Algorithms for Discrete and Continuous Multicommodity Flow Network Interdiction Problems

Churlzu Lim

J. Cole Smith

Department of Systems and Industrial Engineering
The University of Arizona
Tucson, AZ 85721

The authors gratefully acknowledge financial support from the *Air Force Office of Scientific Research* (F49620-03-1-0477), and from the *Office of Naval Research* (N66001-01-1-8925). The authors also thank two anonymous referees and an Associate Editor for their remarks, which improved the presentation of this paper.

Abstract

We consider a network interdiction problem on a multicommodity flow network, in which an attacker disables a set of network arcs in order to minimize the maximum profit that can be obtained from shipping commodities across the network. The attacker is assumed to have some budget for destroying (or "interdicting") arcs, and each arc is associated with a positive interdiction expense. In this paper, we examine problems in which interdiction must be discrete (i.e., each arc must either be left alone or completely destroyed), and in which interdiction can be continuous (the capacities of arcs may be partially reduced). For the discrete problem, we describe a linearized model for optimizing network interdiction that is similar to previous studies in the field, and compare it to a penalty model that does not require linearization constraints. For the continuous case, we prescribe an optimal partitioning algorithm along with a heuristic procedure for estimating the optimal objective function value. We demonstrate on a set of randomly generated test data that our penalty model for the discrete interdiction problem significantly reduces computational time when compared to that consumed by the linearization model.

1 Introduction

In the network interdiction problem, a *leader* (also called as an *interdictor*), partially or fully destroys arcs of some network in order to block the *follower*'s (also called *evader* or *enemy*) flows, delay the delivery length of a supply, detect a stealth traverse, or decrease the follower's profit function. When the partial destruction of an arc is allowed, we refer to this action as *continuous* interdiction; otherwise, we call it *discrete* interdiction. We are particularly concerned with a multicommodity network flow interdiction problem in which revenue is generated by the successful shipment of commodities, and costs are incurred due to arc-flow expenses (see Wood [34] for example). In this problem, the follower makes a profit by delivering multiple commodities to certain destinations. On the other hand, the leader attempts to minimize the follower's profit by destroying arcs, subject to a budget limitation on the maximum (weighted) number of arcs that can be interdicted. We consider both discrete and continuous interdiction actions in this paper.

The network interdiction problem has been studied for more than four decades with broad applicability to military and homeland security operations. Without considering interdiction costs, Wollmer [33] proposed an algorithm that discretely interdicts a prescribed number of arcs in a network in order to minimize the follower's maximal flow. Introducing interdiction costs for a planar graph, McMasters and Mustin [26] generalized this approach to solve a typical military operational problem, in which a daily air strike is planned for interdicting an enemy's supply lines. For a similar situation in which no assumption of graph planarity is made, Ghare et al. [16] proposed a branch-and-bound method. This problem was further studied by Wood [34], who provided an integer programming formulation for a discrete interdiction problem. This work also contributed an extension of the model to allow for continuous interdiction, multiple sources and sinks, undirected networks, multiple interdiction resources, and multiple commodities. In particular, the continuous interdiction algorithm exploits the fact that dual variables must be binary-valued at optimal dual extreme point solutions in the context of maximum flows. Since the problem studied in this paper lacks this binary dual extreme point structure, we must explore a different approach.

Fulkerson and Harding [15] considered maximizing the shortest source-sink path in the presence of arc-extension costs that have an equivalent context as interdiction costs. They provided a minimum cost network flow formulation for this problem that can be solved via network optimization procedures. Instead of restricting the leader's budget, Golden [18] introduced a length constraint for the shortest path while minimizing the total interdiction cost. He presented a minimum cost network flow problem formulation for this problem. More recently, Israeli and Wood [19] proposed two decomposition algorithms for solving a discrete interdiction case using so-called super-valid inequalities and set covering master problems.

Washburn and Wood [32] considered the case in which the follower selects a single path to traverse, while the leader tries to locate a single sentry device on an arc in order to maximize the probability of detecting the follower's travel. Starting with a two-person zero-sum game formulation, they provided a corresponding maximum flow network problem and an optimal solution recovery scheme for the original game problem. The deterministic interdiction problem of [34] was extended to a stochastic network interdiction problem by Cormican et al. [11], in which the leader minimizes the expected maximum flow of the network under uncertain interdiction success and arc capacities. They provided a two-stage stochastic integer program, and proposed a sequential approximation algorithm for its solution.

Aside from military and homeland security applications, network interdiction models have been used in other areas. Assimakopoulos [5] presented an interdiction model for preventing hospital infections, in which the node set consists of sources, carriers, and entry-points of infecting organisms. Anandalingam and Apprey [3] considered conflict resolution problems that allow multiple followers to react to the leader's decision simultaneously and hierarchically as leaders and followers. They proposed penalty function approaches for solving these problems and presented an application to a water conflict resolution problem for a river that runs through two countries. In this case study, the United Nations plays a role as the leader (or arbitrator) and two countries act as the followers.

Linear multicommodity flow models are well known and have been widely used in practice due to their broad applicability. Assad [4] and Kennington [20] provided early comprehensive surveys on the solution of multicommodity flow problems and their variants. Some important approaches to solving large-scale versions of these problems include dual-ascent [7] and primal-dual algorithms [8], decomposition strategies [23], and basis partitioning approaches [25]. The use of interior-point algorithms for solving multicommodity flow problems is addressed in [24]. Whereas most network interdiction studies are concerned with a single commodity network flow, multicommodity flows are more appropriate in certain situations, as noted in [34]. It is easy to construct scenarios in which an enemy is transporting multiple items through a network, but the best practical use of this research might arise in cases where the follower is trying to ascertain the worst-case scenario that can possibly befall the network under consideration. For instance, multicommodity flow networks are prevalent in airline operations, supply chain networks, and telecommunications applications, and the survivability of these networks is often of paramount importance. One can evaluate the robustness of a multicommodity network by modeling an intelligent leader as a malicious enemy using the techniques developed herein.

In this paper, we consider the case in which the leader minimizes the follower's profit in a multicommodity flow network. While Wood's approach in [34] forms the basis for our line of investigation, we expand on this research in the specific context of multicommodity flow networks, and provide new insights into methods for solving such problems in both discrete and continuous interdiction scenarios.

The remainder of this paper is organized as follows. In Section 2, a formal description of this problem is presented. We first discuss the discrete interdiction case in Section 3 and propose two formulations for its solution. One formulation is an extension of the discussion in Wood [34] that provided a linearization approach for minimizing the follower's maximum flow, while the second introduces a penalty to the follower's objective function that eliminates the leader's variables from the follower's constraints. In Section 4, we consider the continuous interdiction case and present two approaches for optimally solving this problem. One is based on the bilinear programming formulation that can be solved via various solution techniques in the literature, and the other method exploits a tailored partitioning technique that permits an implicit enumeration algorithm. We also propose a heuristic method for solving relatively large-scale problems that are intractable via an exact solution approach. In Section 5, we display detailed computational results for the proposed methods, and we conclude our study in Section 6.

2 Problem Description

In this section, we formally describe the multicommodity flow network interdiction problem (**MFNIP**). Consider a directed graph $G(N, A)$, where N and A denote index sets of nodes and arcs, respectively. Suppose that the follower has a set K of commodities to deliver from their supply nodes to their demand nodes. Each commodity $k \in K$ can have multiple supply nodes, whose set is denoted by S^k. Likewise, let D^k be the set of demand nodes for commodity k. The maximum supply at node $l \in S^k$ is s_l^k, and the maximum demand at node $l \in D^k$ is denoted by d_l^k. For simplicity, we can create dummy arcs with no flow costs or rewards, and very large interdiction costs. This set of arcs allows us to claim that all supplies must be exhausted and all demands must be supplied.

