \setminus A_I$;

/* If the exact solution is found, then it prints the results */

if $(|A_I| = r)$ {

print (" The optimal solution to MXFI(R) for R = ", R, "is ");

print $(\hat{\mathbf{x}}, A_I, z)$;

$prevPoint \leftarrow point$; /*get out of while loop */

$R_{cur} \leftarrow r$;

if ($R_{prev} - R_{cur} = 1$) { $R_{prev} \leftarrow R_{cur}$; }

}

if ($|A_I| > r$) { $\lambda' \leftarrow point$; $R' \leftarrow |A_I|$; $f' \leftarrow flow$; $A'_I \leftarrow A_I$; }

if ($|A_I| < r$) { $\lambda \leftarrow point$; $R \leftarrow |A_I|$; $f \leftarrow flow$;}

}

$\lambda^*_r \leftarrow point$; /* The best Lagrangian multiplier is found */

if $(R_{prev} - R_{cur} > 1)$ {

/* With revised values of λ and λ', we can reuse ProblematicR1() to find a feasible solution and optimality gap for missing values of R */

ProblematicR1($R_{prev}, R_{cur}, f', f, \lambda', \lambda, A_I^{(R_{prev})}$);

/* Uncomment the next statement to invoke Branch() . Notice that $UB(R)$ comes from ProblematicR1() */

/* Branch(R_{prev}, R_{cur}, $\lambda^*_{r+1} - \varepsilon$, $A_I^{(R_{prev})}$, $UB(R)$); */

$R_{prev} \leftarrow R_{cur}$;

}

}

}

In our heuristic above, we derive the formula $point = (f - f' - \lambda R + \lambda' R')/(R' - R)$ in order to find the intersection with the following steps:

1. /* Find $z(point, r)$ using the previous solution's parameters (R', λ', f'). Notice that r is the missing value and slope is $R' - r$ */

   $z(point, r) = (f' - \lambda' r) + (R' - r)(point - \lambda')$,

2. /* Find $z(point, r)$ using the current solution's parameters (R, λ, f). This time, slope is $r - R$ */

   $z(point, r) = (f - \lambda r) + (r - R)(\lambda - point)$,

3. /* Using these two equations, solve for "*point*" */

   $point = (f - f' - \lambda R + \lambda' R') / (R' - R)$.

When Algorithm A runs into one or more problematic *R*-values, ProblematicR2() is called to find the best Lagrangian multiplier for each missing value of *R*. This heuristic seeks λ_R^* over an interval of uncertainty by performing a binary search based on the sign of slope $\sum r_k \hat{x}_k - R$. First, it finds the intersection, *point*, using the formula derived earlier. If $A_I(point) = r$, then MXFI(*r*) is solved exactly. Otherwise, it updates the endpoints as in Derbes's procedure (1997). It keeps updating until it converges to λ_R^* and possibly finds an exact solution. If there are multiple values of $R \in [R_{prev} - 1, R_{cur} + 1]$, ProblematicR1() is invoked to improve the results with the best Lagrangian multipliers for each of these problematic *R*-values. For further improvements, we can a invoke Branch() which applies a restricted branch-and-bound procedure.

If there is only one missing *R*-value, one of two cases occurs, and it only takes one evaluation of $z(\lambda,R)$ to figure this out:

1. If there is only a single breakpoint, then we must compute it exactly when we compute the intersection of the two lines extending upwards from the coordinates $(\lambda_{R'}, z(\lambda_{R'}, R'))$ and $(\lambda_R, z(\lambda_R, R))$ in (λ, z)-space. Let that intersection point be (λ'', z''). The single breakpoint is verified by computing $z(\lambda'', R)$ and finding $z'' = z(\lambda'', R)$.

2. If there are two breakpoints in the interval $(\lambda_{R'}, \lambda_R)$, we compute (λ'', z''), evaluate $z(\lambda'', R)$ and find that $z(\lambda'', R) < z''$. Consequently, the slope of the $z(\lambda, R)$ curve

must lie strictly between one and negative one. Since the slope can only change by integer amounts, we have found a portion of the curve where the slope is zero and will, consequently, have solved MXFI(*R*) optimally.

If there are sequential problematic *R*-values, there are three possibilities:

1. There may not be an exact solution for any of the problematic *R*-values. In this case, the procedure finds λ_R^* and then ProblematicR1() is called with updated parameters to find the best *LB*. In fact, λ and λ' computed earlier by Algorithm A are at some u_k. However, when computed by ProblematicR2(), either λ or λ' becomes the best Lagrangian multiplier, which helps improve *LB* for these problematic *R*-values. To invoke Branch(), we use the following idea: When there are more than one problematic *R*-values at one breakpoint, the best Lagrangian multiplier is the same for all these values. In other words, for $R \in [R'-1, R+1]$, $\lambda_{R+1}^* = \lambda_{R+2}^* = \cdots = \lambda_{R-1}^*$. Therefore, we use only one of them with a slight perturbation, $\lambda_{r+1}^* - \varepsilon$, to invoke Branch(), where $|A_I(\lambda_{r+1}^* - \varepsilon)| = R_{prev} = R'$.

2. The interval of uncertainty might yield an exact solution for some of the problematic *R*-values. If it does, then the procedure prints out the results of those values, and R_{prev} and R_{cur} establish new endpoints for the problematic *R*-values. If $R_{prev} - R_{cur} > 1$, ProblematicR1() and/or Branch() can be invoked to find a feasible solution and optimality gap for problematic $R \in [R_{cur}+1, R_{prev}-1]$ as explained in the previous paragraph. Since $A_I^{(R_{prev})}$ and/or $A_I^{(R_{cur})}$ might change for the problematic $R \in [R_{cur}+1, R_{prev}-1]$, not only *LB* but also *UB* are computed by ProblematicR1().

3. The procedure might solve MXFI(*R*) exactly for all missing *R*-values in the sequence. In our tests in Chapter IV, we observe this in two instances out of 10 test problems.

