INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Solving the Bi-Objective Maximum-Flow Network-Interdiction Problem

Johannes O. Royset, R. Kevin Wood,

To cite this article:
Johannes O. Royset, R. Kevin Wood, (2007) Solving the Bi-Objective Maximum-Flow Network-Interdiction Problem. INFORMS Journal on Computing 19(2):175-184. https://doi.org/10.1287/ijoc.1060.0191

INFORMS Journal on Computing
Vol. 19, No. 2, Spring 2007, pp. 175–184
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1902 | 0175

DOI 10.1287/ijoc.1060.0191

Solving the Bi-Objective Maximum-Flow Network-Interdiction Problem

Johannes O. Royset, R. Kevin Wood
Operations Research Department, Naval Postgraduate School, Monterey, California 93943, USA
{joroyset@nps.edu, kwood@nps.edu}

We describe a new algorithm for computing the efficient frontier of the "bi-objective maximum-flow network-interdiction problem." In this problem, an "interdictor" seeks to interdict (destroy) a set of arcs in a capacitated network that are Pareto-optimal with respect to two objectives, minimizing total interdiction cost and minimizing maximum flow. The algorithm identifies these solutions through a sequence of single-objective problems solved using Lagrangian relaxation and a specialized branch-and-bound algorithm. The Lagrangian problems are simply max-flow min-cut problems, while the branch-and-bound procedure partially enumerates *s-t* cuts. Computational tests reveal the new algorithm to be one to two orders of magnitude faster than an algorithm that replaces the specialized branch-and-bound algorithm with a standard integer-programming solver.

Key words: interdiction; maximum flow; Lagrangian relaxation; cut enumeration
History: Accepted by William Cook, Area Editor for Design and Analysis of Algorithms; received September 2005; revised March 2006; accepted May 2006.

1. Introduction

In the deterministic maximum-flow network-interdiction problem (MXFI), an "interdictor" wishes to restrict an "adversary's" use of a capacitated network. Specifically, the adversary seeks to maximize flow through the network, while the interdictor seeks to minimize that maximum flow by interdicting (destroying) arcs using available interdiction resources (Wood 1993). These resources could be a set of cruise missiles, a given number of aerial sorties, etc. MXFI and related problems appear in the areas of military planning (Whiteman 1999), drug interdiction (Steinrauf 1991), and protecting civil infrastructure against terrorist attacks (Salmeron et al. 2004). The model may be classified as a bilevel, target-selection, or weapons-allocation model; see Matlin (1970), Bracken and McGill (1974), Bracken et al. (1977), and Wen and Hsu (1991). A simpler version of MXFI, the problem of cutting all flow using minimum interdiction resource, provided one of the earliest applications of maximum flows and the max-flow min-cut theorem. The flow was rail traffic from the Soviet Union into eastern Europe; see Harris and Ross (1955) and Ford and Fulkerson (1956, 1957).

This paper studies an extension of MXFI, which we call the "bi-objective maximum-flow network-interdiction problem" (BMXFI). In BMXFI, the interdictor wishes to identify all Pareto-optimal solutions for MXFI, i.e., the efficient frontier, with respect to minimizing post-interdiction maximum flow and minimizing "total interdiction cost," i.e., interdiction-resource expenditure. A bi-objective formulation is important for a military commander who (i) must pre-plan for various resource availabilities given the uncertainty of warfare, or (ii) may wish to consider cost-to-effectiveness or risk-to-effectiveness tradeoffs that are difficult to include directly in an interdiction-planning model.

Steinrauf (1991) and Wood (1993) show that MXFI can be formulated as an integer program (IP). Thus, BMXFI can be solved as a set of IPs for MXFI, one for each plausible level of interdiction resource. But, MXFI is strongly NP-complete and difficult to solve in practice (Wood 1993). Hence, solution times could be unacceptable, especially in a time-constrained, military setting. Benders decomposition can be helpful, because it incorporates easy-to-solve maximum-flow subproblems, but the computational expense becomes prohibitive because so many subproblems must be solved (Cormican 1995). Derbes (1997), Bingol (2001) and Uygun (2002) devise fast heuristics based on Lagrangian relaxation, but these cannot guarantee good solutions. Implicitly, these authors also consider BMXFI, and construct heuristic solution approaches based on weighted-sums scalarization of the two objective functions.

The solution of a sequence of IPs will typically require that a commercial solver be installed on one or more local computers, and that up-to-date licenses be maintained for those computers. But, analysts at

a hastily organized military headquarters should not need to worry about software licenses. Alternatively, an open-source IP solver could be used, but such solvers tend to be less efficient than their commercial counterparts. Consequently, a need exists for a specialized approach to solving BMXFI.

This paper constructs a procedure for solving BMXFI based on weighted-sums scalarization of the objectives, viewed as an application of Lagrangian relaxation. Lagrangian relaxation partially maps out the efficient frontier through a sequence of max-flow min-cut problems. The procedure then fills in the missing pieces of the frontier with a specialized branch-and-bound algorithm that enumerates near-minimum-capacity cuts with respect to Lagrangianized capacities. For simplicity, we phrase our discussion in terms of "Pareto-optimal solutions" and "the efficient frontier." However, for a fixed limit on total interdiction cost, we typically allow a 1% or 5% optimality gap. Thus, we are actually identifying "near-Pareto-optimal solutions" (Carlyle et al. 2003), i.e., the "nearly efficient frontier."

The next section defines MXFI and BMXFI precisely. Sections 3 and 4 show how Lagrangian relaxation and cut enumeration can solve MXFI for specific limits on total interdiction cost. Section 5 presents the full solution procedure, Section 6 presents computational examples, and Section 7 presents conclusions.

2. Problem Formulation

Let $G = (N, A)$ denote a *directed graph* with *node set* N and *arc set* $A \subset N \times N$. Two special nodes $s \neq t$ are identified: s is the *source node* and t is the *sink node*. We also define $FS(i) = \{(i', j') \in A \mid i' = i\}$ and $RS(i) = \{(j', i') \in A \mid i' = i\}$.

Each arc $k \in A$ has a *capacity* $u_k \in \mathscr{Z}^+$ and *interdiction cost* $r_k \in \mathscr{Z}^+$. The interdiction cost is the amount of some resource necessary to destroy arc k, i.e., reduce that arc's capacity to zero. We assume that $r_k > 0$ and $u_k > 0$ for all $k \in A$, since any arc with $r_k = 0$ or $u_k = 0$ may be viewed as "already interdicted" and can be removed from the problem. We also expect that, in a military setting, all r_k will be small integers. For example, they might represent the number of aerial sorties, one, two or three, say, required to ensure destruction of a bridge. We note that only simple variations are required to model an undirected network, partial arc interdictions, and node interdiction (Wood 1993).

An *s-t cut* is a partition (N_s, N_t) of N such that $s \in N_s$ and $t \in N_t$. All cuts in this paper are *s-t* cuts, so we drop "*s-t*" hereafter. Given a cut $C = (N_s, N_t)$, A_C denotes the set of arcs $k = (i, j) \in A$ such that $i \in N_s$ and $j \in N_t$. The *capacity of C*, defined by $\sum_{k \in A_C} u_k$, provides an upper bound on the maximum *s-t* flow in G; deletion of A_C from G disconnects all paths directed from s to t (e.g., Ahuja et al. 1993, p. 185). A cut C is *minimal* if A_C is a minimal disconnecting set. We use $\mathscr{C}$ to denote the collection of all minimal cuts in G.

