Operations Research Letters 45 (2017) 210–216

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Solving the traveling salesman problem with interdiction and fortification

Leonardo Lozano, J. Cole Smith [*], Mary E. Kurz

Department of Industrial Engineering, Clemson University, 100 Freeman Hall, PO Box 340920, Clemson, SC 29634-0920, United States

ARTICLE INFO

Article history:
Received 3 May 2016
Received in revised form
20 February 2017
Accepted 20 February 2017
Available online 1 March 2017

Keywords:
Interdiction
Integer programming
Traveling salesman problem

ABSTRACT

We solve a defender-attacker-defender problem over a traveling salesman problem (TSP), in which the defender first acts to defend a subset of arcs, the attacker then interdicts a subset of undefended arcs (thus increasing their costs), and the defender solves a TSP over the remaining network. Our approach employs an exact approach augmented with a TSP restriction phase to accelerate the convergence of the algorithm. Our computational results show success for the first time in optimally solving defender-attacker-defender TSP problems.

1. Introduction

The traveling salesman problem (TSP) is a well-known $\mathcal{NP}$-hard problem that seeks a minimum-cost Hamiltonian cycle (tour) over a graph [9,12]. We present the traveling salesman problem with interdiction and fortification (TSPIF), modeled as a defender-attacker-defender problem. In the first stage (*fortification*), the defender fortifies a subset of arcs. In the second stage (*attack*), an attacker interdicts a subset of unprotected arcs, thus increasing their cost. In the third stage (*recourse*), the defender solves a TSP defined using the costs resulting from the attack stage. In this context, an attack is not necessarily due to a malicious adversary, but could represent some bounded worst-case scenario on arc cost uncertainty. The TSPIF also arises as an alternative conservative approach to modeling routing problems under uncertainty, in which the road travel times may not be known in advance due to congestion effects [16].

The TSPIF may occur in a defense scenario in which troops need to monitor a set of locations and return to a base. If the troops wish to perform these tasks as quickly as possible, they solve a TSP. An adversary might attempt to impair the troops' movement by degrading (interdicting) roadways or bridges. The adversary's actions could be anticipated by the troops, who secure pathways ahead of time by stationing personnel or other resources to deter interdictions. Hence, the troops act first to fortify arcs, after which the adversary interdicts unfortified arcs, and the troops respond by solving a TSP on the resulting network.

We formally define the TSPIF on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$ is the set of arcs. For each arc $(i, j) \in \mathcal{A}$, let $c_{ij} \geq 0$ be the cost of traversing an uninterdicted arc and $d_{ij} \geq 0$ be the additional cost (delay) incurred when traversing an interdicted arc. Thus, the total cost of traversing an interdicted arc is $c_{ij} + d_{ij}$. Let $\mathbf{w} \in \mathcal{W}$ be the fortification decision variables, where $\mathcal{W} \equiv \left\{\mathbf{w} \mid \mathbf{Tw} \leq \mathbf{b},\ \mathbf{w} \in \{0, 1\}^{|\mathcal{A}|}\right\}$ ensures that the variables are binary and enforces a set of linear constraints that limits the extent to which the defender can fortify arcs. Let $\mathbf{x} \in \mathcal{X}(\mathbf{w})$ be the attack decision variables, where $\mathcal{X}(\mathbf{w}) \equiv \left\{\mathbf{x} \mid \mathbf{T}'\mathbf{x} \leq \mathbf{b}',\ x_{ij} \leq 1 - w_{ij},\ \forall (i, j) \in \mathcal{A},\ \mathbf{x} \in \{0, 1\}^{|\mathcal{A}|}\right\}$ forces the $\mathbf{x}$-variables to be binary, ensures that only unfortified arcs are interdicted, and imposes a set of linear constraints that models the ability of the attacker to interdict arcs. Finally, let $\mathbf{y}$ be a vector of binary arc-selection variables such that $y_{ij} = 1$ if arc (i, j) is used in the optimal tour identified for the recourse problem, and $y_{ij} = 0$ otherwise, for all $(i, j) \in \mathcal{A}$. We restrict $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{Y}$ includes the set of binary vectors $\mathbf{y}$ that correspond to TSP solutions in $\mathcal{G}$. The TSPIF can be formally stated as:

$$z^* = \min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \mathcal{Y}} \sum_{(i,j) \in \mathcal{A}} (c_{ij} + d_{ij} x_{ij}) y_{ij}, \tag{1}$$

where in the objective function (1), the original cost of any arc is increased by d_{ij} when the arc is attacked (i.e., $x_{ij} = 1$).

Previous studies on defender-attacker-defender problems include enumerative schemes [6,7,19,20], decomposition approaches based on supervalid inequalities [15,22], and approaches based on strong duality [5,17,21]. The latter approaches combine the

[*] Corresponding author.
E-mail address: jcsmith@clemson.edu (J.C. Smith).
URL: http://jcsmith.people.clemson.edu (J.C. Smith).

http://dx.doi.org/10.1016/j.orl.2017.02.007
0167-6377/© 2017 Elsevier B.V. All rights reserved.

second- and third-stage problems by taking the dual of the third-stage problem, and then solve the resulting problem using a Benders' decomposition algorithm in which the subproblem is a mixed-integer programming problem (MIP). Since the TSP is a non-convex combinatorial problem, we do not combine the second- and third-stage problems, because no polynomial-size strong dual formulation for the TSP is known to exist. In response to this problem, Lozano and Smith [14] propose a backward sampling framework (BSF) for interdiction problems with fortification in which the recourse problem can take any form. They solve defender-attacker-defender games played over shortest path (SPIF) and capacitated lot sizing problems (CLSIF). The BSF significantly outperforms prior approaches for solving the SPIF, and yields an effective mechanism for solving interdiction and fortification problems. However, the design of practically effective algorithms for interdiction and fortification problems defined over a strongly $\mathcal{NP}$-hard problem like the TSP is still an open research question.

