Shortest-Path Network Interdiction

Eitan Israeli, R. Kevin Wood
Operations Research Department, Naval Postgraduate School, Monterey, California 93943

We study the problem of interdicting the arcs in a network in order to maximize the shortest *s–t* path length. "Interdiction" is an attack on an arc that destroys the arc or increases its effective length; there is a limited interdiction budget. We formulate this bilevel, max–min problem as a mixed-integer program (MIP), which can be solved directly, but we develop more efficient decomposition algorithms. One algorithm enhances Benders decomposition by adding generalized integer cutting planes, called "supervalid inequalities" (SVIs), to the master problem. A second algorithm exploits a unique set-covering master problem. Computational results demonstrate orders-of-magnitude improvements of the decomposition algorithms over direct solution of the MIP and show that SVIs also help solve the original MIP faster. © 2002 Wiley Periodicals, Inc.*

Keywords: interdiction; shortest paths; bilevel program; Benders decomposition

1. INTRODUCTION

Network-interdiction problems involve two opposing forces, a *leader* and a *follower,* who are engaged in a warlike conflict. The follower operates a network in order to optimize some objective function such as moving a supply convoy through the network as quickly as possible or maximizing the amount of materiel transported through the network. The leader attempts to limit the follower's achievable objective value by interdicting arcs, for example, by attacking arcs to destroy them, to slow travel over them, or to reduce their capacity. This paper develops a new model and solution methods for the problem of interdicting a transportation network in order to maximize the shortest path length between two specified nodes.

The topic of network interdiction has received some attention over the years, initially with military applications. For instance, McMasters and Mustin [31] and Ghare et al. [21] developed methods for interdicting a capacitated supply network to disrupt the movements of enemy troops and materiel. Models of drug interdiction have triggered further research [41, 45] as has the need to assess the vulnerability of information networks to interdiction [23, 32].

The network-interdiction problem that we focus on is *Maximizing the Shortest Path* (MXSP). (See the related models of Fulkerson and Harding [18] and Golden [22].) In this problem, a *network user,* that is, the follower, wishes to traverse a path of minimum length (or minimum time, minimum cost, etc.) between two specified nodes, s and t, in a directed network. But, by first attacking the network using limited resources, an *interdictor,* that is, the leader, can destroy certain arcs, or increase their effective length, and thereby increase the follower's shortest s–t path length. MXSP is the interdictor's problem: subject to a limited interdiction budget, interdict arcs in a network to maximize the shortest path length between nodes s and t. We assume that if interdicted arcs are destroyed the interdiction budget is insufficient to disconnect s from t, since, otherwise, the problem degenerates into a much simpler minimum-cut problem.

In MXSP, arc interdiction involves a binary decision with a deterministic outcome: Arc k is interdicted or it is not, interdiction consumes a known quantity of interdiction resource(s), and interdiction always increases the arc's effective length by a prespecified, possibly infinite amount. The *k-most-vital-arcs problem* [13, 30] is a special case of MXSP in which the interdictor seeks to destroy exactly k arcs to interdict the network most effectively. Since that problem is NP-complete [5], it follows that MXSP is NP-complete.

The k-most-vital-arcs problem has received limited attention and we are not aware of effective algorithms for solving it. Malik et al. [30] suggested a potentially useful but theoretically flawed algorithm for the problem, which we discuss in Section 4. Corely and Shaw [13] investigated the *single-most-vital-arc problem,* but this problem is a simple, special case of MXSP which is solvable in polynomial time.

MXSP, and our mathematical-programming approach to solving it, allows more generality than does the k-most-vital-arcs problem. We can use general resource constraints to model arcs that require various amounts of resource to

Received July 1999; accepted May 2002
Correspondence to: R. K. Wood; e-mail: kwood@nps.navy.mil
Contract grant sponsors: Office of Naval Research; Air Force Office of Scientific Research

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2002	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Shortest-Path Network Interdiction	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School Operations Research Department Monterey, CA 93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**15**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

interdict. Also, our methods can be extended to model multiple types of interdiction resources, for example, ground troops, aerial sorties, and cruise missiles, although we focus on models with a single resource type. Our binary interdiction variables may also be more realistic than are the continuous variables of Fulkerson and Harding [18] and Golden [22], where the length of an arc increases linearly with the amount of interdiction resource applied. As evidence, we note that Whiteman [43] incorporated binary interdiction variables in a network-interdiction model for planning aerial interdiction sorties for the United States Strategic Command. His model extends the maximum-flow-based integer-programming models of Wood [45], which are similar in spirit to the current paper's model for MXSP.

Cormican et al. [15] and Israeli [27] studied network-interdiction problems where the success of an attempted interdiction is uncertain and/or some data are uncertain. Uncertainty may be important in some instances, but is beyond the scope of this paper.

MXSP may be viewed as a *bilevel mixed-integer program* (BLMIP) (e.g., [9]), which is a special case of a *static Stackelberg game* [37]. BLMIPs assume that a leader first chooses his actions and, subsequently, a follower reacts optimally knowing those actions. (Unlike BLMIPs, a more general Stackelberg game may continue, alternating "plays" between the leader and follower.) However, the usual BLMIP does not assume a max–min conflict, as does MXSP. For instance, the leader and follower might represent two levels of decision makers in a corporation and, consequently, would have similar objectives.

Israeli [27] reviewed existing algorithms for BLMIPs and observes that none is appropriate for solving MXSP. Some algorithms [7, 42] use a "positive approach," which implies that they function best when there is a strong positive correlation between the leader's and follower's objectives. Such algorithms are likely to be inefficient with diametrically opposed objectives as in MXSP. Furthermore, computational experience is discouraging: Bard and Moore [7] solved problems with, at most, 35 binary variables for the leader; Wen and Yang [42] solved problems with, at most, 15 leader variables; we intend to solve problems with over 1000 leader variables.

Two other techniques for solving BLMIPs [33, 40] may not be inherently positive, but they rely on solving bilevel linear programs (BLLPs), which are typically solved using a positive approach. Moore and Bard's [33] algorithm is complicated and these authors could not solve problems with more than 10 integer leader variables. Vicente et al. [40] only demonstrated that a BLMIP can be converted to a BLLP with an unknown penalty parameter; they reported no computational results.

In fact, only three exact algorithms for BLLPs have been tested on relatively large problems [7, 24, 28] and all used a positive approach. A few exact algorithms [2, 35, 39] do not use positive approaches, but none has been tested on large problems, and all are complicated to implement. For instance, Anandalingam and Apprey [2] called a subroutine to solve a concave minimization problem over a polyhedron [17]. But this minimization requires that the leader's feasible region be embedded in a simplex, preferably a "tight" one, which may not exist.

We conclude that no existing BLMIP algorithms are likely candidates for solving MXSP effectively. Furthermore, the literature provides no algorithms in the bilevel arena that take advantage of the special max–min and shortest-path structure of MXSP. We must devise new techniques.

In this paper, we first define MXSP as a max–min problem and show how to formulate it as a mixed-integer program (MIP). We would prefer to solve MXSP directly through that MIP using simple, off-the-shelf, mathematical-programming software. However, it is intuitively clear, and later demonstrated by computation, that direct solution of the MIP through linear-programming-based branch and bound can be extremely difficult even for modest-sized problems. Therefore, we devised two decomposition-based algorithms for MXSP, along with important enhancements, and demonstrate their computational effectiveness. Decomposition is a natural approach for MXSP because of the easy-to-solve shortest-path subproblems embedded in the model.

Our first algorithm applies Benders decomposition [10] to solve MXSP, much as Cormican [14] solved a maximum-flow network-interdiction problem. This basic algorithm performs poorly, so we develop a generalization of integer cutting planes called "supervalid inequalities" (SVIs) to improve performance. The enhanced algorithm works well when delays caused by interdiction are modest, but the algorithm's Benders cuts are ineffective when delays are large. A second decomposition algorithm ameliorates this difficulty (*i*) by simplifying the master problem to a set-covering problem, (*ii*) by incorporating a greedy heuristic to solve the new master problem, in addition to using an exact algorithm, and (*iii*) by exploiting the special structure of shortest-path problems and suboptimal solutions for the follower. Computational results compare the efficiency of the various solution techniques for MXSP, including the use of SVIs coupled with a branch-and-bound solution of the MIP. We end the paper with conclusions and directions for further research.

2. PROBLEM FORMULATION

MXSP is defined on a directed graph $G = (N, A)$, where N is the set of nodes, and A, the set of arcs. The nominal length of arc $k \in A$ is $c_k \geq 0$; interdiction increases the arc's length to $c_k + d_k$, where $d_k > 0$. If interdiction actually destroys arc k, making it impassable, the same model applies by setting d_k to a sufficiently large value. For instance, if $d_k \equiv d^+ = |N| \max_{k' \in A}\{c_{k'}\}$ for all $k \in A$, a solution in which the follower's shortest-path length is d^+ or greater must indicate that all paths for the follower have been destroyed. We model only a single interdiction re-

source constraint because some of our algorithmic specializations require this, but the basic approaches of Sections 3 and 5 could handle constraints that are more complicated. The mathematical-programming formulation of MXSP is

Problem: MXSP: Maximize the shortest s–t path length in a directed network by interdicting arcs,

Indices: $i \in N$, nodes in G (s is the source node, t is the sink node),
$k = (i, j) \in A$, arcs in G,
$k \in FS(i)$ ($k \in RS(i)$), arcs directed out of (into) node i,

Data: $0 \le c_k < \infty$, nominal integer length of arc k (vector form $\mathbf{c}$),
$0 < d_k < \infty$, added integer delay if arc k is interdicted (vector form $\mathbf{d}$),
$r_k > 0$, resource required to interdict arc k (vector form $\mathbf{r}$),
r_0, total amount of interdiction resource available,

Variables: $x_k = 1$ if arc k is interdicted by the leader; else $x_k = 0$,
$y_k = 1$ if arc k is traversed by the follower; else $y_k = 0$,

Formulation:

$$[\text{MXSP-P}] \quad z^* = \max_{\mathbf{x} \in X} \min_{\mathbf{y}} \sum_{k \in A} (c_k + x_k d_k) y_k$$

$$\text{s.t.} \quad \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = \begin{cases} 1 & \text{for } i = s \\ 0 & \forall\, i \in N \backslash \{s, t\} \\ -1 & \text{for } i = t \end{cases} \quad (1)$$

$$y_k \ge 0 \quad \forall\, k \in A,$$

where $X = \{\mathbf{x} \in \{0, 1\}^{|A|} | \mathbf{r}^T \mathbf{x} \le r_0\}$.

