Discrete Optimization

An exact method for binary fortification games

Markus Leitner [a], Ivana Ljubić [b], Michele Monaci [c], Markus Sinnl [d], Kübra Tanınmış [d,*]

[a] *Department of Operations Analytics, Vrije Universiteit Amsterdam, the Netherlands*
[b] *ESSEC Business School of Paris, France*
[c] *Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy*
[d] *Institute of Production and Logistics Management, Johannes Kepler University Linz, Austria*

A B S T R A C T

A fortification game (FG) is a three-level, two-player hierarchical game, also known as defender-attacker-defender game, in which at the uppermost level, the defender selects some assets to be protected from potential malicious attacks. At the middle level, the attacker solves an interdiction game by depreciating unprotected assets, i.e., reducing the values of such assets for the defender, while at the innermost level the defender solves a recourse problem over the surviving or partially damaged assets. Fortification games have applications in various important areas, such as military operations, design of survivable networks, protection of facilities or power grid protection. In this work, we present an exact solution algorithm for FGs, in which the recourse problems correspond to (possibly NP-hard) combinatorial optimization problems. The algorithm is based on a new generic mixed-integer linear programming reformulation in the natural space of fortification variables. Our new model makes use of *fortification cuts* that measure the contribution of a given fortification strategy to the objective function value. These cuts are generated when needed by solving separation problems, which correspond to (modified) middle-level interdiction games. We design a branch-and-cut-based solution algorithm based on fortification cuts, their strengthened versions and other speed-up techniques. We present a computational study using the knapsack fortification game and the shortest path fortification game. For the latter one, we include a comparison with a state-of-the-art solution method from the literature. Our algorithm outperforms this method and allows us to solve previously unsolved instances with up to 330 386 nodes and 1 202 458 arcs to optimality.

© 2022 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

Fortification games (FGs), also known as *defender-attacker-defender* (DAD) problems, have applications in various important areas, such as military operations, the design of survivable networks, protection of facilities, or power grid protection (Lozano & Smith, 2017; Smith & Song, 2020). A FG is a three-level hierarchical game with two players, usually denoted as *defender* and *attacker*: At the third (innermost) level the defender wants to solve some optimization problem (denoted as *recourse* problem) which depends on some resources (*assets*). At the second level, the attacker can select a subset of the assets to attack while anticipating the optimal response of the defender. Depending on the problem setting, this attack can either destroy the assets, or depreciate their usefulness for the defender. The goal of the attacker is to make

the result of the optimization problem of the defender as worse as possible. Such a two-level attacker-defender problem described so far is known as interdiction game (IG) in which case the actions of the attacker are called *interdictions*. In FGs, at the first (outermost) level, the defender tries the anticipate the decisions of the attacker and to fortify some of its assets against potential attacks. When determining her policy, the defender is aware that fortified assets cannot be interdicted by the attacker and of the possibility of recourse actions at a later time.

We focus on a particular family of fortification games which is defined as follows. Let N be the set of assets, $f_i \geq 0$ be the cost for fortification of each asset $i \in N$, and B_F be the fortification budget. Similarly, let $g_i \geq 0$ be the interdiction cost for each asset $i \in N$, and B_I be the interdiction budget. Finally, let $d_i \geq 0$ denote the depreciation of asset $i \in N$ due to interdiction. In the following, we

* Corresponding author.
  E-mail addresses: m.leitner@vu.nl (M. Leitner), ljubic@essec.edu (I. Ljubić), michele.monaci@unibo.it (M. Monaci), markus.sinnl@jku.at (M. Sinnl), kuebra.taninmis_ersues@jku.at (K. Tanınmış).

will denote by w, x and y the incidence vector of a fortification, an interdiction, and a recourse strategy, respectively. Accordingly, we denote by $W = \{w \in \{0,1\}^n : \sum_{i \in N} f_i w_i \le B_F\}$ the set of feasible fortifications and by $X = \{x \in \{0,1\}^n : \sum_{i \in N} g_i x_i \le B_I\}$ the set of possible interdictions if there would be no fortifications, where $n = |N|$. In addition, we let $X(w) = X \cap \{x \in \mathbb{R}^n : x \le 1 - w\}$ be the set of feasible interdictions for a given fortification strategy w, and we denote by $Y \subseteq \{0,1\}^n$ the feasible region of the recourse problem, which is assumed to be non-empty. Note that the feasible region of the recourse problem does not depend on the interdiction decisions, and is bounded by definition. Using these definitions, the FGs studied in this work can be defined as

$$z^* = \min_{w \in W} \max_{x \in X(w)} \min_{y \in Y} \left\{ c^T y + \sum_{i \in N} d_i x_i y_i \right\}. \tag{0-1 FG}$$

Remark 1. Notice that according to our definition of (0-1 FG), interdiction decisions do not affect the feasible region Y of the recourse problem, and each such decision is associated with a cost increase/penalty $d_i \ge 0$. However, without loss of generality, this definition also allows to model problems in which the attacker prohibits the use of the interdicted assets, i.e., where $Y(x) = Y \cap \{y \in \mathbb{R}^n : y \le 1 - x\}$. In this case, the resulting problem

$$z^* = \min_{w \in W} \max_{x \in X(w)} \min_{y \in Y(x)} c^T y$$

can be reformulated as

$$z^* = \min_{w \in W} \max_{x \in X(w)} \min_{y \in Y} \left\{ c^T y + \sum_{i \in N} \mathcal{M}_i x_i y_i \right\}$$

for sufficiently large $\mathcal{M}_i$ (see, e.g., Fischetti, Ljubić, Monaci, & Sinnl, 2019), i.e., in problem (0-1 FG) we set $d_i := \mathcal{M}_i$.

Finally, we notice that more general types of fortification games have been introduced in the literature. For example, Alderson, Brown, Carlyle, & Wood (2011) considered the case in which the feasible region of the innermost problem depends on w as well (i.e., we have $Y(w, x)$ instead of $Y(x)$), whereas Brown, Carlyle, Salmerón, & Wood (2006) addressed situations in which the third level variables are not necessarily binary and there might exist some available capacity which is invulnerable to the attack (e.g., $y \le u_0 + u(1 - x)$). These more general settings are out of scope of this paper, and we focus on a particular version defined by the problem (0-1 FG).

1.1. Contribution and outline

In this work, we present an exact solution algorithm for (0-1 FG). The algorithm is based on a new generic mixed-integer programming (MIP) formulation for (0-1 FG) which makes use of valid inequalities, denoted as *fortification cuts*. These cuts are used to measure the objective function value for given fortification strategies, and are generated when needed by solving separation problems, which correspond to (modified) IGs.

The detailed contribution and outline is summarized as follows.

- We present a general solution framework for the problem (0-1 FG) whose recourse can be an arbitrary (possibly NP-hard) combinatorial optimization problem with linear objective function and binary variables. Thus, we extend theory and methodology of the existing exact algorithms for fortification games, the majority of which requires the recourse problems to be convex.
- We introduce a single-level MIP reformulation for (0-1 FG) using fortification cuts, present alternative methods for strengthening these cuts, and discuss their efficient separation.

- We propose further algorithmic enhancements, that speed-up the separation problem, based on heuristic solutions of the interdiction problem.
- We describe the application of our generic solution algorithm to two concrete problems, namely the knapsack fortification game and the shortest path fortification game, and present an extensive computational study of the performance of our algorithm. This analysis is aimed at evaluating the contribution of different ingredients of our algorithm as well as at comparing its effectiveness with other exact approaches from the literature.

The paper is organized as follows. In the remainder of this section, we provide an overview of previous and related work, whereas in Section 2 we introduce our reformulation and derive the fortification cut for a given intersection strategy. Moreover, we discuss methods for strengthening a given cut and analyze the associated separation problem. Other general speed-up techniques that are used in our solution algorithm are presented in Section 3. Sections 4.1 and 4.2 describe two applications that can be modeled as an FG. A detailed computational study on both problems is given in Section 5, where we discuss further, problem-specific implementation details, assess the efficiency of the various ingredients of the algorithm, and compare the results of our method with state-of-the-art exact approaches from the literature. Finally, Section 6 draws some conclusions and reports possible new lines of research.

1.2. Previous and related work

Closely related to FGs are IGs, which are two-player two-level Stackelberg games used to model attacker-defender settings. The attacker, who acts first, has limited resources. An attack disables some of the defender's assets, reduces their capacity, or increases their cost. At the lower level, the defender solves the recourse problem over the set of surviving or partially damaged assets. The goal of the attacker is to find an optimal interdiction strategy, while anticipating defender's response. IGs arise in military applications (Brown et al., 2006), in controlling the spread of infectious diseases (Assimakopoulos, 1987; Furini, Ljubić, Malaguti, & Paronuzzi, 2020; 2022; Shen, Smith, & Goli, 2012; Tanınmış, Aras, & Altınel, 2021), in counter-terrorism (Wang, Yin, & An, 2016), or in monitoring of communication networks (Furini, Ljubić, Martin, & Segundo, 2019; Furini, Ljubić, Segundo, & Zhao, 2021). Very often, IGs are defined over networks, in which the attacker reduces the capacities of nodes or edges, or even completely removes some of them from the network (Cochran et al., 2011). Some of the most famous examples of so-called network-interdiction games include interdiction of shortest paths (Israeli & Wood, 2002) or network flows (Akgün, Tansel, & Wood, 2011; Lim & Smith, 2007; Smith & Lim, 2008). However, IGs turn out to be much more difficult if the recourse problem is NP-hard, like maximum knapsack (Caprara, Carvalho, Lodi, & Woeginger, 2016; Della Croce & Scatamacchia, 2020; Fischetti et al., 2019) or maximum clique (Furini et al., 2019; Furini et al., 2021), making the associated IG a Σ_2^P-hard problem (Lodi, Ralphs, & Woeginger, 2014). A separate line of research (which is not addressed in this paper) is dedicated to problems called Stackelberg security games. There, the attacker commits to a mixed (randomized) strategy, whereas the defender responds with a pure strategy, see, e.g., the recent survey in Sinha, Fang, An, Kiekintveld, & Tambe (2018) and also Clempner (2022).

In FGs, the defender tries to "interdict" the attacker, by anticipating the attacker's malicious activities, i.e., the defender tries to protect the most vulnerable assets before the attack is taking place. Even though there exists a large body of literature dedicated to IGs, see, e.g., recent surveys in Beck, Ljubić, & Schmidt (2022); Smith &

Song (2020) and Kleinert, Labbé, Ljubić, & Schmidt (2021), there are very few articles dealing with more difficult three-level optimization problems arising in the context of FGs. When it comes to methodology used to solve FGs, there are several ideas proposed in the literature for deriving single-level reformulations for FGs.

- The first group of methods relies on duality theory and is typically applied to FGs in which the recourse problem is a linear/convex program. Hence, after dualization (and potential linearization) of the recourse problem, the three-level model is turned into a bilevel min-max mixed-integer formulation. The latter is then solved using specialized techniques like Benders decomposition or a cutting plane approach, see, e.g., Brown et al. (2006) and Smith, Lim, & Sudargho (2007), respectively. The major advantage of combining the second and the third-level through dualization is that it allows to derive a bilevel program that can nowadays be solved using some of the state-of-the-art solvers for mixed-integer bilevel optimization (see, e.g., Fischetti, Ljubić, Monaci, & Sinnl, 2017; Fischetti, Ljubić, Monaci, & Sinnl, 2018; Tahernejad, Ralphs, & DeNegre, 2020; Yue, Gao, Zeng, & You, 2019). However, this approach often leads to large bilevel programs, which limits its applicability to instances of modest size.