In this paper we explore the solution of the TSPIF via the BSF. Our contribution analyzes two problem restrictions, where both restrictions serve as a heuristic for the TSPIF and model the situation in which the defender lacks the computational resources to compute an optimal response to an attack. We also demonstrate that these restrictions are instrumental in reducing computational time for solving the TSPIF within an exact two-phase approach. We then use these developments to tailor a BSF-based approach for this problem, and demonstrate the efficacy of our approach using TSP instances from the literature.

2. The backward sampling framework for the TSPIF

Our algorithm is composed of an outer cutting-plane approach that optimizes the defense decisions and an inner sampling-based optimization algorithm that solves the corresponding two-level problem. The inner algorithm restricts the defender to select a recourse decision from a sample of TSP tours over $\mathcal{G}$ and iteratively refines the sample to force finite convergence to an optimal solution.

2.1. Solving the TSPIF

Our outer algorithm evaluates defense vectors $\mathbf{w} \in \mathcal{W}$ that eliminate *critical attacks*. An attack $\hat{\mathbf{x}}$ is critical if

$$z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y}\in\mathcal{Y}} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}\hat{x}_{ij})y_{ij} \geq z^*, \tag{2}$$

i.e., if the objective function value of an optimal TSP tour given an attack vector $\hat{\mathbf{x}}$ is greater than or equal to z^*. We add a *covering constraint* (also known as a "no-good constraint" [4]) of the form $\mathbf{w}^\intercal\hat{\mathbf{x}} \geq 1$ to the fortification problem for each critical attack $\hat{\mathbf{x}}$, forcing the defender to protect at least one of the arcs interdicted by $\hat{\mathbf{x}}$.

Our inner algorithm solves the corresponding two-level interdiction problem for every defense vector $\hat{\mathbf{w}} \in \mathcal{W}$ selected by the outer algorithm:

$$\mathcal{Q}(\hat{\mathbf{w}}) : z^I(\hat{\mathbf{w}}) = \max_{\mathbf{x}\in\mathcal{X}(\hat{\mathbf{w}})} \min_{\mathbf{y}\in\mathcal{Y}} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}x_{ij})y_{ij}. \tag{3}$$

We assume that $\mathcal{X}(\mathbf{w}) \neq \emptyset$, $\forall \mathbf{w} \in \mathcal{W}$, and so an optimal solution to (3) must exist.

Since problem $\mathcal{Q}(\hat{\mathbf{w}})$ does not belong to $\mathcal{NP}$ unless $\mathcal{P} = \mathcal{NP}$, our algorithm relies on a sampling procedure that iteratively solves a series of restricted problems whose solutions converge to an optimal solution to $\mathcal{Q}(\hat{\mathbf{w}})$. Let $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$ be a sample of TSP tours over $\mathcal{G}$ and

$$\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}}) : z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}}) = \max_{\mathbf{x}\in\mathcal{X}(\hat{\mathbf{w}})} \min_{\mathbf{y}\in\hat{\mathcal{Y}}} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}x_{ij})y_{ij} \tag{4}$$

be the restricted problem in which recourse decisions are restricted to $\hat{\mathcal{Y}}$. For a given $\hat{\mathbf{w}} \in \mathcal{W}$ and a sample of tours $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$, we formulate $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ as a MIP. Let $\mathcal{T}^k$ be the set of arcs corresponding to the kth tour in sample $\hat{\mathcal{Y}}$, and let $c(\mathcal{T}^k)$ denote its cost. We formulate $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ as follows:

$$z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}}) = \max \quad z \tag{5}$$

$$\text{s.t.} \quad z \leq c(\mathcal{T}^k) + \sum_{(i,j)\in\mathcal{T}^k} d_{ij}x_{ij} \quad \forall \mathcal{T}^k \in \hat{\mathcal{Y}} \tag{6}$$

$$\mathbf{x} \in \mathcal{X}(\hat{\mathbf{w}}). \tag{7}$$

The objective function (5) maximizes z, which is constrained by (6) to be no more than the least-cost tour in $\hat{\mathcal{Y}}$, after considering additional costs caused by arc interdiction. Constraints (7) restrict $\mathbf{x}$ to the set of feasible attacks.

For any $\hat{\mathbf{w}} \in \mathcal{W}$ and sample of tours $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$, $z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ is an upper bound on $z^I(\hat{\mathbf{w}})$, which is in turn an upper bound on z^*. Lower bounds can be obtained by solving a TSP to calculate $z^R(\hat{\mathbf{x}})$ for any $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$. Combining these inequalities, we obtain:

$$z^R(\hat{\mathbf{x}}) \leq z^I(\hat{\mathbf{w}}) \leq z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}}), \quad \forall \hat{\mathbf{w}} \in \mathcal{W}, \ \hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}}), \ \hat{\mathcal{Y}} \subseteq \mathcal{Y}.$$

Thus, if $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ solves $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ and $z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}}) = z^R(\hat{\mathbf{x}})$, then $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ solves $\mathcal{Q}(\hat{\mathbf{w}})$ [14].