Let $x_k = 1$ if arc k is interdicted, and let $x_k = 0$, otherwise. Also, let $G(\mathbf{x}) = (N, A - \{k \in A \mid x_k = 1\})$, and let $g(\mathbf{x})$ denote the maximum *s-t* flow in $G(\mathbf{x})$. In BMXFI, an *interdiction plan* $\mathbf{x} \in \{0, 1\}^{|A|}$ is *dominated* if there exists another plan $\mathbf{x}'$ such that (i) $g(\mathbf{x}') \leq g(\mathbf{x})$, (ii) $\sum_{k \in A} r_k x'_k \leq \sum_{k \in A} r_k x_k$, and (iii) at least one inequality is strict. An interdiction plan that is not dominated is *Pareto-optimal*. We define BMXFI to be the problem of identifying all Pareto-optimal interdiction plans.

Let y_k denote the flow on arc $k \in A$, and let y_a denote the flow on an artificial "return arc" $a = (t, s) \notin A$. Also, define the standard shorthand notation "$\min_{\mathbf{x} \in X}\{g(\mathbf{x}), h(\mathbf{x})\}$" to mean: Find all Pareto-optimal pairs $(g(\mathbf{x}), h(\mathbf{x}))$, for $\mathbf{x} \in X$, with respect to minimization. We can then formulate BMXFI based on the well-known maximum-flow problem:

BMXFI

$$\min_{\mathbf{x} \in \{0,1\}^{|A|}} \left\{ g(\mathbf{x}), \sum_{k \in A} r_k x_k \right\}, \tag{1}$$

where

$$g(\mathbf{x}) = \max_{\mathbf{y} \in \mathscr{R}^{|A|},\, y_a} \quad y_a \tag{2}$$

$$\text{s.t.} \quad \delta_i y_a + \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = 0 \quad \forall i \in N \tag{3}$$

$$0 \leq y_k \leq u_k(1 - x_k) \quad \forall k \in A, \tag{4}$$

with $\delta_s = -1$, $\delta_t = 1$, and $\delta_i = 0$ for all $i \in N - \{s, t\}$. Equations (2)–(4) define a standard maximum-flow problem with "interdiction-modified arc capacities," $u_k(1 - x_k)$, for all $k \in A$.

We identify Pareto-optimal interdiction plans by isolating one objective function and limiting the value of the other by a constraint. Isolating the maximum-flow objective and imposing a limit R on total interdiction cost, we obtain the single-objective maximum-flow network-interdiction problem:

MXFI(R)

$$z^*(R) \equiv \min_{\mathbf{x} \in X} g(\mathbf{x}), \tag{5}$$

where $X \equiv \{\mathbf{x} \in \{0, 1\}^{|A|} \mid \sum_{k \in A} r_k x_k \leq R\}$.

MXFI(R) can be converted to a simple, minimizing IP by reformulating the problem into this essentially equivalent model (Cormican et al. 1998)

$$z^*(R) \equiv \min_{\mathbf{x} \in X} \max_{\mathbf{y}} \quad y_a - \sum_{k \in A} x_k y_k \tag{6}$$

$$\text{s.t.} \quad (3) \text{ and } 0 \leq y_k \leq u_k, \quad \forall k \in A, \tag{7}$$

and by then taking the dual of the inner problem. (See Wood 1993, also.) An optimal 0-1 solution always exists, so the resulting IP, after a few simplifications, is

MXFI-IP(R)

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

$$\text{s.t.} \quad \alpha_i - \alpha_j + \beta_k + x_k \geq 0 \quad \forall (i,j) = k \in A \tag{8}$$

$$\sum_{k\in A} r_k x_k \leq R \tag{9}$$

$$\text{all variables} \in \{0,1\}$$

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

The full efficient frontier can be identified by solving MXFI-IP(R), using standard IP software, over a sufficiently wide range of "R-values." However, this approach can be computationally costly, so we present an alternative approach in Section 5. This approach is also based on solving MXFI(R), so we first focus on this single-objective problem.

Another useful view of MXFI(R) derives from the max-flow min-cut theorem (Ford and Fulkerson 1956): Replace the max-flow problem that defines $g(\mathbf{x})$ with an equivalent min-cut problem. This leads to the following proposition, in which, without loss of generality, we have interchanged the order of the two minimizations.

Proposition 1. *MXFI(R) can be solved by finding $C^* \in \mathscr{C}$ such that a maximum-capacity, cost-feasible set of arc interdictions in C^* leaves as little uninterdicted capacity as possible. That is, MXFI(R) is equivalent to*

MXFI-C(R)

$$z^*(R) = \min_{C\in\mathscr{C}} \min_{\mathbf{x}} \sum_{k\in A_C} u_k(1-x_k) \tag{10}$$

$$s.t. \sum_{k\in A_C} r_k x_k \leq R \tag{11}$$

$$x_k \in \{0,1\} \quad \forall k \in A_C \tag{12}$$

$$x_k = 0 \quad \forall k \in A - A_C. \tag{13}$$

Proof. As in the dual to the maximum-flow problem, any feasible solution to MXFI-IP(R) identifies a minimal or nonminimal cut $C = (N_s, N_t)$ by setting $\alpha_i = 0$ for all $i \in N_s$, and $\alpha_i = 1$ for all $i \in N_t$ (e.g., Wood 1993). Constraints (8) are satisfied by setting $\beta_k = 1$ or $x_k = 1$ (but not both) for all $k \in A_C$, and by setting $\beta_k = x_k = 0$ for all $k \in A - A_C$. Thus, if we extend $\mathscr{C}$ in (10) to include nonminimal cuts, MXFI-C(R) and MXFI-IP(R), and thus MXFI(R), are equivalent. The validity of restricting $\mathscr{C}$ to minimal cuts is obvious. □

For a given cut C, the inner minimization of MXFI-C(R) comprises a knapsack problem with, for our purposes, small coefficients $r_k \in \mathscr{Z}^+$. Thus, if the number of minimal cuts were small (unlikely in practice), MXFI(R) would be fairly easy to solve: Enumerate all minimal cuts (e.g., Shier and Whited 1986), and solve one knapsack problem for each. If $r_k = 1$ for all $k \in A$, the problem may appear to be easier, because the knapsack problem becomes trivial, but it remains strongly NP-complete (Wood 1993).

3. Lagrangian Relaxation

We approach the solution of MXFI(R) by moving the interdiction-cost constraint (11) into the objective function using Lagrangian relaxation (Derbes 1997). Let λ denote the Lagrangian multiplier associated with (11). Then, for any $R \in \mathscr{Z}^+$ and $\lambda \in [0,\infty)$, we define

MXFI-LR(λ, R)

$$\underline{z}(\lambda,R) \equiv \min_{C\in\mathscr{C}} \min_{\mathbf{x}} \sum_{k\in A_C} u_k(1-x_k) + \lambda\left(\sum_{k\in A_C} r_k x_k - R\right) \tag{14}$$

$$\text{s.t.} \quad x_k \in \{0,1\} \quad \forall k \in A_C \tag{15}$$

$$x_k = 0 \quad \forall k \in A - A_C. \tag{16}$$

This Lagrangian problem yields a lower bound for MXFI(R): For any $R \in \mathscr{Z}^+$ and $\lambda \in [0,\infty)$, $\underline{z}(\lambda,R) \leq z^*(R)$. Note that if a solution $(\widehat{C}, \hat{\mathbf{x}})$ to MXFI-LR(λ,R) satisfies $\sum_{k\in A_{\widehat{C}}} r_k \hat{x}_k = R$, then $\underline{z}(\lambda,R) = z^*(R)$, that is, $\hat{\mathbf{x}}$ is optimal for MXFI(R).