- The second group comprises methods based on enumerating all possible fortification strategies and solving the underlying interdiction problem induced by each strategy using an exact algorithm. Such ideas have been applied in e.g., Cappanera & Scaparra (2011); Church & Scaparra (2007); Scaparra & Church (2008a). This approach is typically useful when the fortification budget is given through a cardinality constraint, where only a small number of items can be fortified. In that case, the size of the search space (in terms of fortification decisions) can be controlled and enumeration of these decisions can be embedded into a branching framework. However, the computational limitations of this method are easily reached for problems where the search space grows too fast, for example when the number of allowed fortified items is too large or when the budget constraint it is replaced by a more general knapsack-like constraint.

- The third method is a sampling-based approach recently proposed by Lozano & Smith (2017). The sampling is applied to the set of possible solutions of the third level, and a single-level reformulation in the space of binary fortification variables (w) is derived. The reformulation contains an exponential number of constraints, which are dynamically separated using a cutting-plane method. Since only a subset of possible recourse solutions is taken into account, the reformulation is not exact and has to be embedded into an iterative framework in which lower and upper bounds are tightened until the convergence criteria are met. In particular, for any given fortification strategy vector w, a subset of sampled third-level solutions is used as a basis to derive valid lower and upper bounds by solving IGs based on the sampled third-level solutions. The overall method is completed by an iterative outer optimization over w-variables, where specific interdiction strategies are eliminated using no-good-cut-like covering inequalities. We note that in this iterative outer optimization the problem in w-variables needs to be solved to integer optimality in each iteration. Thanks to the sampling, this method does not require the recourse problem to be convex, and can be applied to a larger class of problems, involving non-linear, or discrete variables in the third level.

Heuristics are much more prevalent than exact methods in the existing literature on FGs. The most recent generic heuristics for FGs have been proposed by Fakhry, Hassini, Ezzeldin, & El-Dakhakhni (2022). Finally, when it comes to applications, FGs stem from similar settings as for the IGs. Three-level DAD models to protect electric power grids have been used by Brown et al. (2006); Lai, Illindala, & Subramaniam (2019); Xiang & Wang (2018); Yuan, Zhao, & Zeng (2014), and Fakhry et al. (2022). Smith et al. (2007) proposed to design networks that can survive network-flow attacks by using FGs. Similarly, Sarhadi, Tulett, & Verma (2017) used a DAD model for protection planning of freight intermodal transportation networks. Other FG models for protecting transportation networks can be found in Jin, Lu, Sun, & Yin (2015); Starita & Scaparra (2016), and Starita, Scaparra, & O'Hanley (2017). In the context of supply chain networks and protection of facilities, we highlight DAD models presented by Church & Scaparra (2007); Scaparra & Church (2008b), and Zheng & Albert (2018). Recently, a trilevel critical node problem used for limiting the spread of a viral attack in a given network has been considered in Baggio, Carvalho, Lodi, & Tramontani (2021). We point out that our list of applications of FGs is far from being comprehensive, and we refer an interested reader to further references provided in Lozano & Smith (2017); Sarhadi et al. (2017); Starita & Scaparra (2016), and Lai et al. (2019).

In our approach, we reformulate the problem as a single-level MIP in the natural space of fortification variables w. Differently than the approach proposed by Lozano & Smith (2017) our cuts are used in a branch-and-cut scheme and they also do not cut off feasible solutions. Moreover, our approach works on the complete space of the recourse problem and not just by using a sampling of the recourse problem solutions. Thus, contrary to Lozano & Smith (2017) we do not need to iteratively solve MIPs to obtain both upper and lower bounds. Instead, we just need to solve a single MIP, in which we add our fortification cuts as needed. The fortification cuts are obtained by solving IGs which are parameterized in the objective function by the current (linear programming relaxation) solution value of w in a branch-and-bound node. One particular feature of our approach is the fact that IGs at the lower level can be solved using any kind of a black-box algorithm. Specifically, depending on the structure of the interdiction problem, one could employ some of the state-of-the-art methods for certain classes of binary interdiction games (Contardo & Sefair, 2022; Tanınmış & Sinnl, 2022; Wei & Walteros, 2022) as a strategy for separating our fortification cuts.

In this sense, our approach is a generalization of the Benders-like decomposition used in Brown et al. (2006) and Smith et al. (2007) for convex recourse problems. We show that their idea of penalization of the objective function can be efficiently used also for discrete recourse problems (where reformulation by convexity does not work), with tight penalization coefficients and without using big-M penalization terms, as long as fortification decisions are binary. From an algorithmic perspective, our approach is similar to the approach for IGs in Fischetti et al. (2019) as we implement a branch-and-cut procedure in which the cuts are separated using an exact solver of the lower-level problem. However, our approach does not impose any conditions on the structure of the underlying interdiction problem, such as the downward monotonicity used in Fischetti et al. (2019). Instead, the only requirements needed for the validity of our approach are that (i) the fortification decisions are binary, (ii) the fortified items cannot be interdicted, and (iii) the interdiction strategy does not affect the set of possible recourse actions.

2. Solution framework

In this section, we first introduce a single-level reformulation of (0-1 FG) in the space of fortification variables w which uses the family of *fortification cuts*. Afterwards, we present different methods for strengthening these cuts and discuss methods for their separation.

2.1. Single-level reformulation and fortification cuts

Let $\Phi_I(x)$ be the value function of the interdiction game associated to a given interdiction strategy $x \in X$, i.e.,

$$\Phi_I(x) = \min_{y \in Y} \left\{ c^T y + \sum_{i \in N} d_i x_i y_i \right\}.$$

Similarly, let $\Phi_F(w)$ be the value function of the fortification game for a given fortification strategy $w \in W$, i.e.,

$$\Phi_F(w) = \max_{x \in X(w)} \min_{y \in Y} \left\{ c^T y + \sum_{i \in N} d_i x_i y_i \right\} = \max_{x \in X(w)} \Phi_I(x)$$

$$= \max_{x \in X} \left\{ \Phi_I(x) - \sum_{i \in N} M_i x_i w_i \right\}. \tag{1}$$

The last equations holds for sufficiently large coefficients M_i, due to the fact that the linking constraints between the fortification and subsequent interdiction are given by $x \le 1 - w$. Using this notation, problem (0-1 FG) can equivalently be written as

$$z^* = \min_{w \in W} \theta \tag{2}$$

$$\theta \ge \Phi_F(w). \tag{3}$$

Eq. (1) implies that, for any $w \in W$ and $\hat{x} \in X$, we have

$$\Phi_F(w) \ge \Phi_I(\hat{x}) - \sum_{i \in N} M_i \hat{x}_i w_i, \tag{4}$$

which leads to the single level reformulation of the fortification game

$$z^* = \min_{w \in W} \theta \tag{5}$$

$$\theta \ge \Phi_I(\hat{x}) - \sum_{i \in N} M_i \hat{x}_i w_i \qquad \forall \hat{x} \in X. \tag{6}$$

Given the possible exponential number of inequalities (6), this reformulation is suitable for being solved using a cutting-plane or a branch-and-cut approach in which they are initially omitted and then added as needed during the solution process. The resulting formulation in which some of the constraints (6) are relaxed, is referred to as *relaxed master problem*. There are two potential drawbacks in using formulation (5) and (6) in practical applications. First, if not carefully chosen, the values of M_i can lead to very weak dual bounds and potentially underperforming branch-and-bound trees. Second, to separate inequalities (6) one has to solve the middle-level IG, a problem which can be Σ_2^P-hard, for NP-hard recourse problems. To address the first issue, we show in Theorem 1 that tight values for the M_i coefficients can be easily derived, no matter the structure of the recourse problem. We then show how these *fortification cuts*, can be strengthened to tighten the dual bounds. We address the separation issue in Section 2.4.

Theorem 1. *Constraints (6) can be replaced with the following fortification cuts:*

$$\theta \ge \Phi_I(\hat{x}) - \sum_{i \in N} d_i \hat{x}_i w_i \qquad \forall \hat{x} \in X. \tag{7}$$

Proof. We need to show that $\Phi_F(w) \ge \Phi_I(\hat{x}) - \sum_{i \in N} d_i \hat{x}_i w_i$ for all $w \in W$ and $\hat{x} \in X$, i.e., that (4) holds for $M_i = d_i$.

Given $w \in W$ and $\hat{x} \in X$, consider an interdiction policy x' defined by selecting those assets in $\hat{x}$ that are not fortified by w, i.e., $x'_i = \hat{x}_i(1 - w_i)$, for all $i \in N$. Since $x' \le \hat{x}$ and $\hat{x} \in X$, we have $x' \in X$. By applying inequality (4) to w and x', and noting that $x'_i w_i = 0$ for all $i \in N$, we derive that $\Phi_F(w) \ge \Phi_I(x')$ holds.

We complete the proof by showing that

$$\Phi_I(x') \ge \Phi_I(\hat{x}) - \sum_{i \in N} d_i \hat{x}_i w_i. \tag{8}$$

By contradiction, assume that inequality (8) is not satisfied, i.e., $\Phi_I(\hat{x}) - \sum_{i \in N} d_i \hat{x}_i w_i > \Phi_I(x')$, and let $y' \in Y$ be an optimal recourse strategy for interdiction strategy x'. Using the definition of value function $\Phi_I(x')$, we obtain

$$\Phi_I(\hat{x}) - \sum_{i \in N} d_i \hat{x}_i w_i > c^T y' + \sum_{i \in N} d_i x'_i y'_i. \tag{9}$$

Using notations $N_1 = \{i \in N : \hat{x}_i w_i = 1\} = \{i \in N : \hat{x}_i = 1, x'_i = 0\}$, and $N_2 = N \setminus N_1$, inequality (9) can be rewritten as

$$\Phi_I(\hat{x}) > c^T y' + \sum_{i \in N_2} d_i x'_i y'_i + \sum_{i \in N_1} d_i, \tag{10}$$

since $x'_i = 0$ for all $i \in N_1$, and $\hat{x}_i w_i = 1$ for all $i \in N_1$. Since $x'_i = \hat{x}_i$ for all $i \in N_2$, and $d_i \ge d_i \hat{x}_i y'_i$ for all $i \in N$, it follows that

$$\Phi_I(\hat{x}) > c^T y' + \sum_{i \in N_2} d_i \hat{x}_i y'_i + \sum_{i \in N_1} d_i \hat{x}_i y'_i = c^T y' + \sum_{i \in N} d_i \hat{x}_i y'_i \tag{11}$$

which shows that y' is a feasible recourse strategy for interdiction strategy $\hat{x}$ whose cost is strictly smaller than $\Phi_I(\hat{x})$. This contradiction concludes the proof. $\square$

2.2. Strengthening the fortification cuts with enumeration

In this section, we introduce sufficient conditions under which a vector of coefficients produces a valid cut strengthening a given fortification cut.

Theorem 2. *Let $\hat{x} \in X$ be a feasible interdiction strategy and denote by $S(\hat{x}) = \{i \in N : \hat{x}_i = 1\}$ the associated set of interdicted assets. Let $\tilde{d} \in \mathbb{R}^n$ be a vector satisfying the following conditions:*

1. *$\tilde{d}_i \le d_i$, for all $i \in N$,*
2. *$\sum_{i \in P} \tilde{d}_i \ge \Phi_I(\hat{x}) - \Phi_I(x')$ for each $P \subseteq S(\hat{x})$ such that there exists a feasible fortification strategy $w \in W : w_i = 1$ for all $i \in P$, and x' is the interdiction strategy with $x'_i = 0$ for all $i \in P$, and $x'_i = \hat{x}_i$ otherwise.*

Then, the inequality

$$\theta \ge \Phi_I(\hat{x}) - \sum_{i \in N} \tilde{d}_i \hat{x}_i w_i \tag{12}$$

is valid for (0-1 FG) and dominates the fortification cut (7) for interdiction strategy $\hat{x}$.

