Optimal Allocation of Protective Resources in Shortest-Path Networks

Paola Cappanera
Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, 50139 Firenze, Italy, paola.cappanera@unifi.it

Maria Paola Scaparra
Kent Business School, University of Kent, Canterbury CT2 7PE, United Kingdom, m.p.scaparra@kent.ac.uk

This article introduces a game-theoretic approach for allocating protection resources among the components of a network so as to maximize its robustness to external disruptions. Specifically, we consider shortest-path networks where disruptions may result in traffic flow delays through the affected components or even in the complete loss of some elements. A multilevel program is proposed to identify the set of components to harden so as to minimize the length of the shortest path between a supply node and a demand node after a worst-case disruption of some unprotected components. An implicit enumeration algorithm is then developed to solve the multilevel problem to optimality. The approach is streamlined by solving the lower-level interdiction problem heuristically at each node of an enumeration tree and by using some variable fixing rules to reduce the dimension of the lower-level problems. A thorough computational investigation demonstrates that the proposed solution method is able to identify optimal protection strategies for networks of significant size. The paper is concluded with a study of the sensitivity of the solution approach to variations of the problem parameters such as the level of disruption and protective resources and the distribution of the arc lengths and delays.

Key words: network interdiction; multilevel programming; shortest path; resource allocation

History: Received: September 2008; revisions received: January 2009, March 2010, May 2010; accepted: July 2010. Published online in Articles in Advance November 19, 2010.

1. Introduction

Most of today’s societal, economic, and industrial functions rely heavily on the correct and continued operations of lifeline networks such as telecommunication, electric power, transportation, transit, water supply, sewage disposal, and natural gas and petroleum distribution systems. The growing complexity and interdependency of these networked systems have significantly increased their vulnerability to intentional and unintentional threats (e.g., terrorist attacks, sabotage, random failures, weather, natural disasters, and so on). As a consequence, the establishment of effective protection strategies has become an imperative need to lessen the operational impact of infrastructure losses and mitigate the dire and far-reaching consequences of possible system disruptions.

Our focus in this paper is to develop a rigorous mathematical approach for identifying the most cost-effective way of allocating scarce protection resources among the components of a network so as to maximize its robustness to malicious attacks. Although it could be argued that protection investments against rare, extreme events are difficult to justify on economic grounds because of the low probability of their occurrence, the need to implement sound protection strategies is justifiable in consideration of the enormous ripple effects these events may have, not only in terms of economical losses but also in terms of human lives. Furthermore, in light of the variety of hazards to which system facilities are exposed, there are likely to be economies of scope in hazard reduction (Sternberg and Lee 2006).

In this paper, we consider distance-based networks where travel times, route lengths, or shipping costs are the main concern as in the case of emergency transport networks, hazardous materials routing, and some freight distribution systems. We develop a multilevel optimization model that identifies the set of components to harden so as to minimize the length of the shortest path from a supply node to a demand node after a worst-case disruption of some unprotected elements. We assume that if a component is disrupted or attacked, its travel cost is increased. This would eventually render it unusable unless the component is hardened, in which case the attack has no effect. The increase in travel time caused by rerouting in the event of link removal or blockage is a common measure of performance reliability in transport networks (Jenelius, Peterson, and Mattsson 2006; Scott et al. 2006). As in other studies concerned with road network vulnerability (Bell 2000; Bell et al. 2008),
we assume that travel costs and delays are only affected by the link status and not by the link flow. Although this assumption may be restrictive in many practical situations where link capacities, congestion issues, and behavioral responses of travelers may affect travel times, we make this assumption to simplify the mathematical representation of the problem and gather some preliminary insights into possible methodological developments for solving complex multilevel network protection problems. In addition, simple shortest-path constructs occur frequently in the decomposition of more complex problems such as hazardous material transportation problems (Kara and Verter 2004) and bilevel models of taxation applied to highway pricing (Labbé, Marcotte, and Savard 1998). The theoretical findings of this study can therefore be extended to solve problems in these application settings as well.

The study of quantitative approaches for improving infrastructure security and robustness has attracted the interest of a number of researchers and practitioners over the years. Early studies mainly focused on the identification of the most vulnerable and valuable system components as a tool to drive decisions on where to direct funds for hardening and reinforcing systems. The mathematical models developed to this end are known as interdiction problems and aim to identify the sets of assets (e.g., nodes, links, or facilities) that, if they were disabled or lost, have the greatest impact on a system’s ability to perform its intended functions. Several interdiction models have been developed for military and homeland security applications in the last few decades. The seminal work on interdiction modeling is because of Wollmer (1964), who analyzed the problem of removing a specified number of arcs from a network in order to minimize the maximum flow between an origin and a destination node. Fulkerson and Harding (1977) introduced the first shortest-path interdiction model where arc lengths can be partially interdicted, subject to an interdiction budget, so as to maximize the shortest paths between supply and demand points. Golden (1978) investigated a least-cost partial interdiction strategy to ensure a predetermined increase in the shortest-path length. Other variations of interdiction models on shortest-path networks can be found in Corely and Sha (1982); Ball, Golden, and Vohra (1989); Malik, Mittal, and Gupta (1989); Israeli and Wood (2002); and Bayrak and Bailey (2008). For a complete survey of early interdiction problems with different underlying network properties, the reader is referred to Church, Scaparra, and Middleton (2004). More recently, stochastic variants of network flow interdiction problems have appeared in Hemmecke, Schultz, and Woodruff (2003); Held, Hemmecke, and Woodruff (2005); and Cormican, Morton, and Wood (1998) whereas Lim and Smith (2007) extend the study of interdiction models to multicommodity networks. Finally, real applications of interdiction models for the study of electric power system vulnerabilities can be found in Salmeron, Wood, and Baldick (2004, 2009) and Bier et al. (2007). Besides interdiction modeling, other methodologies can be used to identify, screen, and rank critical system elements and accident scenarios (see, for example, Michaud and Apostolakis 2006 for a discussion of probabilistic risk assessment and multiattribute utility theory for identifying critical elements in water supply networks). The use of these techniques, however, is outside the scope of our discussion.

Although interdiction models can provide valuable information about the criticality of some system components, protection strategies that rely solely on this information (for example, by prioritizing the protection of the most critical assets) will often result in suboptimal defense plans (Brown et al. 2006). These kinds of plans, in fact, fail to take into account how the set of critical assets varies in response to the hardening of some components and therefore may miss some crucial interactions and synergies within the system.

As an example of this shortfall, consider the simple network depicted in Figure 1. The two numbers associated with each arc represent, respectively, the arc length under normal circumstances and the delay that occurs along the arc in case of disruption. Assume that the efficiency of the system is measured in terms of distance or travel time from the supply node 1 to the destination node 7. Figure 1(a) shows the source-destination shortest path when all the arcs are fully operational (i.e., no delays occur). The length of the shortest path is 45. Shortest-path interdiction models such as the one in Israeli and Wood (2002) can be used to identify the most critical arcs in this network. As an example, the two arcs that, if interdicted, result in the greatest increase of the shortest-path length are arcs (2, 5) and (3, 6) (see the arcs with a cross in Figure 1(b)). In this case, the postinterdiction shortest path in the network, 1-2-4-7, has a length of 67, an increase of almost 50%.

Now, assume that resources are available to harden two network components. If the defensive resources are employed to protect the two critical links (2, 5) and (3, 6), an intelligent attacker aiming at inflicting maximum damage to the network will then attack links (1, 2) and (1, 4). The shortest path through the network in this worst-case scenario is displayed in Figure 1(c), where protected arcs are indicated with cross-hatched lines. If this protection strategy is implemented, the length of the postinterdiction shortest path, 1-2-4-3-6-7, is equal to 60. Although this protection plan reduces the impact of the disruption, the shortest-path length still increases by more than 30%.
Now, consider an alternative protection strategy where arcs (1, 2) and (3, 6) are hardened instead (Figure 1(d)). The worst-case disruption in response to this protection occurs when arcs (2, 4) and (5, 7) are subsequently attacked. Although the shortest path is the same as in the previous case, its length is now 47. This represents a length increase of less than 5% as compared to the length of the best path when no delays occur. Also, note that this fortification strategy renders the second interdiction ineffective. In fact, the same worst-case path length of 47 can be obtained by attacking arc (2, 4) only. This interdiction generates two paths with length 47 (1-2-4-3-6-7 and 1-2-5-7). Because these two paths only share the protected arc (1, 2), which cannot be interdicted, it is not possible to disrupt both of them with only one interdiction resource still available. The choice of the second interdiction, (5, 7), was arbitrary. Any other unprotected arc could have been chosen.