Let $f(\lambda)$ denote the max-flow value in G given arc capacities $u'_k = \min\{u_k, \lambda r_k\}$. Observing that (i) the objective function in MXFI-LR(λ,R) may also be written as $\sum_{k\in A_C}(u_k + (\lambda r_k - u_k)x_k) - \lambda R$, and (ii) the problem's sole constraints are (15) and (16), a simple solution appears for the inner minimization of MXFI-LR(λ,R): For each $k \in A_C$, set $x_k = 1$ if $\lambda r_k \leq u_k$; otherwise set $x_k = 0$. Hence,

$$\underline{z}(\lambda,R) = \min_{C\in\mathscr{C}}\left(\sum_{k\in A_C} \min\{u_k, \lambda r_k\} - \lambda R\right) \tag{17}$$

$$= f(\lambda) - \lambda R, \tag{18}$$

where the last equality follows from the max-flow min-cut theorem. Thus, evaluating $\underline{z}(\lambda,R)$ is no harder than solving a max-flow problem. We do not use the following fact, but it is interesting to note that the best Lagrangian bound, $\max_{\lambda\geq 0} \underline{z}(\lambda,R)$, can be computed in polynomial time; see Lawler (1976), pp. 94–97, Megiddo (1979), and Derbes (1997).

For brevity in the following sections, let the inner minimizations in MXFI-C(R) and MXFI-LR(λ,R) be defined, respectively, as

$$z^*(R,C) \equiv \min_{\mathbf{x}} \sum_{k\in A_C} u_k(1-x_k) \tag{19}$$

$$\text{s.t.} \sum_{k\in A_C} r_k x_k \leq R$$

$$x_k \in \{0,1\} \quad \forall k \in A$$
$$x_k = 0 \quad \forall k \in A - A_C,$$

and

$$\underline{z}(\lambda, R, C) \equiv \sum_{k \in A_C} \min\{u_k, \lambda r_k\} - \lambda R. \tag{20}$$

4. Cut Enumeration

We will not need to maximize the Lagrangian lower bound $\underline{z}(\lambda, R)$ for every R when solving BMXFI. However, it is useful for now to imagine that, for a given R, we have identified $\hat{\lambda} \geq 0$ such that $\underline{z}(\hat{\lambda}, R)$ is "reasonably close" to $\max_{\lambda \geq 0} \underline{z}(\lambda, R)$. In the process of identifying $\hat{\lambda}$, we will have identified a cut $\widehat{C}$ that establishes that bound, i.e., $\underline{z}(\hat{\lambda}, R) = \underline{z}(\hat{\lambda}, R, \widehat{C})$, and will have found a feasible solution $\hat{\mathbf{x}}$ to MXFI(R) that defines an upper bound $\bar{z}(R) \geq z^*(R)$. In particular, having identified $\widehat{C}$, we can set $\bar{z}(R) = z^*(R, \widehat{C})$.

Now, if $\bar{z}(R) - \underline{z}(\hat{\lambda}, R) \leq \epsilon$ for some prespecified tolerance $\epsilon \geq 0$, we have identified an ϵ-optimal solution for MXFI(R). If not, we can apply Theorem 1, below, to find one. We first establish two lemmas to simplify the theorem's proof.

Lemma 1. *$\underline{z}(\lambda, R, C) \leq z^*(R, C)$ for any $\lambda \in [0, \infty)$, $R \in \mathscr{X}^+$, and $C \in \mathscr{C}$.*

Proof. This follows because (i) $z^*(R, C)$ is the optimal objective value for MXFI(R) defined on a network that consists only of the arcs A_C connected in parallel from s to t, and (ii) $\underline{z}(\lambda, R, C)$ defines a Lagrangian lower bound on the same problem. □

Lemma 2. *Let $\lambda \in [0, \infty)$ and $R \in \mathscr{X}^+$. If $\bar{z}(R)$ is a constant such that $\bar{z}(R) \geq z^*(R)$, then the outer minimization in MXFI-C(R) can be restricted to $C \in \mathscr{C}$ such that $\underline{z}(\lambda, R, C) \leq \bar{z}(R)$ without changing that problem's optimal value.*

Proof. Suppose that $\widehat{C} \in \mathscr{C}$ is a solution of the outer minimization in MXFI-C(R) with $\underline{z}(\lambda, R, \widehat{C}) > \bar{z}(R)$. Then, by Lemma 1,

$$\bar{z}(R) < \underline{z}(\lambda, R, \widehat{C}) \leq z^*(R, \widehat{C}) = z^*(R), \tag{21}$$

which is a contradiction. □

Theorem 1. *Given optimality tolerance $\epsilon \geq 0$, $\lambda \in [0, \infty)$, and $R \in \mathscr{X}^+$, suppose that $\widehat{C} \in \mathscr{C}$ yields upper bound $\bar{z}(R) = z^*(R, \widehat{C})$. Also, define*

$$z^*_\epsilon(R) = \min_{C \in \mathscr{C}(\bar{z}, \lambda, R, \epsilon)} z^*(R, C), \tag{22}$$

where $\mathscr{C}(\bar{z}, \lambda, R, \epsilon) \equiv \{C \in \mathscr{C} \mid \underline{z}(\lambda, R, C) \leq \bar{z}(R) - \epsilon\} \cup \{\widehat{C}\}$. Then, $z^*_\epsilon(R) - z^*(R) \leq \epsilon$, i.e., $z^*_\epsilon(R)$ is an ϵ-optimal objective value for MXFI(R).

Proof. If $z^*(R) \geq \bar{z}(R) - \epsilon = z^*(R, \widehat{C}) - \epsilon$, then $\widehat{C}$ has already yielded an ϵ-optimal solution and the theorem is valid. Suppose not, i.e., suppose $z^*(R) < z^*(R, \widehat{C}) - \epsilon$. Then, $z^*(R, \widehat{C}) - \epsilon$ is a valid upper bound on $z^*(R)$ and thus Lemma 2, with $\bar{z}(R)$ replaced by $z^*(R, \widehat{C}) - \epsilon$, implies that the minimization in (22) yields $z^*(R)$. □

Corollary 1. *If $\widehat{C}$ is a minimizer of (22), then every minimizer $\mathbf{x}^*_{\widehat{C}}$ of (19), with C replaced by $\widehat{C}$, is an ϵ-optimal interdiction plan for MXFI(R).* □

When the Lagrangian procedure of the previous section leaves $\bar{z}(R) - \underline{z}(\hat{\lambda}, R) > \epsilon$, Theorem 1 and Corollary 1 guide us to a solution of MXFI(R): Using $\lambda = \hat{\lambda}$ that approximately maximizes $\underline{z}(\lambda, R)$, (i) enumerate all cuts C having $\underline{z}(\hat{\lambda}, R, C) \leq \bar{z}(R) - \epsilon$, (ii) solve a knapsack problem for each C in order to minimize uninterdicted capacity, i.e., compute $z^*(R, C)$, and (iii) save the solution with the least uninterdicted capacity. Naturally, whenever a better solution is found, the upper bound can be updated, and, in effect then, we may apply Theorem 1 multiple times while searching for an ϵ-optimal solution. (Thus, beginning with a solution that is not ϵ-optimal, and a correspondingly weak upper bound, does not imply that we must then seek an optimal solution, which the theorem might seem to suggest.)

We implicitly enumerate all necessary cuts using the tree-search algorithm in Balcioglu and Wood (2003). That algorithm has this key feature: Given that each arc $k \in A$ has capacity u'_k ($u'_k = \min\{u_k, \lambda r_k\}$, in our case), then each node in the search tree corresponds to a unique cut C such that

$$\sum_{k \in C} u'_k \leq \sum_{k \in \widetilde{C}} u'_k \tag{23}$$

for every "child" cut $\widetilde{C}$ of C. Our use of this algorithm defines a depth-first branch-and-bound algorithm. In contrast to LP-based (linear-programming-based) branch and bound that reoptimizes the lower bound for each child, our algorithm recomputes a valid bound for each child, but does not reoptimize with respect to λ: We have not found that reoptimization improves computation times, in practice. Enumeration strategies other than depth first could be employed, but this strategy is easy to program and works well in practice.