Algorithm 1 presents the proposed approach. Let $\mathcal{C}$ be the set of covering constraints added to the (outer) fortification problem and $\mathcal{W}(\mathcal{C}) = \{\mathbf{w} \in \mathcal{W} \mid \mathbf{w} \text{ satisfies all constraints in } \mathcal{C}\}$. The algorithm starts with an empty set of covering constraints and a global upper bound $\bar{z} = \infty$. Line 2 selects an initial sample of tours over $\mathcal{G}$. The outer while-loop (line 4) is executed until the fortification problem becomes infeasible. Line 5 selects a feasible defense $\hat{\mathbf{w}}$ and lines 6–21 solve the corresponding problem $\mathcal{Q}(\hat{\mathbf{w}})$ with our proposed sampling approach. The inner while-loop (line 7) is executed until the global upper bound cannot be further reduced by the current choice of $\hat{\mathbf{w}}$. Line 9 obtains an upper bound on $z^I(\hat{\mathbf{w}})$ by solving the restricted problem $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}}^i)$ and obtaining an attack vector $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$. Line 10 solves a TSP given the fixed attack $\hat{\mathbf{x}}$. The optimal tour, $\hat{\mathbf{y}}^*$, identified in this step yields a lower bound. Line 11 defines the sample at the next iteration as the solutions in the previous sample along with $\hat{\mathbf{y}}^*$. Line 12 checks if UB_i reduces the current global upper bound; if so, then line 13 updates the global upper bound, and line 14 removes from the sample all tours whose cost is greater than $\bar{z}$. Line 15 determines if attack $\hat{\mathbf{x}}$ is critical by checking if $LB_i \geq \bar{z}$, and if so, line 16 adds a covering constraint to the fortification problem. Finally, if the optimality condition is satisfied (line 18), then line 19 updates the incumbent solution.

2.2. Sampling TSP tours

The only condition on the initial sample to ensure that our algorithm terminates with an optimal solution is that $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$. However, $\hat{\mathcal{Y}}^1$ impacts the performance of the BSF since both the tightness of the upper bounds obtained by solving restricted problems $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}}^i)$ and the number of constraints in formulation (5)–(7) depend on the choice of $\hat{\mathcal{Y}}^1$.

We now describe desirable features for a choice of $\hat{\mathcal{Y}}^1$. Tours in $\hat{\mathcal{Y}}^1$ should be diverse in the sense that they do not contain too many of the same arcs, or else the attacker could interdict many tours in the sample by interdicting a few arcs common to those tours. Tours in $\hat{\mathcal{Y}}^1$ should also be optimal or near-optimal solutions to the TSP when $\hat{\mathbf{x}} = \mathbf{0}$. Finally, if $|\hat{\mathcal{Y}}^1|$ is too large, then formulation (5)–(7) will be large as well, and may potentially be too difficult to solve.

Attempting to achieve a balance between the desirable features listed, we propose a genetic algorithm (GA) based on the NSGA-II framework [8] in which each solution has two objectives, both of

Algorithm 1 Backward sampling framework for the TSPIF

```
1: Set the global upper bound $\bar{z} = \infty$ and covering constraints set $\mathcal{C} = \emptyset$    ▷ Initialization
2: Select $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$ as a sampling of tours from $\mathcal{G}$, and compute their objective values
3: Set counter $i = 0$
4: while $\mathcal{W}(\mathcal{C}) \neq \emptyset$ do    ▷ Main while-loop
5:     Select any $\hat{\mathbf{w}} \in \mathcal{W}(\mathcal{C})$
6:     Initialize $LB_i = -\infty$
7:     while $LB_i < \bar{z}$ do
8:         Set $i = i + 1$
9:         Solve $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}}^i)$, set $UB_i = z^I(\hat{\mathbf{w}}, \hat{\mathcal{Y}}^i)$, and record an optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$
10:        Solve $LB_i = z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}\hat{x}_{ij})y_{ij}$ and obtain an optimal tour $\hat{\mathbf{y}}^*$
11:        Set $\hat{\mathcal{Y}}^{i+1} = \hat{\mathcal{Y}}^i \cup \{\hat{\mathbf{y}}^*\}$
12:        if $UB_i < \bar{z}$ then
13:            Update global upper bound $\bar{z} \leftarrow UB_i$
14:            Remove from $\hat{\mathcal{Y}}^{i+1}$ all tours having cost greater than $\bar{z}$
15:        else if $LB_i \geq \bar{z}$ then    ▷ A critical attack has been identified
16:            Add the covering constraint $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to $\mathcal{C}$
17:        end if
18:        if $LB_i = UB_i = \bar{z}$ then
19:            Update the incumbent solution $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) \leftarrow (\hat{\mathbf{w}}, \hat{\mathbf{x}}, \hat{\mathbf{y}})$
20:        end if
21:    end while
22: end while
23: Return $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$
```

which are to be minimized. The first objective is the tour length with respect to the uninterdicted graph. The second objective is a measure of the individual solution's similarity to some reference set of TSP tours. We include in our reference set all tours whose length is not more than ϵ percent greater than the best tour-length seen so far. We compute our second objective as the number of times each solution arc appears in the reference set, divided by the total number of arcs in the set population. A solution that has no arcs in common with any tour in the reference set is in some manner "maximally different" and is desirable, having a second objective of 0. (The implementation details for the GA are available from the authors.)

Alternatively, one simple option is to seed $\hat{\mathcal{Y}}^1$ with one TSP tour. In this case we solve the TSP when $\hat{\mathbf{x}} = \mathbf{0}$, and use only that tour in our initial sample.

2.3. Alternative restrictions for the recourse problem

We now present two restrictions for the recourse problem that model the case in which the defender must expend a quick response to an attack, rather than expending the computational resources required to compute an optimal response. These restrictions are also instrumental in devising a more computationally effective exact TSPIF algorithm.

