Computers & Operations Research 138 (2022) 105558

Contents lists available at ScienceDirect

Computers and Operations Research

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

# An iterative security game for computing robust and adaptive network flows

Supriyo Ghosh [a,*,1], Patrick Jaillet [b]

[a] Institute for Infocomm Research, A*STAR, Singapore
[b] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 02139, United States of America

ARTICLE INFO

*Keywords:*
Network flows
Game theory
Network resilience
Robust optimization
Large-scale optimization

ABSTRACT

We study the robust and adaptive maximum network flow problem in an uncertain environment where the network parameters (e.g., capacities) are known and deterministic, but the network structure (e.g., edges) is vulnerable to adversarial attacks or failures. We propose a robust and sustainable network flow model to effectively and proactively counter plausible attacking behaviors of an adversary operating under a budget constraint. Specifically, we introduce a novel scenario generation approach based on an iterative two-player game between a defender and an adversary. We assume that the adversary always takes a best myopic response (out of some feasible attacks) against the current flow scenario prepared by the defender. On the other hand, we assume that the defender considers all the attacking behaviors revealed by the adversary in previous iterations of the game in order to generate a new conservative flow strategy that is robust (maximin) against all those attacks. This iterative game continues until the objectives of the adversary and the administrator both converge. We show that the robust network flow problem to be solved by the defender is NP-hard and that the complexity of the adversary's decision problem grows exponentially with the network size and the adversary's budget value. We propose two principled heuristic approaches for solving the adversary's problem at the scale of a large urban network. Extensive computational results on multiple synthetic and real-world data sets demonstrate that the solution provided by the defender's problem significantly increases the amount of flow pushed through the network and reduces the expected amount of lost flow over four state-of-the-art benchmark approaches.

## 1. Introduction

Network flow problems have been widely investigated from various points of views by many researchers and remain a central theme in operations research and computer science. Ahuja et al. (1993) provide a comprehensive study of theory, algorithms, and applications of network flow problems. Network flow problems have numerous applications in critical infrastructure network design and operations including urban transportation, water management, oil pipeline, energy distribution, and telecommunication systems. Modern critical infrastructure networks have extensively installed a broad range of automated devices (e.g., sensors are installed at the road intersections for real-time traffic monitoring and intelligent traffic light management) to improve real-time operations. While the proliferation of automated devices in critical infrastructure networks provides many benefits (e.g., real-time monitoring, better sensing, data-centric planning, etc.) to the authorities, they are exposed to new security challenges, e.g., traffic light manipulation, tampering of sensors, destruction of transformers, etc. Several illustrations of cyber or physical attacks such as tampering of traffic monitoring sensors (Zetter, 2012; Reilly et al., 2014; Cerrudo, 2014), manipulation of signal controllers (Ghena et al., 2014; Jacobs, 2014) have been reported recently. By manipulating sensory components of an edge or by physically attacking it, an adversary can either break the edge completely or modify its capacity, to alter the network structure.

There can be two principal ways to counter these adversarial disruptions and to develop resilient critical infrastructure networks: (a) Reactive approach—the network administrator immediately dispatches available resources to recover the compromised edges for fast network restoration; and (b) Proactive approach—the network administrator strategically circulates the flow to maximize the amount of flow that can be pushed to the terminal node under worst-case adversarial attack. The adversarial attacks can also be countered using a combination of both the reactive and the proactive approaches. This paper focuses on proactive and robust operation for resilient control of a critical infrastructure network where the network parameters (e.g., capacities)

* Corresponding author.
*E-mail addresses:* supriyog.2013@phdis.smu.edu.sg (S. Ghosh), jaillet@mit.edu (P. Jaillet).

[1] Most of this work was done while the first author was with Singapore MIT Alliance for Research and Technology (SMART) Centre for Future Mobility (FM), and with IBM Research AI, Singapore Lab.

https://doi.org/10.1016/j.cor.2021.105558
Received 2 December 2019; Received in revised form 24 August 2021; Accepted 9 September 2021
Available online 25 September 2021
0305-0548/

are known, but the network structure (e.g., edges) is vulnerable to adversarial attacks or failures. We next present a few real-world applications to motivate the study of robust and adaptive network flow problems with edge failures.

Let us consider a problem of a crude oil distribution network that links the production units to consumption centers via a number of intermediate pump stations. The traditional manual pipeline management methods do not consider that the pipelines may collapse. For example, if a pipeline segment is bombed or attacked during a war, it severely affects the entire oil distribution system as well as other industries. Therefore, it is crucial to manage a crude oil distribution network in a way that is competent in dealing with such an adversarial situation to reduce the shortage of crude oil at consumption centers, possibly through rerouting flows but not rebuilding the network (Bertsimas et al., 2013). The challenge is to design a network flow strategy which utilizes the edge capacities efficiently under normal condition, but also preserves residual capacity to maximize the flow through the network by rerouting flows under worst-case adversarial attacks. Another real-world motivating example is a supply-chain network that links source to sinks via several hubs that are connected by railways or roads. One can imagine a similar adverse scenario where a few segments may collapse due to natural disaster or attacks. Other motivating domains that anticipate robustness in a flow solution include modern cyber–physical systems where the automated devices installed in nodes and edges are vulnerable to failures.

In this paper, we model the network flow problem in the framework of robust and adaptive optimization by considering an ominous form of uncertainty, the possibility for failure of edges in the network topology. Robust optimization (Ben-Tal and Nemirovski, 1998; Bertsimas and Sim, 2003) is a broad research area that deals with problems where input data can vary within an uncertainty set. A robust solution in general either aims to optimize a particular worst-case objective value or tries to minimize a regret function by considering the uncertainty in the input data. On the other hand, adaptive optimization models (Ben-Tal et al., 2004) are used to address multi-stage decision-making problems under uncertainty. Existing adaptive optimization models (Atamtürk and Zhang, 2007; Poss and Raack, 2013) solve network flow problems by considering different demand uncertainty sets. In contrast to these existing works on demand robustness, we consider network flow problems with edge failures that are motivated by network interdiction problems.