Proof. Let $\hat{x} \in X$ be a feasible interdiction strategy, $\tilde{d} \in \mathbb{R}^n$ be a vector satisfying Conditions 1 and 2 above, and $w \in W$ be any feasible fortification strategy. Using $\hat{x}$ and w, define $P = \{i \in N : \hat{x}_i = 1, w_i = 1\} \subseteq S(\hat{x})$, which clearly satisfies the description in Condition 2 since the condition must hold for each $P \subseteq S(\hat{x})$ associated with some $w \in W$. In addition, consider an interdiction strategy x' such that $x'_i = \hat{x}_i(1 - w_i)$ for all $i \in N$. By definition of P and using Condition 2, we get

$$\Phi_I(\hat{x}) - \sum_{i \in N} \tilde{d}_i \hat{x}_i w_i = \Phi_I(\hat{x}) - \sum_{i \in P} \tilde{d}_i \le \Phi_I(x') \le \theta, \tag{13}$$

where the last inequality follows from the fact that $\theta \ge \Phi_F(w) \ge \Phi_I(x')$ since $x' \in X(w)$, as in the proof of Theorem 1. Thus, Eq. (12) is valid since (13) holds for all $w \in W$.

Finally, Condition 1 ensures that cut (12) dominates its counterpart (7) associated with the same interdiction strategy $\hat{x}$. $\square$

The theorem above provides conditions for determining stronger cuts. While the trivial choice of $\tilde{d} = d$ is always feasible with respect to these conditions, a tighter cut could potentially be

obtained by finding a vector $\tilde{d}$ which is minimal with respect to some norm while still satisfying the conditions, since smaller $\tilde{d}$ would imply a tighter cut. We note that for a given interdiction strategy $\hat{x}$, checking whether a given vector $\tilde{d}$ satisfies to Condition 2 may be computationally challenging. Indeed, one has to evaluate the recourse value of the attacker solution $S(\hat{x}) \setminus P$ for each $P \subseteq S(\hat{x})$ whose elements can be contained in a feasible fortification strategy. In Section 3.3, we describe a heuristic approach to efficiently obtain a non-trivial vector $\tilde{d}$ satisfying the conditions of Theorem 2.

Remark 2. The right-hand-side of Condition 2 in Theorem 2 can be replaced by $\min\{\Phi_I(\hat{x}) - \underline{\Phi}_I(x'), \sum_{i \in P} d_i\}$, where $\underline{\Phi}_I(x')$ is a lower bound on $\Phi_I(x')$. For the problems whose recourse level is computationally hard to solve, a dual bound could be used to produce a faster strengthening procedure. Although the resulting cut may be weaker than the one obtained using $\Phi_I(x')$, it is still as good as (7) since $\tilde{d}_i \leq d_i$, for all $i \in N$.

2.3. Strengthening the fortification cuts with a lower bound

In this section we show an alternative and computationally less expensive way to obtain stronger fortification cuts. The theorem below assumes that a valid lower bound on z^* is available. For $\alpha \in \mathbb{R}$, we define $(\alpha)^+ = \max\{\alpha, 0\}$.

Theorem 3. *Let $\hat{x} \in X$ be a feasible interdiction strategy and $\underline{z}$ be a lower bound on z^*. Then, the inequality*

$$\theta \geq \Phi_I(\hat{x}) - \sum_{i \in N} \min\{(\Phi_I(\hat{x}) - \underline{z})^+, d_i\}\hat{x}_i w_i \quad (14)$$

is valid for (0-1 FG) and dominates the fortification cut (7) for interdiction strategy $\hat{x}$.

Proof. Since $\underline{z}$ is a lower bound on $z^* = \min_{w \in W} \Phi_F(w)$, we have $\Phi_F(w) \geq \underline{z}$, for all $w \in W$. Let $L = \{i \in N : \Phi_I(\hat{x}) - \underline{z} < d_i\}$ be the set of indices for which the coefficient in fortification cut (14) is smaller than the one in (7). Then we have

$$\Phi_I(\hat{x}) - \sum_{i \in N} \min\{(\Phi_I(\hat{x}) - \underline{z})^+, d_i\}\hat{x}_i w_i = \Phi_I(\hat{x})$$

$$- \sum_{i \in L}(\Phi_I(\hat{x}) - \underline{z})^+ \hat{x}_i w_i - \sum_{i \in N \setminus L} d_i \hat{x}_i w_i.$$

If $\hat{x}_i w_i = 0$ for all $i \in L$, then (14) is equivalent to (7). Otherwise, we obtain

$$\Phi_I(\hat{x}) - \sum_{i \in L}(\Phi_I(\hat{x}) - \underline{z})^+ \hat{x}_i w_i - \sum_{i \in N \setminus L} d_i \hat{x}_i w_i \leq \Phi_I(\hat{x}) - (\Phi_I(\hat{x}) - \underline{z})^+ \leq \underline{z},$$

$$(15)$$

which leads to $\Phi_F(w) \geq \underline{z} \geq \Phi_I(\hat{x}) - \sum_{i \in N} \min\{(\Phi_I(\hat{x}) - \underline{z})^+, d_i\}\hat{x}_i w_i$. □

Remark 3. Notice that $z^* \geq \min_{y \in Y} c^T y$, i.e., the objective value of the fortification problem cannot be smaller than the recourse objective value in case of no interdiction. Thus, a valid lower bound on z^* to be used for deriving strengthened cuts (14) can be computed as $\underline{z} = \min_{y \in Y} c^T y$.

The following corollary shows that the results provided in Theorems 2 and 3 above can be combined to obtain a valid cut. It follows from the validity of (12) and inequality (15).

Corollary 1. *Let $\hat{x} \in X$ be a feasible interdiction strategy, $\tilde{d} \in \mathbb{R}^n$ be a vector satisfying the conditions in Theorem 2, and $\underline{z}$ be a lower bound on z^*. Then, the inequality*

$$\theta \geq \Phi_I(\hat{x}) - \sum_{i \in N} \min\{(\Phi_I(\hat{x}) - \underline{z})^+, \tilde{d}_i\}\hat{x}_i w_i \quad (16)$$

is valid for (0-1 FG).

Remark 4. When $\Phi_I(\hat{x}) - \underline{z}$ is small compared to the original cut coefficient d, it is likely that most of the final coefficients after combined strengthening will be equal to $\Phi_I(\hat{x}) - \underline{z}$, which renders the effort to initially compute $\tilde{d}_i$ values unnecessary. In this case, one could opt for the lower bound based strengthening which can be obtained in constant time if $\underline{z}$ is available.

2.4. Separation of fortification cuts

As already mentioned, our approach for solving the problem (0-1 FG) is a branch-and-cut algorithm based on reformulation (5) with (strengthened) fortification cuts (7). At each node of the branch-and-cut tree, valid inequalities are added when violated, to ensure correctness of the algorithm, improve the dual bound at the node, and possibly allow fathoming. Although in principle one could add to the formulation *any* valid violated inequality, a cut that is maximally violated is typically sought. Thus, given a solution, say (w^*, θ^*), for the linear programming (LP)-relaxation of the current model, one is required to solve the separation problem, which is an IG,

$$\Phi_F(w^*) = \max_{x \in X(w^*)} \min_{y \in Y} \left\{ c^T y + \sum_{i \in N} d_i x_i y_i \right\} \quad (17)$$

and check whether $\Phi_F(w^*) > \theta^*$ or not. Notice that this separation problem itself is a mixed-integer bilevel linear program, which is usually NP-hard and possibly even Σ_2^P-hard (Caprara, Carvalho, Lodi, & Woeginger, 2014) in case of an integer recourse problem. Separation problem (17) can be reformulated as follows using Benders decomposition as done by Israeli (1999).

$$(\text{SEP}) \quad \Phi_F(w^*) = \max_{x,t} \quad t \quad (18)$$

$$t \leq c^T \hat{y} + \sum_{i \in N} d_i \hat{y}_i x_i \qquad \forall \hat{y} \in \hat{Y} \quad (19)$$

$$\sum_{i \in N} g_i x_i \leq B_I \quad (20)$$

$$x_i \leq 1 - w_i^* \qquad \forall i \in N \quad (21)$$

$$x \in \{0, 1\}^n \quad (22)$$

Here $\hat{Y}$ denotes the set of extreme points of the convex hull of the feasible solutions of the recourse problem, see, e.g., Cochran et al. (2011); Fischetti et al. (2019). In our generic framework, we solve (SEP) using a branch-and-cut scheme in which we add violated *interdiction cuts* (19) when needed, for both integer and fractional values of x variables. Solving the separation problem yields a violated fortification cut if $\Phi_F(w^*) > \theta^*$, while one can conclude that (w^*, θ^*) is a feasible solution to (2), (3) otherwise.

Strengthening the interdiction cut Given the computational complexity of the separation problem (SEP), one may be interested in determining a violated, but not necessarily maximally violated fortification cut. To this aim, it is enough to find a feasible solution to (SEP) having an objective value strictly larger than θ^*. Hence, we propose to reformulate the separation problem by imposing an *artificial* upper bound, a real number $\ell > \theta^*$, on its optimal objective value. To this end, the interdiction cuts (19) can be strengthened as shown in the following theorem.

Theorem 4. *Given a solution (w^*, θ^*) and a real number $\ell > \theta^*$, consider the following formulation:*

$$(\text{SEP-L}) \quad \Phi_F^\ell(w^*) = \max_{x,t} \quad t$$

$$t \leq c^T \hat{y} + \sum_{i \in N} \min\{(\ell - c^T \hat{y})^+, d_i\}\hat{y}_i x_i \quad \forall \hat{y} \in \hat{Y}$$

$$(23)$$

$$(20)\text{-}(22)$$

Let (x^*, t^*) be an optimal solution to (SEP-L). If $\Phi_F^\ell(w^*) > \theta^*$, then $\Phi_F(w^*) > \theta^*$ and x^* gives a fortification cut (7) which is violated at (w^*, θ^*). Otherwise, (w^*, θ^*) is a feasible solution to (0-1 FG).

Proof. Observe that the problem (SEP-L) is always feasible and bounded, hence there exists an optimal solution (x^*, t^*) with the associated objective value $\Phi_F^\ell(w^*)$, and $t^* = \Phi_F^\ell(w^*) \le \Phi_F(w^*)$. The inequality $\Phi_F^\ell(w^*) \le \Phi_F(w^*)$ follows from the fact that the only difference between (SEP) and (SEP-L) is the replacement of constraints (19) from (SEP) with constraints (23) in (SEP-L), and in (23) the individual coefficients of the x-variables are either the same as in (19) or smaller. Therefore, if $\Phi_F^\ell(w^*) > \theta^*$, then also $\Phi_F(w^*) > \theta^*$ and thus x^* gives a violated cut. Otherwise, i.e., if $\Phi_F^\ell(w^*) \le \theta^*$, then there exists some $\hat{y}$ for each $x \in X(w^*)$ that makes the RHS of (23) less than or equal to θ^*. Moreover, (23) and (19) are identical at such $\hat{y}$ and x pairs, which means $\Phi_F(w^*) = \Phi_F^\ell(w^*) \le \theta^*$. In this case there exists no violated cut at (w^*, θ^*). $\square$

In other words, this strengthened interdiction cut may project the true objective value of the recourse problem to a smaller value, but if the optimal objective value $\Phi_F(w^*)$ of (SEP) is strictly greater than θ^*, then the optimal objective value $\Phi_F^\ell(w^*)$ of (SEP-L) is also greater than θ^*. As a result, if (SEP-L) yields a solution $(\hat{x}, \hat{t})$ with $\hat{t} > \theta^*$, then $\hat{t}$ could be less than $\Phi_I(\hat{x})$ which is the constant part of the fortification cut (7). Thus, the recourse problem should be solved for $\hat{x}$ one more time to obtain $\Phi_I(\hat{x})$. Note that Lozano & Smith (2017) also propose a strengthening procedure for interdiction cuts, however using a global upper bound on z^*. Although this approach is valid in our case too, our strengthening is more aggressive since ℓ is not necessarily an upper bound, but is treated as one.