We use a single index $h \in A$ to index our arcs. Each arc has a finite flow capacity $u_h > 0$. The follower receives rewards when delivering commodities to designated destinations. However, conveying commodities through an arc incurs a flow cost. For a compact presentation of these rewards and flow costs, we will use a unified value r_h^k. All values r_h^k for each $h \in A$ and $k \in K$ include a (negative) flow cost component when one unit of commodity k is conveyed through this arc. If the to-node of $h \in A$ is a demand node of commodity k, a positive reward is added to the flow cost when one unit flow of commodity k is made. Finally, note that a positive flow of commodity k on an arc h whose from-node is one of destinations of commodity k implies that the follower uses this destination as a transhipment node to deliver commodity k to another demand node. Hence, we take back the reward that was gained when this commodity was delivered to the from-node by subtracting this reward from r_h^k.

The leader destroys arcs in order to minimize the follower's optimal reward, subject to a budget of B. The complete interdiction of arc $h \in A$ incurs a cost of b_h. We denote the leader's and follower's decision variables by x_h and y_h^k, respectively. Variables x_h $\forall h \in A$ denote the percentage of arc h destroyed by the leader (i.e., the capacity of arc h is reduced by $x_h \times 100\%$, where $x_h = 1$ represents the complete interdiction of h). Follower variables y_h^k represents the follower's flow of commodity $k \in K$ through arc $h \in A$ after interdiction is completed. Let e denote a vector of $|A|$ ones, and let e_h denote a unit coordinate vector of length $|A|$ with a one in the h^{th} position, $\forall h \in A$. Let $f(h)$ and $t(h)$ denote the from-node and to-node of arc $h \in A$, respectively. Also, we will use $FS(i) \equiv \{h \in A : f(h) = i\}$ and $RS(i) \equiv \{h \in A : t(h) = i\}$ to denote forward- and reverse-stars of each node $i \in N$. Finally, let T denote a transpose operator. Now, we may formally state MFNIP as follows.

MFNIP:

$$\underset{x \in X}{\text{Minimize}} \quad \text{maximize} \sum_{h \in A} \sum_{k \in K} r_h^k y_h^k \tag{1a}$$

$$\text{subject to} \quad \sum_{i \in FS(l)} y_i^k - \sum_{j \in RS(l)} y_j^k = 0 \qquad \forall k \in K,\ \forall l \in N \setminus (S^k \cup D^k) \tag{1b}$$

$$\sum_{i \in FS(l)} y_i^k - \sum_{j \in RS(l)} y_j^k = s_l^k \qquad \forall k \in K,\ \forall l \in S^k \tag{1c}$$

$$\sum_{i \in FS(l)} y_i^k - \sum_{j \in RS(l)} y_j^k = -d_l^k \qquad \forall k \in K,\ \forall l \in D^k \tag{1d}$$

$$\sum_{k \in K} y_h^k \le u_h(1 - x_h) \qquad \forall h \in A \tag{1e}$$

$$y_h^k \ge 0 \quad \forall h \in A,\ \forall k \in K, \tag{1f}$$

where

$$X \equiv \left\{ x \in R^{|A|} : \sum_{h \in A} b_h x_h \le B,\ 0 \le x_h \le 1\ \forall h \in A \right\}. \tag{1g}$$

The leader's budget constraint appears in X. The follower's problem, given fixed values of x-variables, is a multicommodity flow problem. Constraints (1b-d) enforce commodity flow balance conditions at each node. The primary difficulty in solving MFNIP arises in the capacity constraints (1e), where the leader controls arc capacities. Note that if $b^T e \le B$, there exists an optimal solution in which $x_h = 1\ \forall h \in A$. Also, if $b_h \le 0$ for some $h \in A$, we can always fix $x_h = 1$. Therefore, without loss of generality, we assume that $b^T e > B$ and $b_h > 0\ \forall h \in A$.

3 Discrete Interdiction

In this section, we assume that arcs are completely destroyed if interdicted, i.e.,

$$x \in X_I \equiv \left\{ x : \sum_{h \in A} b_h x_h \le B,\ x_h \in \{0, 1\}\ \forall h \in A \right\}. \tag{2}$$

Replacing X by X_I from MFNIP, we call the resulting problem **BMFNIP** (binary MFNIP). We present two mixed-integer programming formulations for this problem in the following subsections, in which we exploit the assumption of binariness of the x-variables to overcome the computational difficulties associated with nonlinear programming.

3.1 Integer Bilinear Reformulation

In this subsection, we reformulate BMFNIP as a mixed-integer bilinear programming problem. Recall that given any leader solution $\widehat{x}$, the follower's (or *inner*) problem is a multicommodity flow problem. Since each constraint in (1b), (1c), and (1d) corresponds to a node $l \in N$ and commodity $k \in K$, let π_l^k denote the dual variable associated with one of these constraints. Furthermore, let ϕ_h denote the dual variable associated with (1e) for an arc $h \in A$. Then, we have the following linear dual of the inner problem.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{k \in K} \sum_{l \in S^k} s_l^k \pi_l^k - \sum_{k \in K} \sum_{l \in D^k} d_l^k \pi_l^k + \sum_{h \in A} u_h \phi_h - \sum_{h \in A} u_h \widehat{x}_h \phi_h && \text{(3a)} \\
\text{subject to} \quad & \pi_{f(h)}^k - \pi_{t(h)}^k + \phi_h \geq r_h^k, \ \forall k \in K, \ \forall h \in A && \text{(3b)} \\
& \pi_l^k \ \text{ unrestricted } \forall k \in K, \ \forall l \in N, \quad \phi_h \geq 0 \ \forall h \in A. && \text{(3c)}
\end{aligned}
$$

Let Θ denote the dual feasible region constrained by (3b) and (3c). Then, BMFNIP can be formulated as the following mixed-integer bilinear programming problem:

$$
\begin{aligned}
\textbf{IBLP:} \ \text{Minimize} \quad & \sum_{k \in K} \sum_{l \in S^k} s_l^k \pi_l^k - \sum_{k \in K} \sum_{l \in D^k} d_l^k \pi_l^k + \sum_{h \in A} u_h \phi_h - \sum_{h \in A} u_h x_h \phi_h && \text{(4a)} \\
\text{subject to} \quad & x \in X_I \text{ and } (\pi, \phi) \in \Theta. && \text{(4b)}
\end{aligned}
$$

IBLP can be solved via standard linearization techniques (see [1, 10, 17, 21, 28] for example). One of the simplest techniques is as follows. For each bilinear term $x_h \phi_h \ \forall h \in A$, where x_h is binary and ϕ_h is a nonnegative variable with an upper bound of $\overline{\phi}_h$, we substitute this product as a single variable $w_h = x_h \phi_h$. To enforce this relationship, we add the following constraints to the problem:

$$
\begin{aligned}
& w_h - \phi_h \leq 0 && \text{(5a)} \\
& w_h - \overline{\phi}_h x_h \leq 0. && \text{(5b)}
\end{aligned}
$$

If $x_h = 0$, then w_h cannot be positive, while if $x_h = 1$, then w_h is restricted to be no more than ϕ_h. Since the w-variables only appear in the objective function of IBLP, and since their objective coefficients are negative, then w_h will take on its maximum permissible value, which is given by $x_h \phi_h$. Hence, we need not include the two sets of lower-bounding constraints, $\phi_h + \overline{\phi}_h x_h - w_h \leq \overline{\phi}_h$ and $w_h \geq 0 \ \forall h \in A$, that would usually be required to establish the desired linearization.

Remark 1. Note that in order to apply this linearization technique, we must obtain upper bounds on $\phi_h \ \forall h \in A$. Recall that ϕ_h is a shadow price of the arc capacity constraint $y_h \leq u_h(1 - x_h)$. Increasing an arc capacity by one unit permits at most one more unit of flow through the network. Since $r_h^k \geq 0$ only when the to-node of arc h is a demand node of commodity k, a simple upper bound on ϕ_h can be set as $\overline{M} = \max_{k \in K}\{\overline{M}^k\}$, where $\overline{M}^k = \max_{h \in RS(D_k)}\{r_h^k\}$ for $k \in K$.