This result not only demonstrates that hardening the most critical components may be a suboptimal choice, but also that securing arcs in an optimal way can significantly reduce the impact of a worst-case disruption and squander offensive resources.

This simple example highlights the importance of explicitly including protection decisions into mathematical models. Different mathematical theories can be employed for the analysis of optimal defensive strategies. One of the major factors driving the choice of appropriate modeling tool is the type of threats to which infrastructure systems are exposed. In particular, reliability-theoretic models are more appropriate for planning the optimal defense of systems that face probabilistic risk (e.g., act of nature) whereas game-theoretic models are more suitable when allocating resources to counter strategic risk (e.g., malicious attacks) (Golany et al. 2009). Modeling willful attacks, in fact, poses some unique challenges, primarily because intelligent attackers can adapt their course of action to counter the defender’s strategy. In several recent studies, this interaction between defender and attacker has been envisaged as a game and represented mathematically as a multilevel model.

As an example, Brown et al. (2006) apply attacker-defender models to determine the value of protecting assets in real-world infrastructure systems such as power grids and petroleum pipeline networks. Protective plans for electric power grids are also analyzed in Holmgren, Jelenius, and Westin (2007), where the authors propose a two-player zero-sum game to allocate protective resources so as to minimize the expected damage under different attacking strategies. The allocation of security resources to a water supply network is investigated in Qiao et al. (2007). The objective of this model is to allocate a security budget to maximize an attacker’s marginal cost of inflicting damage through the destruction of network components. A similar objective is used in Azaiez and Bier (2007), where the authors assume that the defender attempts to deter attacks by making them as costly as possible. This type of objective is particularly suitable to model problems where the amount of resources available to the attacker is not known to the defender. A variation of this model can be found in Hausken (2008). In this work, defensive and offensive strategies are analyzed to improve the reliability of series and parallel systems. The reliability of each system component is assumed to depend on the level of defensive and offensive investments.
An interesting model framework, where defensive investments are allocated to counter both terrorism and natural disaster, is proposed in Zhuang and Bier (2007). They compare a sequential formulation and a simultaneous formulation of a defender-attacker game and discuss policy implications and important insights into the nature of equilibrium defensive strategies.

Finally, Scaparra and Church (2008a) propose a defender-attacker model to identify optimal ways of protecting facilities in service and supply systems so as to minimize the demand weighted distance between customers and facilities after a worst-case disruption. Single-level reformulations of this problem are provided in Church and Scaparra (2006) and Scaparra and Church (2008b), whereas Liberatore, Scaparra, and Daskin (2011) present a stochastic version of the model where the number of possible attacks is uncertain.

The contribution of this paper is to apply a game-theoretic framework to identify optimal protection strategies in shortest-path networks. More specifically, we build upon the shortest-path interdiction model proposed in Israeli and Wood (2002) and add to it an additional layer that explicitly models protection decisions. The resulting model is a trilevel program, which can be reduced to bilevel and single-level programs via duality and reformulation. The single-level program requires the explicit enumeration of all possible interdiction plans and therefore can only be used for problem instances of modest size. We hence discuss a solution approach for the bilevel program. The proposed approach builds on and extends the recursive tree search algorithm developed by Scaparra and Church (2008a) for facility protection interdiction problems. We provide some theoretical results related to the stability of this methodology when it is applied to shortest-path protection problems, and we streamline the approach to address the new challenges introduced by the underlying network structure of our model.

One of the difficulties encountered when dealing with network problems is related to the existence of alternative optimal solutions to the lower-level interdiction problem. Whereas facility interdiction problems such as the r-interdiction median problem (Church, Scaparra, and Middleton 2004) do not generally admit multiple optima, in our model several interdiction plans often generate the same min-cost path. The presence of multiple optimal lower-level solutions in bilevel programs poses stability problems (see, for example, Erkut and Gzara 2008 for a discussion of solution stability in hazmat transport network design problems). We provide a formal proof showing that the branching strategy used in the tree search algorithm always produces stable optimal solutions, even for network problems.

A second difficulty arising in network problems as opposed to facility problems is that the number of components that can be interdicted and protected is usually much larger because both nodes and links can fail rather than just facilities. We therefore propose several techniques to accelerate the algorithm by reducing the number of tree nodes to be evaluated during the search and by reducing the computational effort at each node. These include the introduction of a secondary objective in the lower-level interdiction problem, some variable fixing rules, and the use of a greedy heuristic and some supervalid inequalities. We also introduce an alternative formulation for the shortest-path interdiction model given in Israeli and Wood (2002) that when embedded within the defender-attacker bilevel construct, is remarkably more robust. Finally, we suggest a hybrid approach that selects on the fly the most promising interdiction formulation according to the structure of the problem at hand.

A thorough computational investigation demonstrates that the proposed streamlined approach is able to find optimal solutions for networks of significant size. We discuss the sensitivity of different variants of the algorithm to variations of the problem parameters such as the level of offensive and defensive investments and the distribution of the arc costs and delays. Finally, we study the impact on the solution quality of using an heuristic solution to the inner problem rather than a true attacker optimum to guide the tree search.

The paper is organized as follows. Section 2 presents the multilevel model. Section 3 describes the solution methodology. Section 4 introduces an alternative formulation for the lower-level attacker problem. Section 5 presents the computational results and §6 contains concluding remarks.

2. A Multilevel Formulation
Given a directed graph $G = (N, A)$ where $N$ is the set of nodes and $A$ is the set of arcs, the shortest-path interdiction problem with fortification (SPIF) aims at identifying the set of linkages to harden in such a way that the increase in the shortest path from a supply node $s$ to a demand node $t$ because of a worst-case interdiction of some unprotected linkages is minimized.

Note that although we make the assumption that the critical components that can be interdicted and protected are the network linkages, it is easy to prove that problems where the critical components are the nodes can be reduced to critical arc models by suitably augmenting the underlying graph (Corely and Chang 1974). Hence, we describe the more general case of arc protection and interdiction.
Let $c_k$ be the nominal length of arc $k$ and $d_k$ the delay (or penalty or cost increase) to ship flow through $k$ if the arc is interdicted. Arc lengths and arc delays are assumed to be nonnegative integers. The complete loss of an arc can be captured in the model by choosing $d_k$ sufficiently large (Israeli and Wood 2002). We assume that both offensive and protective resources are limited so that at most $R$ components can be attacked and at most $Q$ components can be hardened against interdiction. We also assume that an attack on a protected component has no effect, so the attacker should always hit unprotected components to optimize the use of his resources.

The multilevel formulation of SPIF uses the following sets of decision variables:

$$ z_k = \begin{cases} 1 & \text{if arc } k \text{ is hardened}, \\ 0 & \text{otherwise}; \end{cases} $$

$$ s_k = \begin{cases} 1 & \text{if arc } k \text{ is interdicted}, \\ 0 & \text{otherwise}; \end{cases} $$

$$ y_k = \begin{cases} 1 & \text{if flow is shipped through arc } k, \\ 0 & \text{otherwise}. \end{cases} $$

The set of feasible strategies of the defender can be defined in terms of the $z_k$ protection variables as $D = \{ z \in [0,1]^{|A|} | \sum_{k \in A} z_k \leq Q \}$. Similarly, the set of feasible strategies of the attacker can be defined in terms of the $s_k$ interdiction variables as $I = \{ s \in [0,1]^{|A|} | \sum_{k \in A} s_k \leq R \}$.