In a classical network interdiction problem (Wood, 1993; Cormican et al., 1998; Sullivan and Smith, 2014), a felonious entity carries illegal goods through the network and an interdictor places resources (e.g., security personnel) to inspect a subset of edges to detect and prevent the illegal activity. Network interdiction problems are typically modeled using a two-stage optimization framework and therefore, many two-player sequential and simultaneous game-theoretical models have been proposed to solve these problems (Washburn and Wood, 1995; Dahan and Amin, 2015; Guo et al., 2016). In contrast, we consider a three-stage optimization problem which falls within the category of defender–attacker–defender model, often called as network fortification problem (Church and Scaparra, 2007; Scaparra and Church, 2008a; Lozano and Smith, 2017), in which the operator fortifies the network before the interdictor executes her action. However, in contrast to the objectives in network interdiction or fortification problems, our goal is to find a robust and adaptive flow strategy whereby an administrator is able to maximize the operational efficiency of the network under worst-case adversarial attacks, possibly through flow rerouting.

Our work is motivated by the three-stage robust and adaptive maximum network flow problem introduced by Bertsimas et al. (2013). For computing an adaptive maximum flow solution, they assume that the flow can be adjusted in the third stage after edge failure occurred, but the adjusted flow is always bounded by the initial flow assigned to an edge. They further propose a linear optimization model to approximately solve the adaptive maximum flow problem. However, we experimentally show that the proposed approximate solution performs poorly when the flow from an attacked edge is allowed to reroute through adjacent edges with residual capacity, as required in many practical flow applications. In addition, to make the problem more realistic, we assume that the administrator faces a cost for routing flows through an edge, which increases the complexity of the optimization problem for computing an adaptive maximum flow solution.

Due to the aforementioned challenges, it is hard to formulate a tractable optimization problem for computing a robust and adaptive maximum flow solution using a two-stage robust model, as done in Bertsimas et al. (2013). Therefore, we treat the problem of computing a robust and adaptive maximum flow strategy as the result of a two-player iterative game between a network administrator and an adversary. We assume that the adversary is operating under a budget constraint (i.e., the number of attacked edges is bounded by a threshold value) and therefore, the adversary has a finite number of possible attacking choices. The assumption of a budget constraint for the adversary is valid in many real-world applications and therefore, several existing works (Bertsimas et al., 2013; Altner et al., 2010; Dahan et al., 2018) assume a budget constraint for the adversary to control the conservatism of robust solutions. As an example of practical consideration about such a constraint, an adversary would need to be present physically within the geographical proximity in order to manipulate vehicle monitoring sensors which indirectly impact the traffic light timings (Cerrudo, 2014), implying a (human) resource budget constraint for the adversary. While the adversary's budget value can be learned efficiently from previous attacking behaviors in case of repeated attacks, we also experimentally show that the initial estimation of the budget value can be done using sensitivity analysis (i.e., by observing the outcomes with varying budget value).

For the administrator, a feasible flow strategy should satisfy the flow conservation constraints at the nodes and the capacity constraints at the edges. The objective of the administrator is to maximize the ultimate flow that can be pushed to the terminal node while minimizing the total routing cost. On the other hand, the objective of the adversary is to identify an attack that minimizes the objective value of the administrator. As the strategy space of the administrator is extremely large and the adversary's strategy space grows exponentially with the network size and the budget value, it is impossible to compute an equilibrium solution with *ex-ante* enumeration of all the pure strategies. In order to tackle the large strategy space of both the players, we propose a novel incremental strategy generation approach which is motivated by the idea of double oracle algorithm (McMahan et al., 2003; Jain et al., 2011). In each iteration of the game, the adversary, who is restricted to a budget constraint, acts as a follower and optimally disrupts the current network flow strategy prepared by the network administrator. In turn, the administrator acts as a leader and generates a new network flow strategy which is robust (maximin) against all the attacking behaviors revealed in previous iterations. This iterative game continues until the objective values of the players converge to the same value and they start repeating their previous actions. At that point, as the adversary repeats her previous attacks, the administrator has already considered these for generating the final flow strategy and therefore, it is a robust (maximin) response to the adversary's best response. In that sense, the iterative game has reached convergence. Note that, unlike the solutions generated by the double oracle algorithm (Jain et al., 2011), that produce a randomized edge interdiction strategy for the defender, our solution converges to a robust pure flow strategy for the administrator.

As the administrator model introduces additional constraints related to current attack on top of the decision model from the previous iteration, the objective value of the administrator reduces monotonically over the iterations. Due to more conservative flow, the adversary's ability to disrupt the flow strategy reduces over iterations and the objective value of the adversary increases. These two objective values are guaranteed to converge to the same value at some point. We show

that the game converges to a maximin optimal flow solution for the administrator and therefore, the objective value of the administrator will at least be the value to which the game has converged for any realization of the feasible attacks.

We show that the robust and adaptive network flow problem to be solved by the administrator is NP-hard. Moreover, we empirically observe that the complexity of the adversary's decision problem grows exponentially with the network size and adversary's budget value. Therefore, we propose two principled heuristic approaches for solving the complex decision problem of the adversary at the scale of a large urban network. The first heuristic is an accelerated greedy approach where we identify one edge at a time in an incremental fashion so as to minimize the objective value of the administrator for a given flow strategy, until the budget constraint of the adversary is exhausted. For the second heuristic, we partition the network into disjoint sub-networks and identify a set of edges to attack within the adversary's budget constraint by solving the corresponding sub-problems. We iteratively solve this process with random partitioning of the network and learn a set of best possible candidate edges to attack and finally, solve the adversary's decision problem to choose the best edges (within budget constraint) from these candidate edges. In each iteration of the game, we execute both heuristics and choose the one with better solution quality as the adversary's decision. By leveraging the computational effectiveness of the proposed heuristics, our solution approach can scale gracefully to large-scale problems while providing a consistent performance gain over four following benchmark approaches: (i) The administrator sends maximum flow through the network without considering any attacks; (ii) The administrator computes a flow solution using a myopic one-step reasoning against the adversary's behavior; (iii) The administrator proactively computes a robust maximum flow solution to improve the worst-case performance (Bertsimas et al., 2013) ; and (iv) The administrator computes an approximate adaptive maximum flow solution using a linear optimization model from Bertsimas et al. (2013).