D. RESTRICTED BRANCH-AND-BOUND

Using procedure ProblematicR2() to solve MXFI, we may be able to reduce the optimality gap by a substantial amount or solve MXFI exactly, for certain values of *R*.

However, there might be some problematic *R*-values for which the optimality gap is still large. For these *R*-values, we can resort to branch-and-bound.

The usual idea of the branch-and-bound in the setting of integer programming is the following: Assume that we have a feasible, binary integer-programming problem (BIP). We can solve the BIP as a linear program (LP) (i.e., the linear-programming relaxation of BIP) by ignoring the integer restrictions. If the LP optimal solution contains only binary-valued variables we have solved the BIP. Otherwise, we select a fractional variable, say x_k, and branch on it. That is, we create two new LPs, LP_1 and LP_2, by introducing the constraints $x_k = 0$ and $x_k = 1$, respectively. An optimal solution to the original problem is contained in the feasible region to at least one of these two problems. Next, we select one of the problems, say LP_1, solve it (the LP relaxation that is), and evaluate its objective. If the objective is no better than the value of some incumbent integer solution (which we obtained heuristically, say), then that problem is *fathomed by bounds.* If the problem is infeasible, it is *fathomed by infeasibility*. If the solution is integer then we have *fathomed by integrality* (and we save this solution as the incumbent if it is better than any we have seen before). If none of those cases occur for LP_1, we select a fractional variable and recursively branch on it. Branch and bound stops when all nodes are fathomed: The best solution found is optimal. (Note that some subproblems that have not been solved may be fathomed using information from other subproblem solutions. For instance, if an optimal integer solution to LP_1 is found and its objective value equals that of LP, then LP_2 is implicitly fathomed by bounding.)

But in our scheme for solving MXFI, there are no fractional values. Thus, we need a different partitioning scheme. We will base the partition on, essentially, setting $x_k = 0$ or $x_k = 1$ for $k \in A_I(\lambda')$, where $A_I(\lambda') > R$. In this case, "forcing arc *k* into the solution" is equivalent to setting $x_k = 1$ and "forcing arc *k* out of the solution" corresponds to setting $x_k = 0$. And, we will consider a node fathomed if, at that node:

1. $|A_I| = R$ (equivalent to being fathomed by integrality; an optimal solution to the subproblem has been found), and

2. $z(\lambda, R) > UB$. (Fathomed by bounding.)

If we use this technique in our problem, our enumeration tree might look like the following, given $A_I(\lambda') = \{m, n, r\}$:

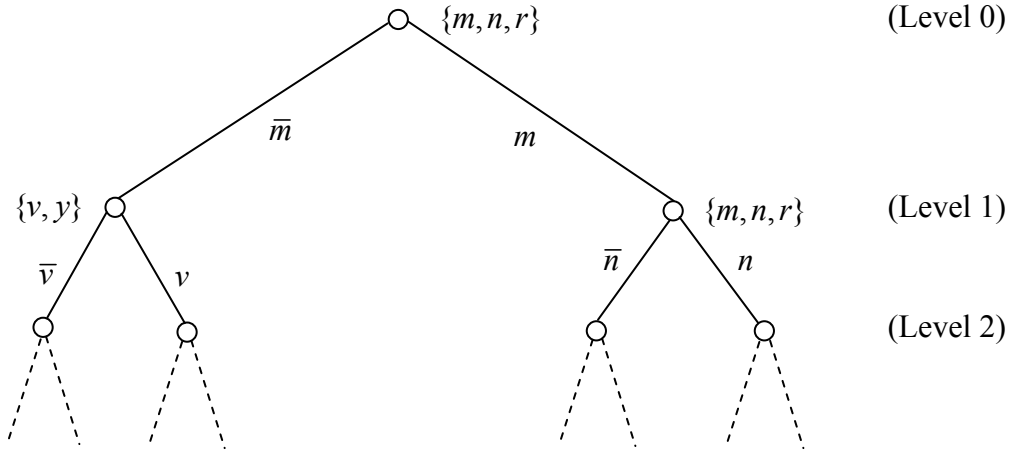

Figure 4. Sample Enumeration Tree for $R = 1$. Solutions are given next to nodes in curly braces and the branching decision is next to the branches: " The initial solution at the root node at level 0 is $\{m, n, r\}$. "$\bar{m}$" and "m" represent the exclusion and inclusion of arc m, respectively. Note that none of the nodes are fathomed in this sample enumeration tree.

In Figure 4, every time we force an arc in, we obtain the same solution. For example, when arc m is included for the solution at Level 1, we get the solution $\{m, n, r\}$ from Level 0. But, it is unnecessary to consider a branch with $\{m, n, r\}$, because we have already evaluated this solution. So, we modify the procedure as follows:

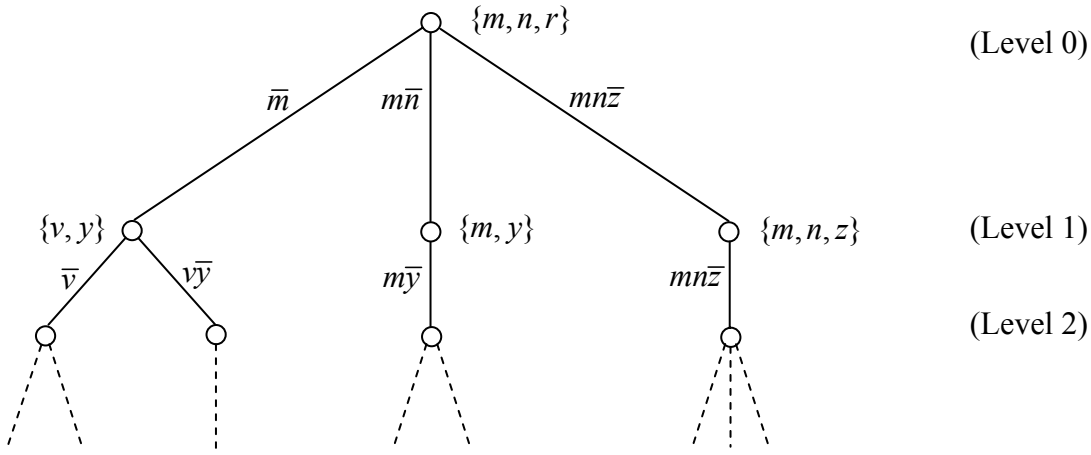

Figure 5. Modified Enumeration Tree for $R = 1$. Solutions are represented next to nodes in {}. $m\bar{n}$ represents the inclusion of arc *m* and exclusion of arc *n*. A full partition of the space of all possible solutions includes {*m, n,* r}, but this branch need not be created, because this solution has already been evaluated at the root level of enumeration tree.