SPIF can then be formulated as the following trilevel problem:

$$(\text{SPIF}) \quad \min_{z \in D, s \in I, y} \max_{\pi} \min_{\nu} \sum_{k \in A} (c_k + d_k s_k) y_k,$$

subject to:

$$\sum_{k \in FS(i)} y_k - \sum_{k \in BS(i)} y_k = b_i \quad \forall i \in N, \quad (1)$$

$$s_k \leq 1 - z_k \quad \forall k \in A, \quad (2)$$

$$y_k \geq 0 \quad \forall k \in A. \quad (3)$$

As in standard shortest-path problems, $b_i = 1$, $b_i = -1$, and $b_i = 0$ for all other nodes $i$ in $N$. BS$(i)$ and FS$(i)$ denote the backward star and the forward star of node $i$, respectively. The objective function (1) computes the minimum-cost path after the worst-case interdiction of $R$ unprotected components. This cost includes the penalties associated with interdicted arcs. Constraints (2) are standard flow-balance constraints for shortest-path problems. Constraints (3) simply state that protected arcs cannot be interdicted. Finally, constraints (4) represent the nonnegativity requirements for the $y_k$ flow variables. Note that integer restrictions for these variables are not required because the bottom-level problem is a standard shortest-path problem for which an optimal integral solution always exists.

This trilevel defender-attacker-user model can be reduced to a bilevel model by taking the dual of the bottom minimization problem. By doing so, the bilevel attacker-user model is collapsed into a single-level attacker problem that inherently incorporates the user’s optimal choice (see Israeli and Wood 2002 for more details). The attacker problem can hence be solved by standard, “off the shelf” optimization software.

Let $\pi$, be the dual variable associated with the flow-balance constraint for node $i$. The reduced defender-attacker bilevel problem is as follows:

$$\min_{\nu} \max_{\pi} \pi_i,$$

subject to:

$$\pi_i - \pi_j - d_k s_k \leq c_k \quad \forall k = (i, j) \in A, \quad (6)$$

$$s_k \leq 1 - z_k \quad \forall k \in A. \quad (7)$$

Note that, in this formulation, we have omitted the dual variable $\pi$, that can be set to zero to offset the presence of at least one redundant constraint in the bottom-level shortest-path problem of SPIF.

The number of levels of the above problem can be further reduced if we explicitly enumerate all possible ways of interdicting $R$ out of the $|A|$ linkages. To this end, let $H_i$ indexed by $h$, be the set of all possible interdiction patterns and let $I_h$ be the set of interdicted arcs in pattern $h$. We denote by $s^h_i$ the interdiction vector associated with pattern $h$, i.e., $s^h = (s^h_1, \ldots, s^h_A)$ with $s^h_k = 1$ if $k \in I_h$ and $s^h_k = 0$ otherwise. Finally, let $\pi^h$ be the vector of shortest-path dual variables associated with a given interdiction pattern $h$. We can then formulate the defender-attacker problem as a single-level problem in the $z$ and $\pi$ variables only:

$$\min \nu,$$

subject to:

$$\nu \geq \pi^h_i \quad \forall h \in H, \quad (9)$$

$$\pi^h_i - \pi^h_j - d_k s^h_k (1 - z_k) \leq c_k \quad \forall k = (i, j) \in A, \forall h \in H, \quad (10)$$

$$\sum_{k \in A} z_k \leq Q, \quad (11)$$

$$z_k \in \{0, 1\} \quad \forall k \in A. \quad (12)$$

Without loss of generality, we can assume that each dual variable $\pi^h_i$ in this formulation represents the postfortification, postinterdiction shortest-path length from source node $s$ to node $i$, given that interdiction pattern $h$ occurs. Although this is not necessarily true for every $h$, it can be easily enforced by adding a properly weighted secondary objective that minimizes the variables $\pi^h_i$. 

Cappanera and Scaparra: Optimal Allocation of Protective Resources in Shortest-Path Networks Transportation Science 45(1), pp. 64–80, © 2011 INFORMS
It should be clear that the use of this single-level model is limited to problems of very modest size because of the rapid growth in the number of variables and constraints with increasing graph sizes and values of \( R \). Nevertheless, this formulation is introduced to prove some theoretical results in §3.

Finding the optimal solution to problems of realistic and practical size requires devising sophisticated approaches tailored to the bilevel structure of the defender-attacker problem (5)–(7). Unfortunately, bilevel optimization problems are not amenable to solution by standard mixed-integer programming methodologies, and no efficient universal algorithm exists for their solution. Furthermore, problem (5)–(7) can be classified as a particularly hard bilevel problem in that the upper-level variables parameterize the constraints of the lower-level problem and integer restrictions appear at both levels (see Moore and Bard 1990 or Dempe 2002 for an in-depth discussion of discrete bilevel problems).

In §3, we describe an implicit enumeration algorithm to solve problem (5)–(7). We also propose an heuristic to solve the lower-level interdiction problem and show how the information gained when solving the problem heuristically can be used to streamline the overall implicit enumeration approach. Finally, we propose some variable fixing mechanisms to further reduce the complexity of the problems solved at each node of the enumeration tree.

### 3. Solution Methodology

#### 3.1. An Implicit Enumeration Algorithm

We solve SPIF using an implicit enumeration scheme that exploits solutions of the lower-level interdiction problem to reduce the number of defender strategies to be evaluated (Scaparra and Church 2008a). The algorithm is built upon the simple observation that an optimal fortification set must include at least one of the components that are interdicted in an optimal solution to the interdiction problem without fortification (Church and Scaparra 2006). This observation can be used recursively at each node of the enumeration tree to narrow the set of the defender’s variables on which to branch. The general structure of the implicit enumeration algorithm is outlined next, followed by a formal proof of the correctness of the branching strategy.

In the pseudocode below, we denote by \( \hat{z} \) and \( \hat{s} \) the vectors of fortification and interdiction variables, respectively. Let \( r \) be the root node of the search tree and \( S \) the set of search tree nodes to be visited. Each node \( n \) has the two following sets associated with it:

- \( C_n \): the set of candidate arcs to be hardened
- \( F_n \): the set of fortifications already made.

Finally, we denote by SPI(\( F_n \)) the conditional shortest-path interdiction problem with the additional restriction that the arcs in \( F_n \) cannot be interdicted. We recall that each SPI(\( F_n \)) is a standard mixed-integer programming problem that can be solved by general purpose mixed-integer programming (MIP) solvers.

**Procedure ImplicitEnumeration**

**Initialization Phase**

Set \( F_n = \emptyset \).

Set \( z = 0 \), \( \hat{s} = \hat{s} \), and \( \hat{\pi} = \hat{\pi} \); \( C_r = \{ j : \hat{s}_j = 1 \} \); \( S = \{ r \} \).

**Iterative Phase**

while \( S \neq \emptyset \) do

begin

- Select \( n \in S \);
- Select \( k \in C_n \);
- Generate node \( n_0 \) with \( F_{n_0} = F_n \) and \( C_{n_0} = C_n \setminus \{ k \} \).
- If \( C_{n_0} \neq \emptyset \) then \( S = S \cup \{ n_0 \} \);
- Generate node \( n_1 \) with \( F_{n_1} = F_n \cup \{ k \} \);
- Solve SPI(\( F_{n_1} \)) for \( \hat{s}, \hat{\pi} \) and objective value \( \hat{\pi}_1 \);
- If \( |F_{n_1}| = Q \) or \( \hat{s} = 0 \) then leaf node if \( \hat{s}_i < \hat{\pi}_i \) then update best solution \( \hat{s} = \hat{s} \); \( \hat{\pi} = \hat{\pi} \); for each \( j \in A \), if \( j \in F_{n_1} \) then \( \hat{z}_j = 1 \) else \( \hat{z}_j = 0 \); endif;
- Else \( C_{n_1} = \{ j | \hat{s}_j = 1 \} \); \( S = S \cup \{ n_1 \} \);

end

Return \(( \hat{z}, \hat{s}, \hat{\pi} )\).

The algorithm is implemented as a binary search tree. The generation of a node \( n_0 \) represents the fact that the value of the selected variable for branching (\( z_k \)) is fixed to zero. In this case, either the set of candidate fortifications is empty and the node is fathomed, or the new node \( n_0 \) is added to the set of tree nodes to be visited. The generation of a node \( n_1 \) corresponds to fixing \( z_k \) to one, in which case we solve a conditional shortest-path interdiction problem with the new set of fortifications and generate a new set of variables to branch on. A leaf node is reached when exactly \( Q \) fortification variables are fixed to one or when, after the protection of the components in \( F_{n_1} \), any attack on \( R \) of the remaining linkages does not worsen the shortest path. In the latter case, we can force the MIP solver to return an empty set of interdictions (i.e., \( \hat{s} = 0 \)) by adding a secondary objective to SPI, as explained at the end of this section.