We outline the cut-enumeration algorithm next, using $\lceil h \rceil$ to denote the smallest integer at least as large as the real number h. This ceiling function appears because, given our integral data, a nonintegral lower bound can always be rounded up to the nearest integer.

Procedure Enumerate. /* Solves MXFI(R) for specified $R \in \mathscr{X}^+$. */

Input Data. Data for MXFI(R), feasible solution $\hat{\mathbf{x}}(R)$ and corresponding upper bound $\bar{z}(R)$, parameters $\hat{\lambda} \geq 0$ and $\epsilon \geq 0$, and global lower bound $\lceil \underline{z}(\hat{\lambda}, R) \rceil$.

Output. An ϵ-optimal interdiction plan $\hat{\mathbf{x}}(R)$ for MXFI(R).

0. Set arc capacities $u'_k \leftarrow \min\{u_k, \hat{\lambda} r_k\}$ for all $k \in A$;

1. Begin the tree-search, cut-enumeration algorithm (Balcioglu and Wood 2003);

2. For each cut C encountered,

(a) If $\bar{z}(R) - \lceil \underline{z}(\hat{\lambda}, R, C) \rceil = \bar{z}(R) - \lceil (\sum_{k \in A_C} u'_k - \hat{\lambda} R) \rceil \leq \epsilon$, go to **Next**;

(b) Compute $z^*(R, C)$ and its optimizer $\mathbf{x}^*_C$;

(c) If $z^*(R, C) < \bar{z}(R)$, set $\bar{z}(R) \leftarrow z^*(R, C)$ and $\hat{\mathbf{x}}(R) \leftarrow \mathbf{x}^*_C$;

(d) If $\bar{z}(R) - \lceil \underline{z}(\hat{\lambda}, R, C) \rceil \leq \epsilon$ go to **Next**;

(e) For each child cut $\widetilde{C}$ of C,

(i) Continue the tree search recursively at Step 2 with $C \leftarrow \widetilde{C}$;

(ii) Upon returning from exploring $\widetilde{C}$, if $\bar{z}(R) - \lceil \underline{z}(\hat{\lambda}, R, C) \rceil \leq \epsilon$ go to **Next**;

/* The above check may be useful if $\bar{z}(R)$ has improved. */

(f) **Next:** If C is the initial cut in the tree, or if $\bar{z}(R) - \lceil \underline{z}(\hat{\lambda}, R) \rceil \leq \epsilon$, halt the enumeration and Return $\hat{\mathbf{x}}(R)$ and $\bar{z}(R)$;

(g) Backtrack from C.

We note that Procedure Enumerate may actually identify some nonminimal cuts that must be searched recursively, but which are not candidates for yielding optimal solutions. Our implementation adjusts for this, but we ignore the issue above for simplicity.

5. A Complete Algorithm

We now return to BMXFI. The full efficient frontier for BMXFI can be explored by solving MXFI(R) for all possible R-values, for example by repeated application of Procedure Enumerate. We construct a more efficient approach, however, which exploits the relationship between MXFI(R) and MXFI(R'), for R close to R'.

5.1. Overview

Using the arguments in the proof of Proposition 1, we find that BMXFI is equivalent to

BMXFI-C

$$\min_{C \in \mathscr{C}} \min_{\mathbf{x}} \left\{ \sum_{k \in A_C} u_k(1 - x_k), \sum_{k \in A_C} r_k x_k \right\} \tag{24}$$

$$\text{s.t. } x_k \in \{0, 1\} \quad \forall k \in A_C \tag{25}$$

$$x_k = 0 \quad \forall k \in A - A_C. \tag{26}$$

In view of this and ignoring the constant λR, we see that MXFI-LR(λ, R) is the reduction of BMXFI-C by means of weighted-sums scalarization of its objectives, using (unscaled) weights 1 and λ. Hence, the evaluation of $f(\lambda)$ (see (18)) for a range of λ-values is equivalent to solving weighted-sums scalarizations using various weights.

While we cannot expect to identify the full efficient frontier this way (Climaco et al. 1997), we hope that a substantial portion can be. Thus, for various values of λ, we perform the following steps:

1. Using a maximum-flow algorithm, compute

$$f(\lambda) \leftarrow \min_{C \in \mathscr{C}} \sum_{k \in A_C} \min\{u_k, \lambda r_k\} \quad \text{and}$$

$$\widehat{C} \leftarrow \arg\min_{C \in \mathscr{C}} \sum_{k \in A_C} \min\{u_k, \lambda r_k\};$$

2. Set $\hat{x}_k \leftarrow 1$ whenever $u_k \geq \lambda r_k$ and $k \in A_{\widehat{C}}$, and set $\hat{x}_k \leftarrow 0$ otherwise;

3. Set $R \leftarrow \sum_{k \in A} r_k \hat{x}_k$, $z^*(R) \leftarrow \sum_{k \in A} u_k(1 - \hat{x}_k)$, and $\hat{\mathbf{x}}(R) \leftarrow \hat{\mathbf{x}}$.

The solution $\hat{\mathbf{x}}(R)$ is optimal for MXFI(R) since $\sum_{k \in A} r_k \hat{x}_k(R) = R$. For problems MXFI($R$) not immediately solved using these steps, we have a head start because, for any R, $\underline{z}(\lambda, R) = f(\lambda) - \lambda R \leq z^*(R)$ for all values of λ that the procedure examines.

Let $R_{\max}$ be the smallest R such that $z^*(R) = 0$, i.e., the minimum amount of interdiction resource that can force the adversary's maximum flow to 0. The corresponding interdiction plan is found by setting $\lambda = \min_{k \in A} u_k / r_k$ in the previously described procedure. Also, for any $\lambda > \max_{k \in A} u_k / r_k$, the solution obtained corresponds to $R = 0$. Thus, it suffices to examine values of λ in the range $[\min_{k \in A} u_k / r_k, \Delta + \max_{k \in A} u_k / r_k]$ for some $\Delta > 0$.

Since $f(\lambda)$ is a piecewise-linear function with no more than $R_{\max}$ break points, a procedure akin to Benders decomposition could map out $\underline{z}(\lambda, R)$ completely by solving at most $R_{\max} + 1$ max-flow problems. The Benders master problem would be simple enough to solve by inspection. However, the sequence of max-flow min-cut problems would involve widely varying arc capacities and would be harder to solve efficiently than one in which arc capacities are nondecreasing. Consequently, we opt for mapping out the function $f(\lambda)$ approximately, starting with a small value for λ and increasing it according to an empirically derived rule.

We motivate that rule with a network consisting of a set of parallel arcs of the form $k = (s, t)$, all with unique ratios u_k / r_k. In this case, $f(\lambda)$ breaks only at points where some arc capacity $\min\{u_k, \lambda r_k\}$ switches from λr_k to u_k, i.e., at $\lambda = u_k / r_k$. Therefore, we can map out the function $f(\lambda)$, and thus the functions $\underline{z}(\lambda, R)$ for all R, by evaluating the maximum flow, or minimum cut capacity, at only these values of λ.

Naturally, the situation is more complex for a general network. But, for simplicity, let us continue to assume that the ratios u_k / r_k are unique. Suppose some $\lambda' > 0$ has been defined, arc capacities $u'_k =$

$\min\{u_k, \lambda' r_k\}$ have been computed, and a minimum cut C is identified such that $u'_k = u_k$ for at least one $k \in A_C$. Without loss of generality, assume $\lambda' r_k \neq u_k$ for any $k \in A$. Now, our example implies that $f(\lambda)$ must break somewhere on the interval $(\lambda', \lambda'']$ where $\lambda'' = \min_{k \in A_C}\{u_k/r_k \mid u_k/r_k > \lambda'\}$, i.e., between its current value and the value where the function would break if C were the only cut in the network. Since G typically contains a huge number of cuts, some other cut may become "the minimum cut" before λ reaches λ''. Thus, increasing λ all the way to λ'' tends to overshoot the next breakpoint. This observation, along with empirical testing, leads us to adopt the following rule for computing the next λ:

$$\lambda = \max\{(1-\phi)\lambda' + \phi\lambda'', \lambda' + \Delta\} \tag{27}$$

for some empirically determined $\Delta > 0$ and $\phi \in (0, 1]$. We find that $\Delta = 0.1 \min_{k \in A} \min\{u_k, r_k\}$, and $\phi = 0.25$ work well in general.