*Our contributions.* The key contributions of this paper are as follows:

1. We formally define the problem of computing a robust and adaptive maximum flow strategy for critical infrastructure networks by exploiting the fact that the flow of a compromised edge might be rerouted through adjacent edges with residual capacity. To solve the problem, we propose an iterative two-player game between a network administrator and an adversary, which is referred to as Network Flow Game (NFG).
2. We develop novel optimization models to solve the decision problem of both players in each iteration of the game. The administrator's optimization model takes into account all the attacking strategies generated by the adversary in previous iterations, and computes a robust flow strategy that maximizes the amount of flow pushed through the network in the worst-case over all the previous attacks. The adversary's decision problem inspects the flow strategy generated by the administrator in the current iteration and generates an attack (out of the feasible attacks under a given budget constraint) that optimally disrupts the current flow strategy.
3. We propose two novel heuristic approaches for solving the complex decision problem of the adversary at the scale of a large urban network. The first heuristic is an accelerated greedy approach that incrementally identifies the best edges to be attacked. The second heuristic is a network partitioning based approach that iteratively identifies a set of best candidate edges to be attacked in the network and then solves the adversary's decision problem over these candidate edges.
4. We provide extensive computational results on multiple synthetic and real-world benchmark data sets to demonstrate that our proposed solution approach scales gracefully to large-scale problems and significantly increases the amount of flow pushed through the network over four state-of-the-art benchmark approaches.

*Structure of the paper.* In Section 2, we elaborate on the relevant research. In Section 3, we formally describe our problem by allowing the flow of an attacked edge to be rerouted through adjacent edges with residual capacity. In Section 4, we demonstrate our proposed iterative two-player game between a network administrator and an adversary. We present the decision problem and optimization models for the adversary and the administrator in Section 4.1 and Section 4.2, respectively. In Section 4.3, we provide the key iterative steps of our overall two-player game. In Section 5, we describe two heuristic approaches to accelerate the solution process of the adversary's decision problem. In Section 5.1, we present the accelerated greedy heuristic and in Section 5.2, we describe the network partitioning based heuristic. We then provide the experimental setup and performance analysis of our proposed approach in Section 6. In Section 6.1, we present the empirical results to verify the utility of our proposed solution approach on small-scale problem instances. In Section 6.2, we demonstrate the experimental results on large-scale problem instances by leveraging the computational effectiveness of the proposed heuristics. In Section 7, we discuss an extension of our proposed solution methodology by relaxing the assumption of forward flow rerouting, and by considering a setting where the administrator is allowed to reroute flows through all the active edges in the network. Finally, we provide concluding remarks in Section 8.

## 2. Related work

Given the practical importance of ensuring robustness in design and operation of critical infrastructure systems and evaluating the resilience of such systems, several game-theoretic models have been proposed to model the attacker–defender interactions (Manshaei et al., 2013). We summarize these contributions along four threads of research. Section 2.1 summarizes existing research on applying security game models in critical infrastructure systems. Section 2.2 recaps literatures on using Stackelberg security games for ensuring physical security. Section 2.3 summarizes literature in solving a relevant example of Stackelberg games called network interdiction problem, which is the motivation behind our robust and adaptive maximum network flow problem. Finally, in Section 2.4, we summarize relevant literatures in robust and adaptive optimization, and their applications on network flow problems.

### 2.1. Security games in critical infrastructure systems

Our work closely resembles the broader research theme of network security games in critical infrastructure systems. Security games, that are used to model attacker–defender interactions in a network, have been employed for strategic network design (Laporte et al., 2010; Dziubiński and Goyal, 2013) of critical infrastructure systems (e.g., railways and defense), where the goal is to optimize a certain utility function by considering the possibility of edge failure. Furthermore, security games are widely used to examine the vulnerability of nodes and edges of critical infrastructure networks including transportation (Baykal-Guersoy et al., 2014), shipment of hazardous material (Szeto, 2013) and communication networks (Gueye et al., 2012). Gueye and Marbukh (2012) propose a security game model between an operator and an attacker to quantify the trade-off between vulnerability and security cost for supply–demand networks.

In the context of network flow problems, several recent research works (Dahan et al., 2018; He et al., 2012; Ma et al., 2011; Wu and Amin, 2018; Wu et al., 2018) have designed simultaneous attacker–defender games in which the adversary is allowed to disrupt multiple edges within a fixed budget constraint so as to identify critical and vulnerable edges in critical infrastructure networks. Similar to our setting, network flow models that are built upon the *max-flow* and *min-cut* theorem, have been proposed for vulnerability assessment of infrastructure networks (Assadi et al., 2014; Dwivedi and Yu, 2013).

Dahan and Amin (2015) and Dahan et al. (2018) combine network flow models within a simultaneous game framework to learn the attacker–defender interaction. However, none of these approaches consider the option to reroute flows through other paths with residual capacity.

We employ network flow models to represent the decision problems of both the administrator and the adversary, but our approach differs from the simultaneous game methods mentioned in this section as we propose an iterative game to identify a robust flow strategy for the administrator by considering the fact that the flow of an attacked edge can be rerouted through other paths with residual capacity.