In practice, the implicit enumeration algorithm is implemented as a depth-first search that uses recursion and backtracking. At each node \( n \), the variable \( z_k \) to branch on is selected at random. Note that if the conditional lower-level interdiction problems have multiple optimal solutions, the order in which
the branching variables are selected may lead to the
generation and exploration of different sets of tree
nodes. Consequently, the choice of the branching vari-
ables may have an impact on the size of the search
tree and hence on the computing time of the over-
all approach. The identification of more sophisticated
selection strategies for the branching variables, which
take into account the existence of multiple optimal
solutions to SPI, is still an open problem that deserves
further investigation.

The proposed solution methodology can be easily
modified to handle the problem in which each link
requires a different amount of offensive resources
to be interdicted and there is a limit on the total
resources available for interdiction. To this end,
it is sufficient to replace the attacker cardinality
constraints in the lower-level interdiction problem
with more general resource constraints. Similarly,
the methodology can be used to solve problems with
a budget constraint on the fortification resources
instead of the cardinality constraint defining the set
of feasible protection strategies. In this case, a node
instead of the cardinality constraint defining the set
of feasible protection strategies. In this case, a node

We now state the correctness of the branching
rule used in the tree search approach by proving that,
with this rule, the recursive algorithm is always
able to identify a stable solution to the bilevel pro-
tree search approach by proving that,
with this rule, the recursive algorithm is always
able to identify a stable solution to the bilevel pro-

**Theorem 3.1.** For any interdiction strategy $\hat{s}$ generated
in the search tree, the inequality $\hat{s}z \geq 1$ is supervalid for
the defender-attacker problem.

**Proof.** Suppose that $(\hat{s}, \hat{\pi})$ is the solution to SPI($F_n$)
at a generic node $n$ of the search tree and $\hat{\pi}_i$ is its
objective value. Let $\hat{z}$ be the defender’s strategy
associated with node $n$ (i.e., for each $j \in A$, $\hat{z}_j = 1$ if $j \in F_n$,
and $\hat{z}_j = 0$ otherwise). Hence, the triplet $(\hat{z}, \hat{s}, \hat{\pi})$ rep-
resenting a feasible solution to the defender-attacker prob-
lem (5)–(7). Let $v^*$ be the objective value of an optimal
solution $(z^*, s^*, \pi^*)$. Suppose also that the incumbent
solution $(\hat{z}, \hat{s}, \hat{\pi})$ is not optimal (i.e., $\hat{\pi}_i > v^*$). Finally,
let $\hat{s} = s^h$ for some $h \in H$ and $\pi^h$ be the shortest path
in response to $z^*$ and $s^h$. Observe that $\hat{s}z^* = 0$ or $\hat{s}z^* \geq 1$
because $s$ and $z$ are 0–1 vectors. By contradiction, we
assume that $\hat{s}z^* = 0$. So, we have

$$v^* \geq \pi^h_i$$

(this is true for any $h \in H$,

see constraints (9) in problem (8)–(12));
approach may greatly benefit from a reduction in computing time needed to solve any given instance of the attacker problem. We now describe how to solve the interdiction problem more efficiently so as to improve the scalability of the proposed solution methodology.

3.2. Heuristic Solutions to the Interdiction Problem

Although the validity of our branching strategy relies upon finding optimal solutions to the interdiction problems, we use an heuristic approach to generate valid lower bounds and supervalid inequalities for the attacker problems. The approach consists of a greedy strategy that starts with an empty set of interdictions and iteratively adds components according to a greedy rule. At each iteration, the newly selected component for interdiction is the linkage on the current shortest path whose delay results in the maximum length increase of the $s-t$ shortest path in the graph.

An outline of the greedy procedure is provided below. In the pseudocode, $\mathcal{P}$ denotes the set of shortest paths processed during the heuristic execution, $P^m$ is the shortest path after the selection of $m-1$ interdictions, with $1 \leq m \leq R$, and $L^m$ denotes its length. Finally, $P_r$ is the shortest path computed in the graph $G = (N, A)$ when arc $k$ is removed and $L_k$ is the length of path $P_r$.

Initialization Phase

Let $P$ be the shortest path in $G = (N, A)$ and $L$ its length; Set $P^1 = P$; $L^1 = L$; $\mathcal{P} = \{P\}$; $\bar{s}_j = 0$ for each $j \in A$.

Iterative Phase

for $m = 1, \ldots, R$ do begin

for each arc $k$ in $P^m$ s.t. $\bar{s}_k = 0$ do begin

$\hat{P}_k = \text{ComputeShortestPath} (N, A\backslash \{k\})$;

$\bar{v}_k = \min\{L^m + d_k, L_k\}$;

end

$\hat{k} = \arg \max_{k \in P^m} \{\bar{v}_k\}$;

$c_{\hat{k}} = c_k + d_{\hat{k}}$; $\bar{s}_{\hat{k}} = 1$;

if $\bar{v}_\hat{k} = L_k$ then \( \backslash \backslash \) the shortest path in response to $\bar{s}$ has changed

$P^{m+1} = \hat{P}_k$; $L^{m+1} = L_k$; $\mathcal{P} = \mathcal{P} \cup \{P^{m+1}\}$;

else $P^{m+1} = P^m$; $L^{m+1} = \bar{v}_k$

end

Return $\bar{s}, \mathcal{P}, P^{R+1}, L^{R+1}$.

It should be noted that in the heuristic step where the new arc $k$ is selected, there may be several arcs whose interdiction results in the same maximum value of $\bar{v}_k$. In this event, we break ties by selecting the arc that, when interdicted, generates an $s-t$ shortest path different from the current one. Further ties are resolved by selecting the arc with the highest delay. The selection step also needs be modified when the heuristic is used at a generic node $n$ of the search tree other than the root node. In this case, we must not process the arcs in $P^m$ that have already been fortified, i.e., the arcs in the set $F_n$.

At termination, $P^{R+1}$ is the shortest path in response to the interdiction plan $\bar{s}$. Its length $L^{R+1}$ represents a lower bound to the optimal solution of the attacker problem. We supply the mixed-integer programming solver with this bound so as to cut off some of the branch-and-bound nodes at an early stage. Note that when $R = 1$, the solution found by the greedy heuristic is optimal and therefore there is no need to call the MIP solver.

To further reduce the number of branches needed to solve the MIP formulation of the attacker problem, we use the set of shortest paths $\mathcal{P}$ to generate the following supervalid inequalities, which are then added to the SPI($F_n$) models:

$$\sum_{k \in P^m} s_k \geq 1, \quad \forall m = 1 \cdots |\mathcal{P}|. \quad (14)$$

Constraints (14) simply force the interdiction of at least one of the arcs in each path generated by the heuristic.

3.3. Variable Fixing Rules

In this section, we present two different rules that can be used to eliminate some interdiction variables from the attacker problems, thus accelerating the solution time of the implicit enumeration procedure.

3.3.1. Bellman Fixing Rule. Let $\pi^i_j$ denote the length of the shortest path from the source node $s$ to a generic node $i$ in the original graph $G = (N, A)$ without any interdictions. Let $\eta^i_j$ denote the length of the shortest path from the source node $s$ to node $i$ in graph $G$ where all the arcs are interdicted.

**Proposition 3.1.** If $\eta^i_j - \pi^i_j \leq c_{ij}$, then the interdiction variable $s_k$ associated with the arc $k = (i, j)$ can be set to zero.

This proposition is based on the simple observation that, if in the graph where all the arcs are interdicted, node $j$ can be reached through a path that is shorter than the shortest path through node $i$ in the graph without interdictions, then interdicting arc $(i, j)$ is never beneficial for the attacker. Therefore, in the initialization phase of the implicit enumeration algorithm, we check the above Bellman-like condition for each arc $k$ in the graph and exclude the interdiction variables associated with the arcs satisfying this condition from further consideration.
3.3.2. K-Shortest-Paths (KSP) Fixing Rule. Assume that $\pi'_t$ is the objective value of the optimal solution to the SPI problem obtained at the root node of the enumeration tree. We denote by $\mathcal{P}_r$ the set of $s-t$ paths in $G = (N, A)$ whose nominal length is less than or equal to $\pi'_t$. Let $f_k$ denote the frequency of the appearance of arc $k$ in the $s-t$ paths in $\mathcal{P}_r$.