In Section 3.2, we propose various additional strategies to speed up the separation procedure.

3. Algorithmic details for an efficient implementation

In this section we address the most important algorithmic elements necessary for an efficient implementation of our solution framework.

3.1. Initialization

We set an initial lower bound on θ by solving the recourse problem with no interdiction as mentioned in Remark 3, i.e., we add the constraint $\theta \ge \min_{y \in Y} c^T y$ to our model.

We also initialize our model with a set of initial fortification cuts. These cuts are produced using the interdiction strategies computed by Algorithm 1. This algorithm takes an optimal recourse solution y^0 in the absence of interdiction as input, and iterates over the assets used in this solution. For each such asset i, an interdiction strategy is obtained via first interdicting i, then deciding the remaining interdiction actions in a greedy fashion. In the greedy part, we pick one element with maximum depreciation-cost ratio among all those that can still be added to the current interdiction strategy without exceeding the interdiction budget. Notice that the number of initial cuts, i.e., the interdiction strategies that Algorithm 1 generates, depends on the support size of the initial recourse solution y^0. Notice that the main loop of Algorithm 1 is repeated at most $|N|$ times. Thus, assuming that each recourse problem can be solved in time at most T_R and using notation $I_{max} = \max_{S \subseteq N, \sum_{i \in S} g_i \le B_I} |S|$ for the maximum number of items that can be interdicted, its rumtime is $O(|N| I_{max} T_R)$.

Algorithm 1 InitialFortificationCuts.

Input: An optimal recourse solution y^0 in case of no interdiction
Output: A set of initial fortification cuts (7)

1: **for** each $i \in N : y_i^0 = 1$ **do**
2:     Initialize theinterdiction strategy $\hat{x} \leftarrow e_i$, define $\hat{B}_I \leftarrow B_I - g_i$
3:     Initialize the candidate set for interdiction $R_N \leftarrow N \setminus \{i\}$
4:     **while** $R_N \ne \emptyset$ **do**
5:         Solve the recourse problem for $\hat{x}$ to obtain an optimal solution y^*
6:         Obtain the new candidate set $R_N = \{i \in N : \hat{x}_i = 0, y_i^* = 1, g_i \le \hat{B}_I\}$
7:         **if** $R_N \ne \emptyset$ **then**
8:             Select $i^* = \arg\max_{i \in R_N} \frac{d_i}{g_i}$
9:             Set $\hat{x}_{i^*} \leftarrow 1$, and update the budget $\hat{B}_I \leftarrow \hat{B}_I - g_{i^*}$
10:    Generate a fortification cut (7) for $\hat{x}$

3.2. On Implementing the separation of fortification cuts

We solve the formulation (5) and (6) by means of a branch-and-cut algorithm, in which feasible solutions (w^*, θ^*) of the relaxed master problem such that w^* is integer are separated when needed. Let (w^*, θ^*) be a feasible solution of the relaxed master problem at the current branching node, e.g., the solution of the LP-relaxation at the current node or a heuristic solution, where w^* is integer. Generating a violated fortification cut associated to (w^*, θ^*) (if such exists) corresponds to finding $\hat{x} \in X(w^*)$ such that $\Phi_I(\hat{x}) > \theta^*$.

Greedy integer separation. To search for such $\hat{x}$, we first invoke the greedy heuristic described in Algorithm 2. This procedure iter-

---

Algorithm 2 GreedyInterdiction.

Input: Current solution (w^*, θ^*) of the relaxed master problem
Output: $\hat{x} \in X(w^*)$, possibly giving a violatedcut (7)

1: Initialize theinterdiction strategy $\hat{x} \leftarrow \vec{0}$ andremaining interdiction budget$\hat{B}_I \leftarrow B_I$
2: Initialize the candidate set for interdiction $R_N \leftarrow N$
3: **while** $R_N \ne \emptyset$ **do**
4:     Solve the recourse problem for $\hat{x}$ to obtain an optimal solution y^*
5:     Obtain the new candidate set $R_N = \{i \in N : \hat{x}_i = 0, y_i^* = 1, g_i \le \hat{B}_I\}$
6:     **if** $R_N \ne \emptyset$ **then**
7:         Select $i^* \in \arg\max_{i \in R_N} \frac{d_i(1 - w_i^*)}{g_i}$
8:         Set $\hat{x}_{i^*} \leftarrow 1$, and update the budget $\hat{B}_I \leftarrow \hat{B}_I - g_{i^*}$
9: Return $\hat{x}$

---

atively defines an interdiction strategy (initially $\hat{x} = \vec{0}$) and a candidate set for interdiction (initially including all assets). At each iteration, the recourse problem associated with the currently interdicted assets is solved, the candidate set is updated accordingly, and an asset is selected and added to the interdiction strategy. The procedure ends when the candidate set for interdiction is empty, meaning that no asset that has been selected in the recourse strategy can be added to the interdiction strategy. Thus, the runtime of Algorithm 2 is bounded by the product of the maximum number of items that may be interdicted simultaneously and time time for solving each recourse problem, i.e., it is $O(I_{max} T_R)$ when using the notation introduced above.

If this `GreedyInterdiction` heuristic does not produce a violated cut, the separation problem (SEP) described in Section 2.4 is solved by means of a branch-and-cut algorithm. We now present

some strategies that are used to improve the performance when solving the model (SEP).

Solution limit as a stopping criterion As already observed, any feasible solution to (SEP) with an objective value larger than θ^* yields a violated cut. Accordingly, one can avoid the computation of a maximally violated cut, and halt the execution of the enumerative algorithm for (SEP) as soon as the incumbent objective value becomes larger than θ^*. In general, a solution limit $s_{max} \geq 1$ can be imposed to prematurely terminate the solution of (SEP) once at least s_{max} incumbent solutions with an objective value larger than θ^* are found.

Setting a lower cutoff value Another strategy that could be helpful in solving (SEP) is to set the lower cutoff value of the used MIP solver to $\theta^* + \epsilon$ where ϵ is a small constant. In this way, one can prune all the nodes with an upper bound less than $\theta^* + \epsilon$, i.e., nodes that cannot produce a violated interdiction cut. If there is no $x \in X(w^*)$ with $\Phi_I(x) \geq \theta^* + \epsilon$, then (SEP) becomes infeasible due to the lower cutoff value, which can be considered as an indicator of bilevel feasiblity of (w^*, θ^*) since $X(w) \neq \emptyset$ for any value of w. In that case, no cuts are added. With this lower cutoff strategy, whenever the incumbent fortification strategy is updated, the associated interdiction strategy is not readily available. Therefore, when an optimal/incumbent solution (w^*, θ^*) to (0-1 FG) is found, it is possible to obtain the optimal attacker response by iterating over all the fortification cuts added to the master problem, i.e., previously obtained interdiction strategies $\hat{x}$. Any $\hat{x} \in X(w^*)$ with $\Phi_I(\hat{x}) = \theta^*$ is an optimal response to w^*. If none is found, the attacker problem is solved optimally for w^*.

While solving (SEP), both integer and fractional x^* are separated. The lower cutoff and the solution limit strategies described above are employed to speed up the separation process. Based on our preliminary experiments, we set the solution limit parameter s_{max} to one. Finally, all feasible solutions to the recourse problem obtained after solving (SEP) in previous iterations are stored and added as initial cuts of the form (19) in future solving attempts.

We remark that the same enhancements can be applied to the alternative separation model (SEP-L) presented in Section 2.4.

3.3. On implementing the strengthening of the fortification cut

In order to strengthen a fortification cut based on Theorem 2, one may solve an LP, e.g., $\min_{\tilde{d} \in \mathbb{R}^n} \sum_{i \in S(\hat{x})} \tilde{d}_i$ subject to the conditions of Theorem 2, after computing all the RHS values appearing in Condition 2. Our preliminary experiments showed that solving this LP was rather time consuming. We thus developed a heuristic method, reported in Algorithm 3, which is more time efficient and which produces cuts that are usually very similar to the cuts

---

Algorithm 3 EnumerativeStrengthening.

Input: $\hat{x} \in X$
Output: Valid coefficients $\tilde{d}$ for the strengthened fortification cut (12)

1: Initialize the strengthened cut coefficients as $\tilde{d} \leftarrow \vec{0}$
2: **for all** $\emptyset \neq P \subseteq S(\hat{x})$ s.t. $\sum_{i \in P} f_i \leq B_F$ **do**
3:    Define attacker solution x' such that $x'_i = 0$ for all $i \in P$, and $x'_i = \hat{x}_i$ otherwise
4:    Solve the recourse problem to obtain $\Phi_I(x')$, compute $\Delta_P = \Phi_I(\hat{x}) - \Phi_I(x')$
5:    **while** $\sum_{i \in P} \tilde{d}_i < \Delta_P$ **do**
6:       Randomly pick $i \in P$ such that $\tilde{d}_i < d_i$, set $\tilde{d}_i \leftarrow \tilde{d}_i + \min\{d_i - \tilde{d}_i, \Delta_P - \sum_{i \in P} \tilde{d}_i\}$
7:    **if** $\tilde{d} = d$ **then**
8:       **return** $\tilde{d}$
9: **return** $\tilde{d}$

---

obtained by solving the LP. This procedure receives an interdiction strategy $\hat{x}$ as input and returns strengthened coefficients $\tilde{d}$ that satisfy the condition in Theorem 2.

The heuristic works as follows: initially, all coefficients are set to zero. Then, all sets $P \subseteq S(\hat{x})$ associated with a feasible w (i.e., $\sum_{i \in P} f_i \leq B_F$) are considered and, for each set, the corresponding attacker solution x' is defined. Then, the cost reduction $\Delta_P = \Phi_I(\hat{x}) - \Phi_I(x')$ with respect to the recourse cost for $\hat{x}$ is split among $\tilde{d}$ coefficients, provided these figures satisfy Condition 1 of Theorem 2. Note that unnecessary computations can be avoided by considering only the subsets P such that there is a feasible recourse solution with $y_i = 1$, for all $i \in P$, as the combined effect of fortifying any two assets on the recourse objective can be larger than the sum of their individual effects only if they can simultaneously be used in a recourse solution. In order to do that, one may exploit the structures of the recourse problems, if known. This is explained in more detail in Section 5 within the problem specific implementation aspects of the two applications we describe next. In addition, any iteration could be halted whenever the sum of the current $\tilde{d}$ coefficients is larger than the maximum value Δ_P could take, i.e., $\Phi_I(\hat{x}) - \min_{y \in Y} c^T y$.

For what concerns the strengthening of fortification cuts based on a lower bound as described in Section 2.3, we initialize the global lower bound as $\underline{z} = \min_{y \in Y} c^T y$. This value is updated dynamically as the best known lower bound obtained from the LP relaxation objective value changes in the branching tree. Whenever the fortification cut is strengthened with $\underline{z}$, the cut is *globally valid* and added as a global cut. In addition, let (w^*, θ^*) be an optimal solution of the LP-relaxation at the current branching node; if $\theta^* > \underline{z}$, i.e, the lower bound of the current branch is tighter than the global one, a *locally valid* and tighter cut may be derived by using θ^* as the strengthening bound.

4. Applications

In this section, we present the two applications for which we conduct a numerical study. A main difference between the two test-cases is in the computational complexity of the associated recourse problems; indeed, this turns out to be an NP-hard problem in the first case and a polynomially solvable problem in the second one.