### 2.2. Stackelberg security games

In this section, we summarize existing research on a relevant class of iterative security games that focuses on identifying optimal Stackelberg strategies (Tsai et al., 2010; Kar et al., 2017). These leader–follower based Stackelberg security games (SSGs) have been applied successfully in many real-world mobile patrolling applications ranging from security patrolling (Pita et al., 2008; Brown et al., 2014) to wild-life protection (Fang et al., 2016) to opportunistic crime (Zhang et al., 2016). The fundamental concept behind these problems is to identify a randomized patrolling strategy that efficiently allocates security personnel on a network to defend against adversarial events. Similar to the challenges faced by our proposed iterative two-player game, solving these large normal-form SSGs is practically infeasible, and therefore, sophisticated methods are needed to speed up the solution process by exploiting specific domain structure.

To solve the iterative SSGs for large-scale infrastructure protection, Jain et al. (2010) propose a combination of column generation and branch-and-bound algorithm that exploits the network flow representation of the problem. In a similar direction, Guo et al. (2016) propose a column and constraint generation algorithm to approximately solve the network security game. Jain et al. (2011) employ a double oracle algorithm to solve the SSGs, which motivates our incremental strategy generation approach. In contrast to enumerating the entire exponential sized strategy space for the players, the proposed double oracle algorithm updates the oracle for the attacker and the defender by adding their pure strategy best response and then, solves an optimization model in each iteration of the game to compute a randomized edge interdiction strategy for the restricted game consisting of pure strategy oracles of both the players. The final solution at the convergence is a randomized patrolling strategy that can be used as a security personnel schedule, as the randomization increases the uncertainty faced by the attacker.

However, these approaches cannot be readily employed to solve our problem as we have exponentially sized strategy space, and for each pair of attacker and defender strategy, we need to solve an optimization model (due to the fact that the flow can be rerouted through other paths in case of adversarial attacks) to compute the payoff value. Furthermore, we assume that the adversary is more powerful and has a perfect knowledge of the flow strategy chosen by the administrator (which in practice does not change over time) and therefore, we need to identify a robust pure strategy for the administrator.

### 2.3. Network interdiction and fortification games

Robust and adaptive network flow problems are motivated by network interdiction problems, which are examples of Stackelberg games. In classical network interdiction problems, a malicious entity carries illegal goods through the network and a security agency deploys a set of interdictors to prevent the illegal activity. Due to its practical importance in defense and drug enforcement, a wide variety of research papers have addressed both deterministic (Wood, 1993) and stochastic (Cormican et al., 1998) network interdiction problems. The earlier works in addressing network interdiction problems (Wollmer, 1964; Ratliff et al., 1975; Ball et al., 1989) propose methodologies for identifying $n$ "most vital" edges in the network. Similar to our problem formulation, several research papers employed sequential and simultaneous two-player game-theoretical models to interdict a maximum flow in a network (Washburn and Wood, 1995; Sullivan and Smith, 2014). Bertsimas et al. (2016) introduce an arc-based and a path-based formulation to solve a two-stage sequential network interdiction game where the defender iteratively chooses an interdiction strategy to disrupt the current flow executed by an adversary. In contrast to these two-stage minimax games, we consider a three-stage robust maximum flow problem in which the administrator can modify the flow in the third stage, possibly through rerouting.

Our three-stage robust maximum flow problem can be considered as a defender–attacker–defender (DAD) based network fortification problem, which is a three-stage network interdiction problem. DAD models have been used to generate robust decisions for facility protection (Church and Scaparra, 2007), resource allocation in a shortest path network (Cappanera and Scaparra, 2011) and critical infrastructure resilience (Alderson et al., 2011). These three-stage DAD models are difficult to solve, and therefore, several approaches are proposed to convert them into single or two-stage problems. Church and Scaparra (2007) and Scaparra and Church (2008a,b) reformulate the DAD model as a single-level problem by enumerating all possible attack plans, and solve it with efficient implicit enumeration algorithm. Several approaches combine the second and third stage problems of a network fortification game using duality mechanism (Alderson et al., 2013, 2015), and solve the reformulated problem using Benders decomposition (Brown et al., 2006) or cutting-plane approach (Smith et al., 2007). Lozano and Smith (2017) recently propose to iteratively refine the samples from third stage solution space so as to solve the bilinear interdiction problems, which are then added as cuts in the first stage defender model. In contrast to these DAD models, where the first stage (the defender fortifies the network) decision variables are binary valued, our first stage decision variables (identifying a feasible flow strategy) are continuous. We combine the second and third stage problems using duality mechanism and iteratively generate samples from the second stage solution space (i.e., attack plans) to improve the first stage defender solution.

In network interdiction problems, the goal is to determine the worst-case scenario, assuming the decision maker is in a position to act after the realization of uncertainty. The operator can either fortify the network components partially against the worst-case scenario or rebuild some components after the failure is realized. Our work differs from this thread of research as we are interested in proactively identifying those flow decisions that are robust against any plausible attacks.

### 2.4. Robust and adaptive optimization

The last thread of research which is complementary to our work presented in this paper is on robust and adaptive optimization. Robust optimization is a broad research area and the solution methodology varies for different problem settings (Ben-Tal and Nemirovski, 1998; Bertsimas et al., 2011, 2018). In a robust optimization framework, the input data of a problem can vary within an uncertainty set. A robust optimization model in general either aims to optimize a particular worst-case or regret-based objective value, or generates a solution that remains feasible for every realization of data within the uncertainty set. Several studies have considered network flow problems within the framework of robust optimization (Bertsimas and Sim, 2003; El Ghaoui et al., 1998; Ben-Tal and Nemirovski, 1998). The design of a robust optimization model varies with different objective functions such as worst-case (Vorobyov et al., 2003; Ghosh et al., 2016), regret (Ahmed et al., 2013) and risk sensitive (Adulyasak and Jaillet, 2015) objectives. The design of a robust optimization solution and the level of conservatism also vary with different types of uncertainty sets such as ellipsoidal (El Ghaoui et al., 1998; Ben-Tal and Nemirovski, 2000; Bertsimas and Sim, 2004b), polyhedral (Bertsimas and Sim, 2003, 2004a)

and interval (Li and Azarm, 2008; Ghosh et al., 2019) uncertainty sets. Our proposed iterative game is designed for robust network flow problems with worst-case objective and interval uncertainty set.