The first restriction is inspired by very large-scale neighborhood search algorithms [1]. We start with a base tour $\mathbf{y}^*$ obtained by solving the TSP to optimality given costs $c_{ij},\ \forall (i,j) \in \mathcal{A}$. For the symmetric case, the defender recourse responses are restricted to belong to the set of tours that can be obtained by performing a series of so-called disjoint 2-opt swaps on $\mathbf{y}^*$. (See Remark 1 for an extension of this idea for the asymmetric case.) To understand the concept of disjoint 2-opt swaps, we first order the nodes in tour $\mathbf{y}^*$ as $v_{(1)}, v_{(2)}, \dots, v_{(|\mathcal{N}|)}$. Let $v_{(|\mathcal{N}|+1)} \equiv v_{(1)}$. A 2-opt swap is performed by identifying two tour indices i and j, where $i \geq 1$, $j \leq |\mathcal{N}| + 1$, and $i + 3 \leq j$. The tour formed by a 2-arc swap replaces arcs $(v_{(i)}, v_{(i+1)})$ and $(v_{(j-1)}, v_{(j)})$ with arcs $(v_{(i)}, v_{(j-1)})$ and $(v_{(i+1)}, v_{(j)})$ in the original tour. Arcs $(v_{(k)}, v_{(k+1)})$, $k = i + 1, \dots, j - 2$, would now be traversed in the opposite direction after the symmetric 2-opt arc swap. (See Fig. 1 for an illustration.) A set of 2-opt swaps is disjoint if the 2-opt swaps are performed over indices $(i_1, j_1), (i_2, j_2), \dots, (i_k, j_k)$ such that $j_h \leq i_{h+1},\ \forall h = 1, \dots, k - 1$.

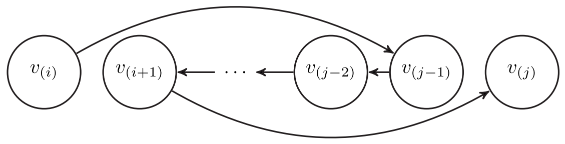

Fig. 1. Graphical representation of a 2-arc swap for the symmetric case.

We model the disjoint 2-opt swap restriction on $\mathbf{y}^*$ by transforming the recourse problem into a shortest path problem defined over a new graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$. The set of nodes $\mathcal{N}' = \{1, \dots, |\mathcal{N}| + 1\}$ represents each ordered node in tour $\mathbf{y}^*$, where $|\mathcal{N}| + 1$ is a duplicate of the first node. The set of arcs $\mathcal{A}' = \mathcal{A}'_1 \cup \mathcal{A}'_2$ comprises two kinds of arcs. Arcs in $\mathcal{A}'_1 = \{(i, i+1) \mid i \in \mathcal{N}',\ i \leq |\mathcal{N}|\}$ correspond to arcs in the original tour $\mathbf{y}^*$. Accordingly, we define their cost as $c'_{ij} = c_{v_{(i)}v_{(j)}}$ and delay for a given attack $\mathbf{x}$ as $d'_{ij} = d_{v_{(i)}v_{(j)}} x_{v_{(i)}v_{(j)}}$, for all $(i,j) \in \mathcal{A}'_1$. Arcs in $\mathcal{A}'_2 = \{(i,j) \mid \forall i = 1, \dots, |\mathcal{N}| - 2,\ j = i + 3, \dots, |\mathcal{N}| + 1\}$ represent a 2-opt swap as illustrated in Fig. 1. For arc $(i,j) \in \mathcal{A}'_2$ the cost and delay for a given attack $\mathbf{x}$ are defined as:

$$c'_{ij} = c_{v_{(i)}v_{(j-1)}} + c_{v_{(i+1)}v_{(j)}} + \sum_{k=i+2}^{j-1} c_{v_{(k)}v_{(k-1)}} \quad \forall (i,j) \in \mathcal{A}'_2 \tag{8}$$

$$d'_{ij} = d_{v_{(i)}v_{(j-1)}} x_{v_{(i)}v_{(j-1)}} + d_{v_{(i+1)}v_{(j)}} x_{v_{(i+1)}v_{(j)}} + \sum_{k=i+2}^{j-1} d_{v_{(k)}v_{(k-1)}} x_{v_{(k)}v_{(k-1)}} \quad \forall (i,j) \in \mathcal{A}'_2, \tag{9}$$

respectively. Note that the third term in Eqs. (8) and (9) accounts for the arcs traversed from $v_{(i+1)}$ to $v_{(j-1)}$ in the original tour, though in the reverse direction.

Every path from 1 to $|\mathcal{N}| + 1$ in $\mathcal{G}'$ corresponds to a TSP tour in the original graph $\mathcal{G}$. These paths encode the original tour $\mathbf{y}^*$ along with all solutions that can be obtained via disjoint 2-opt swaps from that tour.

Remark 1. For the asymmetric case, arcs in $\mathcal{A}'_1$ are given as before. Arcs $(i,j) \in \mathcal{A}'_2$ represent a 3-arc swap that replaces arcs $(v_{(i)}, v_{(i+1)})$, $(v_{(q)}, v_{(q+1)})$, and $(v_{(j-1)}, v_{(j)})$ with arcs $(v_{(i)}, v_{(q+1)})$, $(v_{(j-1)}, v_{(i+1)})$, and $(v_{(q)}, v_{(j)})$ in the original tour, where

$$q \in \operatorname*{argmin}_{i+1 \leq \bar{q} \leq j-2} \{c_{v_{(i)}v_{(\bar{q}+1)}} + c_{v_{(j-1)}v_{(i+1)}} + c_{v_{(\bar{q})}v_{(j)}} - c_{v_{(i)}v_{(i+1)}} - c_{v_{(q)},v_{(q+1)}} - c_{v_{(j-1)}v_{(j)}}\}. \tag{10}$$

Note that q is chosen in (10) so that the perturbed route corresponding to arc $(i,j) \in \mathcal{A}'_2$ is as close to optimal as possible with respect to the uninterdicted graph.