4.1. Knapsack fortification game

The first case-study we consider is the knapsack fortification game (KFG). The classical 0-1 knapsack problem (KP) is one of the most studied discrete optimization problems, because of its practical and theoretical relevance, and because it arises as a subproblem in many more complex problems (see, e.g., Kellerer, Pferschy, & Pisinger, 2004). In this problem, we are given a set N of items, the ith item having profit $d_i \geq 0$ and weight $a_i \geq 0$ and a knapsack having capacity b. Its bilevel counterpart, also known as the knapsack interdiction game, is commonly used as a benchmark for testing solvers for discrete bilevel optimization; see, e.g., Caprara et al. (2014, 2016); Contardo & Sefair (2022); Della Croce & Scatamacchia (2020); DeNegre (2011); Fischetti et al. (2017, 2019); Tang, Richard, & Smith (2016). A relevant application of the knapsack interdiction game is described in DeNegre (2011): Company D (a defender) wishes to determine its marketing strategy for a given set N of potential target regions, where a_i represent the cost needed to employ a marketing campaign in region $i \in N$ and d_i represents the profit achieved by targeting this region. The available marketing budget is b. We also assume that there exists a competitor, company A (an attacker) who can interdict the marketing strategy of company D by targeting the same regions. For company A the

cost for targeting the region $i \in N$ is $g_i \geq 0$, and the overall available budget is $B_I > 0$. If both companies A and D target the same region, the profit of company D drops to zero. The goal of company A is to target a subset of regions from N, under the given budget constraint, in such a way that the maximum profit achieved by company D is minimized.

In the fortification knapsack game studied in this paper, company D is aware of the interdiction strategy of company A and tries to take some preventive measures in order to protect itself from adversarial actions of company A. Company D searches to reinforce its own marketing campaign with some additional more aggressive and more costly marketing activities that will prevent company A from targeting certain regions from the set N. The reinforcement/fortification cost for region $i \in N$ is given as $f_i \geq 0$, and the overall fortification budget is $B_F > 0$.

Hence, in the three-level fortification version of KP, first the defender chooses a set of items to fortify against interdiction while respecting the fortification budget. Then, the attacker interdicts some of the non-fortified items within the interdiction budget. Lastly, the defender determines the recourse action, by solving a KP over non-interdicted items, i.e., by choosing a subset of non-interdicted items giving the maximum profit while not exceeding the knapsack capacity. The feasible regions of the first and the second levels are denoted by $W = \{w \in \{0, 1\}^n : \sum_{i \in N} f_i w_i \leq B_F\}$ and $X(w) = X \cap \{x \in \mathbb{R}^n : x \leq 1 - w\}$, respectively, where $X = \{x \in \{0, 1\}^n : \sum_{i \in N} g_i x_i \leq B_I\}$. The tri-level game is thus formulated by the following program:

$$z^* = \max_{w \in W} \min_{x \in X(w)} \max_{y} \sum_{i \in N} d_i y_i \tag{24}$$

$$\sum_{i \in N} a_i y_i \leq b \tag{25}$$

$$y_i \leq 1 - x_i \qquad \forall i \in N \tag{26}$$

$$y_i \in \{0, 1\} \qquad \forall i \in N \tag{27}$$

Since the recourse problem satisfies the down-monotonicty assumption (Fischetti et al., 2019), the interdiction constraints (26) can be replaced by penalty terms in the objective function with $M_i = d_i$ as explained in Remark 1. With necessary transformations for a defender with a maximization objective, the single level formulation of the KFG becomes

$$\text{KFG}: \quad z^* = \max_{w \in W} \; \theta \tag{28}$$

$$\theta \leq \Phi_I(\hat{x}) + \sum_{i \in N} d_i \hat{x}_i w_i \qquad \forall \hat{x} \in X. \tag{29}$$

We note that the KFG is Σ_3^p-hard already for the special case $d_i = a_i$, for all $i \in N$ as shown by Nabli, Carvalho, & Hosteins (2022).

4.2. Shortest path fortification game

The second application we consider is the shortest path fortification game (SPFG), originally introduced by Cappanera & Scaparra (2011) and later also considered by Lozano & Smith (2017). This is a three-level extension of the shortest-path interdiction game of Israeli & Wood (2002). The latter is defined on a directed graph $G = (V, A)$ where V and A denote the set of nodes and arcs, respectively. For each arc $(i, j) \in A$, we are given $c_{ij} \geq 0$ which is the nominal cost of traversing this arc, and $d_{ij} \geq 0$ which is the additional traversing cost due to its interdiction. An interdictor tries to maximize the length of the shortest $s - t$ path by interdicting at most $B_I > 0$ arcs of the network. To motivate SPFG, Cappanera & Scaparra (2011) show that fortifying the arcs found through the shortest-path interdiction model (also called *the most critical arcs*)

may be suboptimal, thus highlighting the importance of explicitly including protection decisions into a three-level fortification model. Thus, in the SPFG, the outer-most level consists in selecting a set of arcs to protect from the interdiction by an attacker in order to minimize the length of the shortest $s - t$ path. Then, the attacker interdicts some of the non-fortified arcs, with the goal of maximizing the shortest path in the interdicted graph. At a third level, a recourse step is executed by solving a shortest path problem in the interdicted network. The mathematical formulation of the game is given as

$$z^* = \min_{w \in W} \max_{x \in X(w)} \min_{y} \sum_{(i, j) \in A} (c_{ij} + d_{ij} x_{ij}) y_{ij} \tag{30}$$

$$\sum_{j:(i,j) \in A} y_{ij} - \sum_{j:(j,i) \in A} y_{ji} = \begin{cases} 1 & i = s \\ 0 & i \in V \setminus \{s, t\} \\ -1 & i = t \end{cases} \quad \forall i \in V \tag{31}$$

$$y_{ij} \geq 0 \qquad \forall (i, j) \in A. \tag{32}$$

The feasible regions of the first and second level problems are described by cardinality constraints, i.e., $W = \{w \in \{0, 1\}^m : \sum_{(i,j) \in A} w_{ij} \leq B_F\}$, $X = \{x \in \{0, 1\}^m : \sum_{(i,j) \in A} x_{ij} \leq B_I\}$, and $X(w) = X \cap \{x \in \mathbb{R}^m : x \leq 1 - w\}$, where m denotes the the number of arcs. The fortification game is reformulated by the following single level formulation following the steps described in Section 2.

$$\text{SPFG}: \quad z^* = \min_{w \in W} \; \theta \tag{33}$$

$$\theta \geq \Phi_I(\hat{x}) - \sum_{(i,j) \in A} d_{ij} \hat{x}_{ij} w_{ij} \qquad \forall \hat{x} \in X \tag{34}$$

where X includes all feasible interdiction patterns. Notice that replacing $\Phi_I(\hat{x})$ with $\sum_{(i,j) \in A}(c_{ij} + d_{ij}\hat{x}_{ij})\hat{y}_{ij}$, where $\hat{y}$ is an optimal path under costs $c_{ij} + d_{ij}\hat{x}_{ij}$, gives the following fortification cut:

$$\theta \geq \sum_{(i,j) \in A} \left(c_{ij}\hat{y}_{ij} + d_{ij}(\hat{y}_{ij} - w_{ij})\hat{x}_{ij} \right). \tag{35}$$

Let (i, j) be an arc such that $\hat{x}_{ij} = 1$ and $\hat{y}_{ij} = 0$. Then, its contribution to the right-hand side cost reduces to $-d_{ij}w_{ij}$. This means that if (i, j) would be fortified (and thus not interdicted), the shortest path length would be reduced by at most d_{ij}.

5. Computational results

In this section we discuss the results of our computational study which has two main objectives. First, to assess the computational performance of the proposed algorithms by comparison with the existing literature. Second, to analyze the computational effectiveness of the proposed algorithmic enhancements and determine the limitations of the new approach. We present computational results on two data sets from the literature for each of the two case-studies we consider. Our algorithm has been implemented in C++, and makes use of IBM ILOG CPLEX 12.10 (in its default settings) as mixed-integer linear programming solver. All experiments are run on a single thread of an Intel Xeon E5-2670v2 machine with 2.5 gigahertz processor and using a memory limit of 8 gigabyte. Unless otherwise indicated, a time limit of one hour was imposed for each run. Throughout this section we use the following notation to describe the components of our algorithm: bound based strengthening (B), enumerative strengthening (E), greedy integer separation (G), strengthening of interdiction cuts (I), and any combination of these letters to denote the settings involving the corresponding methods. The detailed results of our experiments are provided in the online supplement.

5.1. Results for the KFG

In the following, we first describe problem specific implementation details of the B&C for the KFG. Then, we discuss the results of our experiments on two data sets derived from publicly available instances for the Knapsack Interdiction Problem.

5.1.1. Implementation details for the KFG

While solving (SEP) for the KFG, separation of integer solutions accounts to solving a KP. Though specific efficient codes are available for the exact solution of KP instances (see, e.g., procedure combo in Martello, Pisinger, & Toth, 1999), we solve this problem as a general MIP. We perform the separation of fractional solutions using the classical greedy algorithm for the knapsack problem; notice that, an exact separation of fractional solutions is not needed for the correctness of the algorithm. Similarly, in Step 4 of Algorithm 2, the recourse is solved greedily. Preliminary experiments showed that the strengthening of interdiction cuts, though producing violated cuts, leads to attacker solutions of poor quality, which reduces the strength of the fortification cuts. Besides, we did not observe any significant reduction in the solution time of (SEP) for a given defender (fortification) solution, thus, for the KFG we focus on the algorithm configurations without interdiction cut strengthening.

As to the enumerative strengthening procedure described in Section 2.2, instead of solving the recourse problem exactly (i.e., solving the KP as an MIP) we only solve its LP relaxation, cf. Step 4 of Algorithm 3. In this way we obtain, with limited computational effort, a valid dual bound that can be used for strengthening according to the observation in Remark 2. As described in Section 3.3, a set P satisfying the conditions for strengthening can be removed from consideration if it includes items that cannot be used together in some feasible recourse solution. To eliminate such P, we simply check the knapsack constraint for the recourse solution y having $y_i = 1$, $\forall i \in P$ and $y_i = 0$, $\forall i \notin P$. If it is violated, we move to the next iteration. During the solution of any instance, if 10 successive trials of enumerative strengthening do not produce a strictly tighter cut, we do not employ this feature for the newly generated cuts.

5.1.2. TRS knapsack interdiction dataset

In this section, we consider the 150 instances introduced by Tang et al. (2016) for the Knapsack Interdiction Problem, and add a fortification level involving a cardinality constraint on the number of items that can be fortified. In the original instances, the number of items is $n \in \{20, 22, 25, 28, 30\}$ and maximum number of items to interdict is $B_I \in \{\lceil n/4 \rceil, \lceil n/2 \rceil, \lceil 3n/4 \rceil\}$. The profit d_i and weight a_i of each item are sampled uniformly in $\{1, \ldots, 100\}$ and the knapsack capacity is $b = \lceil \frac{n-B_I}{2n} \sum_{i=1}^{n} a_i \rceil$. 10 random instances are available for each parameter combination. We use the fortification budget levels $B_F = 3$ and $B_F = 5$, thus producing a set of 300 benchmark instances, denoted as TRS. We solved each instance with several algorithm settings, including components B, E, and G in an incremental way. In Fig. 1, we show the cummulative distribution of root gaps and solution times for the TRS instances. It can be seen that bound based strengthening slightly reduces the solution times although it does not produce consistent improvements in terms of root gaps. The next component, enumerative strengthening, improves both measures, while its effect on the root gaps is more evident. It turns out that enhancing the integer separation by the greedy algorithm improves the performance as well, in particular for what concerns the maximum time needed for solving the instances, which is significantly smaller with the setting BEG than with the other settings considered. All 300 instances are solved optimally within 100 seconds under all four settings. With BEG, the maximum solution time is 18 seconds (cf. Fig. 1).