Some additional notation and comments are as follows:

1. $\mathbf{x}^*$ will denote an optimal interdiction plan.
2. Flow-balance constraints (1), in variables $\mathbf{y}$, route one unit of flow from s to t; the inner minimization is a standard shortest-path model with arc lengths $c_k + x_k d_k$.
3. c_k is the nominal length of arc k and $c_k + d_k$ is the arc's length if it is interdicted; d_k is finite and comprises such factors as repair time or the length of a local detour or, if sufficiently large, represents complete destruction of arc k.
4. r_k is typically a small positive integer; it tells us, for instance, how many weapons are required to interdict arc k. Thus, r_0 is the total number of such weapons available.
5. Optimal solutions in $\mathbf{y}$ are assumed to be integral since the inner minimization of [MXSP-P] is a standard shortest-path model, which has integral solutions.

We also note that it is straightforward to extend [MXSP-P] to handle undirected networks and/or node interdiction and to disallow interdiction of certain arcs.

If we (*i*) fix $\mathbf{x}$, (*ii*) take the dual of the inner minimization in [MXSP-P], (*iii*) make a few simple modifications, and (*iv*) then release $\mathbf{x}$, the following MIP results:

$$[\text{MXSP-D}] \quad z^* = \max_{\mathbf{x}, \pi} \pi_t - \pi_s$$

$$\text{s.t.} \quad \pi_j - \pi_i - d_k x_k \le c_k \quad \forall\, k = (i, j) \in A$$

$$\pi_s = 0$$

$$\mathbf{x} \in X.$$

We may assume that $\pi_s = 0$ because the inner minimization of MXSP has at least one redundant flow-balance constraint (as do all network flow models containing a balance constraint for each node). Also, note that the dual variables π are unconstrained in sign and, having reversed their indicated signs compared to convention, we may interpret π_i as the postinterdiction shortest-path length from s to i.

[MXSP-D] is essentially the model explored by Fulkerson and Harding [18] and by Golden [22], except that those authors required variables $\mathbf{x}$ to be continuous. Thus, their model is a simple linear program (LP). Fulkerson and Harding [18] suggested solving the dual of that LP as a parametric min-cost flow model. This approach does not appear to be useful for solving the LP relaxation of [MXSP-D], because bounds $\mathbf{x} \le \mathbf{1}$ needed by that relaxation cannot be incorporated.

In theory, we can solve MXSP by solving [MXSP-D] directly, that is, using a standard LP-based branch-and-bound algorithm—and we will test this approach. However, our decomposition approaches exploit the rapid solvability of shortest-path problems, which branch and bound cannot, and these approaches lead to a generalization of integer cutting planes which achieves speedups in both the decomposition and direct solutions. Unfortunately, both the direct method and our "basic decomposition" suffer from weak LP relaxations when the d_k are large. Another decomposition approach described in Section 5 helps mitigate this difficulty. Interestingly, techniques derived from that approach are also useful in the basic decomposition when solving problems having d_k of moderate magnitude.

3. A BASIC DECOMPOSITION ALGORITHM

Our first decomposition algorithm for MXSP applies Benders decomposition directly to [MXSP-P]. (Equivalently, we can decompose [MXSP-D] as follows: Fix $\mathbf{x}$, take the dual of the problem with respect to π, and release $\mathbf{x}$ so that a max–min problem results; see Garfinkel and Nemhauser [19, pp. 135–143]). Let $\hat{\mathbf{y}} \in \{0, 1\}^{|A|}$ denote the

arc-path incidence vector corresponding to an s–t path P, that is, $\hat{y}_k = 1$ implies that arc k is in P and, otherwise, $\hat{y}_k = 0$. Let $\hat{Y}$ denote a collection of arc-path incidence vectors corresponding to a subset of all simple s–t paths in G. (A path is "simple" if it contains no cycles.) For simplicity, we refer to $\hat{\mathbf{y}}$ as "a path" and to $\hat{Y}$ as "a set of paths." Also, let $D = \text{diag}(\mathbf{d})$ and define

$$[\text{Master}(\hat{Y})\text{-1a}] \quad z_{\hat{Y}} = \max_{\mathbf{x} \in X} z$$

$$\text{s.t.} \quad z \le \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}} \quad \forall\, \hat{\mathbf{y}} \in \hat{Y} \tag{2}$$

$$[\text{SP-Sub}(\hat{\mathbf{x}})] \quad z_{\hat{\mathbf{x}}} = \min_{\mathbf{y}} \sum_{k \in A} (c_k + \hat{x}_k d_k) y_k$$

$$\text{s.t.} \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = \begin{cases} 1 & i = s \\ 0 & i \in N \backslash \{s, t\} \\ -1 & i = t \end{cases}$$

$$y_k \ge 0 \quad \forall\, k \in A.$$

Let Y denote the set of all simple s–t paths. For fixed $\mathbf{x} = \hat{\mathbf{x}}$, a solution to the inner minimization of [MXSP-P], that is, a solution to [SP-Sub($\hat{\mathbf{x}}$)], always occurs at $\hat{\mathbf{y}} \in Y$. Therefore, [Master($\hat{Y}$)-1a] is an equivalent, path-based formulation of [MXSP-P] when $\hat{Y} = Y$. (Actually, the formulations are equivalent even for continuous $\mathbf{x} \ge \mathbf{0}$.) However, we hope to solve [MXSP-P], at least approximately, by sequentially generating only a small fraction of the paths of Y within a decomposition algorithm.

It will be important later to keep track of the *interdiction/response pair* ($\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$), which denotes a specific interdiction plan $\hat{\mathbf{x}}$ and an optimal, but not necessarily unique, response $\hat{\mathbf{y}}$ by the follower to $\hat{\mathbf{x}}$. With respect to a set of such pairs $\hat{X}\hat{Y}$, we let $\hat{X} = \{\hat{\mathbf{x}} | (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}\}$ and $\hat{Y} = \{\hat{\mathbf{y}} | (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}\}$. A simple decomposition algorithm to solve MXSP can now be stated:

Algorithm 1: Basic Benders decomposition algorithm for MXSP.

Input: An instance of MXSP and allowable optimality gap ε.

Output: An ε-optimal interdiction plan $\mathbf{x}^\varepsilon$ for MXSP.

Step 0: $\hat{X}\hat{Y} \leftarrow \emptyset$; $\underline{z} \leftarrow -\infty$; $\bar{z} \leftarrow \infty$; $\hat{\mathbf{x}} \leftarrow \mathbf{0}$;

Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for $\hat{\mathbf{y}}$ and objective value $z_{\hat{\mathbf{x}}}$;
$\hat{X}\hat{Y} \leftarrow \hat{X}\hat{Y} \cup (\hat{\mathbf{x}}, \hat{\mathbf{y}})$;
If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$;
If $\bar{z} - \underline{z} \le \varepsilon$ then go to Step 3;

Step 2: Solve [Master($\hat{Y}$)-1a] for $\hat{\mathbf{x}}$ and objective value $z_{\hat{Y}}$;
$\bar{z} \leftarrow z_{\hat{Y}}$; If $\bar{z} - \underline{z} > \varepsilon$ then go to Step 1;

Step 3: $\mathbf{x}^\varepsilon \leftarrow \mathbf{x}'$; Print $\mathbf{x}^\varepsilon$; Stop.

The correctness of the algorithm, as in any Benders decomposition algorithm, is based on the following observations:

1. The subproblem finds an optimal follower's response for any interdiction plan $\hat{\mathbf{x}}$. Hence, $z_{\hat{\mathbf{x}}}$ gives a lower bound on the leader's optimal objective value.
2. [Master(Y)-1a] is equivalent to [MXSP-P], so, when $\hat{Y} \subseteq Y$, [Master($\hat{Y}$)-1a] is a relaxation of [MXSP-P] and $z_{\hat{Y}}$ is an upper bound on the interdictor's optimal objective value. [The master-problem constraints (2) are called "Benders cuts."]
3. If the subproblem produces the same solution (a simple s–t path) twice, the upper and lower bounds match and the algorithm terminates. The algorithm converges in a finite number of iterations because the number of simple s–t paths is finite.

Run times in our test problems are reduced up to 40% by generating all optimal responses for a given $\hat{\mathbf{x}}$ and by adding all the corresponding cuts to the master problem, rather than generating a single response and cut. Specifically, for fixed $\hat{\mathbf{x}}$, we use the algorithm of Byers and Waterman [11] to find all shortest paths $\hat{\mathbf{y}}_l$, $l = 1, \ldots, h$, and add the cuts $z \le \mathbf{c}^T\hat{\mathbf{y}}_l + \mathbf{x}^T D\hat{\mathbf{y}}_l$, $l = 1, \ldots, h$. The resulting master problems are harder to solve, but this is usually offset by a reduced number of iterations (repetitions of Steps 1 and 2) for the overall algorithm.

4. SUPER-VALID INEQUALITIES FOR BENDERS DECOMPOSITION

This section develops a generalization of integer cutting planes to improve the efficiency of Algorithm 1 and demonstrates the improved algorithm with an example.

4.1. Basic Theory

To strengthen the LP relaxation of [Master($\hat{Y}$)-1a], we introduce the concept of a *supervalid inequality* (SVI), which may be viewed as a generalization of the standard "valid inequality" (which is essentially the same as an "integer cutting plane" or "cut," e.g., Nemhauser and Wolsey [34, pp. 205–295], and Wolsey [44, pp. 113–124]). Standard valid inequalities applied to the master problem would not eliminate any feasible $\mathbf{x}$, but would, normally, reduce the size of the feasible region of that problem's LP relaxation. On the other hand, our SVIs typically make the most recent solution $\hat{\mathbf{x}}$ infeasible and may make other integer solutions infeasible, thereby reducing the size of the feasible region of the LP relaxation under consideration. But these inequalities are guaranteed not to eliminate any optimal solutions unless the incumbent is itself optimal.

Definition 1. *Let* $\mathbf{v}$ *and* $\mathbf{x}$ *denote, respectively, the vectors of continuous and integer variables in an MIP, let* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *be two conforming vectors of constants, respectively, and let* w_0 *be a scalar constant. The inequality* $\mathbf{w}_1^T\mathbf{v} + \mathbf{w}_2^T\mathbf{x}$

$\geq w_0$ *is supervalid for this MIP, that is, it is a supervalid inequality for the MIP, if* (*i*) *adding that inequality to the MIP does not eliminate all optimal solutions or* (*ii*) *an incumbent solution to the MIP*, $(\mathbf{v}', \mathbf{x}')$, *is* (*already*) *optimal*. ∎

Hooker [25] and Atamtürk et al. [3] also used inequalities that are invalid in the standard sense; Hooker calls them "nonvalid cuts." In particular, feasible solutions may be eliminated through their inequalities, although, unlike SVIs, no optimal solutions are. In theory, SVIs can be used, when solving an MIP like [MXSP-D] by branch and bound, to help fathom nodes in the enumeration tree. In fact, we do just that in Section 7.3. However, we develop and demonstrate SVIs in the context of [Master($\hat{Y}$)-1a].

Theorem 1. *For any Benders cut* $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$ *in* [*Master*($\hat{Y}$)-1*a*], *the Type-I inequality* $\hat{\mathbf{y}}^T\mathbf{x} \geq 1$ *is supervalid for* [*Master*(*Y*)-1*a*].