Adaptive optimization models are used to address multi-stage decision-making problems under uncertainty. Ben-Tal et al. (2004) introduce a two-stage adaptive optimization model where a decision is made before uncertainty is realized, which is followed by another set of decisions. They show that the adaptive counterpart of this optimization problem is NP-hard in general and therefore, several approximation methods have been proposed subsequently to tackle this problem (Ben-Tal et al., 2004; Bertsimas et al., 2011; Bertsimas and Goyal, 2010). In the context of network flow problems, adaptive optimization models have been proposed to tackle different demand uncertainty sets such as polyhedral (Mattia, 2013; Poss and Raack, 2013) and interval (Atamtürk and Zhang, 2007) uncertainty sets, and these problems are shown to be NP-hard. On the contrary to these existing works on demand robustness, we consider network flow problems with edge failures.

Our work is motivated by the robust and adaptive network flow problem introduced by Bertsimas et al. (2013). They assume that the flow at an edge can be adjusted after edge failure occurred, but the adjusted flow should be upper bounded by the initially assigned flow value. They propose a linear optimization model to solve the complex three-stage adaptive network flow problem quickly and approximately. However, we experimentally show that the proposed approximate solution performs poorly when the flow from an attacked edge is allowed to reroute through other paths with residual capacity. In addition, we assume that the administrator faces a cost for routing flows through an edge, which increases the complexity of the optimization problem for computing a robust and adaptive maximum flow solution.

## 3. Problem formulation

We begin with a formal definition of the Network Flow Game (**NFG**). Let $G = \langle \mathcal{V}, \mathcal{E} \rangle$ denote a network, where $\mathcal{V}$ symbolizes the set of nodes, and $s, t \in \mathcal{V}$ represent the source node and the terminal node, respectively. Let $\mathcal{E}$ denote the set of edges, where $e_{ij} \in \mathcal{E}$ represents a directed edge from node $i$ to node $j$. We also denote an edge simply as $e \in \mathcal{E}$ and assume that it has a finite capacity $U_e$, corresponding to the maximum amount of flow that can be sent through the edge. We introduce an artificial edge $e_{ts}$ with infinite capacity between the terminal node and the source node, which is excluded in the edge set $\mathcal{E}$. In addition, we introduce the following notations:

- $\delta_v^+$: The set of incoming edges to node $v$. For the source node $s$, $\delta_s^+$ includes the artificial edge $e_{ts}$.
- $\delta_v^-$: The set of outgoing edges from node $v$. For the terminal node $t$, $\delta_t^-$ includes the artificial edge $e_{ts}$.
- $\Lambda_v$: The set of all possible edges that lie in one of the directed walks from node $v$ to the terminal node $t$.
- $\Lambda_\sigma = \bigcup_{v \in \sigma} \Lambda_v$: The set of all possible edges that lie in one of the directed walks from any of the nodes of set $\sigma$ to the terminal node $t$.
- $\Gamma$: Budget for the adversary. That is to say, a maximum of $\Gamma$ edges can be attacked.

A flow scenario $x$ is a function $x : \mathcal{E} \to \mathbb{R}_{\geq 0}$ that assigns a non-negative flow value $x_e$ to each edge $e \in \mathcal{E}$ such that the following capacity and flow conservation constraints (1) are ensured:

$$\begin{aligned} \sum_{e \in \delta_v^+} x_e - \sum_{e \in \delta_v^-} x_e &= 0 && \forall v \in \mathcal{V} \setminus \{s\} \\ 0 \leq x_e &\leq U_e && \forall e \in \mathcal{E} \end{aligned} \tag{1}$$

Let $\mathcal{X}$ denote the set of all possible flow scenarios that satisfy the flow conservation constraints (1). An attack $\mu$ is a function that maps the edges to a binary value, where $\mu_e$ is set to 1 if the edge $e \in \mathcal{E}$ is attacked by the adversary and 0 otherwise. The set of possible attacks that satisfy the budget constraint of the adversary is denoted by $\Psi$.

$$\Psi := \left\{ \mu = (\mu_e)_{e \in \mathcal{E}} \in \{0,1\}^{|\mathcal{E}|} \Big| \sum_{e \in \mathcal{E}} \mu_e \leq \Gamma \right\} \tag{2}$$

We now describe the NFG problem in the context of critical infrastructure networks, where the flow of an attacked edge might be rerouted through adjacent edges with residual capacity. In the crude oil distribution pipeline application, for a given network with fixed edge capacities, the initial flow might partially or fully utilize the edge capacities. However, once this network structure is modified due to edge failures, it is feasible to reroute the flows of an attacked edge to optimally utilize the remaining pipes. We primarily assume that the adjusted flow in an edge can utilize the residual capacity only for rerouting flows from an attacked edge in a forward walk (i.e., a walk from the source node of an attacked edge to the terminal node). This assumption is motivated by the simple properties of liquid flow in which due to the pressure in pipes, additional flows from an attacked edge should naturally be rerouted in a forward direction through adjacent edges with residual capacity. It should also be noted that our proposed solution methodology performs equally well even if we relax the assumption of forward flow rerouting, and allow the flow of an attacked edge to be rerouted through any active edges in the network, as demonstrated in Section 7.