In this modified enumeration tree, we begin with the exclusion of the first arc *m* in the initial interdiction set and we obtain the solution $\{v, y\}$ as a result. Since $|A_I| \neq 1$ for this solution, two new subproblems are introduced by adding the condition $\bar{v}$ or $v\bar{y}$. This process continues for all nodes until each subproblem node is fathomed. When all the nodes are fathomed, the best feasible solution encountered is declared optimal.

Because of limited time for this study, we restrict the enumeration tree to a depth of one (Level 1). So, for $A_I = \{k_1, k_2, \ldots\}$, *LB* is computed as:

$$LB = \min\{\max_{\lambda} z(\lambda, R \mid x_{k_1} = 0), \max_{\lambda} z(\lambda, R \mid x_{k_1} = 1, x_{k_2} = 0), \ldots\}.$$

If the *LB* comes from a node at which $|A_I| = R$, A_I is the optimal solution to MXFI(*R*). This is true because all the other nodes are fathomed by bounding.

We find feasible solutions for each branch created as follows: If $|A_I| > R$, we "uninterdict" arcs $k \in A_I$, in order of increasing capacity, until $|A_I| = R$. If $|A_I| < R$, we interdict arcs $k \in \{A_C \setminus A_I\}$, in order of decreasing capacity, until $|A_I| = R$. The feasible solution with the lowest *UB* becomes the best feasible solution.

If there is more than one missing value of *R*, we apply this procedure sequentially as follows (A^+ denotes the set of arcs to be included):

Procedure Branch(R' , R, λ , $A_I^{(R')}$, $UB(r)$)

Input: Network G= (N, A), source node s, sink node t,

R	current R-value correctly solved,
R'	previous R-value correctly solved ($R' > R + 1$),
λ	value of parameter such as $\lambda = \lambda^*_{R+1} - \varepsilon$,
$A_I^{(R')}$	interdiction set that corresponds to $\hat{\mathbf{x}}(R')$,
$UB(R)$	feasible objective value for $\hat{\mathbf{x}}(R)$.

Output:

$\hat{\mathbf{x}}(R)$	feasible (possibly optimal) solution for MXFI(R),
$A_I^{(R)}$	interdiction set that corresponds to $\hat{\mathbf{x}}(R)$,
$LB(R)$	possibly improved lower bound on optimal objective value $z(R)$,
$UB(R)$	possibly improved feasible objective value for $\hat{\mathbf{x}}(R)$,
$absGap(R)$	for problematic R: absolute optimality gap $UB(R) - LB(R)$,
$relGap(R)$	for problematic R: relative optimality gap $100 \times absGap(R)/LB(R)$.

{

for ($r = R' - 1$ to $R + 1$) { /* Solve for all missing values of R */

$u'_k \leftarrow \min\{u_k, \lambda_{R'} r_k\}$ for all $k \in A$; /* adjust arc capacities */

$LB(r) \leftarrow \infty$;

$A^+ \leftarrow \emptyset$; /* This will be the set of arcs forced into the solution */

for (a = 1 to $r + 1$) { /* Include and exclude the arcs in $A_I(r)$ */

Let k be the a^{th} arc in $A_I(r)$;

If ($k \notin A^+$) { /* if k is not already forced in */

$u'_k \leftarrow u_k$; /* forces arc k out of the cut, i.e., sets $\hat{x}_k = 0$ */

}

$(A_C, f) \leftarrow$ maxFlow (G, s, t, $\mathbf{u}'$);

$\hat{A}_I \leftarrow \{k \in A_C \mid u'_k = \lambda\}$;

$A_I \leftarrow \hat{A}_I \cup A^+$;

$\hat{x}_k \leftarrow 1 \; \forall k \in A_I$; $\hat{x}_k \leftarrow 0 \;\; \forall k \in A \setminus A_I$;

$UB_{local} \leftarrow f - |\hat{A}_I| \lambda_{R'}$;

/* Find a feasible solution */

if ($|A_I| > r$) {

for ($|A_I|$ down to r) {

$k_{\min} \leftarrow \arg\min_{k \in A_I} u_k$;

$UB_{local} \leftarrow UB_{local} + u_{k_{\min}}$;

$A_I \leftarrow A_I - \{k_{\min}\}$;

$\hat{x}_k \leftarrow 0$;

}

}

/* Find a feasible solution */

if ($|A_I| < r$) {

for (r down to $|A_I|$) {

$k_{\max} \leftarrow \arg\max_{k \in \{A_C / A_I\}} u_k$;

$UB_{local} \leftarrow UB_{local} - u_{k_{\max}}$;

$A_I \leftarrow A_I + \{k_{\max}\}$;

$\hat{x}_k \leftarrow 1$;

}

}

$LB_{local} = f - \lambda_{R'}(r - a + 1)$;

$LB(r) = \min\{LB(r), LB_{local}\}$

$UB(r) = \min\{UB(r), UB_{local}\}$

$u'_k = 0$; /* Set $x_k = 1$ in the following subproblems */

$A^+ \leftarrow A^+ \cup \{k\}$;

}

$absGap(r) \leftarrow UB(r) - LB(r)$;

$relGap\ (r) \leftarrow 100 \times absGap(r)/LB(r)$;

print (" For problematic R = ", r, "the approximate soln. to MXFI(R) is: ");

print ($\hat{\mathbf{x}}$, A_I , $UB(r)$, $LB(r)$, $absGap(r)$, $relGap(r)$);

}

}

In this procedure, we force an arc out of a solution by taking its capacity back to its original level, $u_k > \lambda_{R'}$. This ensures that it will never be seen as interdicted since $\hat{A}_I \equiv \{ k \in A_C \mid u'_k = \lambda_{R'} \}$. "Forcing in" is accomplished by setting $u'_k = 0$, which means that arc *k* is destroyed and cannot be used.