Proof. The statement presupposes that X is nonempty and that some feasible interdiction/response pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ has generated the Benders cut $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$. Let $\hat{z} = \mathbf{c}^T\hat{\mathbf{y}} + \hat{\mathbf{x}}^T D\hat{\mathbf{y}}$. If the incumbent solution is optimal, then any inequality in $\mathbf{x}$ is supervalid, so suppose not, that is, suppose that the incumbent has led to $\underline{z} < z^*$. Let $(z^*, \mathbf{x}^*)$ denote an optimal solution to [Master(Y)-1a] and note that $\hat{\mathbf{y}}^T\mathbf{x}^* = 0$ or $\hat{\mathbf{y}}^T\mathbf{x}^* \geq 1$. When $\hat{\mathbf{y}}^T\mathbf{x}^* = 0$,

$z^* \leq \mathbf{c}^T\hat{\mathbf{y}} + (\mathbf{x}^*)^T D\hat{\mathbf{y}}$ is true for any $\hat{\mathbf{y}} \in \hat{Y} \in X$,
$= \hat{z} + (\mathbf{x}^*)^T D\hat{\mathbf{y}} - \hat{\mathbf{x}}^T D\hat{\mathbf{y}}$ because $\hat{z} = \mathbf{c}^T\hat{\mathbf{y}} + \hat{\mathbf{x}}^T D\hat{\mathbf{y}}$,
$\leq \hat{z}$ because $\hat{\mathbf{y}}^T\mathbf{x}^* = 0$ implies that $(\mathbf{x}^*)^T D\hat{\mathbf{y}} = 0$, and because $\mathbf{d} > \mathbf{0}$, $\hat{\mathbf{y}} \geq \mathbf{0}$, and $\mathbf{x}^* \geq \mathbf{0}$ imply that $\hat{\mathbf{x}}^T D\hat{\mathbf{y}} \geq 0$,
$\leq \underline{z}$ because $\hat{\mathbf{x}}$ need not be the incumbent solution, and
$< z^*$ by assumption; but this is a contradiction.

Therefore, if the incumbent solution is not optimal, $\hat{\mathbf{y}}^T\mathbf{x}^* \geq 1$ must be true for every optimal solution $(z^*, \mathbf{x}^*)$. Thus, by definition, the inequality $\hat{\mathbf{y}}^T\mathbf{x} \geq 1$ is supervalid. ∎

Note that if $\hat{\mathbf{y}}^T\hat{\mathbf{x}} = 0$ for a pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ the inequality $\hat{\mathbf{y}}^T\mathbf{x} \geq 1$ makes $\hat{\mathbf{x}}$ infeasible. Thus, $\hat{\mathbf{y}}^T\mathbf{x} \geq 1$ need not be a valid inequality in the standard sense.

Theorem 1 has a simple interpretation: If an objective value better than $\underline{z}$ is ever to be obtained, we must "raise the ceiling" imposed by $\mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$ in the Benders cut $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$ to at least $\hat{z}$ (which is no greater than $\underline{z}$) by forcing $\mathbf{x}^T D\hat{\mathbf{y}}$ to be sufficiently large.

Sometimes, interdiction of a single arc does not raise the ceiling high enough and the following corollary applies:

Corollary 1. *Given the Benders cut* $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$, *order the* $d_k\hat{y}_k > 0$ *so that* $d_{k_1}\hat{y}_{k_1} \geq d_{k_2}\hat{y}_{k_2} \cdots \geq d_{k_L}\hat{y}_{k_L}$. *Then, if* $\underline{z} \geq \mathbf{c}^T\hat{\mathbf{y}} + \sum_{l=1}^{p} d_{k_l}\hat{y}_{k_l}$ *for* $p \leq L$, *the Type-I SVI of Theorem* 1 *can be tightened to* $\hat{\mathbf{y}}^T\mathbf{x} \geq p + 1$.

Proof. Analogous to Theorem 1, if the current solution is not optimal, then constraints $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}^T\mathbf{x} \leq p$ and the condition $\underline{z} \geq \mathbf{c}^T\hat{\mathbf{y}} + \sum_{l=1}^{p} d_{k_l}\hat{y}_{k_l}$ imply that $z^* \leq \mathbf{c}^T\hat{\mathbf{y}} + \sum_{l=1}^{p} d_{k_l}\hat{y}_{k_l} \leq \underline{z}$ for any optimal solution $(z^*, \mathbf{x}^*)$. This is a contradiction, so $\hat{\mathbf{y}}^T\mathbf{x} \geq p + 1$ must be supervalid. ∎

We next develop "Type-II SVIs" which may be viewed as "lifted" versions of Type-I SVIs (although they will sometimes be identical).

Theorem 2. *Let* $\hat{X}\hat{Y}$ *be a set of interdiction/response pairs generated up to some point in Algorithm* 1, *and suppose that* $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$. *Define* $\tilde{\mathbf{y}} = (diag(\mathbf{1} - \hat{\mathbf{x}}))\hat{\mathbf{y}}$. *Then, the Type-II inequality* $\tilde{\mathbf{y}}^T\mathbf{x} \geq 1$ *is supervalid for* [*Master*(*Y*)-1*a*].

Proof. If the interdictor is ever to see a response from the follower that is different and longer than $\hat{\mathbf{y}}$, then he must interdict some arc that was traversed in $\hat{\mathbf{y}}$ but not interdicted. This is precisely the set of arcs in the *uninterdicted subpath* defined by $\tilde{\mathbf{y}}$. ∎

Given $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$, the Type-I SVI $\hat{\mathbf{y}}^T\mathbf{x} \geq 1$ and corresponding Type-II SVI $\tilde{\mathbf{y}}^T\mathbf{x} \geq 1$ will be identical if the follower traverses no interdicted arcs in $\hat{\mathbf{y}}$, that is, if $\hat{\mathbf{y}}^T\hat{\mathbf{x}} = 0$, so that $\tilde{\mathbf{y}} = \hat{\mathbf{y}}$. But if $\hat{\mathbf{y}}^T\hat{\mathbf{x}} > 0$, then $\tilde{\mathbf{y}} \leq \hat{\mathbf{y}}$ and $0 = \tilde{y}_k < \hat{y}_k = 1$ for at least one arc k, and, therefore, the Type-II SVI is stronger than its Type-I counterpart. This strengthening is analogous to "lifting" for valid inequalities (e.g., Nemhauser and Wolsey [34, pp. 261–267]), so we borrow that term here. Type-II SVIs can be lifted through other considerations described next.

Theorem 3. *Suppose that the pair* $(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1) \in \hat{X}\hat{Y}$ *in Algorithm* 1 *generates the Type-II SVI* $\tilde{\mathbf{y}}_1^T\mathbf{x} \geq 1$. *Let* $\tilde{K}_1 = \{k | \tilde{y}_{1k} = 1\}$ *and let* $\tilde{K}_2$ *be any subset of* $\tilde{K}_1$ *such that* $\sum_{k\in\tilde{K}_2} d_k + (\mathbf{c}^T\hat{\mathbf{y}}_1 + \hat{\mathbf{x}}_1^T D\hat{\mathbf{y}}_1) \leq \underline{z}$. *Then, the following inequality is supervalid*: $\tilde{\mathbf{y}}_2^T\mathbf{x} \geq 1$, *where* $\tilde{y}_{2k} = \tilde{y}_{1k}$ *for all* $k \in \tilde{K}_1 \backslash \tilde{K}_2$ *and* $\tilde{y}_{2k} = 0$, *otherwise*.

Proof. To achieve any solution of value greater than $\underline{z}$, we must find $\mathbf{x}$ such that $\mathbf{c}^T\hat{\mathbf{y}}_1 + \mathbf{x}^T D\hat{\mathbf{y}}_1 > \underline{z}$, which requires that some arc from $\tilde{\mathbf{y}}_1$, other than or in addition to those with indices $\tilde{K}_2$, be interdicted. ∎

The inequality $\tilde{\mathbf{y}}_2^T\mathbf{x} \geq 1$ is a *lifted Type-II SVI* and should replace its progenitor $\tilde{\mathbf{y}}_1^T\mathbf{x} \geq 1$.

Naturally, $\tilde{K}_2$ should be a maximal set with the property that $\sum_{k\in\tilde{K}_2} d_k + (\mathbf{c}^T\hat{\mathbf{y}}_1 + \hat{\mathbf{x}}_1^T D\hat{\mathbf{y}}_1) \leq \underline{z}$. In practice, we order the indices of $\tilde{K}_1 = \{k_1, k_2, \ldots, k_p\}$ so that $d_{k_1} \leq d_{k_2} \leq \cdots \leq d_{k_p}$ and define $\tilde{K}_2 = \{k_1, k_2, \ldots, k_{p'}\}$ such that $\sum_{k\in\tilde{K}_2} d_k + (\mathbf{c}^T\hat{\mathbf{y}}_1 + \hat{\mathbf{x}}_1^T D\hat{\mathbf{y}}_1) \leq \underline{z}$, but $\sum_{k\in\tilde{K}_2\cup\{k_{p'+1}\}} d_k + (\mathbf{c}^T\hat{\mathbf{y}}_1 + \hat{\mathbf{x}}_1^T D\hat{\mathbf{y}}_1) > \underline{z}$. The ordering may not be unique, and there may be other ways to define maximal subsets $\tilde{K}_2$, so it may be possible to lift $\tilde{\mathbf{y}}_1^T\mathbf{x} \geq 1$ in different ways to produce different lifted SVIs. This nonunique lifting is akin to the sequence-dependent lifting of valid inequalities, for

example, Padberg [36] and Balas and Zemel [4]. In practice, we only attempt to lift an SVI with respect to a single ordering.

The enhanced version of Algorithm 1 generates at least one Type-I SVI and one Type-II SVI in each iteration. These SVIs may be identical, but typically diverge through tightening and/or lifting. Lifting and/or tightening is attempted as soon as an SVI is generated and whenever $\underline{z}$ changes during the course of the algorithm.

4.2. Algorithm 1 with SVIs: An Example

Here, we demonstrate the application of SVIs to Algorithm 1 through an example. In every iteration, we add to the master problem one Benders cut, one Type-I SVI, and one Type-II SVI and then attempt to tighten or lift those SVIs along with previously generated SVIs. The final version of Algorithm 1 depends on techniques developed in the context of a decomposition discussed later, so we revisit Algorithm 1 in Section 6.