We first provide the additional setup needed to describe the problem. If the adversary attacks an edge $e \in \mathcal{E}$ (i.e., $\mu_e = 1$), then the capacity of the edge is modified from $U_e$ to $m_e$. We set the value of $m_e$ to 0, if the edge is completely blocked. We assume that the administrator faces a routing cost for transporting the flow through edges. Let $p_e$ denote the cost for routing one unit of initial flow through edge $e \in \mathcal{E}$ and the reward for successfully getting one unit of flow to the terminal node is assumed to be 1. Moreover, we assume that the administrator faces an additional $p_e$ cost for rerouting one unit of flow through edge $e$. For example, in the oil distribution pipeline application, the operational cost for routing initially allocated flow remains unchanged after disruption. That is to say, the administrator incurs same amount of routing cost for the uninterrupted flow that reaches the terminal node, and the routing cost for interrupted initial flow due to edge failures, that determines the rate of network disruption from the equilibrium at the normal condition, is considered as penalty to the administrator for the amount of adjustments required in the daily operations. Therefore, the flow rerouting (e.g., utilizing residual capacity) incurs additional operational cost for management of extra load than it is originally planned for regular daily operations. If the sum of initial routing cost and rerouting cost (due to attacks) for pushing one unit of flow to the terminal node exceeds 1, then an optimal strategy is to send zero flow through the network. To avoid such trivial situation, we assume that the value of $p_e$ is always upper bounded by $\frac{1}{2L}$, where $L$ represents the maximum number of edges in a source to destination path. The cost for routing one unit of flow through the artificial edge $p_{e_{ts}}$ is set to 0.

In this setting, given an initial flow $x$ and a resulting attack $\mu$, we can compute the maximum adaptive value of $x$ for the administrator after rerouting flows, $M(x, \mu)$, by solving the following linear optimization (LO) model (3a)–(3f). Let $y_e$ denote the resulting amount of flow going through edge $e \in \mathcal{E}$ and $z_e$ denote the amount of additional flow that is being rerouted through edge $e$ due to the disruption from the attack $\mu$. The objective function (3a) of the LO model computes the trade-off between maximizing the ultimate flow that can be pushed to the terminal node (which is equivalent to the flow, $y_{(t,s)}$ of the artificial edge $e_{ts}$) and minimizing the total rerouting cost of the additional flow from the attacked edges[2].

$$M(x, \mu) = \max \left\{ y_{(t,s)} - \sum_{e \in \mathcal{E}} p_e z_e \right\} - \sum_{e \in \mathcal{E}} p_e x_e \tag{3a}$$

[2] As the input flow $x$ is fixed, the initial flow routing cost (i.e., $\sum_{e \in \mathcal{E}} p_e x_e$) is constant in the objective function (3a).

$$\text{s.t.} \sum_{e\in\delta_v^+} y_e - \sum_{e\in\delta_v^-} y_e \geq 0 \qquad \forall v \in \mathcal{V}\setminus\{s\} \qquad (3b)$$

$$y_e \leq x_e \qquad \forall e \notin \Lambda_{\sigma_\mu} \qquad (3c)$$

$$y_e \leq (1-\mu_e)U_e + \mu_e m_e \qquad \forall e \in \Lambda_{\sigma_\mu} \qquad (3d)$$

$$z_e \geq y_e - x_e \qquad \forall e \in \Lambda_{\sigma_\mu} \qquad (3e)$$

$$y_e \geq 0; z_e \geq 0 \qquad \forall e \in \mathcal{E} \qquad (3f)$$

As a portion of the flow might be lost due to the attack, we employ a weaker notion of flow conservation at the nodes using constraints (3b) to avoid infeasibility issues. Let $\sigma_\mu$ denote the set of source nodes of the attacked edges for an attack $\mu$. As indicated earlier, $\Lambda_{\sigma_\mu}$ represents the set of all possible edges that lie in one of the walks from any of the nodes of set $\sigma_\mu$ to the terminal node. As the flow from an attacked edge can only be rerouted through a forward walk, $\Lambda_{\sigma_\mu}$ represents the set candidate edges that can be utilized for flow rerouting. Therefore, if an edge $e$ does not belong to the set $\Lambda_{\sigma_\mu}$, then constraints (3c) ensure that the value of $y_e$ is bounded by the initial flow value $x_e$. For all the other edges which belong to the set $\Lambda_{\sigma_\mu}$, there can be two possibilities: (a) If the edge $e$ is attacked, then we can send maximum $m_e$ amount of flow through the edge; and (b) If the edge $e$ is not attacked, then we can send maximum $U_e$ amount of flow through the edge. We combine these two possibilities and represent them using constraints (3d). Finally, constraints (3e)–(3f) ensure the amount of rerouted flow for edge $e$, $z_e = \max(0, y_e - x_e)$.

**Example 3.1.** We provide an illustrative example in Fig. 1 to better explain the constraints in the LO model. We consider a network with six nodes and nine edges. We show an initial flow solution $x$ in the first network, where each pair of numbers represents the flow and the capacity value for that particular edge. Assume that the adversary can attack a maximum of one edge in the network. Assume the adversary attacks edge $e_{1t}$ and the resulting flow in that edge is set 0. As the flow of edge $e_{1t}$ can only be rerouted through a forward path that contains edge $e_{13}$ and $e_{3t}$, the resulting flows for all the other edges remain the same. The resulting flow is shown in the second network, where the blue dotted line shows the augmented path through which the flow of the attacked edge is being rerouted.

Given an initial flow strategy $x$ chosen by the administrator, the adversary observes the flow and executes an attack that optimally disrupts the flow $x$. Therefore, for a given flow $x$, an optimal attack $\mu(x)$ and the adversary's objective value (also considered as the adaptive value of $x$) $AV(x)$ are defined as follows:

$$\begin{aligned} \mu(x) &\in \arg\min_{\mu\in\Psi} M(x,\mu) \\ AV(x) &= \min_{\mu\in\Psi} M(x,\mu) \end{aligned} \qquad (4)$$

Finally, the goal of the network administrator is to identify a robust flow strategy which has the maximum adaptive value. Therefore, the objective of the administrator is defined as follows:

$$\max_{x\in\mathcal{X}} AV(x) = \max_{x\in\mathcal{X}} M(x,\mu(x)) \qquad (5)$$

**Proposition 1.** *The problem of computing a robust and adaptive maximum flow strategy to be solved by the administrator from expression (5) is strongly NP-hard.*