Although it is not shown in the pseudo-code for simplicity, if $|A_I| \neq r$, we keep solving max-flow problems for $\lambda \in u_k$ to find $|A_I(\lambda)| = r$. If it doesn't exist, then we find $\lambda_{\min}$ and $\lambda_{\max}$ such that $|A_I(\lambda_{\min})| < r < |A_I(\lambda_{\max})|$. After finding these values, we compute LB_{local} the way it is computed in ProblematicR1( ):

$$LB_{local} = \max\{(f(\lambda_{\max}) - \lambda_{\max}(r-a+1)), (f(\lambda_{\min}) - \lambda_{\min}(r-a+1))\}.$$

If we find $LB = UB$, then MXFI is solved exactly for that value of *R*. Otherwise, we may obtain an improved *LB* and/or *UB*, which decreases the optimality gap.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. COMPUTATIONAL RESULTS

In this chapter, we describe the design of test problems, and provide computational results for our Lagrangian-based procedures to solve MXFI.

A. TEST NETWORK DESIGN

We use grid networks to test our procedures. "NET $a \times b$" denotes such a network, where *a* is the number of nodes on the horizontal axis and *b* is the number of nodes on the vertical axis. The number of nodes in a grid network is $|N| = ab + 2$, and the number of arcs is $|A| = a(5b-4) - 5b + 6$. A representation of a small network is given in Figure 6:

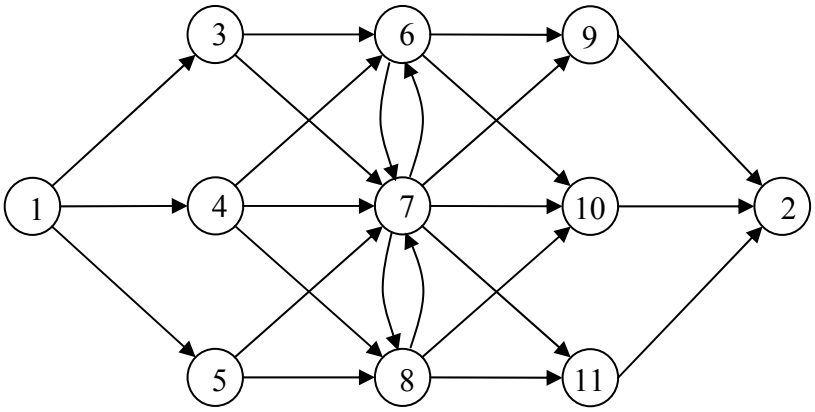

Figure 6. NET 3×3. Arc capacities u_k and resources r_k are not shown in this network. The source node and sink node are numbered 1 and 2, respectively.

As shown in Figure 6, all horizontal arcs lie in the direction of "west to east," and diagonal arcs are oriented in the southeast and northeast directions. There are also vertical arcs, north-pointing and south-pointing, between adjacent nodes in a column of nodes; although these are omitted in the first and last columns, where they would be superfluous for maximizing flow. Arcs going out of the source node and going into the sink node have infinite capacity and cannot be interdicted. (In fact, uninterdictable arcs have $r_k = 999$,

which effectively makes them uninterdictable.) All the other arcs are interdictable. Finite arc capacities u_k are drawn from a discrete uniform distribution with range [1,50].

We use the network generator written in C by Derbes (1997) and modify it for different networks. The network files generated by this program provide the following information about the structure of the network: number of nodes, number of arcs, source and sink nodes. Also, each arc *k* receives a capacity u_k, cost of repair c_k, resource needed to interdict r_k, and "reparability" p_k. However, we use only u_k and r_k in our tests.

B. RESULTS

We use six different grid networks for testing. The smallest one, NET 5×5, has 27 nodes and 86 arcs. The largest one, NET 20×20, has 402 nodes and 1826 arcs. All tests are performed on a 533 MHz Pentium III personal computer with 192 MB of RAM and running the Microsoft Windows XP Professional operating system. All programs are written in the Java 1.3 programming language.

We use three different methods to test the effectiveness of our Lagrangian-based procedures. Each method uses Algorithm A with a different procedure or procedures. Procedure ProblematicR1() is the heuristic that Bingol (2001) uses to find a feasible solution, and a lower bound on $z(\lambda_R^*, R)$. We use ProblematicR2() to find the best Lagrangian multipliers. The procedure Branch() applies the restricted branch-and-bound technique introduced in Chapter III. Method 1 uses ProblematicR1() integrated into Algorithm A; this is the technique Bingol (2001) analyzes. In particular, every time Algorithm A runs into a contiguous set of problematic *R*-values, ProblematicR1() is called to find *UB* and *LB* for each. Method 2 works the same way, but ProblematicR2() is invoked to find λ_R^* for problematic *R*-values before parameters are passed on to ProblematicR1() which is, in this case, only used to find feasible solution. Furthermore, the indicated call to Branch() inside of ProblematicR2() is not made. Method 3 invokes procedure ProblematicR2() exactly as stated in the text, i.e., with a call to Branch() if a positive gap remains.

Since we are really only concerned with problematic R-values, the tables only present: the problematic values R, total run-times, upper bounds $\bar{z}(R)$, lower bounds $z(\lambda, R)$, absolute gaps "absGap," and relative gaps "relGap." Tables also include the uninterdicted maximum flow value f^*.