5.1.3. CCLW knapsack interdiction dataset

Our second benchmark for the KFG is derived from the instances proposed by Caprara et al. (2016) for the Knapsack Interdiction Problem. In this data set, the number of items takes value in $\{35, 40, 45, 50, 55\}$. The profits d_i, weights a_i, and interdiction costs g_i take random values in $[0, 100]$. For each size there are ten instances in which the interdiction budget $B_I = \lceil (ID/11) \sum_{i=1}^{n} a_i \rceil$ increases with the ID of the instance, $1 \leq ID \leq 10$. The knapsack capacity b takes a random value in $\lceil B_I - 10, B_I + 10 \rceil$. In this case too, we introduce an additional fortification level, and impose a cardinality constraint for the defender. We solved these 50 instances for $B_F \in \{3, 5\}$, thus producing a set of 100 benchmark instances denoted as CCLW.

Fig. 2 shows the cumulative distribution of root gaps and solution times for the instances in this benchmark set. It can be seen that bound based strengthening does not have an effect in both measures. This is mostly due to the fact that we are able to produce only a few strengthened cuts, as the value of $\Phi(\hat{x}) - \underline{z}$ is usually larger than d_i in our experiments. Enumerative strengthening does not contribute to decreasing the solution times because of its time complexity, which depends on the interdiction budget; however this strategy decreases the root gap on average. We observed in our experiments that the root gap reduction is larger for smaller budget levels, which can be found in the detailed results provided in the online supplement. We also observe smaller number of B&C nodes when enumerative strengthening is activated, although we do not report details about this kind of measure here. Similar to the results on the TRS data set, the component causing the largest marginal improvement in solution times is greedy integer separation (G). We observe in our experiments that Algorithm 2 is able to find good interdiction strategies quickly, which makes the separation process more efficient compared to the default setting where we stop solving (SEP) once a violated fortification is found. We are able to solve 95 out of 100 instances optimally in one hour when using setting BEG.

5.2. Results for the SPFG

In the following, we first describe problem specific implementation details of the B&C algorithm for the SPFG. Then, we present the results of our experiments on two existing data sets from the literature.

5.2.1. Implementation details for the SPFG

The recourse problem, which is the shortest path problem, is solved via Dijkstra's algorithm (Dijkstra et al., 1959) with a priority queue implementation, wherever needed, e.g., in Algorithms 2 and 3. While solving (SEP) to obtain a fortification cut, both integer and fractional solutions are separated exactly. Separation is carried out in $O(|A| \log |V|)$ time. In addition to the lower cutoff and solution limit strategies, interdiction cut strengthening is used to speed up the solution of (SEP). Recall that any value $l > \theta^*$ can be used to lift the interdiction cut. Since all the problem parameters are integers in out data sets, we choose $l = \lceil \theta^* + \epsilon \rceil$. Remind that fortifying two assets (i.e., arcs) at the same time can be more effective than the sum of their individual effects only if the two arcs can be used together in a recourse solution (see Section 3.3). Accordingly, while executing Algorithm 3, we try to not consider all sets P including pairs of arcs that cannot appear together in an $s - t$ path. Since checking this exactly would be time consuming, we only check if any two arcs in P have the same head or tail and skip the current iteration if there is such an arc pair, which indicates that an $s - t$ path cannot include both of these arcs.
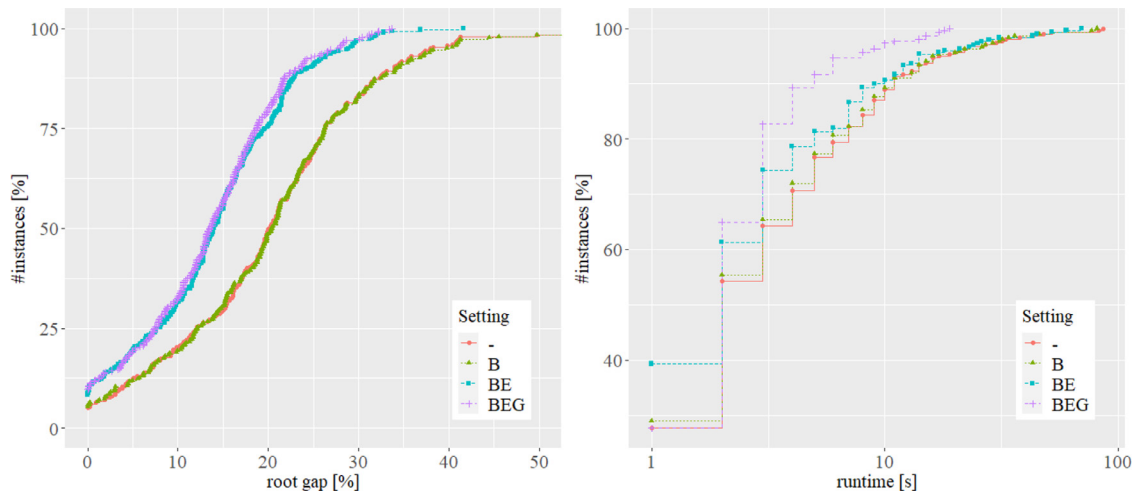
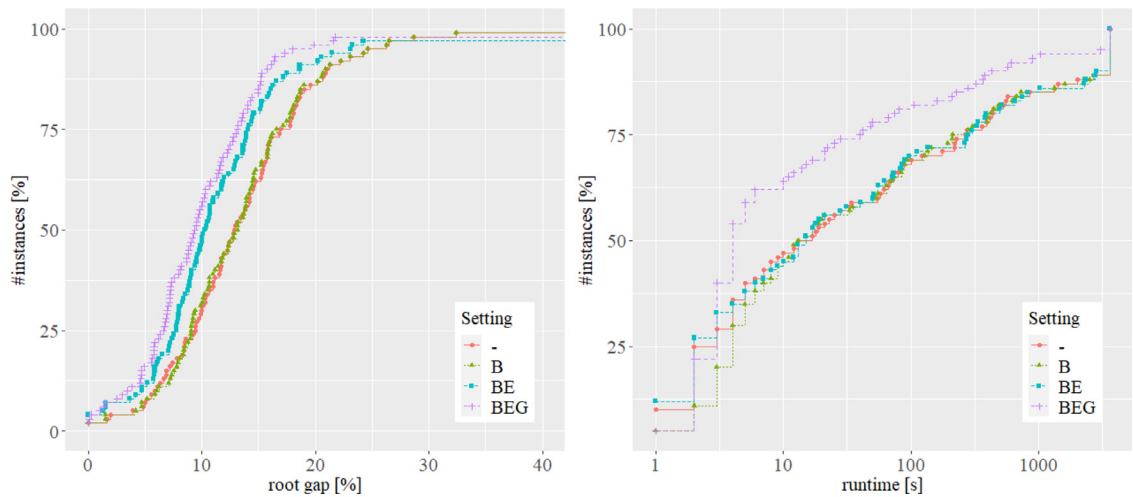**Fig. 1.** Root gaps and solution times of TRS data set instances.

Fig. 2. Root gaps and solution times of CCLW data set instances.

Table 1
The sizes of the grid networks.

Instance	Nodes	Arcs
10×10	102	416
20×20	402	1826
30×30	902	4236
40×40	1602	7646
50×50	2502	12,056
60×60	3602	17,466

5.2.2. Directed grid networks

For the SPFG, the first data set we use is the directed grid network instance set generated by Lozano & Smith (2017) following the topology used by Israeli & Wood (2002) and Cappanera & Scaparra (2011). These instances and the implementation of the solution algorithm proposed in Lozano & Smith (2017) are available at https://doi.org/10.1287/ijoc.2016.0721. In these networks, in addition to a source and a sink node, there are $m \times n$ nodes forming a grid with m rows and n columns. The source (sink) node is linked with each of the grid nodes in the first (last) column. The sizes of the networks are shown in Table 1. We use the cost (c_{ij}) and delay (d_{ij}) values provided with the networks, which were generated randomly in $[1, c_{max}]$ and $[1, d_{max}]$, respectively, using six different c_{max} and d_{max} combinations. For each network size, ten instances with different arc costs/delays are generated. Each instance

is solved for six different fortification and interdiction budget level configurations (B_F, B_I) also used in Lozano & Smith (2017).

In Fig. 3 the cumulative distributions of root gaps and solution times are plotted for three settings: the basic one "−", the best performing setting for KFG instances (BEG), and a variant additionally considering interdiction cut strengthening (IBEG). The plot on the left hand side shows that the root gaps of variant BEG are substantially smaller than those of the basic setting. It is also clear from the plots that the strengthening of interdiction cuts using $l = \lceil \theta^* + \epsilon \rceil$ produces a remarkable improvement. We observed that, using this strategy the number of B&C nodes increased with respect to BEG, since attacker solutions with smaller objective values (and, in turn, weaker fortification cuts) are produced. Nevertheless, in this case, the separation problem can be solved extremely efficiently, which overall leads to improved performances.

Tables 2 and 3 report the average and maximum solution times required by our algorithm with setting IBEG. These figures are compared with the results obtained by the solution algorithm of Lozano & Smith (2017), denoted as LS. Since the implementation of this algorithm is available online, we were able to run it on the same hardware as our algorithm. The algorithm was coded in Java and we ran it with Gurobi 9.2 as MIP solver. In both tables, rows are associated to a network size and a configuration for the fortification and interdiction budget levels, whereas the three blocks of columns refer to different cost-delay configurations ($c_{max} - d_{max}$),
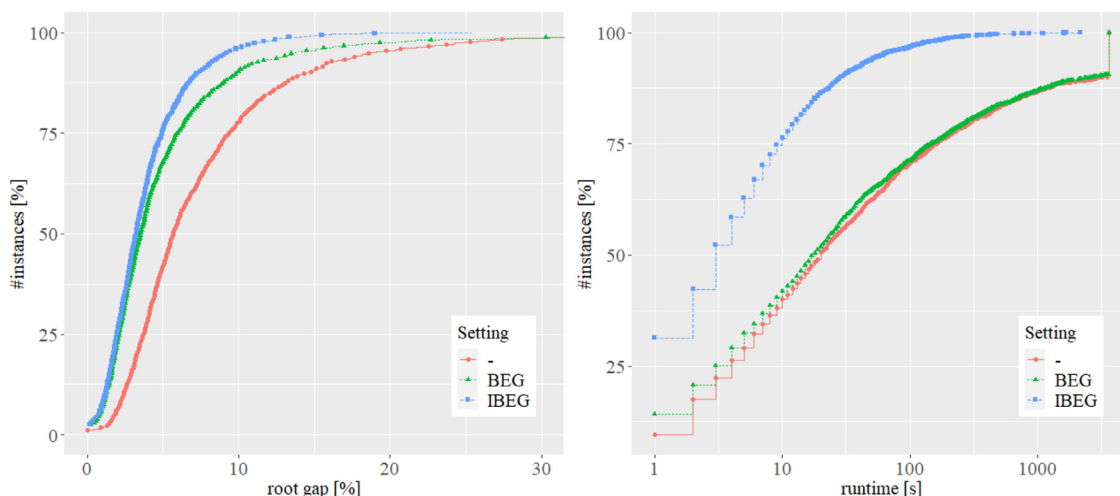
Fig. 3. Root gaps and solution times of Grid networks data set.

Table 2
SPFG Solution times for the directed grid networks with setting IBEG and with algorithm LS (Part 1).