**Proposition 3.2.** If, for some arc $k$, we have that $f_k = 0$, then the interdiction variable $s_k$ can be set to zero.

The rationale behind this proposition is as follows. When none of the network components is hardened, an optimal attack on $R$ linkages can thwart all the $s-t$ paths with length up to $\pi'_t$. When some fortifications are made in subsequent steps of the algorithm, an optimal attack will be less disruptive and will only be able to thwart a subset of these paths. As such, offensive resources should be targeted so as to maximize the number of shortest paths in $P_r$ that can be disrupted. Clearly, resources utilized to interdict arcs that do not appear in any of these paths will be squandered. Consequently, after solving the interdiction problem at the root node, we can restrict the focus of the solution approach only to those arcs that belong to at least one of the paths in $P_r$ and eliminate all the other interdiction variables. In our implementation of the modified implicit enumeration procedure, the paths in $P_r$ were computed by using the shortest-path ranking algorithm proposed in Martins and Santos (2000). This algorithm proved to be very efficient for generating and ranking millions of shortest paths between pairs of nodes in networks of significant size. In our computational experiments, the computing time for generating the paths in $P_r$ was negligible as compared to the computing time of the overall procedure.

4. An Alternative Formulation of the Attacker Problem

The rationale behind Proposition 3.2 also suggests an alternative way of formulating the attacker problem at each node $n$ of the enumeration tree other than the root node.

Let $P_m$ be the $m$th shortest path in $P_r$ and $L_m$ its length. An alternative formulation for SPI($F_n$), denoted in the following as KSPI($F_n$), is as follows:

\[
\begin{align*}
\max \quad & L, \\
\text{s.t.} \quad & L \leq L_m + \sum_{k \in P_m} d_k s_k \quad \forall m \in \mathcal{P}_r, \\
& \sum_{k \in A} s_k \leq R, \\
& s_k \leq 1 - z_k \quad \forall k \in F_n, \\
& s_k \in \{0, 1\} \quad \forall k \in A.
\end{align*}
\]

At optimality, the variable $L$ in the objective function (15) represents the length of the shortest path after the most disruptive interdiction. This is enforced by constraints (16). This formulation has $|A|$ integer variables and $|\mathcal{P}_r| + |F_n| + 1$ constraints. Even here, some of the interdiction variables can be eliminated by using Propositions 3.1 and 3.2. The original formulation of the interdiction problem has $|A|$ integer variables, $|N|$ continuous variables, and $|A| + |F_n| + 1$ constraints. Given that the number of paths in the set $P_r$ is largely problem specific, it is not possible to establish a priori the dominance of one formulation over the other in terms of dimension and difficulty of solution. An advantageous feature of the alternative formulation is that the number of constraints (16) can be reduced every time we solve a new problem at a child node of the enumeration tree. Assume, for example, that $L^*$ is the optimal solution to problem (15)–(19) at node $n$. When we solve the interdiction problem at $n$'s child node, we can safely remove from the formulation all the constraints associated with those paths whose length is strictly greater than $L^*$. The rationale behind this is that if the length of these paths cannot be worsened with the set of interdictions available at node $n$, they certainly cannot be worsened when the set of candidate interdictions is restricted at the child node by hardening an additional component. As we go down in the tree, we can use this problem reduction step iteratively to obtain problems that are increasingly smaller. The use of this alternative formulation becomes, therefore, more attractive when solving problems with larger values of $Q$, because the parameter $Q$ determines the depth of the enumeration tree.

An in-depth discussion of the trade-offs between using formulation (15)–(19) versus the original SPI formulation within the enumeration procedure is given in §5.

Note that the formulation (15)–(19) can be generalized to solve generic instances of SPI by computing an upper bound of the number of shortest paths that can be interdicted with $R$ resources and including a constraint (16) in the formulation for each of these shortest paths.

5. Experimental Results

5.1. Test Problems and Implementation Issues

In our computational testing, we use directed square grid graphs with a number of rows (and columns) in [7, 10, 12, 15]. The grids are generated by following the rules described in Israeli and Wood (2002) so as to obtain networks with the same topology as the ones used to test the shortest-path interdiction model without fortification. The characteristics of the networks in terms of nodes and arcs are given in Table 1.
Given a positive integer $c$ that represents the maximum arc cost, arc costs are uniformly chosen in $[0, c]$. Then, for a given $c$, we consider three different delay distributions: (i) delays uniformly chosen in $[0, c/2]$; (ii) delays uniformly chosen in $[0, c]$; and delays uniformly chosen in $[0, 2c]$. In our experimental analysis, we use two values for $c$ ($c = 10$ and $c = 100$), which sums up to 6 cost-delay combinations for each grid type. For a fixed grid dimension and cost-delay combination, we generate three instances with different random arc attributes. Cardinality constraints on offensive and defensive resources are considered, where the maximum number $Q$ of fortifications belongs to $[3, 5, 7]$ and the maximum number $R$ of interdictions is in $\{1, 2, 3, 4, 5\}$. Each instance is solved for each combination of $Q$ and $R$ in the aforementioned sets. Thus, in total, we solved 1,080 instances. Because solving the biggest grids (15-15) for the largest values of $Q$ and $R$ (i.e., $Q = 7$ and $R = 5$) was significantly more difficult than all the other cases, we divided the analysis in two parts: We first analyze the computational performance of the algorithms on the 1,062 instances of small and medium difficulty (i.e., all grids and parameter combinations except the grids 15-15 with $Q = 7$ and $R = 5$). We then report the results on the remaining 18 critical problems, which were solved with a time limit of 4 hours.

We coded the algorithms in C++ and ran them on a PC with an Intel Core 2, 2.66 GHz processor, and 2 GB of RAM. The commercial solver CPLEX version 11 was used to solve the MIP problems at each node of the enumeration tree.

Two implementations of the implicit enumeration scheme are considered depending on the formulation used for the lower-level interdiction problem at each node of the enumeration tree, namely, $\text{SPI}(F_r)$ or $\text{KSPI}(F_r)$ described, respectively, in §§2 and 4. For each of the two formulations, three variants are investigated. (i) The basic formulations are directly solved with CPLEX without any attempt at reducing the search space. These two versions are referred to as SPINot and KSPINot, respectively. (ii) The implicit enumeration scheme is equipped with the Bellman and KSP variable fixing rules described in §3.3. Additionally, the greedy heuristic is used at the root node. These two versions are referred to as SPIFix and KSPIFix. (iii) In addition to the variable fixing rules, the greedy heuristic and the inequalities (14) are used at each node of the enumeration tree. These two versions are referred to as SPIAll and KSPIAll.

5.2. Algorithmic Performance Comparison

5.2.1. Easy and Medium Difficulty Instances. In Figure 2, a performance comparison is made among the six variants of the implicit enumeration scheme described earlier. More specifically, for each variant, we report the percentage of instances solved in a given time interval where the time classes, represented on the $x$-axis of the bar chart in Figure 2, are as follows: less than 1 second, between 1 second and 1 minute, between 1 minute and 10 minutes, between 10 minutes and 1 hour, between 1 hour and 3 hours, and, finally, over 3 hours. These aggregated results highlight two important facts: (1) the use of the variable fixing rules and of the heuristic allows us to solve more instances in shorter times and (2) the use of the KSPI formulation in the lower-level interdiction problem makes the enumeration approach more efficient. A direct comparison between SPIAll and KSPIAll, for example, shows that the percentage of instances solved in less than 1 minute with KSPIAll is as high as 85.4, whereas for the counterpart SPIAll, the percentage is equal to 79.3. Additionally, the number of instances solved in over one hour is negligible when any of the KSPI-based variants are used.

To analyze in more depth the performance trend of the different approaches, we present some detailed results in Table 2. For a given grid dimension and a given cost-delay distribution, the average computational time, expressed in CPU seconds, is given for each of the six variants of the algorithm. The problem name, shown in the first column, contains four fields that represent, respectively, the number of rows and columns in the grid graph, the maximum cost $c$, and the maximum number $Q$ and $R$ of interdictions.