Table 1 shows results for Method 1. Although all absolute gaps are small, relative gaps can be as large as 25.0% when the divisor $z(\lambda, R)$ is small. Each network that we have tested has yielded one or more problematic values. Run-times are small, the largest one being 31.5 cpu seconds.

Table 2 exhibits results for Method 2 using the same networks. Since the best Lagrangian mulitiplier is not necessarily an integer, $z(\lambda_R^*, R)$ might have fractional values. "absGap" and "relGap" are the results computed with these fractional values. However, optimal objective values must be integral, so it makes sense to round $z(\lambda_R^*, R)$ up to the nearest integer. Thus, "$\lceil$absGap$\rceil$" and "$\lceil$relGap$\rceil$" denote absolute and relative gaps computed after this rounding. Of 36 problematic R-values found with Method 1, Method 2 solves 16 exactly. The largest remaining relative gap is 14.3% for $R = 51$ in NET 20×20. Run-times increase only slightly for the networks that have a few problematic R-values. If ProblematicR2() is invoked many times, the increase is about 25%.

Table 3 presents the results of Method 3 which invokes procedure Branch(). We only pass on to this procedure the problematic R-values remaining after Method 2. With this method, the number of remaining problematic R-values values goes down to four for NET 20×20. The largest relative gap drops to 7.1%, since a better feasible solution is found by Branch(). There can be, unfortunately, a drastic increase in run-time; for example, the run time increases from 31.5 seconds to 4013.7 seconds for NET 20×20.

To explore the effectiveness of these methods further, we use NET 8×15 with restricted capacities. There are four types of restricted capacities, denoted 16[1,2], 8[1,4], 4[1,8], 2[1,16]: The notation "$p[1,q]$" indicates that the random capacities are drawn uniformly from the set $\{p, 2p, \ldots, qp\}$. Bingol (2001) finds problem instances like these, with few distinct capacities, to be particularly difficult. That is, they tend to yield large

optimality gaps. We apply the same three methods as before: Table 4 includes only problematic *R*-values with relative gaps higher than 5% remaining after Method 1.

We start with a total of 16 problematic values from Method 1. The largest relative gap from Method 1 is 20%. By using Method 2, the number of problematic *R*-values goes down to eight, with the largest gap being 6.1%. Four of these problematic values are solved exactly by Method 3. However, there is no improvement in the relative gaps of the remaining four.

$G = (N, A)$	f^*	R	Run Time (cpu sec.)	Problematic R-Values	$\bar{z}(R)$	$z(\lambda, R)$	absGap_1	relGap_1
NET 5× 5	234	[1,13]	0.1	6	55	53	2	3.8%
				2	159	152	7	4.6%
NET 8× 8	416	[1,22]	0.4	17	18	17	1	5.9%
				16	29	27	2	7.4%
				15	39	37	2	5.4%
				14	48	47	1	2.1%
				6	202	200	2	1.0%
				5	228	227	1	0.4%
NET 8× 12	690	[1,34]	1.0	15	189	188	1	0.5%
NET 8× 15	841	[1,43]	1.9	40	5	4	1	25.0%
				39	7	6	1	16.7%
				34	27	26	1	3.8%
				24	146	145	1	0.7%
				23	165	162	3	1.9%
				22	181	180	1	0.6%
				17	279	278	1	0.4%
				6	590	588	2	0.3%
NET 15× 15	878	[1,43]	7.5	38	11	10	1	10.0%
				10	431	430	1	0.2%
				9	463	462	1	0.2%
				3	686	684	2	0.3%
NET 20× 20	1173	[1,58]	31.5	53	8	7	1	14.3%
				51	16	14	2	14.3%
				50	19	18	1	5.6%
				41	83	80	3	3.8%
				40	93	89	4	4.5%
				39	101	98	3	3.1%
				38	110	108	2	1.9%
				37	119	118	1	0.8%
				32	175	174	1	0.6%
				17	508	507	1	0.2%
				16	542	541	1	0.2%
				15	576	575	1	0.2%
				14	611	609	2	0.3%
				11	712	711	1	0.1%
				7	859	858	1	0.1%

Table 1. Test Results for Method 1. This is essentially Bingol's procedure (Bingol 2001) which uses Algorithm A with ProblematicR1(), but not ProblematicR2() or Branch(). f^* shows maximum flow without interdiction. The column labeled "*R*" gives the range of interdiction resource levels tested; he maximum flow goes to 0 when *R* equals the second of the two numbers. Run times cover the whole procedure, but only problematic *R*-values remaining after the procedure is run are shown. $\bar{z}(R)$ is the upper bound obtained from the best solution with ProblematicR1() and $z(\lambda, R)$ is the largest value of Lagrangian function obtained from MXFI-LRD. $z(\lambda, R)$ is always integer since $\lambda = u_k$ for $k \in A$. "absGap_1" is $\bar{z}(R) - z(\lambda, R)$ and "relGap_1" is $100 \times \text{absGap}_1 / z(\lambda, R)$.