Cost and delay attributes are defined analogous to the symmetric case:

$$c'_{ij} = c_{v_{(i)}v_{(q+1)}} + c_{v_{(j-1)}v_{(i+1)}} + c_{v_{(q)}v_{(j)}} + \sum_{k=q+1}^{j-2} c_{v_{(k)}v_{(k+1)}} + \sum_{k=i+1}^{q-1} c_{v_{(k)}v_{(k+1)}} \quad \forall (i,j) \in \mathcal{A}'_2$$

$$d'_{ij} = d_{v_{(i)}v_{(q+1)}} x_{v_{(i)}v_{(q+1)}} + d_{v_{(j-1)}v_{(i+1)}} x_{v_{(j-1)}v_{(i+1)}} + d_{v_{(q)}v_{(j)}} x_{v_{(q)}v_{(j)}}$$

$$+\sum_{k=q+1}^{j-2} d_{v(k)v(k+1)}x_{v(k)v(k+1)} + \sum_{k=i+1}^{q-1} d_{v(k)v(k+1)}x_{v(k)v(k+1)}$$
$$\forall (i,j) \in \mathcal{A}_2'. \quad \square$$

We also consider a second restriction that constrains the defender to solve the recourse problem using the Lin–Kernighan heuristic (LKH) [13]. Note that the first restriction is modeled as a network flow problem, which can then be solved using existing network interdiction and fortification algorithms [6,11]. On the other hand, the second restriction will most likely yield a stronger upper bound given the success of the LKH in obtaining near-optimal TSP solutions, but the restriction cannot practically be modeled as a linear program.

2.4. Two-phase approach

We devise a two-phase approach that first solves a restriction of the TSPIF to identify a set of covering constraints, an initial sample of tours, and an upper bound on z^*. This first phase is based on the solution of a heuristic TSP restriction, and provides a warm start to exactly solve the problem using Algorithm 1 in a second phase.

Algorithm 2 describes our proposed two-phase approach. Line 1 solves a restriction of the TSPIF using a variation of Algorithm 1 in which one of the proposed restrictions in Section 2.3 is used to compute recourse solutions. We record an optimal solution, $(\mathbf{w}^0, \mathbf{x}^0, \mathbf{y}^0)$, and the set of all critical attacks explored, $\hat{\mathcal{X}}$, solving this restricted problem. Note that even though attacks in $\hat{\mathcal{X}}$ are critical for the restricted problem, they are not necessarily critical for the original (exact) problem. Line 2 solves to optimality the interdiction problem corresponding to $\mathbf{w}^0$ (using our inner sampling-based algorithm) and line 3 updates the upper bound and incumbent solution accordingly. Lines 5–11 explore all attacks $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$ to generate the initial sample and possibly identify covering constraints. For every $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$, line 6 solves a TSP to find an optimal recourse tour $\hat{\mathbf{y}}^*$ and calculates $z^R(\hat{\mathbf{x}})$. Line 7 adds $\hat{\mathbf{y}}^*$ into the initial sample. Line 8 determines if attack $\hat{\mathbf{x}}$ is critical by checking if $z^R(\hat{\mathbf{x}}) \geq \bar{z}$. If so, then line 9 adds a covering constraint to the fortification problem. Finally, line 12 continues solving the TSPIF using Algorithm 1, starting with an initial sample $\hat{\mathcal{Y}}^1$, a covering constraint set $\mathcal{C}$, and an upper bound $\bar{z}$.

Algorithm 2 Two-phase algorithm for the TSPIF

```
1:  Obtain an optimal solution $(\mathbf{w}^0, \mathbf{x}^0, \mathbf{y}^0)$ to a restriction of the
    TSPIF and let $\hat{\mathcal{X}}$ be the set of all critical attacks identified in the
    solution process                                          ▷ Begin phase one
2:  Obtain an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to $\mathcal{Q}(\mathbf{w}^0)$
3:  Set $\bar{z} = z^I(\mathbf{w}^0)$ and update the incumbent solution $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) \leftarrow$
    $(\mathbf{w}^0, \mathbf{x}^*, \mathbf{y}^*)$
4:  Initialize sample $\hat{\mathcal{Y}}^1 = \emptyset$ and covering constraints set $\mathcal{C} = \emptyset$
    ▷ Begin phase two
5:  for $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$ do
6:      Solve $z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}} \sum_{(i,j) \in \mathcal{A}} (c_{ij} + d_{ij}\hat{x}_{ij})y_{ij}$ and obtain an optimal
    tour $\hat{\mathbf{y}}^*$
7:      Add $\hat{\mathbf{y}}^*$ into $\hat{\mathcal{Y}}^1$
8:      if $z^R(\hat{\mathbf{x}}) \geq \bar{z}$ then          ▷ A critical attack has been identified
9:          Add the covering constraint $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to $\mathcal{C}$
10:     end if
11: end for
12: Solve the TSPIF using Algorithm 1 warm-started with $\hat{\mathcal{Y}}^1$, $\mathcal{C}$,
    and $\bar{z}$
```

3. Computational results

We coded our algorithm in Java, using Eclipse SDK version 4.4.2, and executed all computational experiments on a machine having an Intel Core i7-3537U CPU (two cores) running at 2.00 GHz with 8 GB of RAM on Windows 8. We solve the TSP instances using CONCORDE [2,3], all other optimization problems using Gurobi 5.6.0, and use the LKH implementation provided by [10].