**Proof.** The adaptive maximum flow problem introduced by Bertsimas et al. (2013) is a related problem where the routing cost is assumed to be 0 and one can adjust the flow solution after every realization of edge failure by assuming that the adjusted flow is bounded by the initial flow assigned to an edge. Bertsimas et al. (2013) show that the adaptive maximum flow problem is strongly NP-hard by reducing it to a classical network interdiction problem (Wood, 1993). Our robust and adaptive maximum flow problem can be reduced to the adaptive maximum flow problem if we set the routing cost to 0 and restrict the adjusted flow for an edge to be upper bounded by the initial flow assigned to it. Due to this trivial reduction to the adaptive maximum flow problem which is strongly NP-Hard, our problem is also strongly NP-Hard. ■

## 4. Solution approach

In this section, we describe an iterative sequence of best-response plays between the administrator and an adversary to solve the problem defined in Section 3. In the first iteration of the game, the administrator assumes that no attack is planned in the system and computes a flow solution that maximizes her objective value. Next, the adversary finds an optimal attack to disrupt the flow solution computed by the administrator. In the subsequent iterations, the administrator considers all the attacks revealed by the adversary in previous iterations and finds a robust (maximin) flow strategy against those attacks. The adversary always computes a best myopic disruption against the flow solution revealed by the administrator in the current iteration. This iterative process continues until the objectives of both the players converge to the same value, or the players start repeating their previous strategies. We now describe the details of optimization problems to be solved by the adversary and by the administrator during each iteration of the game.

### 4.1. Optimization problem for the adversary

Once the administrator reveals a flow scenario $\bar{x}$, the adversary solves the problem (4) to generate an attack that optimally disrupts the flow $\bar{x}$ by considering the fact that the administrator can reroute flow from an attacked edge through other forward paths with residual capacity. We now provide an alternative formulation of the problem (4) to mathematically represent the notion of forward rerouting paths. The set of expressions (6a)–(6h) illustrates the alternative decision problem for the adversary, $\mathcal{P}_{\text{ADV}}$. The objective function (6a) corresponds to finding an attack $\mu$ that minimizes the resulting objective value of the administrator, represented by the inner maximization problem. Constraints (6b)–(6c) ensure flow conservation and capacity constraints on the flow variables $y$. Let $\rho_e$ denote the set of edges, which if attacked, can have a portion of their flows rerouted through edge $e$, i.e., $\rho_e = \{(u,v) = e' \in \mathcal{E} | e \in \Lambda_u\}$. Let $\pi_e$ denote a variable which is set to 0 if none of the edges in the set $\rho_e$ are attacked, and otherwise the value of $\pi_e$ is set to 1. Constraints (6e) and (6g) determine the value of $\pi$. Constraints (6d) enforce that the flow passing through an edge $e$ is bounded by the given flow $\bar{x}_e$ if none of the edges in $\rho_e$ are attacked (i.e., $\pi_e = 0$). Finally, as the inner objective function minimizes the rerouting cost, constraints (6f) are sufficient alone to accurately compute the value of $z_e$.

$$\mathcal{P}_{\text{ADV}} = \min_{\mu\in\Psi}\Big\{\max\ y_{(t,s)} - \sum_{e\in\mathcal{E}} p_e \cdot z_e\Big\} \qquad (6a)$$

$$\text{s.t.} \sum_{e\in\delta_v^+} y_e - \sum_{e\in\delta_v^-} y_e \geq 0 \qquad \forall v \in \mathcal{V}\setminus\{s\} \qquad (6b)$$

$$y_e \leq (1-\mu_e)U_e + \mu_e m_e \qquad \forall e \in \mathcal{E} \qquad (6c)$$

$$y_e \leq (1-\pi_e)\bar{x}_e + \pi_e \cdot U_e \qquad \forall e \in \mathcal{E} \qquad (6d)$$

$$\pi_e \leq \sum_{e'\in\rho_e} \mu_{e'} \qquad \forall e \in \mathcal{E} \qquad (6e)$$

$$z_e \geq y_e - \bar{x}_e \qquad \forall e \in \mathcal{E} \qquad (6f)$$

$$\pi_e \in \{0,1\} \qquad \forall e \in \mathcal{E} \qquad (6g)$$

$$y_e \geq 0; z_e \geq 0 \qquad \forall e \in \mathcal{E} \qquad (6h)$$

**Proposition 2.** *The integrality constraints (6g) can be relaxed to $0 \leq \pi_e \leq 1$, without compromising the feasibility or optimality of the optimization problem (6a)–(6h).*

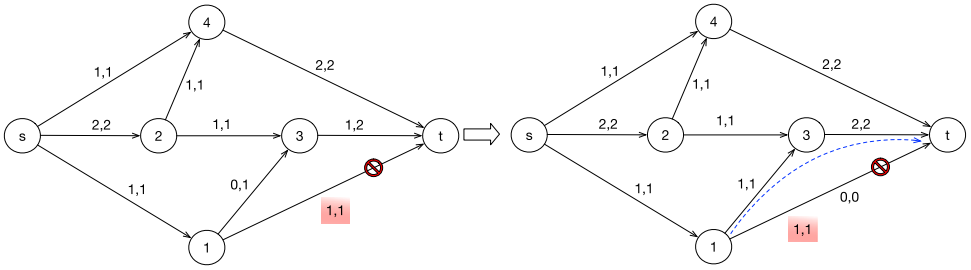


**Fig. 1.** Illustration of resulting flow after an attack in a network.

**Proof.** As $\mu$-variables are binary, constraints (6e) enforce that the value of $\pi_e$ is set to 0 if none of the edges in set $\rho_e$ are attacked. On the other hand, if some edges in set $\rho_e$ are attacked, then the excessive flow of those edges might be rerouted through edge $e$ and therefore, the actual flow through edge $e$ can take any value between $\bar{x}_e$ and $U_e$. As $\pi_e$ is used by constraints (6d) to only enforce an upper bound on the flow variable $y_e$, even with continuous relaxation, $\pi_e$ will take a value of 1 to maximize the inner objective of expression (6a). ■