However, a well-known consideration in improving the tightness of mixed-integer programming formulations is to use the smallest valid upper bound values obtainable. An intermediate tightening scheme considers each individual commodity $k \in K$ and forms a corresponding graph $G_k(N, A)$ in which the distance (or flow cost) of each arc $h \in A$ is set as r_h^k. Note that the length of a simple path from a supply node $i \in S^k$ to a demand node $j \in D^k$ in the graph G_k represents the profit obtained by delivering one unit of commodity k through this simple path. Let LP_{ij}^k denote the length of a longest simple path, i.e., the maximum unit profit when delivering one unit of commodity k from the supply node i to the demand node j. (This path can be determined in polynomial time due to the absence of positive-cost cycles in G_k, $\forall k \in K$.) If no path from $i \in S^k$ to $j \in D^k$ exists, we simply put $LP_{ij}^k = -\infty$. Then, the total reward increment gained by delivering one more unit of commodity k is at most $\widehat{M}^k = \max_{i \in S^k,\ j \in D^k} \big\{\max\{LP_{ij}^k, 0\}\big\}$. Therefore, we have $\widehat{M} = \max_{k \in K}\{\widehat{M}^k\}$ as a valid upper bound on ϕ_h $\forall h \in A$.

The strongest possible tightening scheme that we practically recommend attempts to determine the largest increase in the objective function value due to an increase specifically in the capacity of arc $h \in A$ when routing commodity $k \in K$ between each origin node $i \in S^k$ to each destination node $j \in D^k$. However, it is not possible to polynomially determine the longest simple path between i and j that uses $h \in A$ using arc costs r_h^k as given in graph G_k (even without positive-cost cycles), unless $P = NP$ (see [13]). Instead, we find the length of the longest path from i to $f(h)$ in G_k ($LP_{i,f(h)}^k$), and the length of the longest path from $t(h)$ to j in G_k ($LP_{t(h),j}^k$). We accordingly compute $\widetilde{M}_h^k = \max_{i \in S^k,\ j \in D^k} \Big\{\max\{LP_{i,f(h)}^k + r_h^k + LP_{t(h),j}^k, 0\}\Big\}$, and if necessary, $\widetilde{M}_h = \max_{k \in K}\{\widetilde{M}_h^k\}$ $\forall h \in A$ and $\widetilde{M} = \max_{h \in A}\{\widetilde{M}_h\}$.

It is not hard to prove that $\overline{M} \geq \widehat{M} \geq \widetilde{M}$; however, the amount of computations required to compute $\overline{M}$ is less than that required to compute $\widehat{M}$, which is slightly less than that required to compute $\widetilde{M}$. We will investigate the computational implications of the use of these bounds in Section 5. □

Using any of the foregoing upper bounds on ϕ as discussed in Remark 1 together with the simple linearization scheme in (5a,b), we can linearize IBLP as follows.

$$\textbf{ILP:}\ \text{Minimize}\quad \sum_{k \in K}\sum_{l \in S^k} s_l^k \pi_l^k - \sum_{k \in K}\sum_{l \in D^k} d_l^k \pi_l^k + \sum_{h \in A} u_h \phi_h - \sum_{h \in A} u_h w_h \tag{6a}$$

$$\text{subject to}\quad x \in X_I \tag{6b}$$

$$(\pi, \phi) \in \Theta \tag{6c}$$

$$\text{Constraints (5a, b)} \quad \forall h \in A. \tag{6d}$$

This problem is a linear mixed-integer programming problem that has $|N||K|+2|A|$ continuous and $|A|$ binary variables, and $|A|(|K|+2)+1$ structural constraints.

3.2 Penalty Formulation

In this subsection, we present an equivalent reformulation to BMFNIP in which the leader's variables appear only in the objective function of the follower's problem and serve to penalize the follower's use of interdicted arcs. Letting M_h^k be some large constant value, consider the following nonlinear penalty formulation:

NPF:

$$\underset{x\in X_I}{\text{Minimize}}\ \text{maximize} \quad \sum_{h\in A}\sum_{k\in K}(r_h^k - M_h^k x_h)y_h^k \tag{7a}$$

$$\text{subject to} \quad \sum_{k\in K} y_h^k \le u_h \qquad \forall h \in A \tag{7b}$$

$$\text{Flow balance constraints (1b, c, d, f).} \tag{7c}$$

Lemma 1. Let $M_h^k > \widetilde{M}_h^k$ as described in Remark 1, and for any $\widehat{x} \in X_I$, let $\overline{y}(\widehat{x})$ be an optimal solution to the follower's problem in NPF. Then for any $\widehat{x} \in X_I$ we have that $\overline{y}(\widehat{x})_h^k = 0\ \forall k \in K$ if $\widehat{x}_h = 1$.

Proof. If $\overline{y}(\widehat{x})_h^k > 0$ for some $k \in K$, then we have a positive flow of commodity k along paths that contain arc h. Note that the cumulative reward of one unit of commodity k flow along any path using arc h cannot exceed $\widetilde{M}_h^k$. Since $M_h^k > \widetilde{M}_h^k$, a strictly better feasible solution exists in which $y(\widehat{x})_h^k = 0$, and in which flows are moved from paths containing arc h to the dummy arcs in the network that represent the shortage of commodity flows. This completes the proof. □

Proposition 1. Let $M_h^k > \widetilde{M}_h^k$. Given any $\widehat{x} \in X_I$, let BMFNIP($\widehat{x}$) and NPF($\widehat{x}$) denote the follower's problems in BMFNIP and NPF, respectively. Then, BMFNIP($\widehat{x}$) has an optimal solution $y(\widehat{x})$ if and only if $y(\widehat{x})$ is optimal to NPF($\widehat{x}$). Moreover, the optimal objective function value to BMFNIP($\widehat{x}$) matches that of NPF($\widehat{x}$).

Proof. We first show that the optimal objective value to NPF($\widehat{x}$) is no more than the optimal objective value to BMFNIP($\widehat{x}$). Let $y(\widehat{x})$ be an optimal solution to BMFNIP($\widehat{x}$) with objective value $v^\star$. Since NPF($\widehat{x}$) is a relaxation of BMFNIP($\widehat{x}$), $y(\widehat{x})$ is also feasible to NPF($\widehat{x}$). Also, we have $\sum_{h\in A}\sum_{k\in K}(r_h^k - M_h^k\widehat{x}_h)y(\widehat{x})_h^k = v^\star$ since $\sum_{h\in A}\sum_{k\in K} r_h^k y(\widehat{x})_h^k = v^\star$ and Lemma 1 guarantees that $\sum_{h\in A}\sum_{k\in K} M_h^k\widehat{x}_h y(\widehat{x})_h^k = 0$. Hence, the optimal objective value to NPF($\widehat{x}$) does not exceed $v^\star$.