			(10-5)				(10-10)				(10-20)			
			IBEG		LS		IBEG		LS		IBEG		LS	
	B_F	B_I	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10×10	3	3	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.2
	4	3	0.1	0.1	0.1	0.3	0.1	0.2	0.2	0.2	0.1	0.1	0.2	0.2
	3	4	0.1	0.2	0.2	0.3	0.2	0.2	0.3	0.4	0.2	0.2	0.3	0.6
	5	4	0.2	0.2	0.3	0.5	0.2	0.3	0.4	0.5	0.2	0.4	0.4	0.6
	4	5	0.2	0.3	0.4	0.7	0.2	0.3	0.6	1.0	0.3	0.4	0.7	1.5
	7	5	0.3	0.6	1.1	1.7	0.4	0.6	1.1	1.5	0.6	1.0	1.3	1.9
20×20	3	3	0.3	0.5	0.4	1.0	0.3	0.6	0.6	2.1	0.3	0.8	0.6	2.6
	4	3	0.3	0.5	0.5	1.0	0.4	0.8	0.7	2.5	0.4	0.9	0.7	2.7
	3	4	0.4	1.0	0.6	1.3	0.4	0.8	1.0	2.5	0.5	0.8	1.2	5.2
	5	4	0.6	1.4	0.9	1.8	0.8	1.8	1.5	5.1	0.7	1.4	1.7	7.3
	4	5	0.8	1.2	2.1	5.6	0.9	1.6	2.7	9.9	0.9	1.4	3.0	11.4
	7	5	1.7	3.3	3.3	5.9	2.4	3.4	5.1	16.7	2.6	3.9	7.1	31.5
30×30	3	3	0.6	1.0	0.9	1.2	0.6	1.2	1.4	3.8	0.6	1.0	3.6	19.0
	4	3	0.6	0.9	1.1	1.8	0.8	1.3	1.8	4.6	0.8	1.2	3.9	19.6
	3	4	1.2	1.8	1.9	2.7	1.4	2.9	5.6	27.7	1.4	3.1	8.0	35.1
	5	4	1.9	3.4	2.7	3.9	2.1	3.1	5.8	16.4	2.0	2.9	9.2	35.9
	4	5	2.7	4.9	6.4	9.9	2.9	5.9	19.2	110.5	3.4	8.7	21.5	108.7
	7	5	5.6	9.7	12.3	24.5	7.2	13.8	29.2	146.2	7.6	12.8	28.0	115.1
40×40	3	3	1.7	3.5	2.7	6.0	1.8	4.3	5.4	26.9	1.8	4.3	7.8	43.3
	4	3	1.9	3.0	3.6	7.3	2.4	4.6	6.8	32.9	2.2	3.8	9.0	45.6
	3	4	2.7	5.1	3.9	6.3	3.3	8.2	37.4	267.9	3.3	9.3	63.6	540.0
	5	4	4.9	8.4	9.1	30.5	5.3	12.3	42.2	260.8	5.9	14.4	69.9	549.6
	4	5	6.0	11.3	13.7	33.4	10.5	38.8	247.3	1943.5	9.5	30.9	377.7	2786.6
	7	5	15.6	32.4	26.5	76.0	22.9	66.8	486.4	3514.0	27.9	124.1	516.7	4106.4
50×50	3	3	2.1	3.4	4.4	11.1	2.1	4.2	10.7	38.0	2.2	4.7	13.3	41.6
	4	3	2.8	6.8	6.6	25.2	2.8	5.0	19.7	82.2	2.7	5.3	14.4	45.4
	3	4	3.3	8.1	13.1	83.7	5.0	11.8	53.4	240.4	4.8	17.0	52.5	371.5
	5	4	5.9	9.6	15.8	87.6	7.6	20.2	112.9	627.1	8.5	28.1	61.1	416.2
	4	5	9.5	24.0	38.6	276.0	17.7	75.9	219.6	1068.9	14.6	67.5	112.6	858.0
	7	5	24.8	45.9	77.9	518.4	33.5	124.3	407.3	2325.3	38.1	149.2	102.3	624.7
60×60	3	3	4.0	8.1	11.4	30.4	4.4	7.4	21.8	65.6	4.1	6.4	32.0	73.7
	4	3	5.4	13.1	12.6	33.9	5.9	13.3	32.7	143.7	6.1	18.2	38.0	95.8
	3	4	7.1	13.9	15.1	28.2	10.5	18.1	69.6	299.4	11.1	23.3	95.0	426.7
	5	4	12.7	27.9	21.2	54.8	16.7	30.6	134.4	905.8	16.6	28.8	115.8	575.2
	4	5	18.8	26.3	40.5	76.6	35.5	107.3	285.4	915.7	30.4	75.1	265.7	1056.2
	7	5	45.7	80.6	108.6	351.8	87.0	185.0	624.8	3440.4	74.2	127.9	474.1	2155.0

i.e., (10-5), (10-10), and (10-20) in Table 2; (100-50), (100-100), and (100-200) in Table 3.

Table 2 contains the results for smallest three cost-delay configurations, and shows that IBEG yields significantly smaller average and maximum solution times for most of the instances. In 104 (107) out of 108 configurations, IBEG produces a strictly smaller average (maximum) solution time. For many instances of larger sizes our algorithm is faster than LS by one order of magnitude. Overall, the performance improvement of IBEG over LS becomes more pronounced as the c_{max} and d_{max} values increase.

Table 3 gives results for the largest cost-delay configurations. For these instances, again IBEG outperforms LS for most of the instances. Here, in 100 (103) out of 108 configurations, IBEG produces a strictly smaller average (maximum) solution time. As the instances get more difficult, IBEG becomes more advantageous. For the most difficult group of instances with $n = 60$ and $(c_{max} - d_{max})$

Table 3
SPFG solution times for the directed grid networks with setting IBEG and with algorithm LS (Part 2).

		(100-50)				(100-100)				(100-200)			
		IBEG		LS		IBEG		LS		IBEG		LS	
B_F	B_I	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10 × 10	3	0.1	0.2	0.1	0.2	0.3	0.4	0.2	0.3	0.1	0.2	0.2	0.2
	3												
	4	3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.2	0.3	0.2	0.3
	3	4	0.3	0.4	0.3	0.5	0.2	0.4	0.4	0.6	0.3	0.4	0.4	0.9
	5	4	0.3	0.4	0.4	0.5	0.3	0.7	0.5	0.8	0.3	0.5	0.5	0.9
	4	5	0.4	0.5	0.8	1.3	0.5	0.7	1.0	2.0	0.6	0.9	1.1	2.4
	7	5	0.5	1.0	1.3	2.4	0.8	1.2	1.7	3.3	0.9	1.7	1.9	3.4
20 × 20	3	3	0.5	0.8	0.7	1.9	0.5	0.7	0.7	1.1	0.5	0.9	0.7	1.5
	4	3	0.6	0.9	0.9	1.7	0.6	1.1	0.8	1.2	0.7	1.2	0.8	1.7
	3	4	1.0	1.5	2.1	5.4	1.0	1.3	1.8	4.4	0.9	1.6	1.8	4.6
	5	4	1.7	3.1	2.6	4.8	1.8	3.0	2.4	6.1	1.7	3.3	2.4	5.7
	4	5	2.4	3.4	5.8	14.5	2.7	4.7	7.7	22.6	2.4	4.5	7.8	25.2
	7	5	6.3	9.6	7.7	12.3	5.9	8.9	9.4	22.9	5.5	8.7	9.5	28.7
30 × 30	3	3	1.2	2.3	2.9	17.5	1.5	3.1	1.9	4.0	1.3	2.2	2.4	6.0
	4	3	1.4	2.9	2.1	9.5	1.7	2.9	2.7	6.8	1.6	2.5	2.7	6.8
	3	4	2.5	5.1	4.6	13.7	2.3	3.3	5.1	14.7	2.4	3.4	4.8	10.8
	5	4	5.1	12.7	7.3	22.3	4.0	7.2	7.6	24.0	3.9	6.1	7.1	22.6
	4	5	6.7	10.0	19.0	97.4	6.0	10.1	22.3	52.6	6.3	10.8	19.6	75.3
	7	5	19.8	47.5	29.9	89.2	16.1	23.4	49.1	218.1	16.1	23.8	42.6	175.1
40 × 40	3	3	2.3	4.3	3.4	6.7	2.8	4.5	6.9	31.0	2.4	3.6	6.1	18.9
	4	3	3.0	3.9	4.1	6.6	3.2	7.0	9.3	53.3	3.2	7.3	7.5	23.7
	3	4	6.6	11.8	11.3	37.0	6.1	13.9	14.9	40.1	4.9	9.0	13.0	21.0
	5	4	11.0	22.7	16.3	50.3	10.0	18.6	27.2	137.6	8.6	14.4	22.4	85.4
	4	5	21.3	49.9	32.5	78.3	22.9	92.3	75.2	277.6	13.7	33.1	44.9	138.1
	7	5	56.8	95.1	53.0	128.1	57.9	144.5	157.2	934.0	50.4	174.6	114.4	705.3
50 × 50	3	3	4.3	10.1	10.0	33.0	5.6	18.4	36.4	234.9	5.2	19.4	40.5	182.0
	4	3	5.6	12.4	10.1	27.5	6.0	17.3	49.5	359.1	6.2	23.3	40.8	173.8
	3	4	12.9	37.0	28.7	98.3	15.0	42.9	108.8	763.7	9.5	25.8	76.2	268.2
	5	4	27.3	89.4	61.1	204.3	26.6	102.6	417.3	3678.8	20.0	52.2	104.7	496.7
	4	5	57.3	206.9	120.4	494.4	69.4	251.0	253.2	1436.7	58.5	227.9	309.5	1545.9
	7	5	281.0	1097.6	238.5	1099.6	135.7	402.5	912.6	7746.5	93.4	334.8	469.2	3025.0
60 × 60	3	3	8.2	25.0	14.0	40.9	9.2	28.2	52.1	338.1	8.3	22.0	62.3	250.7
	4	3	11.4	24.4	17.6	58.2	11.4	34.7	76.2	552.1	12.9	41.9	68.4	297.4
	3	4	21.2	33.8	44.9	92.0	25.7	104.6	260.6	1790.4	23.2	99.1	180.7	866.1
	5	4	45.5	169.6	64.9	194.8	52.7	246.0	254.0	1726.6	33.1	110.1	202.4	1029.8
	4	5	92.5	365.0	148.2	391.5	167.4	854.4	1281.5	7442.6	222.9	1677.1	1120.4	5620.4
	7	5	271.8	761.7	329.8	1325.5	368.6	1608.8	1256.7	6658.2	364.7	2149.1	1493.8	8691.8

Table 4
Results for instance set `road network`.

					IBEG			LS		
Instance	Nodes	Arcs	B_F	B_I	Avg(s.)	Max(s.)	Solved	Avg(s.)	Max(s.)	Solved
DC	9559	39,377	3	3	9.5	15.1	9	35.0	99.2	9
			4	3	14.2	29.4	9	39.9	101.8	9
			3	4	18.1	32.2	9	94.2	453.7	9
			5	4	45.3	68.4	9	104.0	426.5	9
			4	5	51.5	86.3	9	523.6	2959.7	9
			7	5	123.7	230.7	9	608.4	2126.0	9
RI	53,658	192,084	3	3	88.2	234.2	9	137.7	588.3	9
			4	3	164.3	289.3	9	144.2	570.1	9
			3	4	285.7	960.1	9	463.6	3031.8	9
			5	4	389.0	1432.8	9	465.4	2621.2	9
			4	5	1084.4	5766.4	9	1896.3	13758.4	9
			7	5	1565.7	6042.8	9	2092.1	TL	8
NJ	330,386	1,202,458	3	3	858.3	1997.3	9	988.5	5442.5	9
			4	3	1078.8	2906.5	9	2057.4	TL	8
			3	4	1576.3	2849.0	9	2217.3	TL	8
			5	4	2968.8	8052.1	9	2557.4	TL	8
			4	5	5695.8	TL	8	2702.0	TL	8
			7	5	10042.3	TL	7	3885.4	TL	8
Total							159			156

is (100-100) or (100-200), the average solution times with IBEG are 80% and 78% smaller, respectively, than their LS counterparts. Moreover, IBEG solves all instances within the time limit of one hour.