5.2. Algorithm Interdict

Based on the discussion and outline above, we now state a complete algorithm for BMXFI:

Algorithm Interdict. /* Solves BMXFI. */
Parameters. Absolute tolerance $\epsilon \geq 0$ and algorithm controls $\phi \in (0, 1]$ and $\Delta > 0$.
Input Data. Network $G = (N, A)$, with interdiction costs $r_k \in \mathscr{Z}^+$ and arc capacities $u_k \in \mathscr{Z}^+$ for all $k \in A$. Source node $s \in N$ and sink node $t \in N$.
Output. An optimal or ϵ-optimal solution $\hat{\mathbf{x}}(R)$ to MXFI(R) for each $R \in \{0, 1, \ldots, R_{\max}\}$.

0. Set $\lambda \leftarrow \min_{k \in A}\{u_k/r_k\}$ and $\Theta \leftarrow \varnothing$;
/* Θ represents the set of R-values with known solutions. */

1. While $\lambda \leq \max_{k \in A}\{u_k/r_k\} + \Delta$
/* Step 1 roughly maps out $f(\lambda)$ and identifies, we hope, a large number of solutions $\hat{\mathbf{x}}(R)$ for different R-values. */

(a) Solve the maximum flow problem on G using arc capacities $u'_k = \min\{u_k, \lambda r_k\}$, and find a corresponding minimum-capacity cut C;

(b) Set $\hat{x}_k \leftarrow 1$ for all $k \in A_C$ such that $u_k \geq \lambda r_k$, and set $\hat{x}_k \leftarrow 0$ otherwise;

(c) Set $R \leftarrow \sum_{k \in A} r_k \hat{x}_k$;

(d) If $R \notin \Theta$, /* A new solution has been found. */ set $\Theta \leftarrow \Theta \cup \{R\}$, $\lambda(R) \leftarrow \lambda$, $z^*(R) \leftarrow \sum_{k \in A} u_k(1 - \hat{x}_k)$, and $\hat{\mathbf{x}}(R) \leftarrow \hat{\mathbf{x}}$;

(e) If $\lambda = \min_{k \in A}\{u_k/r_k\}$, set $R_{\max} \leftarrow R$;

(f) If $\min\{u_k, \lambda r_k\} = u_k$ for all $k \in A_C$, break from "While" loop; Else, set $\lambda \leftarrow \max\{(1-\phi)\lambda + \phi \min_{k \in A_C}\{u_k/r_k \mid u_k/r_k \geq \lambda\}, \lambda + \Delta\}$; /* See (27) */

2. For each $R \in \{0, 1, \ldots, R_{\max}\} - \Theta$
/* Step 2 refines lower bounds for R-values whose ϵ-optimal solutions remain unidentified. It may also identify some new, absolutely optimal solutions. */

(a) Select $R', R'' \in \Theta$, $R' < R < R''$, to bracket R as tightly as possible;

(b) Set $\lambda(R) \leftarrow \arg\max_{\lambda \in \{\lambda(R'), \lambda(R'')\}} \underline{z}(\lambda, R)$;

(c) Compute a new λ-value, $\lambda''' \leftarrow (z^*(R') - z^*(R''))/(R'' - R')$;
/* λ''' estimates $\arg\max_{\lambda \geq 0} \underline{z}(\lambda, R)$ using subgradients of $\underline{z}(\lambda, R)$. */

(d) If $\lambda''' \notin (\lambda(R''), \lambda(R'))$ continue the "For" loop;
/* The "continue" means that λ''' cannot yield an improved lower bound, so just proceed to the next R-value. */

(e) Compute $\widehat{C}$ and $\hat{\mathbf{x}}$ by performing Steps 1(a)–(b), with $\lambda = \lambda'''$, and set $\underline{z}(\lambda''', R) \leftarrow \underline{z}(\lambda''', R, \widehat{C})$;

(f) Set $\lambda(R) \leftarrow \arg\max_{\lambda \in \{\lambda(R), \lambda'''\}} \underline{z}(\lambda, R)$;

(g) If $R = \sum_{k \in A} r_k \hat{x}_k$,
/* A new optimal solution has been found. */ set $\Theta \leftarrow \Theta \cup \{R\}$, $z^*(R) \leftarrow \underline{z}(\lambda(R), R)$, and $\hat{\mathbf{x}}(R) \leftarrow \hat{\mathbf{x}}$;

3. For each $R \in \{0, 1, \ldots, R_{\max}\} - \Theta$ in increasing order /* Step 3 identifies ϵ-optimal solutions for R-values that have eluded exact solution in Steps 1 and 2. Depending on ϵ, it may or may not be necessary to actually call the cut-enumeration (branch-and-bound) procedure. */

(a) Set initial values $\bar{z}(R) \leftarrow \bar{z}(R-1)$, $\hat{\mathbf{x}}(R) \leftarrow \hat{\mathbf{x}}(R-1)$, $\hat{\lambda} \leftarrow \lambda(R)$; /* $\underline{z}(\hat{\lambda}, R)$ is now also defined */

(b) If $\bar{z}(R) - \lceil \underline{z}(\hat{\lambda}, R) \rceil \leq \epsilon$, then continue the "For" loop;

(c) Call **Procedure Enumerate** with inputs $\bar{z}(R)$, $\hat{\mathbf{x}}(R)$, $\hat{\lambda}$, $\lceil \underline{z}(\hat{\lambda}, R) \rceil$, to return ϵ-optimal $\hat{\mathbf{x}}(R)$;

4. /* Output solution. */

(a) For all $R \in \Theta$, Print(R, "Optimal solution is," $\hat{\mathbf{x}}(R)$);

(b) For all $R \in \{0, 1, \ldots, R_{\max}\} - \Theta$, Print($R$, "$\epsilon$-optimal solution is," $\hat{\mathbf{x}}(R)$).

Step 2 of Algorithm Interdict estimates the best lower bound $\max_{\lambda \geq 0} \underline{z}(\lambda, R)$ for $z^*(R)$ and the corresponding maximizer $\lambda(R)$ by using $z^*(R')$, $z^*(R'')$ and a slope estimate of $\underline{z}(\lambda, R)$ with respect to λ, where R' and R'' bracket R. In particular, Step 2(c) uses the fact, from (18), that $\underline{z}(\lambda, R) \leq \underline{z}(\lambda, \rho) + (\rho - R)\lambda$ for any $\rho \in \mathscr{Z}^+$ and $\lambda \geq 0$, as well as the fact that $\underline{z}(\lambda(\rho), \rho) = z^*(\rho)$ for $\rho = R', R''$. Step 2 may not maximize the lower bound precisely, but optimizing the bound has not proved computationally worthwhile. We also note that Steps 1 and 2 are actually integrated in the algorithm's implementation, and have been separated above for the sake of clarity.

Step 3(a) in Algorithm Interdict exploits the fact that an upper bound on $z^*(R')$ is also an upper bound on $z^*(R)$ for $R > R'$. Hence, the initial upper bound for each R is computed without significant computational cost.

Since $R_{\max}$ is finite and the network G has a finite number of cuts, Algorithm Interdict solves BMXFI in finite computing time. Given that the number of cuts

can be exponential in the size of G, the algorithm's worst-case complexity is exponential, however.