Next, we show that the optimal objective value for BMFNIP($\widehat{x}$) is no more than the optimal objective value to NPF($\widehat{x}$), and thus, that these values are equal. Let $y(\widehat{x})$ be an optimal solution to NPF($\widehat{x}$) having objective value $v^\star$. Since Lemma 1 guarantees that $y(\widehat{x})^k_h = 0$ for $\widehat{x}_h = 1$, $y(\widehat{x})$ is a feasible solution to BMFNIP($\widehat{x}$) with objective value $\sum_{h\in A}\sum_{k\in K} r^k_h y(\widehat{x})^k_h = v^\star$, which provides a lower bound on the optimal objective value of BMFNIP($\widehat{x}$). This completes the proof. □

Remark 2. Observe that if $M^k_h = \widetilde{M}^k_h$ for each $h \in A$ and $k \in K$, we can only guarantee that the optimal objective function value for NPF matches that of BMFNIP. From the proof of Proposition 1, a feasible solution can be obtained by simply decreasing follower flows on interdicted arcs and restoring the remaining flows on the graph (perhaps via dummy arcs) to satisfy the flow balance constraints. Hence, we can use $\widetilde{M}^k_h$ as our values of M^k_h, $\forall h,\ k$. Naturally, if $\widetilde{M}^k_h$ has not been computed, we can validly use any upper bound on $\widetilde{M}^k_h$ in its place, such as $\widehat{M}^k$ or $\overline{M}^k$ as given in Remark 1. □

Given $\widehat{x} \in X_I$, NPF($\widehat{x}$) is a linear program with a bounded feasible region. Hence, there exists an extreme point optimal solution to NPF($\widehat{x}$). A key feature of this problem is that the feasible region of NPF(x) does not change for different values of x. Hence, when solving NPF, we could directly apply the Benders cutting plane algorithm, in which the master program is a mixed-integer programming problem having binary variables x and an additional continuous variable, and the corresponding subproblem is a continuous multicommodity flow problem. However, note that we would need to solve a mixed-integer master problem in which Benders cuts are added at each iteration. Therefore, the cutting plane algorithm tends to be slow for this type of problems (see [27]). (We observed a slow convergence of this behavior in a preliminary computational investigation.)

Instead of solving NPF via the cutting plane method, we investigate the dual of NPF($\widehat{x}$):

$$\text{Minimize} \quad \sum_{k\in K}\sum_{l\in S^k} s^k_l \pi^k_l - \sum_{k\in K}\sum_{l\in D^k} d^k_l \pi^k_l + \sum_{h\in A} u_h \phi_h \tag{8a}$$

$$\text{subject to} \quad \pi^k_{f(h)} - \pi^k_{t(h)} + \phi_h \geq r^k_h - M^k_h \widehat{x}_h,\ \forall k \in K,\ \forall h \in A \tag{8b}$$

$$\pi^k_l \ \text{ unrestricted } \forall k \in K,\ \forall l \in N, \quad \phi_h \geq 0\ \forall h \in A. \tag{8c}$$

Releasing $\widehat{x}$ as variables x, we have the following mixed-integer linear programming problem that is equivalent to NPF.

$$\textbf{PF:}\ \text{Minimize}\quad \sum_{k\in K}\sum_{l\in S^k} s_l^k\pi_l^k - \sum_{k\in K}\sum_{l\in D^k} d_l^k\pi_l^k + \sum_{h\in A} u_h\phi_h \tag{9a}$$

$$\text{subject to}\quad \pi_{f(h)}^k - \pi_{t(h)}^k + \phi_h + M_h^k x_h \geq r_h^k,\ \forall k\in K,\ \forall h\in A \tag{9b}$$

$$\pi_l^k\ \text{ unrestricted } \forall k\in K,\ \forall l\in N,\quad \phi_h\geq 0\ \forall h\in A \tag{9c}$$

$$x\in X_I. \tag{9d}$$

Constraints (9b) have the interpretation of deactivating a dual constraint corresponding to an arc $h\in A$ when arc h has been interdicted. Observe that this strategy cannot simply be applied to the continuous interdiction case, since Proposition 1 does not generally hold true for $x\in X$. Unlike IBLP as given in (4), we do not need linearization processes for solving PF. Note that this problem has $|N||K|+|A|$ continuous and $|A|$ binary variables, and $|A||K|+1$ structural constraints. This is a slightly smaller formulation than ILP, which has $|A|$ additional continuous variables and $2|A|$ additional constraints, due to the need for linearization. The relative tightness of these two formulations is established in the following proposition.

Proposition 2. Consider the linear programming relaxation to ILP (called ILPLP) and the linear programming relaxation to PF (called PFLP), and let $\nu(\bullet)$ denote the optimal objective function value of problem $\bullet$. Then if the upper bounds to ϕ_h, $\forall h\in A$ in ILP and the values M_h^k, $\forall h\in A,\ k\in K$ are all equal to some constant value μ, then $\nu(\text{ILPLP}) = \nu(\text{PF})$.

Proof. Consider any solution $(\widehat{x},\widehat{\pi},\widehat{\phi},\widehat{w})$ to ILPLP. A solution $(\widetilde{x},\widetilde{\pi},\widetilde{\phi})$ to PFLP can be constructed as follows. First, set $\widetilde{x}=\widehat{x}$ and $\widetilde{\pi}=\widehat{\pi}$. Observe that for any optimal solution to ILPLP, we have $\widehat{w}_h=\min\{\widehat{\phi}_h,\mu\widehat{x}_h\}$ $\forall h\in A$. If $\widehat{\phi}_h\leq\mu\widehat{x}_h$ for $h\in A$, then set $\widetilde{\phi}_h=0$. In this case, note that all constraints (9b) corresponding to h, $\forall k\in K$, are satisfied since $\widetilde{\pi}_{f(h)}^k-\widetilde{\pi}_{t(h)}^k+\mu\widetilde{x}_h\geq\widehat{\pi}_{f(h)}^k-\widehat{\pi}_{t(h)}^k+\widehat{\phi}_h\geq r_h^k$. Also, the contribution of $u_h(\widehat{\phi}_h-\widehat{w}_h)$ to the objective of ILPLP and the contribution of $u_h\widetilde{\phi}_h$ to the objective of PFLP are both zero. On the other hand, if $\widehat{\phi}_h>\mu\widehat{x}_h$, then set $\widetilde{\phi}_h=\widehat{\phi}_h-\mu\widehat{x}_h$. Constraints (9b) are easily shown to be satisfied for this h and all $k\in K$ as well by definition of $\widetilde{\phi}_h$, and $u_h(\widehat{\phi}_h-\widehat{w}_h)$ is contributed to both objective functions. Hence, $\nu(\text{PF})\leq\nu(\text{ILP})$. To see that $\nu(\text{PF})=\nu(\text{ILP})$, consider an optimal solution $(\widetilde{x},\widetilde{\pi},\widetilde{\phi})$ to PFLP, and note that a feasible solution $(\widehat{x},\widehat{\pi},\widehat{\phi},\widehat{w})$ with the same objective function value to ILPLP can be obtained by setting $\widehat{x}=\widetilde{x}$, $\widehat{\pi}=\widetilde{\pi}$, $\widehat{\phi}=\widetilde{\phi}+\mu\widetilde{x}$, and $\widehat{w}=\mu\widetilde{x}$. A similar argument reveals that this is a feasible solution to ILPLP with the same objective function value. This completes the proof. $\square$

Remark 3. Proposition 2 indicates that the two formulations for BMFNIP are equally tight when a constant upper bound of μ is used for each formulation. This situation would arise if we decide to use $\overline{M}$ as determined by Remark 1. However, suppose we use the tightest possible bounds available, which are $\widetilde{M}_h$ for $h \in A$ as upper bounds to ϕ_h in ILP, and $\widetilde{M}_h^k$ for M_h^k in PF. Since $\widetilde{M}_h \geq \widetilde{M}_h^k$ for each $h \in A$ and $k \in K$, the linear programming relaxation to PF becomes at least as tight as the linear programming relaxation to ILP. The key observation regarding these two models is that not only is PF smaller than ILP (in terms of the number of variables and constraints), but it permits the use of upper bounds on each individual combination of $h \in A$ and $k \in K$, thereby permitting us to tighten the feasible region beyond what can be accomplished by a similar strategy for ILP. $\square$

4 Continuous Interdiction

In this section, we examine exact and heuristic procedures for solving MFNIP as given in (1). Since the leader's variables are continuous, we can no longer resort to standard linearization procedures to solve the interdiction problem by a single integer program. Hence, it appears that the continuous interdiction problem is actually more difficult to solve than the discrete interdiction problem. However, since $b^T e > B$ by assumption, we can state the following proposition without proof, which will ultimately allow us to decompose MFNIP into a series of linear integer programming problems.