5.2.3. Real road networks

For the second part of the SPFG experiments, we considered the real road network data sets of the cities Washington (DC), Rhode Island (RI), and New Jersey (NJ), used by Raith & Ehrgott (2009) and Lozano & Smith (2017). While the original undirected data sets are available at http://www.diag.uniroma1.it//~challenge9/data/tiger/, we used the data sets provided by Lozano & Smith (2017) where each edge is replaced by two arcs to get directed networks and in which connectivity is enforced by additional high-cost arcs. The costs of the arcs are equal to the distances and the delay is 10,000 for each arc. On each of the three

networks, the SPFG is solved for nine distinct $s - t$ pairs and using the six fortification and interdiction budget combinations used for the grid networks, resulting in 162 instances. As these are the most challenging instances considered in our study, the time limit is set to four hours for each run. Table 4 reports the sizes of the networks as well as average and maximum solution times over the nine instances with the same size and budget combination for IBEG and LS, as reported in Lozano & Smith (2017). The table shows that our default setting IBEG is able to solve 159 over 162 instances optimally within the time limit. Notice that our B&C algorithm solves three instances more than the sampling based algorithm by Lozano & Smith (2017), which was run on the same machine as IBEG and produced significantly smaller solution times than the ones reported in their paper. In addition, when both algorithms solve all the instances, IBEG is significantly faster than LS: the average computing time and the maximum computing times are below 48% and 72% on the average, respectively, of their counterpart for LS.

6. Conclusion

In this study we address fortification games (FGs), i.e., defender-attacker-defender problems which involve fortification, interdiction and recourse decisions, respectively. These problems are interesting from a theoretical viewpoint and have many real-world applications, in, e.g., military operations or the design of robust networks. We introduce *fortification cuts*, which allow for a single-level exact reformulation of this complex trilevel optimization problem, and a solution approach which uses the cuts within a branch-and-cut algorithm and works in the space of the fortification variables only.

While mainly focusing on the version where interdicting an asset depreciates its usefulness for the defender, we also show that our methodology is directly applicable to FGs with complete destruction of interdicted assets. After introducing the basic fortification cuts, we present two strengthening procedures that can be used to produce stronger inequalities, and give a detailed description of the separation procedure. We provide numerical results for two relevant test-cases, namely knapsack fortification games and shortest path fortification games. Due to the different structures and computational complexity of their recourse problems, we observe different effects of our algorithmic components in our computational study, and try to give insight about possible reasons.

Finally, we point out that our solution scheme is generic, and it leaves space for problem specific improvements. For example, for a given FG, one could solve the associated separation problem by means of a state-of-the-art exact method or using some specialized heuristic instead of the greedy method proposed in this paper. Moreover, problem-specific valid inequalities or strengthening procedures of the fortification cuts could also be possible.

Acknowledgements

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.10.038

References

Akgün, I., Tansel, B. Ç., & Wood, R. K. (2011). The multi-terminal maximum-flow network-interdiction problem. *European Journal of Operational Research, 211*(2), 241–251.

Alderson, D. L., Brown, G. G., Carlyle, W. M., & Wood, R. K. (2011). Solving defender-attacker-defender models for infrastructure defense. *Technical report*. Naval Postgraduate School Monterey CA Dept of Operations Research.

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

Baggio, A., Carvalho, M., Lodi, A., & Tramontani, A. (2021). Multilevel approaches for the critical node problem. *Operations Research, 69*(2), 486–508.

Beck, Y., Ljubić, I., & Schmidt, M. (2022). A survey on bilevel optimization under uncertainty. http://www.optimization-online.org/DB_HTML/2022/06/8963.html.

Brown, G., Carlyle, M., Salmerón, J., & Wood, K. (2006). Defending critical infrastructure. *Interfaces, 36*(6), 530–544.

Cappanera, P., & Scaparra, M. P. (2011). Optimal allocation of protective resources in shortest-path networks. *Transportation Science, 45*(1), 64–80.

Caprara, A., Carvalho, M., Lodi, A., & Woeginger, G. J. (2014). A study on the computational complexity of the bilevel knapsack problem. *SIAM Journal on Optimization, 24*(2), 823–838.

Caprara, A., Carvalho, M., Lodi, A., & Woeginger, G. J. (2016). Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing, 28*(2), 319–333.

Church, R. L., & Scaparra, M. P. (2007). Protecting critical assets: The r-interdiction median problem with fortification. *Geographical Analysis, 39*(2), 129–146.

Clempner, J. B. (2022). Reveling misleading information for defenders and attackers in repeated Stackelberg security games. *Engineering Applications of Artificial Intelligence, 110*, 104703.

Cochran, J. J., Cox, L. A., Keskinocak, P., Kharoufeh, J. P., Smith, J. C., & Wood, R. K. (2011). Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science.*

Contardo, C., & Sefair, J. A. (2022). A progressive approximation approach for the exact solution of sparse large-scale binary interdiction games. *INFORMS Journal on Computing, 34*(2), 890–908.

Della Croce, F., & Scatamacchia, R. (2020). An exact approach for the bilevel knapsack problem with interdiction constraints and extensions. *Mathematical Programming, 183*(1), 249–281.

DeNegre, S. (2011). *Interdiction and discrete bilevel linear programming.* Lehigh University Ph.D. thesis.

Dijkstra, E. W., et al. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271.

Fakhry, R., Hassini, E., Ezzeldin, M., & El-Dakhakhni, W. (2022). Tri-level mixed-binary linear programming: Solution approaches and application in defending critical infrastructure. *European Journal of Operational Research, 298*(3), 1114–1131.

Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research, 65*(6), 1615–1637.

Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming, 172*(1-2), 77–103.

Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2019). Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing, 31*(2), 390–410.

Furini, F., Ljubić, I., Malaguti, E., & Paronuzzi, P. (2020). On integer and bilevel formulations for the k-vertex cut problem. *Mathematical Programming Computation, 12*, 133–164.

Furini, F., Ljubić, I., Malaguti, E., & Paronuzzi, P. (2022). Casting light on the hidden bilevel combinatorial structure of the capacitated vertex separator problem. *Operations Research, 70*(4), 2399–2420.

Furini, F., Ljubić, I., Martin, S., & Segundo, P. S. (2019). The maximum clique interdiction problem. *European Journal of Operational Research, 277*(1), 112–127.

Furini, F., Ljubić, I., Segundo, P. S., & Zhao, Y. (2021). A branch-and-cut algorithm for the edge interdiction clique problem. *European Journal of Operational Research, 294*(1), 54–69.

Israeli, E. (1999). *System interdiction and defense..* Operations Research Department, Naval Postgraduate School Monterey CA Ph.D. thesis..

Israeli, E., & Wood, R. K. (2002). Shortest-path network interdiction. *Networks: An International Journal, 40*(2), 97–111.

Jin, J. G., Lu, L., Sun, L., & Yin, J. (2015). Optimal allocation of protective resources in urban rail transit networks against intentional attacks. *Transportation Research Part E: Logistics and Transportation Review, 84*, 73–87.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems.* Berlin, Germany: Springer.

Kleinert, T., Labbé, M., Ljubić, I., & Schmidt, M. (2021). A survey on mixed-integer programming techniques in bilevel optimization. *EURO Journal on Computational Optimization, 9*, 100007.

Lai, K., Illindala, M., & Subramaniam, K. (2019). A tri-level optimization model to mitigate coordinated attacks on electric power systems in a cyber-physical environment. *Applied Energy, 235*, 204–218.

Lim, C., & Smith, J. C. (2007). Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions, 39*(1), 15–26.

Lodi, A., Ralphs, T. K., & Woeginger, G. J. (2014). Bilevel programming and the separation problem. *Mathematical Programming, 146*(1), 437–458.

Lozano, L., & Smith, J. C. (2017). A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing, 29*(1), 123–139.

Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science, 3*, 414–424.

Nabli, A., Carvalho, M., & Hosteins, P. (2022). Complexity of the multilevel critical node problem. *Journal of Computer and System Sciences, 127*, 122–145.

Raith, A., & Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Computers and Operations Research, 36*(4), 1299–1331.

Sarhadi, H., Tulett, D. M., & Verma, M. (2017). An analytical approach to the protection planning of a rail intermodal terminal network. *European Journal of Operational Research, 257*(2), 511–525.

Scaparra, M. P., & Church, R. L. (2008a). A bilevel mixed-integer program for critical infrastructure protection planning. *Computers and Operations Research, 35*(6), 1905–1923.

Scaparra, M. P., & Church, R. L. (2008b). An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research, 189*(1), 76–92.

Shen, S., Smith, J. C., & Goli, R. (2012). Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization, 9*(3), 172–188.

Sinha, A., Fang, F., An, B., Kiekintveld, C., & Tambe, M. (2018). Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18* (pp. 5494–5501). International Joint Conferences on Artificial Intelligence Organization.

Smith, J. C., & Lim, C. (2008). *Algorithms for network interdiction and fortification games*. In A. Chinchuluun, P. M. Pardalos, A. Migdalas, & L. Pitsoulis (Eds.) (pp. 609–644)). New York, NY: Springer New York.

Smith, J. C., Lim, C., & Sudargho, F. (2007). Survivable network design under optimal and heuristic interdiction scenarios. *Journal of Global Optimization, 38*(2), 181–199.

Smith, J. C., & Song, Y. (2020). A survey of network interdiction models and algorithms. *European Journal of Operational Research, 283*(3), 797–811.

Starita, S., & Scaparra, M. P. (2016). Optimizing dynamic investment decisions for railway systems protection. *European Journal of Operational Research, 248*(2), 543–557.

Starita, S., Scaparra, M. P., & O'Hanley, J. R. (2017). A dynamic model for road protection against flooding. *Journal of the Operational Research Society, 68*(1), 74–88.

Tahernejad, S., Ralphs, T. K., & DeNegre, S. T. (2020). A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Mathematical Programming Computation, 12*(4), 529–568.

Tang, Y., Richard, J.-P. P., & Smith, J. C. (2016). A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization, 66*(2), 225–262.

Tanınmış, K., & Sinnl, M. (2022). A branch-and-cut algorithm for submodular interdiction games. *INFORMS Journal on Computing, 0*(0), toappear.

Tanınmış, K., Aras, N., & Altınel, İ. K. (2021). Improved x-space algorithm for min–max bilevel problems with an application to misinformation spread in social networks. *European Journal of Operational Research, 297*(1), 40–52.

Wang, Z., Yin, Y., & An, B. (2016). Computing optimal monitoring strategy for detecting terrorist plots. In D. Schuurmans, & M. P. Wellman (Eds.), *Proceedings of the thirtieth AAAI conference on artificial intelligence, February 12–17, 2016, Phoenix, Arizona, USA* (pp. 637–643). AAAI Press.

Wei, N., & Walteros, J. L. (2022). Integer programming methods for solving binary interdiction games. *European Journal of Operational Research, 302*(2), 456–469.

Xiang, Y., & Wang, L. (2018). An improved defender–attacker–defender model for transmission line defense considering offensive resource uncertainties. *IEEE Transactions on Smart Grid, 10*(3), 2534–2546.

Yuan, W., Zhao, L., & Zeng, B. (2014). Optimal power grid protection through a defender–attacker–defender model. *Reliability Engineering and System Safety, 121*, 83–89.

Yue, D., Gao, J., Zeng, B., & You, F. (2019). A projection-based reformulation and decomposition algorithm for global optimization of a class of mixed integer bilevel linear programs. *Journal of Global Optimization, 73*(1), 27–57.

Zheng, K., & Albert, L. A. (2018). An exact algorithm for solving the bilevel facility interdiction and fortification problem. *Operations Research Letters, 46*(6), 573–578.