Table 1 Grid Dimensions

<table>
<thead>
<tr>
<th>Grid</th>
<th>Nodes</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-7</td>
<td>51</td>
<td>188</td>
</tr>
<tr>
<td>10-10</td>
<td>102</td>
<td>416</td>
</tr>
<tr>
<td>12-12</td>
<td>146</td>
<td>618</td>
</tr>
<tr>
<td>15-15</td>
<td>227</td>
<td>996</td>
</tr>
</tbody>
</table>

Figure 2 Frequency Analysis
As to be expected, the overall results show that the SPIAll performance on the 10 × 10 and the 12 × 12 grids. The computing times to solve the instances in the first block can be as high as double the times to solve instances of equal size in the second block (see, for example, the instance 10-10-10-20 versus the instance 10-10-100-200 or the instance 12-12-10-20 versus the instance 12-12-100-200). The impact of the cost-delay range on the performance of the SPI variants seems to be negligible for the 15 × 15 grids. On the other side, the performance of the KSPI implementations clearly deteriorates when wider cost ranges are considered. This is particularly evident for the bigger instances in the test bed (e.g., 15-15-100-200).

Overall, the preeminence of the KSPI-based variants seems quite overwhelming if we exclude the 15-15-100-200 instances. Possible causes that may render these instances more difficult to solve by the KSPI implementations are discussed in the next paragraphs.

In Table 3, we provide some information related to the fixing rules and the KSP paths generated by the approaches. The level of aggregation of the results in Table 3 is the same as in Table 2. For each grid dimension and each cost-delay combination, Table 3 provides the following information: the average number (KSPPath) of s − t shortest paths whose cost does not exceed the upper bound to the trilevel defender-attacker-user problem objective computed at the root node via the SPI formulation with no fortifications; the average percentages of noninterdictable arcs computed by the Bellman-like conditions (%Bfix) and by the k-shortest-path tools (%Kfix); and, in the last column, the average percentage of arcs fixed by both the rules described above. First of all, we observe that the KSP fixing rule is very effective in reducing the search space for both easy and critical instances and is more powerful than the Bellman fixing rule. Nevertheless, the Bellman fixing rule is always able to identify some additional variables to fix that cannot be identified by

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Time Comparison Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProbName</td>
<td>SPIAll</td>
</tr>
<tr>
<td>7-7-10-5</td>
<td>0.44</td>
</tr>
<tr>
<td>7-7-10-10</td>
<td>3.05</td>
</tr>
<tr>
<td>7-7-10-20</td>
<td>5.90</td>
</tr>
<tr>
<td>10-10-10-5</td>
<td>32.20</td>
</tr>
<tr>
<td>10-10-10-10</td>
<td>79.36</td>
</tr>
<tr>
<td>10-10-10-20</td>
<td>392.65</td>
</tr>
<tr>
<td>12-12-10-5</td>
<td>108.78</td>
</tr>
<tr>
<td>12-12-10-10</td>
<td>545.66</td>
</tr>
<tr>
<td>12-12-10-20</td>
<td>1,008.21</td>
</tr>
<tr>
<td>15-15-10-5</td>
<td>69.09</td>
</tr>
<tr>
<td>15-15-10-10</td>
<td>635.08</td>
</tr>
<tr>
<td>15-15-10-20</td>
<td>927.57</td>
</tr>
<tr>
<td>7-7-10-50</td>
<td>2.50</td>
</tr>
<tr>
<td>7-7-10-100</td>
<td>1.50</td>
</tr>
<tr>
<td>7-7-10-200</td>
<td>3.55</td>
</tr>
<tr>
<td>10-10-100-50</td>
<td>11.38</td>
</tr>
<tr>
<td>10-10-100-100</td>
<td>85.23</td>
</tr>
<tr>
<td>10-10-100-200</td>
<td>218.75</td>
</tr>
<tr>
<td>12-12-100-50</td>
<td>86.57</td>
</tr>
<tr>
<td>12-12-100-100</td>
<td>243.70</td>
</tr>
<tr>
<td>12-12-100-200</td>
<td>579.62</td>
</tr>
<tr>
<td>15-15-100-50</td>
<td>69.09</td>
</tr>
<tr>
<td>15-15-100-100</td>
<td>865.63</td>
</tr>
<tr>
<td>15-15-100-200</td>
<td>916.69</td>
</tr>
<tr>
<td>Average</td>
<td>287.18</td>
</tr>
</tbody>
</table>

and the maximum delay d. Each row of Table 2 is thus averaged over 45 instances (3 instances times 15 Q − R combinations) when the grid dimension is in [7, 10, 12] and over 42 instances for the 15-row grids (3 instances times 14 Q − R combinations). The results are grouped according to the value of c: In the first block of rows, results are shown for costs ranging in [0, 10] while the second group of rows relates to instances characterized by a higher variability of cost coefficients that vary in the range [0, 100]. In each row, the most efficient approach is displayed via a bold entry. In the last row of Table 2, average computational times over all the instances are given.
using the KSP fixing rule alone, as shown by comparing the fourth and the last columns of Table 3.

By looking at the KSPPath column in Table 3, we can observe that the average number of KSP paths for the instances in the class 15-15-100-200 is significantly higher than the KSPath number for the other classes. This explains the greater difficulty of the KSPI formulation at solving these problems. Recall, in fact, that whereas the number of constraints in the SPI formulation of the interdiction problem strictly depends on the number of arcs, in the KSPI formulation the number of constraints is determined by the number of KSP shortest paths identified at the root node. This observation suggests the use of a hybrid approach, where the interdiction problem formulation to be used in subsequent nodes of the enumeration tree (SPI or KSPI) is selected on the fly at the root node after the number KSPath is computed. If this number is significantly higher than the number of arcs (as is the case for some of the 15-15-100-200 problems), then the SPI formulation should be selected. Conversely, if the number of KSP paths is in the same order of magnitude as the number of arcs, then the KSPI formulation should be used.

It is important to note that the number of KSP paths is largely problem specific and may vary considerably even for networks of the same size and with the same cost and delay distributions. This is true, for example, for the class 15-15-100-200, where the high KSPath number and consequently the high average computing times of the KSPI algorithms are mainly because of one of the three instances. In support of this observation, we provide in Table 4 the disaggregated results for two instances of type 15-15-100-200: In the first column, the name of the instance is given where the last field refers to the instance index ranging in \{0, 1, 2\}. This is followed by the number \(Q\) of fortifications and the number \(R\) of interdictions. The next four columns give the information relative to the fixing rules in the same format as in Table 3. Finally, the computational time expressed in CPU seconds is reported for the two most efficient implementations of each formulation type, i.e., SPIAll and KSPIAll. These disaggregated results reveal that KSPIAll performs better than SPIAll on the instance 15-15-100-200-0. Note that, in this case, the number of KSPPaths is never over 3,500 and there are some \(Q - R\) difficult combinations (see, for example, the 5-4, 5-5, and 7-4 cases) that can be solved by KSPIAll in about one-fourth of the time required by SPIAll. A completely different behavior can be observed on the instance 15-15-100-200-1. Here, there are three cases (the 3-4, 5-4, and 7-4 cases) where the time required by KSPIAll is remarkably greater than the one required by SPIAll. Additionally, there are 2 cases for which KSPIAll needs a huge time (see the 3-5 and 5-5 cases for which the times are, respectively, about 4 and over 15 hours).

In all these 5 cases, the poor performance of KSPIAll is because of the high number of KSPPaths that can reach the value 36,805. These results seem to thus corroborate the fact that a hybrid approach may be used successfully to overcome the weaknesses of the SPI-based and the KSPI-based approaches.

Finally, note that the number of KSP paths depends on the amount of interdiction resources: The larger the value of \(R\), the greater the number of shortest paths that can be interdicted with the \(R\) resources. This may render the use of the KSPI formulation more critical for solving problems with large values of the parameter \(R\). Nonetheless, the range of values considered in our computational tests (\(R\) between 1 and 5) seems to be the most interesting from a practical point of view. The simultaneous loss of a much larger number of components, in fact, seems quite unrealistic or at least quite unlikely to occur for most problem situations. On the other side, the parameter \(Q\) may vary in a much broader range, given the high variability of the amount of defensive resources that may be allocated for protection purposes.