Proposition 3. There exists an optimal solution $(x^\star, y^\star)$ to MFNIP such that $b^T x^\star = B$.

Hence, from now on, we replace the knapsack inequality in X by an equality, and denote this restricted feasible region by X_C (i.e., $X_C \equiv \{x : b^T x = B,\ 0 \leq x \leq e\}$). We examine exact and heuristic methods for MFNIP in the subsequent subsections.

4.1 Exact Algorithm for MFNIP

Applying the same approach as in Section 3.1, we have the following (continuous) bilinear program.

$$\begin{aligned} \textbf{BLP:}\ \text{Minimize} \quad & g(x, \pi, \phi) && \text{(10a)} \\ \text{subject to} \quad & x \in X_C \text{ and } (\pi, \phi) \in \Theta, && \text{(10b)} \end{aligned}$$

where $g(x, \pi, \phi)$ is stated in (4a). Note that fixing x yields a linear program in terms of (π, ϕ), and vice versa. Hence, the problem BLP has an optimal solution $(x^\star, \pi^\star, \phi^\star)$ such that $x^\star$ and $(\pi^\star, \phi^\star)$ are extreme points of X_C and Θ, respectively (see [29] for example).

BLP can be solved via various algorithms for disjointly constrained bilinear programming problems (see [2, 6, 22, 29, 31] for examples). One notable algorithm is presented by Alarie et al. [2], who combine the concavity cuts of Tuy [30] with the branch-and-bound method of Audet et al. [6]. In the first phase of this approach, concavity cuts (sometimes referred as Tuy cuts) are generated at a nondegenerate extreme point of X_C in order to remove a part of the cone defined by binding constraints, in the hope that these cuts eliminate as many ineligible extreme points as possible. If this phase cannot find an optimal solution, a branch-and-bound phase is executed to complete the procedure.

While this procedure converges in finite time to an optimal solution, it suffers from two primary drawbacks. One, the procedure is computationally slow due to the necessary identification of adjacent extreme points and elimination of suboptimal extreme points via concavity cuts. Two, the cutting plane generation phase of the procedure is practically quite difficult to implement. While this algorithm is one of the best alternatives available for general bilinear programming problems, we develop an alternative approach for the specific class of problems encountered in this paper.

As an alternative strategy, we present a partitioning algorithm that partitions MFNIP into $|A|$ subproblems. First, consider the following lemma.

Lemma 2. Consider the polyhedral set X_C. Then, for each extreme point $\widehat{x}$ of X_C, there exists a single basic variable $\widehat{x}_r$ such that $\widehat{x}_r \in [0, 1]$, while all other variables are nonbasic at their lower bounds of zero or upper bounds of one.

Proof. See [12]. $\square$

From Lemma 2, if we designate x_r as a basic variable that can have any value between 0 and 1, then the remaining $|A| - 1$ nonbasic variables must have binary values in order for this point to be an extreme point of X_C. Suppose that x_r is designated as a basic variable, and consider the following subproblem.

$$\begin{aligned}
\textbf{SP: } \text{Minimize} \quad & g(x, \pi, \phi) && \text{(11a)}\\
\text{subject to} \quad & b^T x = B && \text{(11b)}\\
& 0 \le x_r \le 1 && \text{(11c)}\\
& x_h \in \{0, 1\} \quad \forall h \in A \setminus \{r\} && \text{(11d)}\\
& (\pi, \phi) \in \Theta. && \text{(11e)}
\end{aligned}$$

From the equality in (11b), we have

$$x_r = \frac{B - \sum_{h \in A \setminus \{r\}} b_h x_h}{b_r}. \tag{12}$$

Substituting x_r in (11a) with (12) and noting that $B - b_r \leq \sum_{h\in A\setminus\{r\}} b_h x_h \leq B$, we have the following mixed-integer bilinear program, which does not contain the continuous variable x_r.

$$\begin{aligned}
\textbf{SP}(\boldsymbol{r})\textbf{:}\ \text{Minimize}\quad & \sum_{k\in K}\sum_{l\in S^k} s_l^k \pi_l^k - \sum_{k\in K}\sum_{l\in D^k} d_l^k \pi_l^k + \sum_{h\in A} u_h \phi_h - \sum_{h\in A\setminus\{r\}}^{n} u_h x_h \phi_h \\
& + \left(\frac{u_r \phi_r}{b_r}\right) \sum_{h\in A\setminus\{r\}} b_h x_h - \frac{B u_r \phi_r}{b_r} && \text{(13a)} \\
\text{subject to}\quad & \sum_{h\in A\setminus\{r\}} b_h x_h \geq B - b_r && \text{(13b)} \\
& \sum_{h\in A\setminus\{r\}} b_h x_h \leq B && \text{(13c)} \\
& x_h \in \{0,1\} \quad \forall h \in A\setminus\{r\} && \text{(13d)} \\
& (\pi, \phi) \in \Theta. && \text{(13e)}
\end{aligned}$$

Using the linearization method in (5) to replace the nonlinear terms $x_h\phi_h$ with w_h $\forall h \in A\setminus\{r\}$, and $x_h\phi_r$ with v_h $\forall h \in A\setminus\{r\}$, we have the following linear mixed-integer program.

$$\begin{aligned}
\textbf{MILP}(\boldsymbol{r})\textbf{:}\ \text{Minimize}\quad & \sum_{k\in K}\sum_{l\in S^k} s_l^k \pi_l^k - \sum_{k\in K}\sum_{l\in D^k} d_l^k \pi_l^k + \sum_{h\in A} u_h \phi_h - \sum_{h\in A\setminus\{r\}} u_h w_h \\
& + \left(\frac{u_r}{b_r}\right) \sum_{h\in A\setminus\{r\}} b_h v_h - \frac{B u_r \phi_r}{b_r} && \text{(14a)} \\
\text{subject to}\quad & \text{Constraints (13b)} - \text{(13e)} && \text{(14b)} \\
& w_h - \phi_h \leq 0 \quad \forall h \in A\setminus\{r\} && \text{(14c)} \\
& w_h - \overline{\phi}_h x_h \leq 0 \quad \forall h \in A\setminus\{r\} && \text{(14d)} \\
& \phi_r + \overline{\phi}_r x_h - v_h \leq \overline{\phi}_r \quad \forall h \in A\setminus\{r\} && \text{(14e)} \\
& v_h \geq 0 \quad \forall h \in A\setminus\{r\}. && \text{(14f)}
\end{aligned}$$

Once again, the lower bounds on w_h, $\forall h \in A\setminus\{r\}$, have been removed from the formulation. Similarly, in the linearization of $v_h = \phi_r x_h$, $\forall h \in A\setminus\{r\}$, we need not state the upper bounding constraints on v_h, since these variables appear only in the objective function multiplied by a positive number, and in the bounding constraints. Hence, the equations $v_h \leq \overline{\phi}_r x_h$ and $v_h \leq \phi_r$ that would normally be included in a linearization are removed in this formulation.

Note that given a solution to SP(r), the value of x_r can be recovered by (12). Let $[x(r), \pi(r), \phi(r)]$ and $z(r)$ denote an optimal solution and optimal objective value of SP(r), respectively. Then, $r^\star \in \operatorname{argmin}_{r\in A}\{z(r)\}$ yields an overall optimal solution $x(r^\star)$ and objective value $z(r^\star)$. The partitioning algorithm for the continuous interdiction problem is given as follows.