Our set of test instances consists of 100 instances derived from 10 networks (5 symmetric and 5 asymmetric) from TSPLIB [18]. In every instance the cost coefficient for arc $(i,j) \in \mathcal{A}$ corresponds to the distance between nodes i and j in the original network. The delay coefficient for arc $(i,j) \in \mathcal{A}$ is initially taken to be a random integer uniformly distributed between $[1, c_{ij}]$. We generate 10 instances with random arc delay coefficients for each of the original networks. We define the defender's feasible region as $\mathcal{W} \equiv \left\{\mathbf{w} \mid \mathbf{e}^\intercal\mathbf{w} \leq Q,\ \mathbf{w} \in \{0,1\}^{|\mathcal{A}|}\right\}$, which enforces a cardinality constraint on the number of fortified arcs and ensures that the variables are binary. We also define $\mathcal{X}(\mathbf{w}) \equiv \left\{\mathbf{x} \mid \mathbf{e}^\intercal\mathbf{x} \leq B,\ x_{ij} \leq 1 - w_{ij},\ \forall (i,j) \in \mathcal{A},\ \mathbf{x} \in \{0,1\}^{|\mathcal{A}|}\right\}$, which ensures that a maximum of B unfortified arcs are interdicted, and forces the $\mathbf{x}$-variables to be binary.

3.1. Solving the TSPIF

We compared four versions of the proposed algorithm. The first one (one-tour sampling) initializes the sample as a single tour that optimizes the TSP when no arcs have been interdicted. The second one (GA sampling) implements the proposed GA sampling scheme. The third one (two-phase 2-opt) implements the two-phase algorithm in Section 2.4 with the 2-opt restriction, and the fourth one (two-phase LKH) implements the two-phase algorithm with the LKH restriction. We solve each instance three times with different budget configurations (Q, B) in $\{(3,3), (5,4), (4,5)\}$ for a total of 300 experiments for each version of the algorithm.

Table 1 shows the results for these experiments. The first five rows present results for symmetric instances and the last five rows for asymmetric instances. The columns present the average CPU time in seconds obtained over 10 instances derived from the same network (Avg), the largest CPU time obtained over those runs (Max), and the number of instances solved within a four-hour time limit (# solved) for the four versions of the algorithm. We calculate the average CPU time only among the instances solved within the time limit and report an overall CPU time average for the symmetric and asymmetric instances. For each row, the best average and maximum CPU times are highlighted in bold.

Table 1 shows that two-phase LKH outperforms the other algorithms. For the symmetric instances, two-phase LKH is on average about 40% faster than one-tour sampling and two-phase 2-opt, and about 65% faster than GA sampling. However, there is one instance from network gr96 that none of the algorithms solve within the time limit. For the harder asymmetric instances, two-phase LKH is on average more than two times faster than the other algorithms. The maximum CPU times follow a similar behavior.

3.2. Assessing the effectiveness of the proposed restrictions

Table 2 compares two-phase 2-opt and two-phase LKH when $Q = 4$ and $B = 5$, which was the most difficult budget configuration in Table 1. We omitted instance gr96, which was not solved to optimality. Column "z^*" presents the average optimal objective value obtained over the instances derived from the same network. The remaining columns show the average upper bound obtained at the end of phase one ($\bar{z}$), the average objective function value gap, calculated as $(\bar{z} - z^*)/z^* \times 100$ (% Gap), the average CPU time in seconds for phase one (I), the average CPU time for

Table 1
Comparing four different versions of our proposed algorithm for solving the TSPIF.

Instance	Arcs	Q	B	One-tour sampling			GA sampling			Two-phase 2-opt			Two-phase LKH		
				Avg	Max	# solved	Avg	Max	# solved	Avg	Max	# solved	Avg	Max	# solved
bayg29	406	3	3	10	22	10	15	30	10	12	24	10	**7**	**16**	10
		5	4	48	103	10	61	123	10	47	105	10	**40**	**87**	10
		4	5	99	302	10	119	218	10	101	213	10	**82**	**167**	10
hk48	1128	3	3	10	24	10	18	37	10	13	26	10	**7**	**17**	10
		5	4	63	213	10	94	330	10	72	224	10	**50**	**156**	10
		4	5	151	**666**	10	212	878	10	197	821	10	**140**	721	10
brazil58	1653	3	3	15	30	10	24	48	10	19	60	10	**12**	**23**	10
		5	4	63	149	10	107	261	10	72	232	10	**55**	**138**	10
		4	5	137	**368**	10	215	738	10	159	527	10	**118**	458	10
eli76	2850	3	3	32	53	10	42	72	10	43	74	10	**22**	**51**	10
		5	4	174	**295**	10	231	460	10	174	**295**	10	**119**	298	10
		4	5	379	**797**	10	565	1294	10	445	892	10	**378**	1172	10
gr96	4560	3	3	222	523	10	238	628	10	235	554	10	**124**	**389**	10
		5	4	1913	9088	10	2061	9534	10	1938	8250	10	**1411**	**7762**	10
		4	5	3351	> 14,400	9	3938	> 14,400	9	3391	> 14,400	9	**2249**	> 14,400	9
Overall average				444			529			461			**321**		
Asymmetric instances															
br17	272	3	3	**2**	3	10	3	3	10	3	3	10	**2**	**2**	10
		5	4	6	7	10	**5**	**6**	10	8	10	10	**5**	**6**	10
		4	5	7	9	10	**6**	**8**	10	10	11	10	**6**	**8**	10
p43	1806	3	3	222	287	10	197	248	10	194	242	10	**111**	**153**	10
		5	4	409	620	10	421	681	10	464	589	10	**246**	**361**	10
		4	5	565	690	10	556	735	10	606	804	10	**328**	**493**	10
ry48p	2256	3	3	1007	1193	10	976	1102	10	1267	1657	10	**452**	**995**	10
		5	4	3237	4462	10	3269	4579	10	3666	4810	10	**1280**	**1833**	10
		4	5	6481	8576	10	6648	9353	10	7386	10,321	10	**2758**	**5471**	10
ft53	2756	3	3	53	79	10	54	92	10	66	116	10	**19**	**27**	10
		5	4	194	338	10	206	371	10	202	364	10	**104**	**215**	10
		4	5	405	940	10	420	1010	10	484	833	10	**199**	**499**	10
ftv64	4160	3	3	230	356	10	210	329	10	270	408	10	**95**	**184**	10
		5	4	737	1278	10	706	1088	10	838	1435	10	**370**	**647**	10
		4	5	859	1630	10	921	1828	10	984	1674	10	**429**	**659**	10
Overall average				961			973			1097			**427**		

Table 2
Comparing the performance of the proposed restrictions within the two-phase approach.