Example 1. *Consider a network containing s–t paths* P_1 *and* P_2, *among others*: P_1 *consists of arcs* 1, 2, *and* 3 *and* P_2 *consists of arcs* 1, 4, *and* 5. *Those arcs have the following parameters*: $c_1 = 3$, $c_2 = 1$, $c_3 = 8$, $c_4 = 4$, $c_5 = 6$, $d_1 = 3$, $d_2 = 4$, $d_3 = 5$, $d_4 = 1$, *and* $d_5 = 3$. *Suppose that* P_1 *is the shortest s–t path in the network and*, *hence*, *is returned by the subproblem in the first iteration of the algorithm when* $\hat{\mathbf{x}} = \mathbf{0}$. *Therefore*, $\underline{z} = 12$, *and the first Benders cut of the master problem is*

$$\text{Benders}_1: \quad z \le 12 + 3x_1 + 4x_2 + 5x_3.$$

The associated Type-I and Type II SVIs complete this iteration's master problem:

$$\begin{aligned} \text{I-SVI}_1: &\quad x_1 + x_2 + x_3 \ge 1, \\ \text{II-SVI}_1: &\quad x_1 + x_2 + x_3 \ge 1. \end{aligned}$$

Suppose that the interdictor has enough resource(s) to interdict arcs 1, 2, and 3 together and that this is the solution, with $\bar{z} = 24$, of the first master problem, which consists of Benders_1, I-SVI_1, and II-SVI_1. Further, assume that the shortest s–t path given these interdictions is P_2, so that $\underline{z} = c_1 + d_1 + c_4 + c_5 = 16$. We can therefore lift II-SVI_1 because interdiction of arc 1 alone cannot raise the ceiling on z over the lower bound $\underline{z} = 16$. (We could also lift this constraint by noting that interdiction of arc 2 alone is insufficient to raise the ceiling above 16, but we follow the discussion after Theorem 3 and lift only with respect to the first indication.) The constraints from the first iteration are now

$$\begin{aligned} \text{Benders}_1: &\quad z \le 12 + 3x_1 + 4x_2 + 5x_3, \\ \text{I-SVI}_1: &\quad x_1 + x_2 + x_3 \ge 1, \\ \text{II-SVI}_1: &\quad x_2 + x_3 \ge 1, \end{aligned}$$

and the constraints from the second iteration are

$$\begin{aligned} \text{Benders}_2: &\quad z \le 13 + 3x_1 + 1x_4 + 3x_5, \\ \text{I-SVI}_2: &\quad x_1 + x_4 + x_5 \ge 2, \\ \text{II-SVI}_2: &\quad x_4 + x_5 \ge 1, \end{aligned}$$

where we have already tightened I-SVI_2 by Corollary 1. The new master problem incorporates the six constraints above. Suppose that its solution is $\hat{x}_1 = \hat{x}_3 = \hat{x}_4 = \hat{x}_5 = 1$ and $\hat{x}_2 = 0$, so that $\bar{z} = 20$ and the algorithm continues.

Then suppose that $\underline{z}$ increases to 17 at some later iteration. We can then tighten the right-hand side of I-SVI_1 to 2, because to raise z over $\underline{z}$ we must interdict at least two of the three arcs in P_1. We can also lift II-SVI_2, because interdiction of arc 4 (along with arc 3) cannot raise z over $\underline{z} = 17$. The master problem derived from the first two iterations is now (constraints from any subsequent iterations are omitted)

$$\begin{aligned} \text{Benders}_1: &\quad z \le 12 + 3x_1 + 4x_2 + 5x_3, \\ \text{I-SVI}_1: &\quad x_1 + x_2 + x_3 \ge 2, \\ \text{II-SVI}_1: &\quad x_2 + x_3 \ge 1, \\ \text{Benders}_2: &\quad z \le 13 + 3x_1 + 1x_4 + 3x_5, \\ \text{I-SVI}_2: &\quad x_1 + x_4 + x_5 \ge 2, \\ \text{II-SVI}_2: &\quad x_5 \ge 1. \end{aligned}$$

The algorithm may or may not halt now depending on the other cuts that have been generated and the value of $\bar{z}$ obtained after solving the current master problem.

5. A COVERING DECOMPOSITION

The special decomposition algorithm described here is particularly useful for problems in which interdictions destroy arcs, thereby making them impassable.

5.1. Arc Destruction: A Caveat

Problems in which arcs are actually destroyed by interdiction can be modeled by setting each $d_k = d^+$, where d^+ is a sufficiently large "artificial delay." Unfortunately, Algorithm 1's master problem becomes weaker and harder to solve as d^+ increases in value with respect to c_k; this is clear in theory and is demonstrated empirically in Section 7.2. Thus, it is tempting to guess at a modest value for d^+ and hope that the guess is large enough. But the following example shows that being too conservative with d^+ can lead to difficulties.

Example 2. *Consider the network of Figure* 1 *and suppose that an interdictor can destroy any two arcs in that network. One optimal solution to MXSP interdicts* (s, a) *and* (s, b), *yielding a shortest s–t path length of* 20. *Let* d^+ *be the artificial delay to be added to each interdicted arc*, *and let us use Algorithm* 1 *without SVIs to solve this problem.*

Initially, *the algorithm finds the uninterdicted shortest*

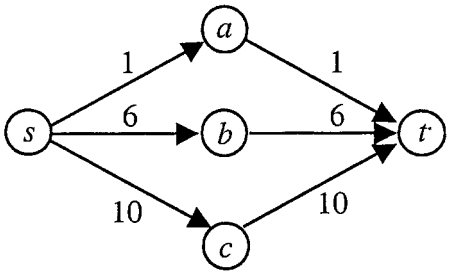

FIG. 1. Network to illustrate difficulties with artificial delays. Numbers next to arcs denote arc lengths.

path s–a–t with length 2. *Given that solution for the follower, the leader interdicts* (s, a) *and* (a, t), *and solves the master problem containing the single Benders cut* $z \leq 2 + d^+x_{sa} + d^+x_{at}$. *The solution yields the "quasi-upper bound" of* $2 + 2d^+$: *This is a "quasi-bound" because it is valid only if* $d^+ \geq 9$. *The follower then finds the shortest path after* (s, a) *and* (a, t) *are removed from the network. The solution is s–b–t, with length* 12, *and the master problem's cuts are*

$$z \leq 2 + d^+x_{sa} + d^+x_{at} \text{ from the first iteration, and}$$

$$z \leq 12 + d^+x_{sb} + d^+x_{bt} \text{ from the second iteration.}$$

There are four cases to consider now:

(a) *If* $d^+ < 5$, *the shortest path from the second iteration has length greater than the quasi-upper bound from the first iteration. We conclude that* d^+ *is too small, increase it, and return to the master problem.*

(b) *If* $5 \leq d^+ \leq 10$, *the optimal master problem objective, that is, the quasi-upper bound, is 12. Since the lower bound and quasi-upper bound match, the algorithm terminates, but with an incorrect solution. In this case, we see no way to recognize that we have chosen* d^+ *too small without solving this NP-complete problem: Does there exist a solution to MXSP with an objective value greater than* $z_{\hat{Y}}$?

(c) *If* $10 < d^+ < 18$, *the master problem interdicts both paths and has the optimal objective value* $2 + d^+$. *The third iteration of the subproblem finds the path s–c–t with length 20, which is larger than is the current quasi-upper bound. Again, we conclude that* d^+ *is too small, increase it, and return to the master problem.*

(d) *If* $d^+ \geq 18$, *the quasi-upper bound is a true upper bound and the algorithm terminates with the optimal solution.* ∎

5.2. The Basic Covering Algorithm

Case (b) above is disturbing. To address this and related issues, we offer an alternative decomposition algorithm that eliminates the delay terms from the master problem. The master problem is simply a "feasibility-seeking set-covering problem":

$$[\text{Master}(\hat{X}\hat{Y}) - 2a] \quad \text{Find} \quad \mathbf{x} \in X$$

$$\text{s.t.} \quad \tilde{\mathbf{y}}^T\mathbf{x} \geq 1 \quad \forall\, (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y} \text{ where } \tilde{\mathbf{y}} \equiv (\text{diag}(\mathbf{1} - \hat{\mathbf{x}}))\hat{\mathbf{y}}.$$

In this context, we call the constraints $\tilde{\mathbf{y}}^T\mathbf{x} \geq 1$ *covering constraints* and each such constraint simply implies that "if the interdictor wishes to force the follower to traverse a path other than $\hat{\mathbf{y}}$ then a new interdiction plan $\hat{\mathbf{x}}'$ must interdict some arc that is not interdicted by $\hat{\mathbf{x}}$ but is used by the follower in response to $\hat{\mathbf{x}}$."

The algorithm associated with [Master($\hat{X}\hat{Y}$)-2a] generates new interdiction plans until the master problem becomes infeasible, at which point we can prove that the best plan found is optimal. We refer to this algorithm as a *covering decomposition.*

Algorithm 2: A covering decomposition algorithm for MXSP.

Input: An instance of MXSP.
Output: An optimal interdiction plan $\mathbf{x}^*$.
Step 0: $\hat{X}\hat{Y} \leftarrow \emptyset$; $\underline{z} \leftarrow -\infty$; $\hat{\mathbf{x}} \leftarrow \mathbf{0}$;
Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for $\hat{\mathbf{y}}$ with objective value $z_{\hat{\mathbf{x}}}$;
$\hat{X}\hat{Y} \leftarrow \hat{X}\hat{Y} \cup (\hat{\mathbf{x}}, \hat{\mathbf{y}})$;
If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$;
Step 2: Attempt to solve [Master($\hat{X}\hat{Y}$)-2a] for feasible solution $\hat{\mathbf{x}}$;
If [Master($\hat{X}\hat{Y}$)-2a] is feasible then go to Step 1;
Step 3: $\mathbf{x}^* \leftarrow \mathbf{x}'$; Print $\mathbf{x}^*$; Stop.

Theorem 4. *Algorithm* 2 *correctly solves MXSP.*

Proof. Define $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \mathbf{c}^T\hat{\mathbf{y}} + \hat{\mathbf{x}}^TD\hat{\mathbf{y}}$ for all $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$. Now, we may view Algorithm 2 as follows: Each time the algorithm reaches Step 1, the follower responds as best possible to the current interdiction plan $\hat{\mathbf{x}}$ with one new s–t path $\hat{\mathbf{y}}$ and, implicitly, with an uninterdicted subpath $\tilde{\mathbf{y}}$. Then, in Step 2, the leader tries to find a plan that interdicts at least one arc in each of the uninterdicted subpaths generated so far. (If such a plan is found, it may or may not force the follower to traverse a path longer than the path forced by the incumbent $\mathbf{x}'$.) The algorithm terminates when the leader fails to interdict all the subpaths represented by $\hat{X}\hat{Y}$. At this point, he cannot force a worse response for the follower than any of responses seen so far, that is, $z^* \leq \max_{(\hat{\mathbf{x}},\hat{\mathbf{y}})\in\hat{X}\hat{Y}} z(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. But $\underline{z} = \max_{(\hat{\mathbf{x}},\hat{\mathbf{y}})\in\hat{X}\hat{Y}} z(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, so the incumbent solution $\mathbf{x}'$ is optimal. ∎

The correctness of Algorithm 2 can also be proved via a variant of Benders decomposition in which the master problem is not solved to optimality; see Israeli [27].