$G = (N, A)$	f^*	R	Run Time (cpu sec.)	Problematic R-Values	$\bar{z}(R)$	$z(\lambda_R^*, R)$	absGap	relGap	$\lceil z(\lambda_R^*, R) \rceil$	absGap_2	relGap_2	relGap_1
NET 5× 5	234	[1,13]	0.1	6	55	53.5	1.5	2.8%	54	1	1.9%	3.8%
				2	159	152.5	6.5	4.3%	153	6	3.9%	4.6%
NET 8× 8	416	[1,22]	0.7	17	18	17.2	0.8	4.7%	18	-	-	5.9%
				16	29	27.4	1.6	5.8%	28	1	3.6%	7.4%
				15	39	37.6	1.4	3.7%	38	1	2.6%	5.4%
				14	48	47.8	0.2	0.4%	48	-	-	2.1%
				6	202	200	2.0	1.0%	200	2	1.0%	1.0%
				5	228	227	1.0	0.4%	227	1	0.4%	0.4%
NET 8×12	690	[1,34]	1.0	15	189	188.5	0.5	0.3%	189	-	-	0.5%
NET 8 ×15	841	[1,43]	2.3	40	5	4.3	0.7	16.3%	5	-	-	25.0%
				39	7	6.7	0.3	4.5%	7	-	-	16.7%
				34	27	26.5	0.5	1.9%	27	-	-	3.8%
				24	146	145.5	0.5	0.3%	146	-	-	0.7%
				23	165	163	2.0	1.2%	163	2	1.2%	1.9%
				22	181	180.5	0.5	0.3%	181	-	-	0.6%
				17	279	278.5	0.5	0.2%	279	-	-	0.4%
				6	590	588.5	1.5	0.3%	589	1	0.2%	0.3%
NET 15×15	878	[1,43]	10.3	38	11	10.5	0.5	4.8%	11	-	-	10.0%
				10	431	430.3	0.7	0.2%	431	-	-	0.2%
				9	463	462.7	0.3	0.1%	463	-	-	0.2%
				3	686	684.5	1.5	0.2%	685	1	0.1%	0.3%
NET 20×20	1173	[1,58]	39.3	53	7	7	-	-	7	-	-	14.3%
				51	16	14	2.0	14.3%	14	2	14.3%	14.3%
				50	19	18	1.0	5.6%	18	1	5.6%	5.6%
				41	83	80.5	2.5	3.1%	81	2	2.5%	3.8%
				40	93	90	3.0	3.3%	90	3	3.3%	4.5%
				39	101	99.5	1.5	1.5%	100	1	1.0%	3.1%
				38	110	109	1.0	0.9%	109	1	0.9%	1.9%
				37	119	118.5	0.5	0.4%	119	-	-	0.8%
				32	175	174.5	0.5	0.3%	175	-	-	0.6%
				17	508	507.8	0.2	0.0%	508	-	-	0.2%
				16	542	541.5	0.5	0.1%	542	-	-	0.2%
				15	576	575.3	0.7	0.1%	576	-	-	0.2%
				14	609	609	-	-	609	-	-	0.3%
				11	712	711.5	0.5	0.1%	712	-	-	0.1%
				7	859	858	1.0	0.1%	858	1	0.1%	0.1%

Table 2. Test Results for Method 2. This method uses Algorithm A and invokes ProblematicR2() and ProblematicR1(). ProblematicR2() finds the best Lagrangian multipliers λ_R^* for problematic R-values, which are then passed on to ProblematicR1() to find feasible solutions. Since $z(R)$ is integer, $\left\lceil z(\lambda_R^*, R) \right\rceil$ is a valid lower bound on $z(R)$. "absGap_2" is $\bar{z}(R) - \left\lceil z(\lambda_R^*, R) \right\rceil$ and "relGap_2" is $100 \times \text{absGap}_2 / \left\lceil z(\lambda_R^*, R) \right\rceil$. "absGap" is $\bar{z}(R) - z(\lambda_R^*, R)$ and "relGap" is $100 \times \text{absGap} / z(\lambda_R^*, R)$. "-" indicates that MXFI is solved optimally. Only two problematic R-values, $R = 53$ and $R = 14$ for NET 20×20, are solved exactly before rounding up $z(\lambda_R^*, R)$ but, overall, more than 50% of the problematic R-values are solved with rounding. relGap_1 is shown for comparison.

$G = (N, A)$	f^*	R	Run Time (cpu sec.)	Problematic R-Values	$\bar{z}(R)$	LB	$\lceil LB \rceil$	absGap_3	relGap_3	relGap_1	relGap_2
NET 5×5	234	[1,13]	0.2	6	55	55	55	-	-	3.8%	1.9%
				2	159	158	158	1	0.6%	4.6%	3.9%
NET 8×8	416	[1,22]	7.0	17	18	18	18	-	-	5.9%	-
				16	29	29	29	-	-	7.4%	3.6%
				15	39	39	39	-	-	5.4%	2.6%
				14	48	48	48	-	-	2.1%	-
				6	202	202	202	-	-	1.0%	1.0%
				5	228	228	228	-	-	0.4%	0.4%
NET 8×12	690	[1,34]	4.4	15	189	189	189	-	-	0.5%	-
NET 8×15	841	[1,43]	25.2	40	5	4.3	5	-	-	25.0%	-
				39	7	6.7	7	-	-	16.7%	-
				34	27	27	27	-	-	3.8%	-
				24	146	146	146	-	-	0.7%	-
				23	165	165	165	-	-	1.9%	1.2%
				22	181	181	181	-	-	0.6%	-
				17	279	279	279	-	-	0.4%	-
				6	590	590	590	-	-	0.3%	0.2%
NET 15×15	878	[1,43]	96.0	38	11	11	11	-	-	10.0%	-
				10	431	431	431	-	-	0.2%	-
				9	463	463	463	-	-	0.2%	-
				3	686	686	686	-	-	0.3%	0.1%
NET 20×20	1173	[1,58]	4013.7	51	15	14	14	1	7.1%	14.3%	14.3%
				50	19	18	18	1	5.6%	5.6%	5.6%
				41	83	83	83	-	-	3.8%	2.5%
				40	93	92	92	1	1.1%	4.5%	3.3%
				39	101	101	101	-	-	3.1%	1.0%
				38	110	110	110	-	-	1.9%	0.9%
				37	119	119	119	-	-	0.8%	-
				32	175	174.5	175	-	-	0.6%	-
				17	508	508	508	-	-	0.2%	-
				16	542	542	542	-	-	0.2%	-
				15	576	576	576	-	-	0.2%	-
				11	712	712	712	-	-	0.1%	-
				7	859	859	859	-	-	0.1%	0.1%

Table 3. Test Results from Method 3. Method 3 is essentially Method 2 except that Branch() is invoked for problematic R-values that Method 2 cannot resolve. $LB = \min\{\max_{\lambda} z(\lambda, R \mid x_{k_1} = 0), \max_{\lambda} z(\lambda, R \mid x_{k_1} = 1, x_{k_2} = 0), \ldots\}$. "$\text{absGap}_3$" is $\bar{z}(\lambda) - \lceil LB \rceil$ and "relGap_3" is $100 \times \text{absGap}_3 / \lceil LB \rceil$. Absolute and relative gaps computed with fractional values are not shown in this table. 12 problematic R-values out of the 16 remaining after Method 2 are solved exactly by Method 3. UB for $R = 51$ of NET 20×20 drops from 16 to 15, because a better feasible solution is found by Branch(). relGap_1 and relGap_2 are also shown in the table for comparison.