Partitioning Algorithm (PA)
Step 0. Let $R = A$. Set the incumbent solution $(\overline{x}, \overline{\pi}, \overline{\phi})$ as a blank vector and the incumbent objective value as $\overline{v} = \infty$.
Step 1. If $R = \emptyset$, terminate the algorithm with the incumbent solution. Otherwise, select any $r \in R$ and put $R = R \setminus \{r\}$.
Step 2. Solve MILP(r) and obtain the optimal solution $(\widehat{x}, \widehat{\pi}, \widehat{\phi})$ with objective function value $\widehat{v}$. If $\hat{v} < \overline{v}$, then set $\overline{v} = \hat{v}$ and $(\overline{x}, \overline{\pi}, \overline{\phi}) = (\widehat{x}, \widehat{\pi}, \widehat{\phi})$. Return to Step 1.

Remark 4. Note that when solving MILP(r), the incumbent objective value $\overline{v}$ can be used as an initial upper bound of the subproblem. Hence, if we have a relatively small incumbent objective value in early stages of the procedure, the solution time for the remaining subproblems could be reduced since the subproblems can be fathomed quickly using this upper bound. Accordingly, we can make an ordering of the subproblems as follows. For each $r \in A$, we solve a linear programming (LP) relaxation of MILP(r) to obtain its optimal value $v_{LP}(r)$. Then, we create a sequence of indices $r_1, \ldots, r_{|A|}$ so that $v_{LP}(r_i) \leq v_{LP}(r_j)$ if $i < j$. The partitioning algorithm is implemented in the order of this sequence, in anticipation of improving the upper bound $\overline{v}$ as quickly as possible. We call this approach the Ordered Partitioning Algorithm (**OPA**), and investigate its efficacy in our computational study. □

4.2 Heuristic for Continuous Interdiction Problems

When we apply the aforementioned partitioning algorithm, we may have to solve $|A|$ mixed-integer programming problems in order to find an exact solution in the worst case. Therefore, these methods may not be suitable for solving large-scale problems. Hence, we suggest the following heuristic method for relatively large-scale problems.

The heuristic approach that is tailored to this problem is to start with no interdiction performed and obtain the follower's objective. We can then examine what happens to the follower's optimal solution by interdicting each arc, one at a time. If an arc cannot be fully interdicted, then we examine the impact on the follower's optimal solution when the remaining budget is spent on interdicting the arc. A logical choice for interdiction would then be an arc $\hat{h} \in A$ that exhibits best ratio of objective decrease to budget consumed when interdicted. (This ratio is denoted by σ in the formal description below.) This process continues until the budget has been exhausted. This algorithm is formally described as follows.

Heuristic Algorithm (HA)

Step 0. Initialize $\overline{x}_h = 0\ \forall h \in A$, define the remaining budget as $\overline{B} = B$, and define the candidate set of arcs that can be interdicted as $\overline{A} = A$.

Step 1. Set $\sigma = 0$, $R = \overline{A}$, and solve the follower's problem given $\overline{x}$. Let v be the optimal objective value of the follower's problem.

Step 2. If $R = \emptyset$, go to Step 4. Otherwise, choose any $h \in R$ and proceed to Step 3.

Step 3. Set $\overline{x}_h = \min\{b_h, \overline{B}\}/b_h$. Solve the follower's problem to obtain the optimal objective value v_h. If $(v - v_h)/\min\{b_h, \overline{B}\} > \sigma$, put $\widehat{h} = h$ and $\sigma = (v - v_h)/\min\{b_h, \overline{B}\}$. Reset $\overline{x}_h = 0$ and return to Step 2.

Step 4. Fix $\overline{x}_{\widehat{h}} = \min\{b_{\widehat{h}}, \overline{B}\}/b_{\widehat{h}}$, update $\overline{B} = \overline{B} - \min\{b_{\widehat{h}}, \overline{B}\}$, and remove $\widehat{h}$ from $\overline{A}$. If $\overline{B} = 0$, terminate with the heuristic solution $\overline{x}$. Otherwise, return to Step 1.

5 Computational Study

In this section, we describe the results of a computational study of our proposed algorithms. In order to generate interdiction problems, we randomly generated linear multicommodity flow instances using a popular generator called *Mnetgen*[1] (see [9, 14]). Since MFNIP is substantially more difficult to solve than linear multicommodity flow instances, our test instances are smaller than those analyzed in [9, 14]. The first set (S1) of test problems contains nine network topologies, and the second set (S2) consists of six topologies. We investigate the performance of ILP and PF on the instances in S1, and analyze PA, OPA, and the heuristic HA on the instances in S2. Each topology is characterized by $|N|$ and $|K|$. For S1, we have $|N| \in \{16, 32, 64\}$ and $|K| \in \{2^1, 2^2, ..., |N|/4\}$, while in S2, we examine combinations of $|N| \in \{8, 12, 16\}$ and $|K| \in \{2, 4\}$. As in [9, 14], half of N are designated as supply nodes, and the other half are demand nodes. Also, we select flow costs from the interval $[1, 10]$, and set the length of the longest chain in a network as $|N|/2$. After creating these linear multicommodity flow problems, we randomly generate rewards and interdiction budgets, and set the interdiction costs equal to the arc capacities. Specifically, the budget value B is selected between 20% and 50% of the total arc capacity in the corresponding network. Moreover, since instances generated by Mnetgen have only flow costs, we randomly generate rewards large enough so that even the delivery through the longest chain incurs positive profit (recall that these values together with flow costs are incorporated into the unit flow profits r_i^k). We generate ten instances for each topology, and report the average performance of our algorithms over these instances. These test problems are summarized in Table 1.

[1]http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html.

In the above test instances, there always exists an arc that directly connects a supply node to a demand node since each node in the network is either a supply or a demand node for all commodities. Hence, to examine the performance of our algorithms on networks that contain transshipment nodes, we also generated two additional sets of random instances from grid network topologies, in which the left-most and right-most nodes in the grid are set as origin and destination nodes, respectively. Furthermore, each commodity has a single origin-destination pair corresponding to the left-most and right-most nodes on a common row of the grid. Thus, the number of transshipment nodes in a origin-destination path is at least the number of columns in the grid minus two. The first set (G1) of test problems contains six network topologies, while the second set (G2) consists of three topologies. Similar to S1 and S2, test problems in G1 are used to investigate ILP and PF, while those in G2 are for PA, OPA, and HA. For each topology, we generated five instances with flow costs and supplies generated from ranges [5,10] and [5,15], respectively. For test problems in G2, a single flow cost per arc is assigned regardless of commodity, while different rewards are generated for demand nodes. Furthermore, for test instances in G1, arc capacities and interdiction costs are independently generated on the intervals [10,20] and [5,10], whereas arc capacities for instances in G2 are selected from the set $\{10, 20, 30\}$ and interdiction costs are set equal to the capacities. The rewards and interdiction budget are randomly generated to guarantee that there exists a path that yields a positive profit to the follower for each commodity, and to ensure that the leader cannot simultaneously interdict all arcs in any origin-destination cut-set. These test problems are summarized in Table 2.

All algorithms were coded in C++ using CPLEX 8.1 Concert Technology with a default solution setting, and were run on a Sun Fire 280R server with 900 Mhz UltraSPARC-III CPU.

5.1 Computational Results for Discrete Interdiction

In this subsection we summarize computational results for discrete interdiction. For both ILP and PF, we experiment with three upper bounds on ϕ_h including $\overline{M}$, $\widehat{M}$ and $\widetilde{M}_h$. Furthermore, we employ three tighter penalty values $\overline{M}^k$, $\widehat{M}^k$, and $\widetilde{M}_h^k$ for PF. Our first experiment analyzes the efficiency of these nine mixed-integer programming models, including the pre-processing time required to find the various M-values. Table 3 displays the average CPU times (in seconds) required to solve instances in S1 using each of these nine implementations.