If arcs are actually destroyed by interdiction, Algorithm 2 can be applied directly with sufficiently large artificial delays representing arc destruction in the subproblem or by simply making interdicted arcs impassable. (In either case,

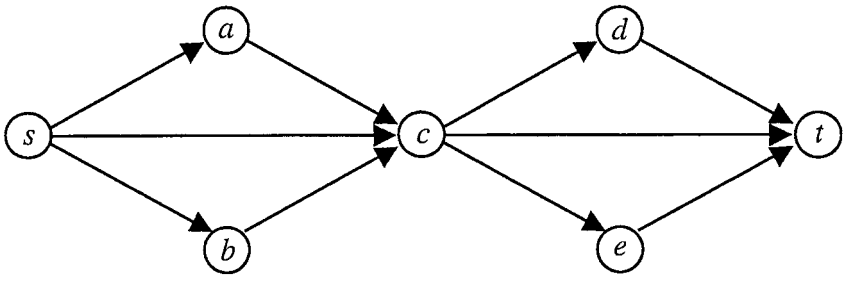

FIG. 2. Network for counterexample. If all arcs lengths are 1 and exactly two arcs can be interdicted, the optimal solution to MXSP interdicts arcs (s, c) and (c, t) which are not part of any cutset.

$\tilde{\mathbf{y}} = \hat{\mathbf{y}}$ for all follower responses $\hat{\mathbf{y}}$, unless it is possible for the interdictor to disconnect s from t, in which case the algorithm can be modified to terminate early.) In fact, when applied to such a problem, Algorithm 2 is similar to the algorithm for the "k-most-vital-arcs-problem" suggested by Malik et al. [30]. There, paths with nondecreasing lengths in the uninterdicted network are enumerated and interdiction is attempted in the network G' consisting of the union of the first l paths, for $l = 1, 2, \ldots$, until this is impossible. Essentially, an algorithm that produces the l^{th}-shortest path in the original network (e.g., [29]) replaces Step 1 in Algorithm 2. However, Malik et al. assumed that an optimal interdiction plan $\mathbf{x}^*$ for the l shortest paths must correspond to a cutset in G', and, thus, $\mathbf{x}^*$ corresponds to an easily identified, minimum-cardinality cutset in G'. Figure 2's counterexample shows that this assumption is incorrect, so the procedure of Malik et al. must be viewed as a heuristic.

We next describe some enhancements to Algorithm 2 to improve efficiency.

5.3. Heuristic Solution of the Master Problem

Since we assume that X includes only a single resource constraint $\mathbf{r}^T\mathbf{x} \leq r_0$, we can solve the following master problem to find a solution to [Master($\hat{X}\hat{Y}$)-2a]:

$$
\begin{array}{lll}
[\text{Master}(\hat{X}\hat{Y})\text{-2b}] & \min\limits_{\mathbf{x}} & \mathbf{r}^T\mathbf{x} \\
& \text{s.t.} & \tilde{\mathbf{y}}^T\mathbf{x} \geq 1 \ \ \forall \ (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y} \\
& & \text{where } \tilde{\mathbf{y}} \equiv (\text{diag}(\mathbf{1} - \hat{\mathbf{x}}))\hat{\mathbf{y}} \\
& & \\
& & \mathbf{x} \in \{0, 1\}^{|A|}.
\end{array}
$$

[Master($\hat{X}\hat{Y}$)-2b] is a standard set-covering problem (SCP), which has an optimal objective value not exceeding r_0 if and only if [Master($\hat{X}\hat{Y}$)-2a] is feasible. However, we need only solve [Master($\hat{X}\hat{Y}$)-2b] for a feasible solution $\hat{\mathbf{x}}$ satisfying $\mathbf{r}^T\hat{\mathbf{x}} \leq r_0$, and this suggests the use of fast heuristics. We use a version of the greedy heuristic for SCPs discussed by, among others, Nemhauser and Wolsey [34]. (Other SCP heuristics exist, e.g., Beasley [8] and Caprara et al. [12], but our choice is easy to implement and provides adequate performance for our test problems.) Thus, whenever we need to solve a master problem in Algorithm 2, we run the greedy heuristic on [Master($\hat{X}\hat{Y}$)-2b]. If a feasible solution, $\hat{\mathbf{x}} \in X$, is found, we proceed to Step 1 of the algorithm. If not, only then do we resort to a slower, exact, branch-and-bound algorithm.

5.4. Local Search

Algorithm 2 typically iterates faster than does Algorithm 1 because the master problems are easier to solve. This is true even when delays d_k are small. On the other hand, Algorithm 2 usually requires more iterations than does Algorithm 1 when the d_k are not too large. Another difficulty with Algorithm 2 is the lack of an upper bound, which could enable early termination with a near-optimal solution. To help overcome these two difficulties, we exploit multiple pairs $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ generated by a local-search procedure for a given $\hat{\mathbf{x}}$, where $\hat{\mathbf{y}}'$ denotes an optimal or suboptimal response to $\hat{\mathbf{x}}$. The key result is

Theorem 5. *Let $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ denote an interdiction plan $\hat{\mathbf{x}}$ and possibly suboptimal response $\hat{\mathbf{y}}'$, that is, $\hat{\mathbf{y}}'$ is a valid path given $\hat{\mathbf{x}}$, but may not be a shortest path. Then, $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ may be added to $\hat{X}\hat{Y}$ without compromising the validity of Algorithm 2 provided that $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \underline{z}$.*

Proof. This follows from the definition of $\underline{z}$. ∎

Before describing the related upper-bounding procedure, we describe a local-search process that generates the "extra pairs" $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$.

Assume that G has arc length vector $\mathbf{c} + D\hat{\mathbf{x}}$. It is well known that finding the shortest paths from s to all other nodes in G is not much more difficult than finding a single, shortest s–t path. So, we first compute and encode the former paths using a standard "shortest-path tree" T and "predecessor function" (e.g., Ahuja et al. [1, pp. 106–107]). Now, let $P(t) = (s, i_1, i_2, \ldots, i_n, t)$ denote the shortest s–t path in T and let $P(j)$ be the shortest s–j path. For every node $i_m \in \{i_1, i_2, \ldots, i_n\}$ and for every arc $(j, i_m) \in A$, $j \notin P(t)$, the procedure **Local_Search**$(\hat{\mathbf{x}}, T)$ then builds the path $(P(j), i_m, i_{m+1}, \ldots, i_n, t)$, represented by its incidence vector $\hat{\mathbf{y}}'$ and returns the corresponding pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$.

We omit pseudocode for this procedure and other ancillary procedures whose implementations are straightforward.

Every pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ returned from **Local_Search**$(\hat{\mathbf{x}}, T)$ with $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \underline{z}$ is introduced into $\hat{X}\hat{Y}$ to yield one more subpath to be covered in the master problem. If $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') > \underline{z}$, $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ is stored in a special set $\hat{X}\hat{Y}^+$. Later, after updating $\underline{z}$ in succeeding iterations, any $(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}\hat{Y}^+$ satisfying $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \underline{z}$ is moved into $\hat{X}\hat{Y}$. Based on Theorem 6 below, the paths contained in $\hat{X}\hat{Y}^+$ can also be used to verify a hypothesized upper bound, for example, $\underline{z} + \varepsilon$ for some $\varepsilon > 0$, which then allows us to solve for ε-optimal solutions.

Theorem 6. *Let $\hat{X}\hat{Y}_{\bar{z}}$ be defined such that $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \bar{z}$ for all $(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}\hat{Y}_{\bar{z}}$, where $\bar{z}$ is a hypothesized upper bound on z^*. Then, if [Master($\hat{X}\hat{Y}_{\bar{z}}$)-2b] is infeasible, $z^* \leq \bar{z}$, that is, the upper bound is valid.*

Proof. Let $\hat{X}Y'_{\bar{z}}$ denote the set of *all* interdiction plans and possible optimal and nonoptimal responses $(\hat{\mathbf{x}}, \hat{\mathbf{y}}')$ such that $z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \bar{z}$. If $z^* > \bar{z}$, we can feasibly interdict all uninterdicted subpaths $\tilde{\mathbf{y}}'$ derived from $(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}Y'_{\bar{z}} \supseteq \hat{X}\hat{Y}_{\bar{z}}$. But, under the assumption that [Master($\hat{X}\hat{Y}_{\bar{z}}$)-2b] is infeasible, this is impossible. Therefore, $z^* \leq \bar{z}$. ∎

So, given valid lower bound $\underline{z}$ derived from some $\hat{\mathbf{x}}$, we can check for ε-optimality of $\hat{\mathbf{x}}$ by (*i*) hypothesizing an upper bound $\bar{z} = \underline{z} + \varepsilon$ for some $\varepsilon > 0$, (*ii*) defining $\hat{X}\hat{Y}_{\bar{z}} = \hat{X}\hat{Y} \cup \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}\hat{Y}^+ | z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \underline{z} + \varepsilon\}$, and, (*iii*) checking feasibility of [Master($\hat{X}\hat{Y}_{\bar{z}}$)-2b]. If it is infeasible, we know that $z^* \leq \underline{z} + \varepsilon$ and $\hat{\mathbf{x}}$ is ε-optimal. When $\varepsilon > 0$ is defined, we generate and solve [Master($\hat{X}\hat{Y}_{\bar{z}}$)-2b] periodically within Algorithm 2 and find that the extra computational overhead is more than offset by a reduced number of iterations.

5.5 Other Improvements

Algorithm 2 adds two procedures that often improve efficiency. One procedure. **Lift**$(\hat{X}\hat{Y}, \underline{z})$, uses the current lower-bound $\underline{z}$ and information on arcs with modest delays to lift the covering constraints based on $\hat{X}\hat{Y}$, just as Type-II SVIs are lifted via Theorem 3: Covering constraints are essentially Type-II SVIs. [The procedure returns the "lifted interdiction/response pairs" derived from $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$ rather than the lifted constraints themselves.] Additionally, we generalize the procedure and call **Lift**$(\hat{X}\hat{Y}, \underline{z} + \varepsilon)$, for $\varepsilon \geq 0$ to create *ε-supervalid inequalities* (ε-SVIs). These inequalities will not eliminate all ε-optimal solutions unless the incumbent solution is already ε-optimal.

Another procedure, not shown, removes redundant rows from the master problem just as such rows are removed in set-covering problems (e.g., Garfinkel and Nemhauser [19, pp. 302–304] and Taha [38, pp. 316–332]). We find it more efficient to do this before sending the master problem to the solver, although the solver can detect and remove some or all of these rows by itself.

5.6. The Enhanced Covering Decomposition

Algorithm 2, with enhancements, is outlined below. The actual implementation reorders certain computations for efficiency's sake.

Algorithm 2E: Enhanced covering decomposition for MXSP.

Input: An instance of MXSP and optimality tolerance $\varepsilon \geq 0$.

Output: An ε-optimal interdiction plan $\mathbf{x}^\varepsilon$ for MXSP.