		Method 1			Method 2			Method 3		
G=(N, A)	Problematic R-Values	$\overline{z}(R)$	$z(\lambda, R)$	relGap$_1$	$\overline{z}(R)$	$\lceil z(\lambda_R^*, R) \rceil$	relGap$_2$	$\overline{z}(R)$	$\lceil LB \rceil$	relGap$_3$
16[1,2]	24	320	304	5.3%	304	304	-	-	-	-
	23	352	320	10.0%	320	320	-	-	-	-
	22	384	336	14.3%	336	336	-	-	-	-
	21	400	352	13.6%	352	352	-	-	-	-
	20	416	368	13.0%	368	368	-	-	-	-
	19	432	384	12.5%	400	394	1.5%	400	394	1.5%
	18	448	400	12.0%	432	420	2.9%	432	420	2.9%
	17	464	432	7.4%	464	446	4.0%	464	446	4.0%
8[1,4]	31	104	96	8.3%	96	96	-	-	-	-
	30	120	104	15.4%	112	110	1.8%	112	112	-
	29	128	120	6.7%	128	123	4.1%	128	123	4.1%
4[1,8]	38	240	200	20.0%	200	200	-	-	-	-
	37	280	240	16.7%	240	240	-	-	-	-
2[1,16]	35	280	260	7.7%	280	264	6.1%	280	280	-
	34	340	320	6.3%	340	329	3.3%	340	340	-
	33	400	380	5.3%	400	393	1.8%	400	400	-

Table 4. Comparative Test Results for NET 8×15 with Restricted Arc Capacities. "Methods" are defined in the text. Arc capacities denoted $p[1, q]$ are drawn from the set $\{ p, 2p, \ldots, qp \}$. "relGap$_1$, relGap$_2$, and relGap$_3$" are the relative gaps obtained from Method 1, 2 and 3, respectively. Only R-values with relative gaps remaining greater than 5% after Method 1 are shown in this table. 50% of problematic R-values are solved exactly by Method 2, and 50% of those remaining are then solved by Method 3. Notice that UB for R = 30 of the network with arc capacities 8[1,4] has dropped from 120 to 112, because R=31 is solved exactly by Method 2, and this makes it possible to find a better feasible solution for R = 30.

V. CONCLUSIONS AND FURTHER RESEARCH

A. SUMMARY

The maximum-flow network-interdiction problem (MXFI) arises when an "interdictor" uses limited interdiction resources to restrict an adversary's use of a network. Conceptually, the interdictor destroys arcs to minimize the maximum flow that the adversary, or "network user," can achieve through the network. In this study, we have explored new Lagrangian-relaxation techniques for solving MXFI for all values of interdiction resource R in a specified range. We assume that one unit of resource is required to interdict an arc, so we only deal with integer values of R.

MXFI can be solved by means of integer programming, but this approach is slow. Bingol (2001) modifies the Lagrangian-relaxation technique developed by Derbes (1997) to solve MXFI for all $R \in [1, |A_{\min}|]$, where $A_{\min}$ denotes a minimum cardinality cut consisting of only interdictable arcs. The Lagrangian-relaxation technique solves a sequence of max-flow problems with related arc capacities, and is therefore quite efficient.

Let $z(R)$ denote the optimal objective value to MXFI for a given R, and let $z(\lambda, R)$ denote the Lagrangian lower bound on $z(R)$ given Lagrangian multiplier λ. Essentially, Bingol (2001) evaluates $z(\lambda, R)$ simultaneously for all R, but only for values of λ equal to perturbed arc capacities. We follow this procedure—the range of λ is correct—but, when $z(\lambda, R) < \bar{z}(R)$ for $\lambda_R^* = \underset{\lambda}{\operatorname{argmax}}\, z(\lambda, R)$, where $\bar{z}(R)$ is the objective-function value for the best feasible solution found. This is accomplished with a search technique based on the slope of $z(\lambda, R)$ with respect to λ. For most test problems, we can decrease the optimality gap for "problematic R-values" by approximately 50%. A value of R is "problematic" if Bingol's procedure leaves $z(\lambda, R) < \bar{z}(R)$. Although this approach focuses on tightening lower bounds, it is also finds better feasible solutions on occasion, and sometimes proves optimality of the best solution. However, some large optimality gaps may remain.

We tackle the remaining problematic *R*-values with a modified branch-and-bound procedure. This procedure uses a partition based on the best interdiction set found, $A_I = \{k_1, k_2, \ldots, k_L\}$. In particular, we solve $\max_{\lambda} z(\lambda, R \mid x_{k_1} = 1, x_{k_2} = 0), \ldots,$ $\max_{\lambda} z(\lambda, R \mid x_{k_1} = 1, x_{k_2} = 1, \ldots, \; x_{k_{L-1}} = 1, x_{k_L} = 0)$. ($\max_{\lambda} z(\lambda, R \mid x_k = 1 \; \forall k \in A_I)$ is already solved.) The minimum lower bound from these problems is a lower bound on the optimal solution. This partitioning scheme can be applied recursively to solve MXFI exactly for a given *R*, but we have only implemented a single application of the partition. That is, we have limited the enumeration tree of this method to one level below the root. Using this technique, we may find a lower bound which equals the upper bound obtained from the best feasible solution. In this case, MXFI is solved. In our tests, we have solved approximately 70% of the problematic *R*-values that remain after applying the first technique. But, this procedure runs slowly because of the need to solve many max-flow problems, so it is best used only when large optimality gaps are encountered.