5.3. Enhancements to the Algorithm

Our implementation of Algorithm Interdict makes two modifications to the basic algorithm to improve computational efficiency. First, maximum flows are calculated for many similar problems, so we exploit "warm starts." We use a variant of the shortest-augmenting-path algorithm of Edmonds and Karp (1972) to solve maximum-flow problems, and an inherent feature of such algorithms is that the maximum flow for a problem with arc capacities $u_k^{(1)}$ provides a feasible initial flow, i.e., a warm start, for a problem with capacities $u_k^{(2)} \geq u_k^{(1)}$ for all $k \in A$. We make use of that feature in Step 1 of the algorithm since arc capacities never decrease as λ increases. We also make use of it in Step 2 of Procedure Enumerate, where the next cut to consider is determined through a warm-started max-flow calculation (Balcioglu and Wood 2003).

The second computational improvement comes from perturbing the ratios u_k/r_k so that $u_k/r_k \neq u_l/r_l$ for all $k, l \in A$, $k \neq l$. Derbes (1997) first notes the benefits of this technique; see also Bingol (2001). We implement perturbations by multiplying every u_k by a large positive integer (in practice, 6×10^5), and by then adding a small positive integer to u_k whenever $u_k/r_k = u_l/r_l$ for some $l \in A$, $l \neq k$. The perturbations are several orders of magnitude smaller than the smallest arc capacity and hence do not change a problem's solutions. This technique typically creates more linear segments in $\underline{z}(\lambda, R)$ as a function of λ. Hence, it is more likely that $\underline{z}(\lambda, R)$ is constant for λ in some interval. Because any λ in this interval yields $\underline{z}(\lambda, R) = z^*(R)$, this technique results in more instances of MXFI(R) being solved optimally in Step 2 of the Algorithm.

6. Computational Results

This section examines the efficiency of Algorithm Interdict for solving BMXFI applied to artificial and real-world network data. The algorithm is coded in Java 1.4.2 (Sun Microsystems 2004), and run on a 3.0 GHz Pentium IV laptop computer with one gigabyte of RAM.

We compare our algorithm's run times to the times required by a hybrid algorithm that looks just like Algorithm Interdict except that a commercial IP solver is called instead of Procedure Enumerate to resolve instances of MXFI-IP(R) for "problematic" values of R. We use CPLEX Version 9.1 (ILOG 2005) as the solver, integrated with our Java code using ILOG Concert Technology (ILOG 2006). We have tested a variety of CPLEX options, and find that it is best to favor "branching up" and to apply mixed-integer rounding cuts aggressively.

The tables of results also report the average number of branch-and-bound nodes, computed over all $R \in \{0, 1, \ldots, R_{\max}\}$, for both algorithms. When CPLEX solves an IP without any enumeration, it records zero nodes enumerated. For consistency, we also record zero nodes when Algorithm Interdict does not invoke Procedure Enumerate for a given R.

The CPLEX version of MXFI-IP(R) exploits $\bar{z}(R-1)$ just as Procedure Enumerate does, and incorporates two enhancements. The first defines an auxiliary integer variable $\tilde{x}$ and an auxiliary "branching constraint" $\sum_{k \in A} x_k = \tilde{x}$ (Appleget and Wood 2000). When not all $r_k = 1$, setting the branching priority highest for $\tilde{x}$, (and second highest for the x_k, and lowest for the other variables) can substantially reduce enumeration. Roughly speaking, this causes the solver to branch on the number of interdictions before branching on individual interdictions. The second enhancement incorporates warm starts for the sequence of closely related LP relaxations that CPLEX must solve. We believe that these enhancements to the IP-based methodology mean that the solution times reported for the hybrid algorithm present a substantial challenge to Algorithm Interdict.

Figure 1 illustrates an instance from the first class of test networks, rectangular grid networks. Letting n_1 and n_2 denote the number of rows and columns of nodes in these networks, $|N| = n_1 n_2 + 2$ and $|A| = 2n_1 + 2((n_2 - 1)n_1 + (n_1 - 1)n_2)$. Arcs connecting the source s and the sink t all have effectively infinite capacities and are invulnerable to interdiction. All other arcs k have capacities u_k that are drawn randomly and independently from the discrete uniform distribution on $[1, 49]$.

We consider three variants of the grid networks, all having common topology but with different numerical data. For variant A1, interdiction costs are $r_k = 1$ for all $k \in A$. Interdiction costs are randomly generated for variants A2 and A3, as specified in Table 1. In this table, k_W, k_E, k_S, and k_N represent westbound, eastbound, southbound, and northbound arcs, respectively. Randomly generated data are statistically independent. Small integers should adequately

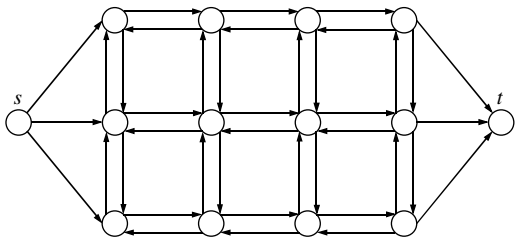

Figure 1 **Rectangular Grid Network with $n_1 = 3$ and $n_2 = 4$**

Table 1 **Probabilities for Pseudo-Random Generation of Interdictions Costs in Grid-Network Variants A1, A2, and A3**

Network	A1	A2	A3
$\text{Prob}(r_{k_W} = 1)$	1	1/4	1/2
$\text{Prob}(r_{k_W} = 2)$	0	3/4	1/2
$\text{Prob}(r_{k_W} = 3)$	0	0	0
$\text{Prob}(r_{k_E} = 1)$	1	0	0
$\text{Prob}(r_{k_E} = 2)$	0	1	1/2
$\text{Prob}(r_{k_E} = 3)$	0	0	1/2
$\text{Prob}(r_{k_S} = 1)$	1	1/4	1/2
$\text{Prob}(r_{k_S} = 2)$	0	3/4	1/2
$\text{Prob}(r_{k_S} = 3)$	0	0	0
$\text{Prob}(r_{k_N} = 1)$	1	1/4	1/2
$\text{Prob}(r_{k_N} = 2)$	0	3/4	1/2
$\text{Prob}(r_{k_N} = 3)$	0	0	0

model interdiction costs in military applications, so $r_k \in \{1, 2, 3\}$ in all problem instances. Note that Algorithm Interdict can be simplified when all $r_k = 1$ because the knapsack problem in Step 2(b) of Procedure Enumerate becomes trivial. We have not implemented code to take advantage of this special case, however.

Tables 2, 3, and 4 display solution times and average number of nodes in the enumeration trees for both algorithms, when using relative optimality tolerances of 1% and 5%. For Algorithm Interdict, this means that ϵ is evaluated as $0.01\lceil \underline{z}(\lambda(R), R)\rceil$ or $0.05\lceil \underline{z}(\lambda(R), R)\rceil$, respectively, instead of as a fixed value.

These tables indicate that Algorithm Interdict runs between one and two orders of magnitude faster than the hybrid algorithm (on average, 24, 42, and 119 times faster on the grid networks A1, A2, and A3, respectively). We note that Algorithm Interdict solves faster than reported for certain problems when using other schemes for determining λ in Step 1(f) of the Algorithm. However, we have not optimized algorithmic performance by specializing the code for particular problems. Figure 2 illustrates the trade-off between maximum flow and total interdiction cost for the case A3 (20×40). The lack of convexity in this figure implies that the efficient frontier cannot be identified though weighted-sums scalarization alone (Climaco et al. 1997).