### 5.2.2. Difficult Instances

In Table 5, we report some summary results obtained on the most critical instances in the test bed. We also include in the comparison the implementation of the algorithm that uses the original interdiction problem as proposed in Israeli and Wood (2002), i.e., the SPI formulation without the secondary objective and none of the speedup

<table>
<thead>
<tr>
<th>ProbName</th>
<th>Arcs</th>
<th>KSPPath</th>
<th>%Bfix</th>
<th>%Kfix</th>
<th>%BKfix</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-7-10-5</td>
<td>188</td>
<td>87.53</td>
<td>26.95</td>
<td>72.73</td>
<td>21.81</td>
</tr>
<tr>
<td>7-7-10-10</td>
<td>188</td>
<td>64.47</td>
<td>14.01</td>
<td>68.72</td>
<td>11.21</td>
</tr>
<tr>
<td>7-7-10-20</td>
<td>188</td>
<td>94.09</td>
<td>7.67</td>
<td>67.63</td>
<td>4.57</td>
</tr>
<tr>
<td>10-10-5</td>
<td>416</td>
<td>118.47</td>
<td>15.95</td>
<td>77.92</td>
<td>14.25</td>
</tr>
<tr>
<td>10-10-10</td>
<td>416</td>
<td>390.33</td>
<td>9.54</td>
<td>72.82</td>
<td>7.34</td>
</tr>
<tr>
<td>10-10-10-5</td>
<td>416</td>
<td>249.13</td>
<td>7.37</td>
<td>74.44</td>
<td>5.66</td>
</tr>
<tr>
<td>12-12-10</td>
<td>618</td>
<td>675.13</td>
<td>11.97</td>
<td>80.87</td>
<td>10.63</td>
</tr>
<tr>
<td>12-12-10</td>
<td>618</td>
<td>450.73</td>
<td>7.71</td>
<td>79.36</td>
<td>6.31</td>
</tr>
<tr>
<td>12-12-20</td>
<td>618</td>
<td>438.27</td>
<td>4.05</td>
<td>76.43</td>
<td>3.03</td>
</tr>
<tr>
<td>15-15-10</td>
<td>996</td>
<td>791.43</td>
<td>10.07</td>
<td>82.96</td>
<td>8.96</td>
</tr>
<tr>
<td>15-15-10</td>
<td>996</td>
<td>223.29</td>
<td>7.56</td>
<td>85.29</td>
<td>6.75</td>
</tr>
<tr>
<td>15-15-20</td>
<td>996</td>
<td>1,551.88</td>
<td>3.25</td>
<td>73.86</td>
<td>2.14</td>
</tr>
<tr>
<td>7-7-100-50</td>
<td>188</td>
<td>33.40</td>
<td>33.05</td>
<td>78.13</td>
<td>30.71</td>
</tr>
<tr>
<td>7-7-100-100</td>
<td>188</td>
<td>99.33</td>
<td>10.46</td>
<td>71.56</td>
<td>8.19</td>
</tr>
<tr>
<td>7-7-100-200</td>
<td>188</td>
<td>110.07</td>
<td>7.27</td>
<td>71.63</td>
<td>4.54</td>
</tr>
<tr>
<td>10-10-100-50</td>
<td>416</td>
<td>204.13</td>
<td>18.99</td>
<td>80.22</td>
<td>16.68</td>
</tr>
<tr>
<td>10-10-100-100</td>
<td>416</td>
<td>196.33</td>
<td>8.65</td>
<td>78.27</td>
<td>7.24</td>
</tr>
<tr>
<td>10-10-100-200</td>
<td>416</td>
<td>220.13</td>
<td>4.97</td>
<td>76.35</td>
<td>3.48</td>
</tr>
<tr>
<td>12-12-100-50</td>
<td>618</td>
<td>686.13</td>
<td>13.27</td>
<td>80.19</td>
<td>12.28</td>
</tr>
<tr>
<td>12-12-100-100</td>
<td>618</td>
<td>242.87</td>
<td>7.28</td>
<td>81.26</td>
<td>6.60</td>
</tr>
<tr>
<td>12-12-100-200</td>
<td>618</td>
<td>632.20</td>
<td>3.45</td>
<td>79.59</td>
<td>2.41</td>
</tr>
<tr>
<td>15-15-100-50</td>
<td>996</td>
<td>791.43</td>
<td>10.07</td>
<td>82.96</td>
<td>8.96</td>
</tr>
<tr>
<td>15-15-100-100</td>
<td>996</td>
<td>534.00</td>
<td>6.09</td>
<td>83.48</td>
<td>5.28</td>
</tr>
<tr>
<td>15-15-100-200</td>
<td>996</td>
<td>3,266.31</td>
<td>3.85</td>
<td>79.37</td>
<td>2.84</td>
</tr>
</tbody>
</table>
expedients. This implementation is simply referred to as SPI. The first two rows in Table 5 report, respectively, the number of instances for which optimality could not be proven within the time limit of four hours and the number of solutions that were, in fact, not optimal. The next two rows display the maximum and average gap for each of the implementations. Finally, the last two rows show the average time employed to solve the 18 problems and the average time computed over only the instances that did not exceed the time limit.

The results on these difficult instances confirm many of the previous findings of the analysis, namely, that the KSPI variants are more efficient in terms of both the number of optimal solutions found and computing time. Although the maximum gap is higher for these implementations, this is entirely because of the problematic instance 15-15-100-200-1 that was discussed in the previous section. Without this instance, the average gap is only 1% for the KSPIFix and KSPIAll and 0% for the KSPINot, which solves all the other problems to optimality. As previously noted, the speedup expedients seem less effective on these variants. Although the average computing time slightly decreases as more expedients are added, the overall solution quality does not necessarily improve. The SPI variants, on the contrary, benefit from all of the proposed expedients. The straightforward SPI version is the slowest, does not solve to optimality 28% of the problems, and has the largest average gap and a maximum gap as high as 6.25%. The SPI version with all the expedients has the minimum average gap and the minimum maximum gap among all the implementations. Overall, the partial average time is reduced by

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Performance Comparison on Two Difficult Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProbName</td>
<td>Q</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-0</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>3</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>5</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>7</td>
</tr>
<tr>
<td>15-15-100-200-1</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Performance Analysis on the 15-15 Grids for Q = 7 and R = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>SPINot</td>
</tr>
<tr>
<td>No. time exceeded</td>
<td>11</td>
</tr>
<tr>
<td>No. suboptimal</td>
<td>5</td>
</tr>
<tr>
<td>Max gap (%)</td>
<td>6.25</td>
</tr>
<tr>
<td>Average gap (%)</td>
<td>0.75</td>
</tr>
<tr>
<td>Average time</td>
<td>10,706.35</td>
</tr>
<tr>
<td>Partial average time</td>
<td>4,902.04</td>
</tr>
</tbody>
</table>
about 30% with the addition of the secondary objective (SPINot) and by an additional 6% with the use of the fixing rules (SPIFix). That time is almost halved when all the expedients are used (SPIAll).

Finally, it is interesting to note that a hybrid approach that selects the interdiction formulation on the fly as suggested in the previous section would have solved all the problems to optimality within the four hours.

5.3. Impact of Offensive and Protective Resources on the Computing Time

In Figure 3, a scalability comparison with respect to the number of protective resources \( Q \) and the number of offensive resources \( R \) is made between SPIAll and KSPIAll on medium-sized grids, namely, the 10 \( \times \) 10 grids for the most critical cost-delay combinations in the test bed, i.e., for costs ranging in [0, 10] and delays in [0, 20], and costs ranging in [0, 100] and delays in [0, 200], respectively. Specifically, in Figure 3, we report on a logarithmic scale the average time computed over the 3 instances of the same type for each of the 15 \( Q-R \) combinations analyzed in our computational study. The values of \( Q \) and \( R \) are displayed along the \( y \)-axis and the \( x \)-axis, respectively. Bars that are below the axes intercept of the logarithmic graph (fixed at one) have a slightly lighter shade. These bars represent average times below one second. The two cost-delay distributions exhibit quite similar behaviors in terms of increase in the computational time as \( Q \) and \( R \) grow.