Step 0: $\hat{X}\hat{Y} \leftarrow \emptyset$; $\hat{X}\hat{Y}^+ \leftarrow \emptyset$; $\underline{z} \leftarrow -\infty$; $\hat{\mathbf{x}} \leftarrow \mathbf{0}$;

Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for shortest-path tree T and objective $z_{\hat{\mathbf{x}}}$;
If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$;
$\hat{X}\hat{Y}^+ \leftarrow \hat{X}\hat{Y}^+ \cup$ **Local_Search**$(\hat{\mathbf{x}}, T)$;
$\hat{X}\hat{Y}' \leftarrow \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}\hat{Y}^+ | z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \leq \underline{z} + \varepsilon\}$;
$\hat{X}\hat{Y}^+ \leftarrow \hat{X}\hat{Y}^+ - \hat{X}\hat{Y}'$;
$\hat{X}\hat{Y}' \leftarrow$ **Lift**$(\hat{X}\hat{Y}', \underline{z} + \varepsilon)$;

Step 2: Attempt to solve [Master($\hat{X}\hat{Y}'$)-2b] for feasible solution $\hat{\mathbf{x}}$.
If [Master($\hat{X}\hat{Y}'$)-2b] is feasible, then go to Step 1.

Step 3: $\mathbf{x}^\varepsilon \leftarrow \mathbf{x}'$; Print $\mathbf{x}^\varepsilon$; Stop.

6. ALGORITHM 1 REVISITED

Algorithm 2's enhancements can also improve Algorithm 1, as can a few other techniques described here. Algorithm 1E is the result.

6.1. SVIs and the Master Problem

Algorithm 1 works best when all optimal responses $\hat{\mathbf{y}}$ for a given $\hat{\mathbf{x}}$ are generated and the corresponding cuts added to the master problem, but suboptimal responses $\hat{\mathbf{y}}$ can also be used. They create "slack Benders cuts" that lie strictly above the function $z_{\mathbf{x}}$ at the points $\mathbf{x} = \hat{\mathbf{x}} \in X$ that generate these cuts. Slack cuts often tighten the master problem away from $\hat{\mathbf{x}}$ and thereby reduce enumeration, so we use **Local_Search** just as in Algorithm 2E to generate the necessary paths. (We could also use the Byers and Waterman algorithm, but find it overly sensitive to parameter settings in our tests.)

The master problem for Algorithm 1E includes both Type-I and Type-II SVIs. To use both types, we maintain the set $\hat{X}\hat{Y}$ as defined in Algorithm 1 and second set $\hat{X}\hat{Y}'$ of interdiction/response pairs that are augmented and lifted by **Local_Search** and **Lift**, respectively. The right-hand sides b for Type-I SVIs are defined as functions by combining Theorem 3 and Corollary 1 as follows: For each $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$, order the $d_k\hat{y}_k > 0$ such that $d_{k_1}\hat{y}_{k_1} \geq d_{k_2}\hat{y}_{k_2} \cdots \geq d_{k_L}\hat{y}_{k_L}$ and define $d_{k_0}\hat{y}_{k_0} = \infty$, $d_{k_{P+1}}\hat{y}_{k_{P+1}} = 0$ and

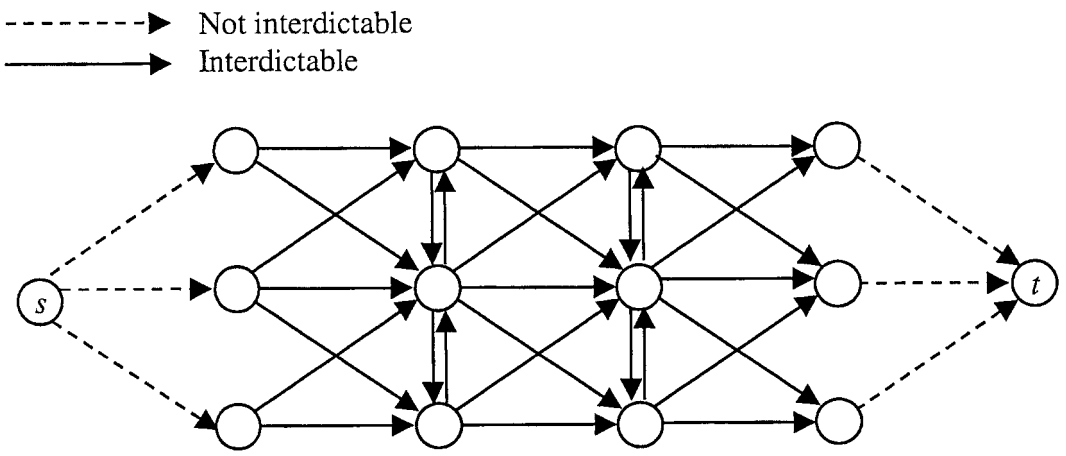

FIG. 3. Example topology of a 3 × 4 test network.

$b(\mathbf{c}^T\hat{\mathbf{y}}, \mathbf{d}, \underline{z}) \equiv p + 1$ if

$$\mathbf{c}^T\hat{\mathbf{y}} + \sum_{l=1}^{p} d_{kl}\hat{y}_{kl} \le \underline{z} < \mathbf{c}^T\hat{\mathbf{y}} + \sum_{l=1}^{p+1} d_{kl}\hat{y}_{kl}.$$

Just as Type-II SVIs are generalized to ε-SVIs, the Type II SVIs are also, simply by using $b(\mathbf{c}^T\hat{\mathbf{y}}, \mathbf{d}, \underline{z} + \varepsilon)$. The master problem for Algorithm 1E is then

$$[\text{Master}(\hat{X}\hat{Y}, \hat{X}\hat{Y}', \underline{z}, \varepsilon)\text{-1b}]$$

$$z_{\hat{X}\hat{Y}}^{\wedge} = \max_{\mathbf{x}\in X} z$$

$$\text{s.t.} \quad z \le \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}} \quad \forall\, \hat{\mathbf{y}} \in \hat{Y}$$

$$\hat{\mathbf{y}}^T\mathbf{x} \ge b(\mathbf{c}^T\hat{\mathbf{y}}, \mathbf{d}, \underline{z} + \varepsilon) \;\; \forall \;\; (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}$$

$$\tilde{\mathbf{y}}^T\mathbf{x} \ge 1 \;\; \forall \;\; (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \hat{X}\hat{Y}' \text{ where } \tilde{\mathbf{y}} \equiv (\text{diag}(\mathbf{1} - \hat{\mathbf{x}}))\hat{\mathbf{y}}.$$

6.2. Algorithm 1E

We outline Algorithm 1E as follows:

Algorithm 1E: The Benders decomposition Algorithm 1, enhanced.

Input: An instance of MXSP and optimality tolerance $\varepsilon \ge 0$.

Output: An ε-optimal interdiction plan $\mathbf{x}^{\varepsilon}$ for MXSP.

Step 0: $\hat{X}\hat{Y} \leftarrow \emptyset$; $\hat{X}\hat{Y}^+ \leftarrow \emptyset$; $\underline{z} \leftarrow -\infty$; $\bar{z} \leftarrow \infty$; $\hat{\mathbf{x}} \leftarrow \mathbf{0}$;

Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for shortest-path tree T and objective $z_{\hat{\mathbf{x}}}$.
If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$;
$\hat{X}\hat{Y}^+ \leftarrow \hat{X}\hat{Y}^+ \cup$ **Local_Search**($\hat{\mathbf{x}}$, T);
$\hat{X}\hat{Y}' \leftarrow \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \in \hat{X}\hat{Y}^+ | z(\hat{\mathbf{x}}, \hat{\mathbf{y}}') \le \underline{z} + \varepsilon\}$;
$\hat{X}\hat{Y}^+ \leftarrow \hat{X}\hat{Y}^+ - \hat{X}\hat{Y}'$;
$\hat{X}\hat{Y}' \leftarrow$ **Lift**($\hat{X}\hat{Y}'$, $\underline{z} + \varepsilon$);

Step 2: Attempt to solve [Master($\hat{X}\hat{Y}$, $\hat{X}\hat{Y}'$, $\underline{z}$, ε)-1b] for optimal solution $\hat{\mathbf{x}}$ and objective $z_{\hat{X}\hat{Y}}$;
If [Master($\hat{X}\hat{Y}$, $\hat{X}\hat{Y}'$, $\underline{z}$, ε)-1b] is infeasible, then go to Step 3;
$\bar{z} \leftarrow z_{\hat{X}\hat{Y}}$; If $\bar{z} - \underline{z} > \varepsilon$ then go to Step 1;

Step 3: $\mathbf{x}^{\varepsilon} \leftarrow \mathbf{x}'$; Print $\mathbf{x}^{\varepsilon}$; Stop.

In addition to using SVIs, solution times for Algorithm 1E are often reduced by not solving the master problem to optimality. This variant of Benders decomposition (e.g., [20]) will converge provided that (*i*) a suboptimal integer master-problem solution $\hat{\mathbf{x}}$ is accepted only if the corresponding objective value z' satisfies $z' > \underline{z}$ and (*ii*) we update $\bar{z}$ only when the master problem is solved to optimality or through the master problem's global upper bound.

Another useful technique for Algorithm 1E is: Do not use Benders cuts in early iterations. Instead, we heuristically solve Algorithm 2's set-covering master problem to find new solutions $\hat{\mathbf{x}}$ until the heuristic fails. Then, we switch to the full master problem [Master($\hat{X}\hat{Y}$, $\hat{X}\hat{Y}'$, $\underline{z}$, ε)-1b]. If this problem is infeasible, or if $\bar{z} - \underline{z} \equiv z_{\hat{X}\hat{Y}} - \underline{z} \le \varepsilon$, the algorithm halts. Otherwise, it proceeds using the full master problem.

7. COMPUTATIONAL EXPERIENCE

7.1. Test Problems and Environment

We test our algorithms on a set of directed grid networks of different size and with randomly generated arc attributes (Fig. 3):

1. There is one source node s and one sink node t.
2. There are $m \times n$ "transshipment nodes" arranged in a grid of m rows and n columns.

TABLE 1. Test problem statistics.

Problem	$m \times n$	a	c	d	r	r_0
1	10×10	396	10	10	5	20
2						30
3						40
4						50
5	12×8	370				25
6	8×12	382				
7	14×7	370				
8	7×14	391				
9	15×10	611				
10	10×15	626				
11	20×12	1,018				
12	12×20	1,042				
13	7×7	188	10	$d^+ = 5$ or 10	1	5
14	7×7					10
15	8×8	238				5
16	8×8					10
17	9×9	312				5
18	9×9					10
19	12×12	594				5
20	12×12					10

Blank table cells repeat values from cells above. Problems 13–20 are k-most-vital-arcs problems for $k = 5$ or $k = 10$ and with artificial delays of $d^+ = 5$ or $d^+ = 10$. ($d^+ = 10$ is "sufficiently large;" $d^+ = 5$ may not be.) $m \times n$, Grid of nodes has m rows and n columns; a, no. interdictable arcs; c, arc costs c_k are integers in the range $[1, c]$; d, arc delays d_k are integers in the range $[1, d]$, except for problems 13–20; r, arc interdiction resources r_k are integers in the range $[1, r]$; r_0, total interdiction resource available.