Table 3 **Total Time to Solve BMXFI, and Average Number of Nodes in Enumeration Trees for Grid Networks A2**

			Network A2			
	Value	Tol. (%)	10×20	20×40	30×60	40×80
Hybrid alg.	Tot. time (sec.)	1	1.2	3.4	459.8	1,619.6
	Tot. time (sec.)	5	1.1	3.4	428.8	1,053.0
	Avg. nodes	1	0.1	0.3	0.5	1.1
	Avg. nodes	5	0.1	0.3	0.5	1.1
Alg. interdict	Tot. time (sec.)	1	0.1	0.7	5.9	23.3
	Tot. time (sec.)	5	0.1	0.7	5.6	13.3
	Avg. nodes	1	13.5	26.6	56.6	200.6
	Avg. nodes	5	10.2	5.6	20.2	31.7

The second collection of test problems uses networks derived from the highways and roads in Maryland, Virginia, and Washington, D.C. (Carlyle and Wood 2005). These data include all interstate, state, and county roads in these areas. Using various arc capacities and interdiction costs, we construct 14 different network problems denoted B1–B7 and C1–C7. Each problem contains multiple sources and sinks that are connected to a super-source and super-sink, respectively, using invulnerable, infinite-capacity arcs.

The "road-B networks" incorporate 9,876 directed road segments, which are modeled as arcs in a directed graph with 3,670 nodes. These segments represent all roads with speed limits of 30 miles per hour or higher. 16 nodes along the region's northern border constitute source nodes, and 14 nodes on the southern border constitute sink nodes. Including auxiliary nodes and arcs, the network comprises 3,672 nodes and 9,906 arcs. Interdiction costs and arc capacities are generated as specified by Table 5, where $\text{Prob}(u_k = \mu)$ is the probability that u_k takes on a value in $\{1, 2, \ldots, 49\}$. Randomly generated data are statistically independent. Deterministic problem instances

Table 2 **Total Time to Solve BMXFI, and Average Number of Nodes in Enumeration Trees for Grid Networks A1**

			Network A1			
	Value	Tol. (%)	10×20	20×40	30×60	40×80
Hybrid alg.	Tot. time (sec.)	1	1.2	4.4	59.7	317.8
	Tot. time (sec.)	5	1.2	4.4	59.7	319.8
	Avg. nodes	1	0.5	0.0	0.1	0.0
	Avg. nodes	5	0.5	0.0	0.1	0.0
Alg. interdict	Tot. time (sec.)	1	0.1	0.3	2.6	6.4
	Tot. time (sec.)	5	0.1	0.7	2.7	6.5
	Avg. nodes	1	6.1	0.2	0.4	7.4
	Avg. nodes	5	3.1	0.2	0.4	3.4

Table 4 **Total Time to Solve BMXFI, and Average Number of Nodes in Enumeration Trees for Grid Networks A3**

			Network A3			
	Value	Tol. (%)	10×20	20×40	30×60	40×80
Hybrid alg.	Tot. time (sec.)	1	2.5	196.5	1,556.5	2,257.0
	Tot. time (sec.)	5	2.5	177.7	651.1	1,219.7
	Avg. nodes	1	0.6	1.4	2.0	0.4
	Avg. nodes	5	0.6	1.0	0.7	0.6
Alg. interdict	Tot. time (sec.)	1	0.1	0.9	7.1	28.3
	Tot. time (sec.)	5	0.1	0.9	5.4	18.7
	Avg. nodes	1	26.8	100.1	86.6	92.1
	Avg. nodes	5	19.3	60.0	25.7	5.9

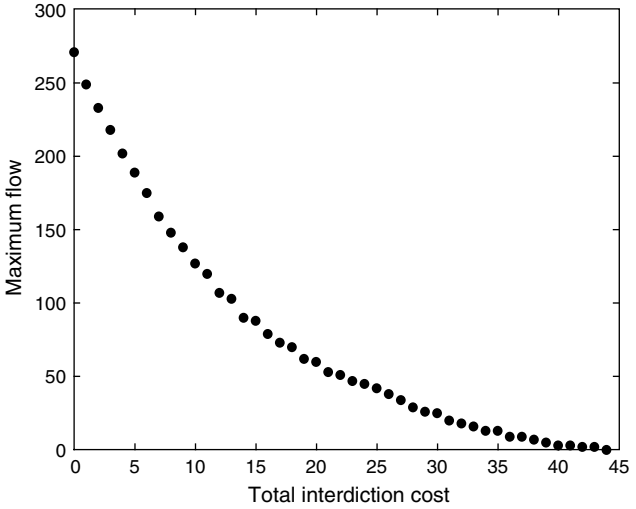

Figure 2 **Solving BMXFI for Case A3** (20×40)**: The Trade-Off Between Maximum Flow and Total Interdiction Cost**

("Deter.") have $u_k = s_k/5$ for all k, where s_k is the speed limit in miles per hour for road segment k, $s_k \in \{30, 45, 50, 55, 65\}$. Also, for B7 and C7, "Deter." indicates that $r_k = 1$ when $s_k = 30$, and otherwise $r_k = 2$.

The "road-C networks" have the same topology as the B-networks, but with different sources and sinks. 152 nodes in the region's center constitute source nodes, and 35 nodes scattered about the periphery constitute sink nodes. The network comprises 3,672 nodes and 10,031 arcs in total. Interdiction costs are generated as specified in Table 5.

A node $i \in N$ in the road networks is said to be a *nonintersection* if $FS(i) = \{(i, j_1), (i, j_2)\}$ and $RS(i) = \{(j_1, i), (j_2, i)\}$. About 40% of the nodes in the road networks are nonintersections because of the level of detail represented: Nonintersections do actually represent intersections, but at least one intersecting road segment has been deleted because its speed limit lies below the cutoff of 30 miles per hour. Clearly, a nonintersection can be deleted and replaced by two arcs, (j_1, j_2) and (j_2, j_1), with appropriate adjustments to the numerical data. Both algorithms use the preprocessed data. (Solution times reported exclude the pre-processing time, but none exceeds 0.1 seconds.)

For the cases in which arc capacities u_k are defined as one fifth of the speed limit (see Table 5), Algorithm Interdict adds a test to ensure that the lower bound does not take on a value of flow that cannot be maximum, i.e., $\underline{z}(\lambda(R), R) \notin \{1, 2, 3, 4, 5, 7, 8, 14\}$. For instance, a nominal lower bound of 4 increases to 6.

Table 5 **Parameters for Generating Road-Network Capacities and Interdiction Costs**

Network	B1, C1	B2, C2	B3, C3	B4, C4	B5, C5	B6, C6	B7, C7
Prob($r_k = 1$)	1	1/2	1/3	1	1/2	1/3	
Prob($r_k = 2$)	0	1/2	1/3	0	1/2	1/3	Deter.
Prob($r_k = 3$)	0	0	1/3	0	0	1/3	
Prob($u_k = \mu$)	1/49	1/49	1/49	Deter.	Deter.	Deter.	Deter.