Having used these two techniques together, we have been able to solve MXFI either exactly or with small optimality gaps for most of the problematic *R*-values encountered. But, we still cannot guarantee an exact solution or a small optimality gap.

B. FURTHER RESEARCH

In this thesis, we have used a restricted modified branch-and-bound procedure that explores the enumeration tree only one level below the root. That is why some problematic *R*-values still remain. A fully recursive implementation of the branch-and-bound algorithm will solve MXFI exactly, at least in theory. Even the restricted branch-and-bound procedure becomes slow because of the need to solve many max-flow problems. Thus, a fully recursive procedure based on this methodology would be even slower. To make such a procedure reasonably efficient, a faster maximum-flow algorithm should be used and techniques for reoptimizing related maximum-flow problems should be explored.

For the sake of simplicity, we have assumed that that $r_k = 1$ for all interdictable arcs *k*. That is, one unit of resource is needed to interdict any interdictable arc. Allowing

$r_k > 1$ (but still integer) will make the problem harder. In this case, a complete branch-and-bound procedure would become more important to possess.

We have assumed that grid networks are adequate representatives for real-world networks, so we have performed our tests on these networks. However, we are uncertain how well these procedures will perform on actual networks. Obviously, future research should apply these techniques to more realistic networks.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ahuja, R.K., Magnanti, T.L., and Orlin, J. B., *Network Flows*, Prentice Hall, New Jersey, 1993, pp. 601-603.

Akgun, I., "The K-Group Maximum-Flow Network Interdiction Problem," Master's Thesis, Naval Postgraduate School, Monterey, California, March 2000.

Balcioglu, A., "An Algorithm for Enumerating the Near-Minimum Weight s-t Cuts of a Graph," Master's Thesis, Naval Postgraduate School, Monterey, California, December 2000.

Bingol, L., "A Lagrangian Heuristic for Solving a Network Interdiction Problem," Master's Thesis, Naval Postgraduate School, Monterey, California, December 2001.

Cormican, K.J., "Computational Methods for Deterministic and Stochastic Network Interdiction Problems," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1995.

Cormican, K.J., Morton, D.P., and Wood, R.K., "Stochastic Network Interdiction," *Operations Research*, Vol. 46, pp. 184-197, 1998.

Derbes, H.D., "Efficiently Interdicting a Time-Expanded Transshipment Network," Master's Thesis, Naval Postgraduate School, Monterey, California, September 1997.

Edmonds, J., and Karp, R.M., "Theoretical Improvements in Algorithmic Efficiency for Network flow Problems," *Journal of the Association for Computing Machinery*, Vol. 19, No. 2, April 1972.

Ford, L.R., Jr., and Fulkerson, D.R., "Maximal Flow through a Network," *Canadian Journal of Mathematics*, 1956.

Ghare, P.M., Montgomery, D.C., and Turner, W.C., "Optimal Interdiction Policy for a Flow Network," *Naval Research Logistics Quarterly*, Vol. 18, pp. 37-45, 1971.

Golden, B., "A Problem in Network Interdiction," *Naval Research Logistics Quarterly*, Vol. 25, pp. 711-713, 1978.

Israeli, E., "System Interdiction and Defense," Doctoral Dissertation, Naval Postgraduate School, Monterey, California, March 1999.

Israeli, E., and Wood, R.K., "Shortest Path Network Interdiction," *Networks*, to appear.

Lubore, S.H., Ratliff, H.D., and Sicilia, G.T., "Finding the n most Vital Links in Flow Networks," *Management Science*, Vol. 21, No. 5, 1975.

McMasters, A.W., and Mustin, T.M., "Optimal Interdiction of a Supply Network," *Naval Research Logistics Quarterly*, Vol. 17, pp. 261-268, 1970.

Morton, D., "Stochastic Network Interdiction of Nuclear Material Smuggling," *Systems and Industrial Engineering Seminar*, September 2001.

Steinrauf, R.L., "Network Interdiction Models," Master's Thesis, Naval Postgraduate School, Monterey, California, September 1991.

Taha, H.A., *Operations Research*, 6th ed., Prentice Hall, New Jersey, 1997, pp. 385-390.

Wevley, C.M., "The Quickest Path Network Interdiction Problem," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1999.

Whiteman, P.S., "Improving Single Strike Effectiveness for Network Interdiction," *Military Operations Research*, Vol. 4, pp.15-30, 1999.

Wollmer, R.D., "Removing Arcs from a Network," *Operations Research*, Vol. 12, pp. 807-1076, 1964.

Wollmer, R.D., "Algorithm for Targeting Strikes in a Lines-of-Communication Network," *Operations Research*, Vol. 18, pp.497-515, 1970.

Wood, R.K., "Deterministic Network Interdiction," *Mathematical and Computer Modeling*, Vol. 17, No. 2, pp. 1-18, 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Professor R. Kevin Wood
   Department of Operations Research, Code OR/Wd
   Naval Postgraduate School
   Monterey, CA

4. LCDR Kelly J. Cormican
   Department of Operations Research, Code OR/CK
   Naval Postgraduate School
   Monterey, CA

5. Kara Kuvvetleri Komutanligi
   Kutuphane
   Bakanliklar, Ankara, TURKEY

6. Kara Harp Okulu Komutanligi
   Kutuphane
   Tuzla, Istanbul, TURKEY

7. Bilkent Universitesi Kutuphanesi
   Bilkent, Ankara, TURKEY

8. Orta Dogu Teknik Universitesi Kutuphanesi
   Balgat, Ankara, TURKEY

9. Bogazici Universitesi Kutuphanesi
   Bebek, Istanbul, TURKEY

10. Top. Utgm. Adnan Uygun
    Kara Kuvvetleri Komutanligi
    Bakanliklar, Ankara, TURKEY