3. There is a 0-length arc from s to each transshipment node in the first column, and there is a 0-length arc from each transshipment node in the last column to t. None of these $2m$ arcs may be interdicted.
4. An arc exists from each node in row r and column c, that is, in grid position (r, c), to the nodes in positions $(r + 1, c)$, $(r - 1, c)$, $(r, c + 1)$, $(r + 1, c + 1)$, and $(r - 1, c + 1)$ provided that (*i*) a node exists in the particular position, and (*ii*) these are not vertical arcs in the first or last columns, which would be superfluous. All these arcs are interdictable.
5. The basic data for each network are
   (a) m and n [so that there are $a = (n - 2)(5m - 4) + 3m - 2$ interdictable arcs, and $2m$ noninterdictable arcs];
   (b) Maximum arc length c, maximum arc delay d, and maximum resource r required to interdict an arc; all these values are positive integers;
   (c) r_0, the total, positive integer, interdiction resource available.
6. The randomly generated, integer data for each interdictable arc k are c_k, d_k, and r_k, uniformly distributed on $[1, c]$, $[1, d]$, and $[1, r]$, respectively.

Table 1 provides summary statistics for the test problems.

We program our algorithms in C using the CPLEX version 6.5 callable library [26] for exact solution of master problems, when needed, and for direct solutions of the MIP [MXSP-D]. Extensive testing indicates that default solver options are best except that "variable selection strategy" is set to "branch based on pseudoreduced cost" in all cases. Computation is performed on an IBM RS-6000 Model 595 workstation with 512 megabytes of RAM. Run times shown are averages across 10 networks of identical topology, but with different random arc attributes.

7.2. Basic Results

Results for problems 1–4 (Table 2) compare solution methods as total interdiction resource changes. All problems are solved to optimality here, that is, $\varepsilon = 0$.

Overall, Algorithms 1E and 2E are competitive with each other and are the fastest of the four procedures by a wide margin. We would also like to point out that

1. Varying arc attributes, while holding the network topology and algorithm fixed, can lead to widely varying solution times for all algorithms: The fastest run among the 10 instances of a problem may be 20 times faster than the slowest. We are still investigating ways to reduce run times for the longer-running problems.
2. All algorithms are sensitive to total interdiction resource r_0. Run times typically increase rapidly as r_0 increases from a small value but then start decreasing for sufficiently large values, beyond those displayed here. This

TABLE 2. Computational results for a network with $100 = 10 \times 10$ transshipment nodes ($a = 396$), $c = 10$, $d = 10$, and $r = 5$; $\varepsilon = 0$.

		[MXSP-D]		Algorithm 1				Algorithm 1E				Algorithm 2E			
Problem	r_0	T	S	T	S	N	P	T	S	N	P	T	S	N	P
1	20	60	36	219	139	53	306	3	3	21	334	4	4	22	341
2	30	651	757	(5)	—	—	—	26	31	34	671	39	51	37	693
3	40	(4)	—	(0)	—	—	—	135	162	52	1101	191	277	54	1167
4	50	(2)	—	(0)	—	—	—	(9)	—	—	—	1349	1781	77	1656

"[MXSP-D]" gives results for the direct, branch-and-bound solution of [MXSP-D]. Nos. in parentheses are the no. problems, out of 10, solved to optimality in under 3600 CPU s. "—" indicates "not applicable" because at least one of the 10 instances could not be solved in the allotted time. T, Average run time, in CPU seconds, for 10 problems; S, standard deviation of T in CPU seconds; N, average number of iterations for the algorithm; P, average number of constraints in master problem when algorithm terminates.

TABLE 3. Computational results for networks with different row-to-column ratios, with $r_0 = 25$, $c = 10$, $d = 10$, and $r = 5$; $\varepsilon = 0$.

			[MXSP-D]		Algorithm 1E				Algorithm 2E			
Problem	$m \times n$	a	T	S	T	S	N	P	T	S	N	P
5	12×8	370	296	369	1	1	15	186	1	1	16	186
6	8×12	382	286	326	72	95	40	949	71	86	40	935
7	14×7	370	131	139	1	1	13	126	1	1	14	127
8	7×14	391	298	387	222	301	54	1668	409	515	58	1768
9	15×10	611	(8)	—	3	3	18	305	3	3	20	311
10	10×15	626	(8)	—	407	475	54	1886	722	965	58	2044
11	20×12	1018	(3)	—	11	12	21	438	7	7	22	438
12	12×20	1042	(0)	—	(3)	—	—	—	(1)	—	—	—

The total no. interdictable arcs is a. See Table 2 for other definitions.

makes sense since increasing interdiction resource allows more combinations of arcs to be interdicted, up to a point. Then, for sufficiently large r_0, all or nearly all arcs can be interdicted and the problem becomes easy.

3. We find that branch and bound does not solve any of the problems much faster even if it identifies an optimal solution immediately and simply needs to prove optimality. For instance, results for problems 3 and 4 do not materially change if an optimal solution is known as the branch-and-bound procedure begins, and solution times improve less than 10% for problems 1 and 2. Evidently, the LP-based bounds improve only slowly as the enumeration tree expands.

Table 3 displays results from runs that explore the sensitivity of the algorithms to a network's row-to-column ratio. Results for Algorithm 1 are omitted because, as in Table 1, they are much worse than for Algorithm 1E and even [MXSP-D]. The decomposition algorithms appear to prefer "tall networks" like the 12×8 network over "long networks" like the 7×14 network. This tendency may result from the greater number of paths in a long network and the potentially greater number of constraints in the corresponding master problems and because there is a positive correlation between the number of potential constraints and the actual number needed to generate a tight master problem.

We next explore how allowing $\varepsilon > 0$ may shorten run times for Algorithms 1E and 2E and, for the sake of comparison, for direct solutions of [MXSP-D]. Improved run times result because (*i*) branch-and-bound enumeration trees can be trimmed for the [MXSP-D] MIP and for Algorithm 1E's master problem, (*ii*) termination tests in the decomposition algorithms are more easily satisfied, and (*iii*) the SVIs in the decomposition algorithms may be more aggressively lifted or tightened as ε-SVIs.

We repeat the computational tests on five of the most difficult problems from Tables 2 and 3 while allowing a 5% optimality gap. (Optimality gaps were described previously in absolute terms. Here, an allowable gap of g % indicates that $100(\bar{z} - \underline{z})/\underline{z} \leq g$). Results are displayed in Table 4. Indeed, all times are reduced with the relaxed optimality criterion, and even branch and bound becomes more competitive.

Table 5 compares several algorithms for the variant of MXSP in which interdiction destroys arcs, making them impassable. The cost of interdiction is fixed to one unit of resource per arc, so we are actually solving the k-most-vital-arcs problem, for $k = 5$ and $k = 10$, in four different network topologies. [MXSP-D] and Algorithm 1 use artificial delays of $d^+ = 5$ and $d^+ = 10$. However, $d^+ = 5$ is often too small and yields invalid solutions, while $d^+ = 10$ is sufficiently large but results in long run times.

Both branch and bound and Algorithm 1 perform poorly; Algorithm 2 is the fastest by a substantial margin. Results for Algorithm 1E are omitted because that algorithm is uniformly slower than is Algorithm 2 here. (We can view Algorithm 1E as Algorithm 2 with Benders cuts added in the master problem. But, these cuts are weak and uninformative when all delays are identical and/or large, and thus the cuts only serve to hinder solution of Algorithm 2's master problem.) Results for Algorithm 2E are omitted because, when d^+ is large, lifting and local search are ineffective for that algorithm's master problem.

As a side note, it is interesting to see that "plain" Benders decomposition, Algorithm 1, is faster than is branch and

TABLE 4. Computational tests on the hardest problems from Tables 2 and 3, with a 5% optimality gap allowed.

	[MXSP-D]		Algorithm 1E		Algorithm 2E	
Problem	T (5%)	T (opt)	T (5%)	T (opt)	T (5%)	T (opt)
3	(9)	(4)	104	135	74	191
4	1378	(2)	604	(9)	799	1349
8	27	298	77	222	201	409
10	301	(8)	113	407	256	2044
12	(8)	(0)	(8)	(3)	(3)	(1)

Times "T (5%)" are the results in CPU seconds averaged over the 10 test problems. For comparison, "T (opt)" gives the average solution times in seconds, taken from Tables 2 and 3, when the same problems are solved to optimality. As before, run times are limited to 3600 seconds, and the nos. in parentheses indicate the no. problems that can be solved under that limit.

TABLE 5. Results for the k-most-vital-arcs problems.

					[MXSP-D]		Algorithm 1		
					$d^+ = 5$	$d^+ = 10$	$d^+ = 5$	$d^+ = 10$	Algorithm 2
Problem	k	$m \times n$	a	OK	T	T	T	T	T
13	5	7 × 7	188	8	6	23	1	5	0.3
14	10	7 × 7	188	2	71	1551	66	(7)	7.1
15	5	8 × 8	238	10	15	84	2	11	0.4
16	10	8 × 8	238	7	161	(2)	222	(0)	10.0
17	5	9 × 9	312	4	24	162	5	26	0.6
18	10	9 × 9	312	10	644	(0)	977	(0)	44.7
19	5	12 × 12	594	6	93	811	8	68	2.1
20	10	12 × 12	594	9*	(6)	(0)	(5)	(0)	446.9

Neither branch and bound nor Algorithm 1 solve all problems correctly when d^+ is too small, that is, when $d^+ = 5$. All problems can, theoretically, be solved correctly by those techniques when $d^+ = 10$, but not necessarily within the 3600-second time limit; nos. in parentheses show the no. solved under that limit. Algorithm 2 does not depend on the artificial delay d^+ and is the fastest of the three techniques tested. The value marked by "*" was determined by increasing the time limit for [MXSP-D] to 10,000 seconds. k, Total no. arcs the interdictor may interdict; d^+, artificial delay; T, average CPU time to solve 10 problem instances; OK, no. problems that solve correctly when $d^+ = 5$.

bound on many of these problems. The Benders master problem is no stronger than is [MXSP-D], so this fact must result from quick solutions of the shortest-path subproblems.

7.3. SVIs to Speed Direct Solutions

This section explores the use of SVIs to speed the direct solution of the MIP [MXSP-D] via branch and bound: We heuristically generate a set of SVIs, add them to the MIP, and then launch the branch-and-bound code. Integer cutting planes have been used in this fashion to help solve general MIPs by branch and bound, starting with Dantzig et al. [16], and continuing to today's commercial solvers. This use is not as sophisticated as "branch and cut" where cutting planes are integrated into the branch-and-bound procedure (e.g., Wolsey [44] pp. 157–160), but success with the simpler technique should bode well for more sophisticated applications. Indeed, the topic of SVIs warrants a full research paper on its own; we only intend to illustrate its potential.