More specifically, both (a) and (b) of Figure 3 highlight that the impact of the number \( R \) of offensive resources is more decisive than the number \( Q \) of protective resources in that: (1) for a fixed value of \( Q \), the computational time of both approaches increases by one order of magnitude for a unitary increase of \( R \); (2) for a fixed value of \( R \), the time required by instances characterized by the biggest value of \( Q \) in the test bed, namely, \( Q = 7 \) is about one order of magnitude larger than the time required by instances with the smallest value of \( Q \) in the set considered, namely, \( Q = 3 \); and (3) when the number of interdictions is small (see \( R = 1 \) and \( R = 2 \) in Figure 3), the computational times are negligible (never over one second) for both approaches independent of the number of fortifications. These results indicate that especially when the expected number of possible losses is low, the proposed method can be used to identify the optimal network components to be hardened for relatively large networks and large protection budgets.

5.4. Impact of Offensive and Protective Resources on the Objective Value

To analyze the potential benefits of investing in protection efforts in the face of disruptions of different sizes, we provide in Figure 4 a visual display of the impact that different levels of protection may have on the efficiency loss of a system. We perform the analysis for two medium instances (grids 10-10-10-20-2 and 10-10-100-200-2). A similar analysis can be conducted for any other grid. Several important insights can be drawn from the graphs in Figure 4. First, they show to what extent a system is vulnerable to interdiction. As an example, if no protective measures are implemented (\( Q = 0 \)) on grid 10-10-10-20-2 (Figure 4(a)), the worst-case loss of a single arc can increase the travel cost by 50%, and the worst-case loss of 5 out of the 416 arcs can cause a cost increment of 80%. By comparison, the network 10-10-100-200-2 (Figure 4(b)) seems less vulnerable; the cost increase because of losses in the range 1 to 5 varies between 15% and 50%. Second, the graphs show the impact of protection efforts for each level of interdiction. For the first network, the optimal protection of a single arc can drastically reduce the impact of disruption from 50% to
20% for a single loss and from 80% to 57% for 5 losses. On the second network, protecting a single arc is less effective for all interdiction sizes and is completely futile for \( R = 1 \) and \( R = 4 \). Hence, these graphs also highlight possible resource wastage. As an additional example, if only one loss is expected on the grid 10-10-10-20-2, there is no added value in protecting two or three linkages, but an additional reduction in efficiency loss can be achieved by protecting four arcs. For both networks, the optimal protection of 10 links renders the system completely immune to disruption against any number of losses in the range considered. Finally, from these graphs, it is possible to deduce what is the minimum amount of resources that must be employed to guarantee that the travel cost does not increase above a given percentage threshold. As an example, a protection plan that guarantees that the travel cost does not increase by more than 30% if 3 links are lost simultaneously (case \( R = 3 \)) requires the protection of 6 linkages in the first grid and 3 links in the second grid.

5.5. Solving the Interdiction Problem Heuristically

Although most protection interdiction studies assume that the attacker always acts optimally, assuming complete rationality on the part of the enemy in real-world scenarios may be unrealistic (Smith, Lim, and Sudargho 2007). For example, the enemy may be unable to compute the optimum attack strategy because of limited tools or information and hence may decide to act according to an heuristic, context-specific strategy. Additionally, when dealing with very large problems, solving the lower-level problem to optimality repetitively may be computationally prohibitive. In this section, we analyze the accuracy of the solutions obtained when the defender optimum is computed based on an heuristic solution to the interdictor problem rather than on the true attacker optimum. In the analysis, we use the solutions obtained with the heuristic described in §3.2.

Table 6 shows some average results obtained on the medium and difficult instances as well as the average gap from the optimal solution and the average computing time for each group of instances. It also reports the average computing time of the KSPIAll algorithm, which on average was the most efficient implementation. It can be observed that the greedy heuristic is quite effective in producing good approximate solutions in a fraction of the time required by the exact methods. As shown in Table 7, the optimal solution is found for 37% of the problems, and in 70% of the cases the gap is below 5%. Additionally, the heuristic approach is robust in that it produces similar gaps for grids of different sizes. The solution quality seems mostly affected by the number of offensive resources, because bigger gaps were observed on the instances with \( R = 5 \). Conversely, the gaps are not particularly sensitive to the amount of protective resources. In terms of computing time, 70% of the instances were solved in less than 1 second, only 1% required more than 1 minute, and none of the instances required more than 2 minutes. In summary, using heuristic solutions to the inner problems seems to be a viable and reliable option for solving large protection interdiction problems, especially if the expected number of possible simultaneous losses is low.
6. Summary and Future Research

We develop a multilevel optimization model, called the shortest-path interdiction problem with fortification (SPIF), to identify optimal protection strategies in shortest-path networks. With a simple example, we show that the explicit inclusion of protection decisions into an optimization model can produce much sounder protection plans than simply using a shortest-path interdiction model (such as the one proposed in Israeli and Wood 2002) to identify the critical components to be hardened. The impact of a worst-case disruption can be significantly reduced and some offensive resources can even be rendered ineffective. Clearly, the benefit of including protection decisions into the model comes at a cost—the resulting model is a trilevel program and its solution requires the devise of sophisticated approaches tailored to this multilevel structure.

We propose an implicit enumeration algorithm for solving SPIF, that exploits the solution to the lower-level interdiction problem to reduce the number of protection strategies to be evaluated. Given that the computation effort of the proposed method is largely determined by the number of lower-level interdiction problems to be solved in the enumeration tree and by the efficiency with which these can be solved, we introduce several expedients to reduce the search tree size and speed up the computation. These include some variable fixing mechanisms and the use of an heuristic and some supervalid inequalities at each node of the search tree. We also propose an alternative formulation for the shortest-path interdiction problem that generally outperforms the formulation proposed in Israeli and Wood (2002) when used within the implicit enumeration algorithm.

An extensive computational exercise demonstrates that the proposed methodology is quite effective at solving protection problems on networks with more than 200 nodes and almost 1,000 arcs. We also show how the distribution of the arc lengths and delays can affect the efficiency of different implementation variants and suggest the use of a hybrid approach to enhance the performance of our solution method. Finally, we study the sensitivity of the approach to variations of the number of fortifications and interdictions, and we show that the effect of the number of offensive resources is more crucial than the number of protective resources.

The need to develop quantitative methods for improving the ability of logistics systems to withstand intentional or accidental disruptions has been widely recognized in the scientific community, and among practitioners, supply chain managers, and government agencies. Although in recent years a few researchers have begun studying mathematical models for the protection of infrastructure systems, the level of sophistication of these models needs to be further improved in order to cope with the complexities of today’s logistics systems. Currently, we are exploring different variations of the SPIF model, which may increase its applicability to solve real-world problems. As an example, we are investigating a stochastic variant of SPIF where the number of possible losses is uncertain and unknown to the defender and the attacks are successful only with a given probability. In another model variation, we assume that the protection of a network component only reduces its probability of disruption and that the extent to which the disruption probability is reduced depends on the level of investment in protective measures. Another important extension involves the use of more complex network flow models in the lower-level user problem such as multicommodity and multiple origin-destination flow problems. We are also exploring the use of different measures of network performances that take into account the network topology as well as link capacities, travel demands, and congestion issues. An example is the network robustness index proposed in Scott et al. (2006). Models using this performance measure are significantly more difficult to solve because they require solving user assignment equilibrium models to reroute all traffic upon interdiction. Nevertheless, the use of heuristic interdiction solutions within a tree search scheme may be a viable approach to obtain good approximate solutions for these complex problems. Finally, we plan to develop multiperiod versions of network protection models where protection resources become available in different periods and the objective is to maximize the discounted future reliability of the transportation system. Models where the offensive and defensive efforts are continuous rather than discrete are also in our agenda of future lines of research.

In summary, we hope that this study lays the ground work for the development of many other models able to identify cost-efficient protection strategies that enhance the reliability and safety of our road networks and other ground transportation systems.

Acknowledgments

This research was supported by Engineering and Physical Sciences Research Council (EPRSC) Grant EP/E048552/1

Table 7 Frequency of Heuristic Gap

<table>
<thead>
<tr>
<th>% Gap</th>
<th>% Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>37.41</td>
</tr>
<tr>
<td>≤ 1</td>
<td>39.44</td>
</tr>
<tr>
<td>≤ 3</td>
<td>47.41</td>
</tr>
<tr>
<td>≤ 5</td>
<td>70.93</td>
</tr>
<tr>
<td>≤ 10</td>
<td>94.07</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>5.93</td>
</tr>
</tbody>
</table>
and is gratefully acknowledged. The authors also thank the associate editor and two anonymous referees for their valuable and constructive comments.

References