Instance	z^*	Two-phase 2-opt						Two-phase LKH					
		$\bar{z}$	% Gap	Time (s)		Cuts		$\bar{z}$	% Gap	Time (s)		Cuts	
				I	II	I	II			I	II	I	II
bayg29	1 702	1 722	1.20	17	84	9	21	1 702	0	80	2	23	0
hk48	12,057	12,140	0.68	60	137	5	25	12,057	0	136	4	26	0
brazil58	26,349	26,477	0.49	49	110	6	22	26,349	0	112	6	29	0
eli76	560	563	0.70	44	401	1	36	560	0	365	13	39	0
gr96	56,951	57,183	0.41	749	2643	4	26	56,951	0	2157	92	28	0
Overall average			0.69	184	675	5	26		0	570	24	29	0
Asymmetric instances													
br17	40	41	1.75	2	8	1	15	40	0	5	2	16	0
p43	5 625	5 630	0.08	107	499	1	19	5 626	0.01	72	257	19	2
ry48p	14,884	14,991	0.72	1188	6198	1	23	14,885	0.01	610	2148	21	2
ft53	7 185	7 243	0.81	78	407	3	28	7 185	0	167	32	27	0
ftv64	1 921	1 929	0.41	140	843	2	34	1 922	0.05	47	381	37	1
Overall average			0.75	303	1591	2	24		0.01	180	564	24	1

phase two (II), the average total CPU time (Total), and the number of covering constraints added at the end of phases one and two (Cuts).

Table 2 shows that for symmetric instances, two-phase 2-opt quickly obtains near-optimal heuristic solutions with an objective function gap less than 1% on average. However, the total time to find an optimal solution by two-phase 2-opt is larger than the time required by two-phase LKH, which finds an optimal solution for every instance at the end of phase one. The 2-opt restriction identifies only a small number of covering constraints compared to the LKH restriction.

We conclude that for symmetric instances the 2-opt restriction is the best choice for a stand-alone heuristic, and the LKH restriction is better when embedded in our two-phase exact algorithm. For the asymmetric instances, the LKH restriction outperforms the 2-opt restriction both as a stand-alone heuristic and within our exact algorithm, finding heuristic solutions with a smaller average gap in less computational time. Both algorithms require considerably more time for phase two on the asymmetric instances.

3.3. *Sensitivity analysis*

We conduct additional sensitivity analysis experiments related to the defender's budget, Q, the attacker's budget, B, and the range of the arc delay coefficient. For this purpose, we select a subset of 10 symmetric instances based on network eli76 and 10 asymmetric instances based on network ftv64, and begin by solving each

Table 3
Measuring the effect of Q and B on execution time and objective.

Q	B	eli76 (symmetric)				ftv64 (asymmetric)			
		Avg	Max	# solved	z^*	Avg	Max	# solved	z^*
0	4	20	57	10	560	21	43	10	1933
2	4	63	134	10	558	95	238	10	1915
4	4	133	382	10	556	242	467	10	1907
6	4	276	699	10	554	454	723	10	1899
8	4	352	486	10	553	676	1087	10	1892
10	4	679	1127	10	552	1004	1501	10	1887
4	2	14	18	10	547	67	130	10	1875
4	4	132	381	10	556	241	464	10	1907
4	6	2193	5874	10	563	594	832	10	1935
4	8	5823	>14,400	6	570[a]	1641	3354	10	1959

[a] Average optimal objective value computed only among the instances solved within the time limit.

Table 4
Measuring the effect of varying the delay coefficient range on execution time.

Delay configuration	bayg29 (symmetric)				p43 (asymmetric)			
	Avg	Max	# solved	z^*	Avg	Max	# solved	z^*
$[1, c_{ij}]$	25	46	10	1686	185	363	10	5624
$[1, 2c_{ij}]$	119	173	10	1712	214	357	10	5624
$[1, 3c_{ij}]$	344	875	10	1722	225	337	10	5626
$[1, M]$	>14,400	>14,400	0	–	749	1505	10	5633

Table 5
Imposing a penalty on the number of times an arc is used in the sample.

Delay configuration	bayg29 (symmetric)				p43 (asymmetric)			
	Avg	Max	# solved	z^*	Avg	Max	# solved	z^*
$[1, c_{ij}]$	39	65	10	1686	346	607	10	5624
$[1, 2c_{ij}]$	122	184	10	1712	455	645	10	5624
$[1, 3c_{ij}]$	250	520	10	1722	462	553	10	5626
$[1, M]$	5180	6826	10	1759	1562	1886	10	5633

instance with intermediate values of $Q = 4$ and $B = 4$. We then vary B and Q to determine the impact that these parameters have on computational time and objective function value. For this experiment we use the two-phase LKH algorithm since it is the best performer among the proposed algorithms. Table 3 presents the results of this experiment.

Table 3 shows that increasing the attacker's budget has a dramatic effect on the execution time of the algorithm. For the symmetric instances, the computational times increase from 14 s to over 2000 s as B grows from 2 to 6, and only six out of ten instances are solved to optimality when $B = 8$. For the asymmetric instances the computational time increases by roughly a factor of three when increasing B by two units. On the other hand, increasing the defender's budget has a less pronounced effect on the computational time. Increasing the defender's budget by ten units decreases z^* by about 1.5% for the symmetric instances and 2.4% for the asymmetric instances. Increasing the attacker's budget by six units results in an objective value increase of roughly 4% for both the symmetric and the asymmetric instances.