To generate the SVIs, we run Algorithm 2 and heuristically solve the set-covering master problem until the heuristic fails. Each set-covering solution, together with **Local_Search**, yields multiple Type-I SVIs. These are tightened, as appropriate, using information about the incumbent solution's value and corresponding Benders cut coefficients (which are generated and saved, but used only for this purpose).

Table 6 displays results for [MXSP-D] with and without SVIs and, for comparison, results for Algorithm 2E. Dramatic improvements are obtained using SVIs in conjunction with the MIP. In fact, branch and bound plus SVIs performs better than does Algorithm 2E in two instances. Improve-

TABLE 6. Computational results for [MXSP-D] with SVIs added.

	[MXSP-D]					[MXSP-D] + SVIs				Algorithm 2E	
Problem	cons	vars	T	S	enum	SVIs	T	S	enum	T	S
1	418	518	60	36	3992	367	12	12	66	4	4
2			651	757	47,417	683	83	100	768	39	51
3			(4)	—	>159K	1158	584	890	3470	191	277
4			(2)	—	>212K	1788	(8)	—	>5221	1349	1781
5	396	492	296	369	22,622	254	8	5	24	1	1
6	400	496	286	326	19,599	1304	87	40	318	57	69
7	396	498	131	139	9822	156	2	2	2	1	1
8	407	505	298	387	19,911	1504	314	234	2110	409	515
9	643	793	(8)	—	>87K	340	24	12	65	3	3
10	648	798	(8)	—	>66K	1815	922	799	6957	722	965
11	991	1191	(3)	—	>86K	438	92	93	569	7	7
12	1022	1238	(0)	—	>97K	3456	(2)	—	>4755	(1)	—

The same 10 instances of each problem that generate the results in Tables 2 and 3 are used here. Results for [MXSP-D] and Algorithm 2E are provided for comparison. T, average CPU time to solve 10 problem instances; S, standard deviation of T; cons, no. constraints in [MXSP-D]; var, no. variables in [MXSP-D]; enum, nodes in the branch-and-bound enumeration tree; SVIs, no. SVIs (new constraints) added to [MXSP-D].

ments are less dramatic when the number of SVIs becomes large, which occurs in the longer networks.

There is a clear trade-off between the reduced enumeration achieved with SVIs and increased solution times for the LP relaxations of the resulting, larger MIPs: Compare the huge reduction in the size of branch-and-bound trees to the lesser improvement in run times, especially with the longer networks. Further research is warranted.

8. CONCLUSIONS

This paper has defined a shortest-path network-interdiction problem, MXSP, on a directed network. The objective of an "interdictor" is to interdict (attack) network arcs, using limited resources, in order to maximize the length of a shortest path between two specified nodes, s and t. Interdiction of an arc increases its effective length or destroys the arc, making it impassable. The ultimate purpose of the interdiction is to delay a "network user" from traveling between s and t.

We have formulated MXSP as a standard MIP, but develop special decomposition algorithms that solve test problems much more quickly than does LP-based branch and bound applied to the MIP. Straightforward Benders decomposition performs poorly, but special techniques, including "supervalid inequalities" (SVIs) for the master problem, improve efficiency significantly. A "covering decomposition," which uses a special set-covering master problem, is particularly useful for problems in which interdiction completely destroys arcs or adds large delays.

An SVI generalizes the concepts of valid inequality and the integer cutting plane by allowing feasible and even optimal solutions to be eliminated. We have also added SVIs to the MXSP MIP, just as valid inequalities might be added, and have demonstrated substantial speed-ups in branch-and-bound solutions. It will be interesting to investigate the use of SVIs for solving other MIPs.

Our decomposition techniques, especially Algorithm 2 and its variants, can be generalized to other network-interdiction problems and to "system interdiction problems." For instance, we may wish to disrupt a segment of an adversary's economy by attacking key components of that economy, for example, power generators and weapons production facilities. We simply require that (*i*) the interdictor's choices be discrete and resource-constrained, (*ii*) interdictions disrupt system activities in a well-defined manner, and (*iii*) given a specific set of interdictions, it is possible to determine an optimal response by the system user.

We are already applying these generalizations to a trilevel "system-defense problem," in particular, the problem of hardening a road network against attack. In this problem, the network user has limited resources to harden (make invulnerable) arcs in his/her network against potential interdiction. The interdictor observes which components of his/her adversary's network have been hardened and solves a shortest-path interdiction problem to determine the best interdiction plan. The network user responds with a new hardening plan, and the decomposition proceeds as described in this paper. In essence, the network user interdicts the interdictor. Our solution approach uses a nested decomposition scheme. We are also pursuing issues of uncertainty in interdiction.

Acknowledgments

The authors thank Gerald Brown, Robert Dell, Karla Hoffman, Alexandra Newman, Javier Salmerón, Alan Washburn, and two anonymous referees for numerous suggestions and edits that have improved this paper.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network Flows, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] G. Anandalingam and V. Aprey, Multi-level programming and conflict resolution, Eur J Oper Res 51 (1991), 233–247.

[3] A. Atamtürk, G.L. Nemhauser, and M.W.P. Savelsbergh, Conflict graphs in solving integer programming problems, Eur J Oper Res 121 (2000), 40–55.

[4] E. Balas and E. Zemel, Facets of the knapsack polytope from minimal covers, SIAM J Appl Math 34 (1978), 119–148.

[5] M.O. Ball, B.L. Golden, and R.V. Vohra, Finding the most vital arcs in a network, Oper Res Lett 8 (1989), 73–76.

[6] J.F. Bard and J.T. Moore, A branch and bound algorithm for the bilevel programming problem, SIAM J Sci Stat Comput 11 (1990), 281–292.

[7] J.F. Bard and J.T. Moore, An algorithm for the discrete bilevel programming problem, Naval Res Log 39 (1992), 419–435.

[8] J.E. Beasley, A Lagrangian heuristic for set-covering problems, Naval Res Log 37 (1990), 151–164.

[9] O. Ben-Ayed, Bilevel linear programming, Comput Oper Res 20 (1993), 485–501.

[10] J.F. Benders, Partioning procedures for solving mixed integer variables programming problems, Num Math 4 (1962), 238–252.

[11] T.H. Byers and M.S. Waterman, Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming, Oper Res 32 (1984), 1381–1384.

[12] A. Caprara, M. Fischetti, and P. Toth, A heuristic algorithm for the set covering problem, Integer programming and combinatorial optimization, Lecture Notes on Computer Science 1084, W.H. Cunningham, S.T. McCormick, and M. Queyranne (Editors), Springer-Verlag, Berlin, 1996, pp. 72–81.

[13] H.W. Corely and D.Y. Shaw, Most vital links and nodes in weighted networks, Oper Res Lett 1 (1982), 157–160.

[14] K.J. Cormican, Computational methods for deterministic and stochastic network interdiction problems, Masters Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 1995.

[15] K.J. Cormican, D.P. Morton, and R.K. Wood, Stochastic network interdiction, Oper Res 46 (1998), 184–197.

[16] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson, Solution of a large scale traveling salesman problem, Oper Res 2 (1954), 393–410.

[17] J.E. Falk and K.R. Hoffman, A successive underestimation method for concave minimization problems, Math Oper Res 1 (1976), 251–259.

[18] D.R. Fulkerson and G.C. Harding, Maximizing the minimum source-sink path subject to a budget constraint, Math Program 13 (1977), 116–118.

[19] R.S. Garfinkel and G.L. Nemhauser, Integer programming, Wiley, New York, 1972.

[20] A.M. Geoffrion and G.W. Graves, Multicommodity distribution system design by Benders decomposition, Mgmt Sci 20 (1974), 822–844.

[21] P.M. Ghare, D.C. Montgomery, and T.M. Turner, Optimal interdiction policy for a flow network, Naval Res Log Q 18 (1971), 37–45.

[22] B.L. Golden, A problem in network interdiction, Naval Res Log Q 25 (1978), 711–713.

[23] M. Grötschel, C.L. Monma, and M. Stoer, Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints, Oper Res 40 (1992), 309–330.

[24] P. Hansen, B. Jaumard, and G. Savard, New branch-and-bound rules for linear bi-level programming, SIAM J Sci Stat Comput 13 (1992), 1194–1217.

[25] J.N. Hooker, Logic-based methods for optimization, Principles and practice of constraint programming, Lecture Notes in Computer Science 874, A. Borning (Editor), Springer-Verlag, Berlin, 1994, pp. 336–349.

[26] ILOG 1999, ILOG CPLEX 6.5, Reference Manual, ILOG, S.A., France.

[27] E. Israeli, System interdiction and defense, Ph.D. Dissertation, Operations Research Department, Naval Postgraduate School, Monterey, CA, 1999.

[28] J.J. Judice and A.M. Faustino, A sequential LCP method for bilevel linear programming, Ann Oper Res 34 (1992), 89–106.

[29] N. Katoh, T. Ibaraki, and H. Mine, An efficient algorithm for the k shortest simple paths, Networks 12 (1982), 411–427.

[30] K. Malik, A.K. Mittal, and S.K. Gupta, The k most vital arcs in the shortest path problem, Oper Res Lett 8 (1989), 223–227.

[31] A.W. McMasters and T.M. Mustin, Optimal interdiction of a supply network, Naval Res Log Q 17 (1970), 261–268.

[32] D. Medhi, A unified approach to network survivability for teletraffic networks: Models, algorithms and analysis, IEEE Trans Commun 42 (1994), 534–548.

[33] J.T. Moore and J.F. Bard, The mixed integer linear bilevel programming problem, Oper Res 38 (1990), 911–921.

[34] G.L. Nemhauser and L.A. Wolsey, Integer and combinatorial optimization, Wiley, New York, 1988.

[35] H. Önal, A modified simplex approach for solving bi-level programming problems, Eur J Oper Res 67 (1993), 126–135.

[36] M. Padberg, A note on zero-one programming, Oper Res 23 (1975), 833–837.

[37] M. Simaan and J.B. Cruz, On the Stackelberg strategy in nonzero-sum games, J Optim Theory Appl 11 (1973), 533–555.

[38] H.A. Taha, Integer programming, Academic Press, New York, 1975.

[39] H. Vaish and C.M. Shetty, A cutting plane algorithm for the bilinear programming problem, Naval Res Log Q 24 (1977), 83–94.

[40] L. Vicente, G. Savard, and J. Judice, Discrete linear bi-level programming problem, J Optim Theory Appl 89 (1996), 597–614.

[41] A. Washburn and K. Wood, Two-person zero-sum games for network interdiction, Oper Res 43 (1994), 243–251.

[42] U.P. Wen and Y.H. Yang, Algorithms for solving the mixed integer two-level linear programming problem, Comput Oper Res 17 (1990), 133–142.

[43] P.S. Whiteman, Improving single strike effectiveness for network interdiction, Mil Oper Res 4 (1999), 15–30.

[44] L.A. Wolsey, Integer programming, Wiley, New York, 1998.

[45] R.K. Wood, Deterministic network interdiction, Math Comput Mod 17 (1993), 1–18.