Table 6 **Total Time to Solve BMXFI, and Number of Nodes in Enumeration Trees for Road-B Networks**

			Network						
	Value	Tol. (%)	B1	B2	B3	B4	B5	B6	B7
Hybrid alg.	Tot. time (sec.)	1	6.9	12.0	8.4	6.9	23.2	13.3	14.6
	Tot. time (sec.)	5	6.9	12.0	8.3	6.8	23.1	13.1	14.4
	Avg. nodes	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Avg. nodes	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Alg. interdict	Tot. time (sec.)	1	0.2	0.3	0.3	0.4	0.5	0.6	1.2
	Tot. time (sec.)	5	0.2	0.3	0.3	0.4	0.4	0.6	0.8
	Avg. nodes	1	0.0	0.1	0.2	0.1	0.3	24.1	55.6
	Avg. nodes	5	0.0	0.1	0.2	0.1	0.3	24.1	0.5

Table 7 **Total Time to Solve BMXFI, and Average Number of Nodes in Enumeration Trees for Road-C Networks**

			Network						
	Value	Tol. (%)	C1	C2	C3	C4	C5	C6	C7
Hybrid alg.	Tot. time (sec.)	1	9.5	13.4	14.2	7.9	9.9	14.9	19.8
	Tot. time (sec.)	5	9.4	13.2	14.1	7.9	9.8	14.7	18.2
	Avg. nodes	1	0.0	0.2	0.0	0.0	0.0	0.0	0.0
	Avg. nodes	5	0.0	0.2	0.0	0.0	0.0	0.0	0.0
Alg. interdict	Tot. time (sec.)	1	0.5	0.7	8.5	0.5	0.5	2.9	5.4
	Tot. time (sec.)	5	0.5	0.7	0.7	0.5	0.5	2.4	0.9
	Avg. nodes	1	0.1	33.8	958.6	0.1	0.5	278.7	389.5
	Avg. nodes	5	0.1	33.8	2.0	0.1	0.2	220.5	0.3

Tables 6 and 7 display solution times for both algorithms applied to the road-B and road-C networks, respectively. On average, Algorithm Interdict remains an order of magnitude faster than the hybrid algorithm (30 and 15 times faster on the B and C networks, respectively). However, the road-C problems include three cases, C3, C6, and C7, with relatively small speed-ups at the 1%-tolerance level. The hybrid algorithm appears only modestly slower in these cases because Algorithm Interdict requires substantial enumeration, while CPLEX requires almost none.

On average, Algorithm Interdict requires more enumeration than does the hybrid algorithm, no doubt because the Lagrangian lower bounds are weaker than CPLEX's cut-enhanced, LP-based bounds. An increased tolerance can significantly reduce the number of branch-and-bound nodes for Algorithm Interdict, while the number of nodes for CPLEX stays small, almost independent of the optimality tolerance.

7. Conclusions

This paper describes a new procedure, "Algorithm Interdict," for solving the bi-objective maximum-flow interdiction problem. The algorithm first identifies a large portion of the efficient frontier using weighted-sums scalarization of the two objectives to be minimized, maximum flow and total interdiction cost.

We interpret this through the theory of Lagrangian relaxation. A specialized branch-and-bound procedure, involving partial cut enumeration, then identifies any missing parts of that frontier.

We have compared Algorithm Interdict to a hybrid algorithm in computational tests; the hybrid algorithm calls a standard integer-programming solver instead of the specialized branch-and-bound procedure, when the Lagrangian solution from Algorithm Interdict is not ϵ-optimal. With rare exceptions, our algorithm is one to two orders of magnitude faster than the hybrid algorithm; on average it is 40 times faster. Its efficiency results from the fact that bounds and *s-t* cuts are computed via the solutions of interrelated, and easily solved maximum-flow problems. The algorithm may require more enumeration than does linear-programming-based branch and bound, but that enumeration is highly efficient. Algorithm Interdict also provides the benefit of not requiring a licensed solver.

We might attempt to reduce enumeration by improving the Lagrangian lower bound. Indeed, it is clear that the standard solver achieves better bounds through the use of integer cutting planes. If such cutting planes could be identified and Lagrangianized with appropriate multipliers, this could improve the lower bound. Wood (1993) identifies some problem-specific cutting planes that could be explored for this purpose.

Acknowledgments

The authors thank an anonymous reviewer for valuable comments and suggestions. The first author thanks the National Research Council for research support. The second author thanks the Office of Naval Research, the Air Force Office of Scientific Research, the Naval Postgraduate School, and the University of Auckland for research support. Both authors thank Gerald Brown for providing road data for computational examples and Matthew Carlyle, Javier Salmeron, and Keith Olson for valuable comments.

References

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

Appleget, J., K. Wood. 2000. Explicit-constraint branching for solving mixed integer programs. M. Laguna, J. L. González-Velarde, eds. *Computing Tools for Modeling, Optimization and Simulation*. Kluwer Academic Publishers, Boston, MA, 245–262.

Balcioglu, A., R. K. Wood. 2003. Enumerating near-min *s-t* cuts. D. L. Woodruff, ed. *Network Interdiction and Stochastic Integer Programming*. Kluwer Academic Publishers, Boston, 21–49.

Bingol, L. 2001. A Lagrangian heuristic for solving a network interdiction problem. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.

Bracken, J., J. T. McGill. 1974. Defense applications of mathematical programs with optimization problems in the constraints. *Oper. Res.* **22** 1086–1096.

Bracken, J., J. E. Falk, F. A. Miercort. 1977. A strategic weapons exchange allocation model. *Oper. Res.* **25** 968–976.

Carlyle, W. M., R. K. Wood. 2005. Near-shortest and *k*-shortest simple paths. *Networks* **46** 98–109.

Carlyle, W. M., J. W. Fowler, E. S. Gel, B. Kim. 2003. Quantitative comparison of approximate solution sets for bi-criteria optimization problems. *Decision Sci.* **34** 63–82.

Climaco, J., C. Ferreira, M. Captivo. 1997. Multicriteria integer programming: An overview of the different algorithmic approaches. J. Climaco, ed. *Multicriteria Analysis*. Springer, Berlin, Germany, 248–258.

Cormican, K. J. 1995. Computational methods for deterministic and stochastic network interdiction problems. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.

Cormican, K. J., D. P. Morton, R. K. Wood. 1998. Stochastic network interdiction. *Oper. Res.* **46** 184–197.

Derbes, H. D. 1997. Efficiently interdicting a time-expanded transshipment network. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.

Edmonds, J., R. M. Karp. 1972. Theoretical improvements in algorithm efficiency for network flow problems. *J. ACM* **19** 248–264.

Ford, L. R., D. R. Fulkerson. 1956. Maximal flow through a network. *Canadian J. Math.* **8** 399–404.

Ford, L. R., D. R. Fulkerson. 1957. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canadian J. Math.* **9** 210–218.

Harris, T. E., F. S. Ross. 1955. Fundamentals of a method for evaluating rail net capacities. Research Memorandum RM-1573, The Rand Corp., Santa Monica, CA.

ILOG. 2005. *ILOG CPLEX 9.1, User's Manual*. ILOG, S.A., Gentilly Cedex, France.

ILOG. 2006. *ILOG Concert Technology*. ILOG, S.A., Gentilly Cedex, France, http://www.ilog.com/products/optimization/tech/concert.cfm.

Lawler, E. L. 1976. *Combinatorial Optimization, Networks and Matroids*. Holt, Rinehart and Winston, New York.

Matlin, S. 1970. A review of the literature on the missile allocation problem. *Oper. Res.* **17** 334–373.

Megiddo, N. 1979. Combinatorial optimization with rational objective functions. *Math. Oper. Res.* **4** 414–424.

Salmeron, J., K. Wood, R. Baldick. 2004. Analysis of electric grid security under terrorist threat. *IEEE Trans. Power Systems* **19-2** 905–912.

Shier, D. R., D. E. Whited. 1986. Iterative algorithms for generating minimal cutsets in directed graphs. *Networks* **16** 133–147.

Steinrauf, R. L. 1991. Network interdiction models. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.

Sun Microsystems. 2004. *Java 1.4.2. Documentation*. Sun Microsystems, Santa Clara, CA, http://www.java.sun.com.

Uygun, A. 2002. Network interdiction by Lagrangian relaxation and branch-and-bound. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.

Wen, U., S. Hsu. 1991. Linear bi-level programming problems—A review. *J. Oper. Res. Soc.* **42** 125–133.

Whiteman, P. S. 1999. Improving single strike effectiveness for network interdiction. *Military Oper. Res.* **4** 15–30.

Wood, R. K. 1993. Deterministic network interdiction. *Math. Comput. Model.* **17** 1–18.