A time limit for solving a single instance is set as 7200 seconds, and exceeding this limit is considered as a failure. If there exists any failure when solving an instance of a topology, the number of instances of that topology solved within the time limit is displayed in parentheses

Table 1: Summary of Test Problems from General Network Topologies.

Set	Topology	Number of Nodes ($\lvert N \rvert$)	Number of Commodities ($\lvert K \rvert$)
S1	DTP1	16	2
	DTP2	16	4
	DTP3	32	2
	DTP4	32	4
	DTP5	32	8
	DTP6	64	2
	DTP7	64	4
	DTP8	64	8
	DTP9	64	16
S2	CTP1	8	2
	CTP2	8	4
	CTP3	12	2
	CTP4	12	4
	CTP5	16	2
	CTP6	16	4

Table 2: Summary of Test Problems from Grid Network Topologies.

Set	Topology	Number of Rows	Number of Columns	Number of Arcs
G1	GTP1	4	6	80
	GTP2	4	10	136
	GTP3	6	6	127
	GTP4	6	10	220
	GTP5	8	6	136
	GTP6	8	10	240
G2	GTP7	4	6	44
	GTP8	4	8	102
	GTP9	4	10	99

instead of an average CPU time. For example, ILP in conjunction with the upper bound $\overline{M}$ terminates within 7200 seconds for six out of ten instances on topology DTP8.

As expected, Table 3 shows that PF consumes less CPU time than does ILP. For example, when solving instances in DPT7, the procedure PF along with $\overline{M}$ consumes only an average of 7.67% of CPU time that was required by ILP with the same upper bound $\overline{M}$. Moreover, ILP does not yield optimal solutions for many of the DTP8 and DTP9 instances within the 7200 second time limit. We also observe that a tighter bound plays a beneficial role when solving relatively large-scale problems. For example, the average solution time consumed by ILP in concert with $\widetilde{M}_h$ is about 50% of the average CPU time of ILP with $\overline{M}$ for solving DTP7. However, the advantage of a tighter bound is reduced when implementing PF. Further tightened bounds (or penalties) $\overline{M}^k$, $\widehat{M}^k$, and $\widetilde{M}_h^k$ exhibit slightly faster CPU times than those of $\overline{M}$, $\widehat{M}$, and $\widetilde{M}_h$, as shown in columns 5–10 in Table 3. While the reduction of computational effort is insignificant for small-sized topologies such as DTP1–DTP7, we observe greater benefits for larger-scale instances in DTP8 and DTP9. Overall, the PF procedure with the tightest bound $\widetilde{M}_h^k$ yields the most promising performance. This advantage occurs despite the fact that the test instances have $|N|/2$ supply and $|N|/2$ demand nodes, and hence the computation of these bounds can become expensive. However, the time required to compute these bounds is small compared to the time required to solve the mixed-integer programs. In particular, it takes less than one second to compute the M-values for DTP1–DTP5, and computing $\widetilde{M}_h$ for the largest instances DTP9 consumes only 11.4 seconds on average.

When these formulations are tested on grid networks, the use of the tightest possible bounds in these models becomes even more important (see Table 4). Without time limits, we experimented with using $\widehat{M}$ and $\widetilde{M}_h$ for ILP and $\widehat{M}^k$ and $\widetilde{M}_h^k$ for PF. For ILP, using the tighter bound $\widetilde{M}_h$ consumes 17% of CPU time required when using $\widehat{M}$ on average. When $\widetilde{M}_h^k$ is employed, PF consumes 7% of the CPU time required for the weaker bound $\widehat{M}^k$ on average. Also, we observe that the relative computational improvement of using PF with $\widetilde{M}_h^k$ over $\widehat{M}^k$ seems to increase as the difficulty of the problem instance increases. This improvement is most dramatic on the GTP6 instances, in which PF along with $\widetilde{M}_h^k$ uses only an average 5% of the CPU time consumed by PF with $\widehat{M}^k$.

Based on these observations, we recommend using PF in concert with the tightest bound $\widetilde{M}_h^k$, especially for the solution of large-scale problems. Furthermore, we expect that this tighter bound greatly enhances the performance of PF when the problem has relatively a large number of transshipment nodes.

Table 3: Average CPU Times for Solving Instances in S1.

Topology	ILP			PF					
	$\overline{M}$	$\widehat{M}$	$\widetilde{M}_h$	$\overline{M}$	$\widehat{M}$	$\widetilde{M}_h$	$\overline{M}^k$	$\widehat{M}^k$	$\widetilde{M}_h^k$
DTP1	0.31	0.31	0.33	0.19	0.16	0.18	0.18	0.17	0.18
DTP2	0.6	0.62	0.66	0.33	0.3	0.34	0.32	0.31	0.32
DTP3	11.7	13.0	10.6	2.6	2.3	2.2	2.3	2.3	2.2
DTP4	10.5	9.7	10.2	2.4	2.5	2.5	2.3	2.1	2.4
DTP5	20.9	22.1	19.3	8.2	7.6	6.9	6.9	6.8	7.1
DTP6	69.3	73.5	61.3	5.1	5.2	5.8	4.9	5.0	5.8
DTP7	226.9	198.4	114.7	17.4	17.8	19.0	19.4	18.5	18.9
DTP8	(6)	(6)	(2)	288.8	231.3	214.6	236.9	222.8	190.7
DTP9	(1)	(2)	(0)	1352.4	1337.6	1274.1	1342.3	1197.2	1197.2

Table 4: Average CPU Times for Solving Instances in G1.

Instance	ILP		PF	
	$\widehat{M}$	$\widetilde{M}_h$	$\widehat{M}^k$	$\widetilde{M}_h^k$
GTP1	14.7	7.9	4.0	4.0
GTP2	55.1	38.6	36.3	10.2
GTP3	847.2	46.2	137.8	13.6
GTP4	1356.3	367.8	367.2	57.2
GTP5	831.7	40.8	17.4	15.0
GTP6	7879.8	1355.2	3144.6	165.2

Table 5: Average CPU Times for Solving Instances in S2.

Topology	PA			OPA		
	$\overline{M}$	$\widehat{M}$	$\widetilde{M}_h$	$\overline{M}$	$\widehat{M}$	$\widetilde{M}_h$
CTP1	8.43	7.92	5.58	8.47	7.88	5.60
CTP2	3.61	3.42	3.20	3.73	3.53	3.28
CTP3	59.65	53.25	38.79	58.80	52.08	37.32
CTP4	194.32	172.69	131.98	190.73	170.44	125.08
CTP5	1029.09	1064.39	653.41	1321.80	1049.91	627.43
CTP6	(9)	(9)	1234.04	(9)	(9)	1232.14

5.2 Computational Results for Continuous Interdiction

For the continuous interdiction problem, we implemented PA and OPA in conjunction with different upper bounds, $\overline{M}$, $\widehat{M}$, and $\widetilde{M}_h$. In Table 5, we report average CPU times for each topology, which include bound computing and ordering times in addition to subproblem solution times. Our first observation is that tighter upper bounds consistently help reduce computational effort. Since continuous test topologies are smaller than discrete ones, bound computation times are negligible (all bounds are obtained within 0.05 seconds in every case). Note that both PA and OPA in conjunction with $\overline{M}$ and $\widehat{M}$ fail to solve one instance in CTP6 within the time limit, whereas PA and OPA with $\widetilde{M}$ terminates within the time limit in all ten instances. When considering only solved problems in CTP6, we observe that employing $\widetilde{M}_h$ over $\overline{M}$ reduces CPU times by 41% for both PA and OPA on average. The overall effectiveness of OPA on improving the fathoming process is slight, which is explained by our observation that each subproblem MILP(r), for each $r \in A$, solved in OPA tends to yield very similar optimal objective values.