We also conduct experiments to measure the effect of increasing the arc delay coefficient range on the execution times. For this purpose, we generate new instances based on symmetric network bayg29 and asymmetric network p43. We generate 20 instances (10 symmetric and 10 asymmetric) having random arc delay coefficients uniformly distributed between $[1, 2c_{ij}]$, 20 instances having delay coefficients between $[1, 3c_{ij}]$, and 20 instances having delay coefficients between $[1, M]$, where $M = 10 \max_{(i,j)\in\mathcal{A}}\{c_{ij}\}$. The latter delay configuration models the case in which an arc may become unavailable when interdicted. Table 4 presents results over this new set of instances. As before, we use intermediate values of $Q = 4$ and $B = 4$.

Table 4 shows that for the symmetric instances, the computational time increases by roughly a factor of 15 as the delay range increases from $[1, c_{ij}]$ to $[1, 3c_{ij}]$, and none of the instances for which the delay coefficients are between $[1, M]$ terminate within the four-hour time limit. On the contrary, for the asymmetric instances the computational time exhibits a moderate increase with respect to the delay coefficient range and the instances having delay coefficients between $[1, M]$ are solved on average in about 12 min. The increased difficulty of solving symmetric instances as the range of the d-parameters grows may be due to the amount of similar tours that exist in the sample, which we expect to be considerably greater for symmetric instances than for asymmetric instances.

To mitigate the extent to which similar tours are included in the sample, we modify the objective function in phase one of our algorithm to include a penalty function based on sample diversity. Let u_{ij} be the number of tours in the sample that use arc (i, j). For a given attack $\hat{\mathbf{x}}$, we set the third-stage objective function in phase one as

$$\min_{\mathbf{y}\in\mathcal{Y}} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}\hat{x}_{ij})y_{ij} + Pu_{ij}y_{ij}, \tag{11}$$

where P is an arbitrary constant penalty. After fine-tuning the two-phase LKH algorithm, we set P equal to $1/2$.

Table 5 shows that including the penalty in the objective function greatly reduces the computational time for solving the symmetric instances having delay coefficients between $[1, M]$, which are now solved in less than 2 h. On the other hand, including the penalty in the objective function has a negative effect on the execution time for the asymmetric instances, which in the worst case increases by roughly a factor of 2.

Acknowledgments

The authors are grateful for the remarks of an anonymous referee and an Associate Editor, whose comments led to an improved version of this paper. The authors also gratefully acknowledge the support of the *Air Force Office of Scientific Research* under grant FA9550-12-1-0353, and the *Office of Naval Research* under grant N000141310036.

References

[1] R.K. Ahuja, Ö. Ergun, J.B. Orlin, A.P. Punnen, A survey of very large-scale neighborhood search techniques, Discrete Appl. Math. 123 (1) (2002) 75–102.
[2] D. Applegate, R. Bixby, V. Chvátal, W. Cook, CONCORDE TSP solver, available at www.math.uwaterloo.ca/tsp/concorde.html.
[3] D. Applegate, R. Bixby, W. Cook, V. Chvátal, On the solution of traveling salesman problems, Rheinische Friedrich-Wilhelms-Universität, Bonn, Germany, 1998.
[4] E. Balas, R. Jeroslow, Canonical cuts on the unit hypercube, SIAM J. Appl. Math. 23 (1) (1972) 61–69.
[5] G.G. Brown, W.M. Carlyle, J. Salmerón, R.K. Wood, Defending critical infrastructure, Interfaces 36 (6) (2006) 530–544.
[6] P. Cappanera, M.P. Scaparra, Optimal allocation of protective resources in shortest-path networks, Transp. Sci. 45 (1) (2011) 64–80.
[7] R.L. Church, M.P. Scaparra, The r-interdiction median problem with fortification, Geogr. Anal. 39 (2) (2007) 129–146.
[8] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[9] M.M. Flood, The traveling-salesman problem, Oper. Res. 4 (1) (1956) 61–75.
[10] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic, European J. Oper. Res. 126 (1) (2000) 106–130.
[11] E. Israeli, R.K. Wood, Shortest-path network interdiction, Networks 40 (2) (2002) 97–111.
[12] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons, New York, 1985.
[13] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (2) (1973) 498–516.
[14] L. Lozano, J.C. Smith, A backward sampling framework for interdiction problems with fortification, INFORMS J. Comput. 29 (1) (2016) 123–139.
[15] J.R. O'Hanley, R.L. Church, Designing robust coverage networks to hedge against worst-case facility losses, European J. Oper. Res. 209 (1) (2011) 23–36.
[16] V. Pillac, M. Gendreau, C. Guéret, A.L. Medaglia, A review of dynamic vehicle routing problems, European J. Oper. Res. 225 (1) (2013) 1–11.
[17] M. Prince, J.C. Smith, J. Geunes, A three-stage procurement optimization problem under uncertainty, Nav. Res. Logist. 60 (1) (2013) 395–412.
[18] G. Reinelt, TSPLIB–A traveling salesman problem library, INFORMS J. Comput. 3 (4) (1991) 376–384.
[19] M.P. Scaparra, R.L. Church, A bilevel mixed-integer program for critical infrastructure protection planning, Comput. Oper. Res. 35 (6) (2008) 1905–1923.
[20] M.P. Scaparra, R.L. Church, An exact solution approach for the interdiction median problem with fortification, European J. Oper. Res. 189 (1) (2008) 76–92.
[21] J.C. Smith, C. Lim, F. Sudargho, Survivable network design under optimal and heuristic interdiction scenarios, J. Global Optim. 38 (2) (2007) 181–199.
[22] S. Starita, M.P. Scaparra, Optimizing dynamic investment decisions for railway systems protection, European J. Oper. Res. 248 (2) (2016) 543–557.