We also report average CPU times for solving instances taken from grid networks in Table 6. Note that among tested topologies, GTP8 is most difficult to solve due to the large number of arcs as well as its relatively high arc density. Similar to the observation above, the use of tighter bounds reveals a modest computational advantage over the use of relatively loose bounds, and OPA provides no significant computational advantage or disadvantage over the use of PA.

Finally, we observed that the proposed heuristic method HA consumes less than one second for solving all instances. Furthermore, HA yields optimal solutions for 19 instances out of 60 CTP instances. Nonetheless, we observed fairly large optimality gaps for other instances

Table 6: Average CPU Times for Solving Instances in G2.

Instance	PA		OPA	
	$\widehat{M}$	$\widetilde{M_h}$	$\widehat{M}$	$\widetilde{M_h}$
GTP7	39.4	34.6	39.4	36.0
GTP8	1056.8	959.4	989.4	981.0
GTP9	282.0	247.6	284.4	249.6

when applied to the general network topologies. Average optimality gaps are 86.0%, 13.2%, 30.7%, 75.4%, 31.9%, and 33.9% for topologies CTP1–CTP6, respectively. However, when this heuristic is applied to instances from grid network topologies GTP7–GTP9, HA yields average optimality gaps of just 6.1%, 1.6%, and 5.3% for topologies GTP7–GTP9. We thus conclude that HA performs well when applied to problems having a relatively simple network structure, but is inconsistent for more complex topologies such as those generated by Mnetgen.

6 Conclusions

In this paper, we proposed two mixed-integer programming formulations, ILP and PF, for the discrete interdiction problem by exploiting upper bounds on the follower's marginal profit. While ILP employs standard linearization techniques to linearize bilinear terms that appear in the objective function of the follower's dual problem, we demonstrated how to avoid this nonlinearity in PF via an equivalent bilevel program in which the interdictor's decision does not perturb the follower's feasible region. We also proved that two formulations are equally tight in the sense that linear programming relaxations yield the same optimal objective value when the same bound on the marginal profit is used. However, since tighter upper bounds can be obtained efficiently, we can tighten PF beyond ILP, and our computational study confirms the substantial computational savings that result from this tightening.

For continuous interdiction, we devised a partitioning algorithm, which uses the fact that an extreme point of the interdictor's feasible region has at most one fractional value. Our computational results demonstrate that the continuous interdiction problem is difficult to solve by our algorithm, and hence becomes intractable as the problem size increases even for problems having a relatively simple structure. Hence, we proposed a heuristic technique that often seems to find the optimal solution within a second and yields small optimality gaps for grid-structured problems. However, the routine appears to be much less reliable for the general topologies generated by Mnetgen.

The most pressing need for future research seems to regard the identification of more effective algorithms for addressing the continuous interdiction problem. A future study might address the continued development of heuristics for the continuous interdiction case, and also for large-scale discrete interdiction instances. Another possible avenue would be to investigate special structures inherent in the continuous bilinear interdiction problem, and explore possibilities for exploiting these special structures within more general global optimization methods for continuous bilinear programming. These topics are left for future research.

References

[1] Adams, W.P. and Forrester, R.J. (2005) A simple recipe for concise mixed 0-1 linearizations. *Operations Research Letters*, **33**(1), 55–61.

[2] Alarie, S., Audet, C., Jaumard, B. and Savard, G. (2001) Concavity cuts for disjoint bilinear programming. *Mathematical Programming*, **90**(2), 373–398.

[3] Anandalingam, G. and Apprey, V. (1991) Multi-level programming and conflict resolution. *European Journal of Operational Research*, **51**(2), 233–247.

[4] Assad, A.A. (1978) Multicommodity network flows - a survey. *Networks*, **8**(1), 37–91.

[5] Assimakopoulos, N. (1987) A network interdiction model for hospital infection control. *Computers in Biology and Medicine*, **17**(6), 413–422.

[6] Audet, C., Hansen, P., Jaumard, B. and Savard, G. (1999) A symmetrical linear maxmin approach to disjoint bilinear programming. *Mathematical Programming*, **85**(3), 573–592.

[7] Barnhart, C. (1993) Dual-ascent methods for large-scale multicommodity flow problems. *Naval Research Logistics*, **40**(3), 305–324.

[8] Barnhart, C. and Sheffi, Y. (1993) A network-based primal-dual heuristic for the solution of multicommodity network flow problems. *Transportation Science*, **27**(2), 102–117.

[9] Cappanera, P. and Fragioni, A. (2003) Symmetric and asymmetric parallelization of a cost-decomposition algorithm for multi-commodity flow problems. *INFORMS Journal on Computing*, **15**(4), 369–384.

[10] Chang, C.T. (2000) An efficient linearization approach for mixed-integer problems. *European Journal of Operational Research*, **123**(3), 652–659.

[11] Cormican, K.J., Morton, D.P. and Wood, R.K. (1998) Stochastic network interdiction. *Operations Research*, **46**(2), 184–196.

[12] Dantzig, G.B. (1955) Upper bounds, secondary constraints, and block triangularity in linear programming. *Econometrica*, **23**(2), 174–183.

[13] Fortune, S., Hopcroft, J. and Wyllie, J. (1980) The directed subgraph homeomorphism problem. *Theoretical Computer Science*, **10**, 111–121.

[14] Fragioni, A. and Gallo, G. (1999) A bundle type dual-ascent approach to linear multicommodity min cost flow problems. *INFORMS Journal on Computing*, **11**(4), 370–393.

[15] Fulkerson, D.R. and Harding, G.C. (1977) Maximizing minimum source-sink path subject to a budget constraint. *Mathematical Programming*, **13**(1), 116–118.

[16] Ghare, P.M., Montgomery, D.C. and Turner, W.C. (1971) Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, **18**(1), 37–45.

[17] Glover, F. (1975) Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, **22**(4), 455–469.

[18] Golden, B. (1978) A problem in network interdiction. *Naval Research Logistics Quarterly*, **25**(4), 711–713.

[19] Israeli, E. and Wood, R.K. (2002) Shortest-path network interdiction. *Networks*, **40**(2), 97–111.

[20] Kennington, J.L. (1978) A survey of linear cost multicommodity network flows. *Operations Research*, **26**(2), 209–236.

[21] Kettani, O. and Oral, M. (1990) Equivalent formulations of nonlinear integer problems for efficient optimization. *Management Science*, **36**(1), 115–119.

[22] Konno, H. (1976) Cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, **11**(1), 14–27.

[23] Mamer, J.W. and McBride, R.D. (2000) A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. *Management Science*, **46**(5), 603–709.

[24] McBride, R.D. (1985) Advances in solving the multicommodity flow problem. *Interfaces*, **28**(1), 32–41.

[25] McBride, R.D. (1985) Solving embedded generalized network flow problems. *European Journal of Operational Research*, **21**, 82–92.

[26] McMasters, A. and Thomas, M. (1970) Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, **17**, 261–268.

[27] Nemhauser, G.L. and Wolsey, L.A. (1988) *Integer and combinatorial optimization*, John Wiley & Sons, New York, NY.

[28] Sherali, H.D. and Adams, W.P. (1999) *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, Kluwer Academic Publishers, Dordrecht.

[29] Sherali, H.D. and Shetty, C.M. (1980) A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. *Mathematical Programming*, **19**, 14–31.

[30] Tuy, H. (1964) Concave programming under linear constraints. *Soviet Mathematics*, **5**(1), 1437–1440.

[31] Vaish, H. and Shetty, C.M. (1976) The bilinear programming problem. *Naval Research Logistics Quarterly*, **23**(2), 303–309.

[32] Washburn, A. and Wood, R.K. (1995) Two-person zero-sum games for network interdiction. *Operations Research*, **43**(2), 243–251.

[33] Wollmer, R. (1964) Removing arcs from a network. *Operations Research*, **12**(6), 934–940.

[34] Wood, R.K. (1993) Deterministic network interdiction. *Mathematical and Computer Modelling*, **17**(2), 1–18.