Unfortunately, the adversary's optimization model cannot be solved directly as a linear program due to the minimax function in objective (6a). However, as the inner maximization problem is a linear program, we can convert the entire problem into a minimization problem by taking the dual of the inner problem. The dual problem $\mathcal{D}_{\text{ADV}}$, which is quadratic nature, is compactly shown using the set of expressions (7a)–(7g), where $\alpha, \beta, \gamma, \zeta, \omega$ and $\eta$ represent nonnegative dual price variables for constraints (6b)–(6g), respectively. We set $\mu$ as variable in the dual problem and incorporate the domain constraints (7e) of $\mu$ to represent the optimization problem of the adversary.

$$\mathcal{D}_{\text{ADV}} = \min_{\alpha,\beta,\gamma,\zeta,\eta,\omega,\mu} \sum_{e\in\mathcal{E}} \beta_e U_e - \sum_{e\in\mathcal{E}} \beta_e \mu_e (U_e - m_e) + \sum_{e\in\mathcal{E}} \gamma_e \bar{x}_e + \sum_{e\in\mathcal{E}} \zeta_e \sum_{e'\in\rho_e} \mu_{e'} + \sum_{e\in\mathcal{E}} \eta_e + \sum_{e\in\mathcal{E}} \omega_e \bar{x}_e \tag{7a}$$

$$\text{s.t. } \beta_e + \gamma_e + \omega_e + \alpha_v - \alpha_w \geq 0 \qquad \forall e := (v,w) \in \mathcal{E} \tag{7b}$$

$$\zeta_e + \eta_e - \gamma_e (U_e - \bar{x}_e) \geq 0 \qquad \forall e \in \mathcal{E} \tag{7c}$$

$$\omega_e \leq p_e \qquad \forall e \in \mathcal{E} \tag{7d}$$

$$\sum_{e\in\mathcal{E}} \mu_e \leq \Gamma \tag{7e}$$

$$\alpha_t = 1; \alpha_s = 0 \tag{7f}$$

$$\alpha_v, \beta_e, \gamma_e, \zeta_e, \eta_e, \omega_e \geq 0; \mu_e \in \{0,1\} \tag{7g}$$

The objective function (7a) – redefined by maximizing the problem (6a)–(6h) over all the variables except for $\mu$ – would be concave in $\mu$. As a concave function over a compact domain achieves the optimal solution at an extreme point of the feasible region, we can relax the binary variables $\mu$ to continuous ones (see e.g., the proof of Lemma 6 of Bertsimas et al. 2013). However, if $\mu$-variables are binary, then the bilinear terms in the objective function (7a) can be represented as *special ordered set* (SOS) constraints, and these constraints are efficiently implemented by existing mixed-integer optimization solvers like CPLEX or Gurobi.

### 4.2. Optimization problem for the administrator

In a fully secured network, the optimal strategy for the network administrator is to adopt one of the *max-flow min-cost* solutions (Ahuja et al., 1993; Orlin, 1997). However, in case of an adversarial attack, a *max-flow min-cost* solution is not necessarily an optimal strategy. The administrator's goal is then to identify a robust flow strategy that maximizes the objective value under worst-case attack, as indicated in problem (5). However, as the strategy space of the adversary increases with network size and budget value, we use an incremental approach and consider the attacks revealed by the adversary in previous iterations only. In the last iteration of the game, as the adversary repeats her previous attacks as the best response, the administrator at this point has already generated a robust flow strategy against these worst-case attacks without considering the entire large strategy space of the adversary.

Given a set of $K$ attacks (the set of indices for these attacks is denoted by $\mathcal{K}$) revealed by the adversary in previous $K$ iterations, the administrator generates a flow strategy that maximizes the minimum objective value over these $K$ attacks. Let $\hat{\mu}^k$ denote the attack $k$ and $\hat{x}$ denote the robust flow strategy against these attacks. Even though the initial flow decision $\hat{x}$ is chosen, it may lead to different outcomes under different attacks, and the actual amount of flow that reaches the terminal node will vary depending on the attack. So, we introduce $\hat{y}^k$ variables to represent the actual flow and $\hat{z}^k$ variables to represent the additional rerouted flow under the attack $k$. $\hat{\pi}^k_e$ is a given input to the administrator's decision model, which is set to 0 if none of the edges in edge set $\rho_e$ (from which the flow can be rerouted to edge $e$) are attacked under attack $k$ and otherwise, it is fixed to 1:

$$\hat{\pi}^k_e = \min(1, \sum_{e'\in\rho_e} \hat{\mu}^k_{e'}) \qquad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{8}$$

We show the entire linear optimization problem for the administrator, $\mathcal{P}_{\text{ADM}}$ compactly using the set of expressions (10a)–(10i). The goal of the administrator is to maximize the worst-case objective value over $K$ previously executed attacks.

$$\max \min_{k\in\mathcal{K}} \hat{y}^k_{(t,s)} - \sum_{e\in\mathcal{E}} p_e \left[\hat{x}_e + \hat{z}^k_e\right] \tag{9}$$

The objective function (9) can easily be linearized using expressions (10a)–(10b), where we maximize a proxy variable $\lambda$ that represents the worst-case objective value over $K$ attacks. Constraints (10c) compute the value of $\hat{z}^k_e$. Constraints (10d)–(10g) enforce the flow preservation and capacity constraints on the flow variables $\hat{x}$ and $\hat{y}^k$. Finally, constraints (10h) enforce that the actual flow going through edge $e$ under attack $k$, $\hat{y}^k_e$ is bounded by the initial robust flow $\hat{x}_e$ if none of the edges in set $\rho_e$ are attacked and otherwise, the upper bound is set to the capacity of the edge, $U_e$.

$$\mathcal{P}_{\text{ADM}} = max \ \lambda \tag{10a}$$

$$s.t. \ \lambda \leq \hat{y}^k_{(t,s)} - \sum_{e\in\mathcal{E}} p_e \left[\hat{x}_e + \hat{z}^k_e\right] \qquad \forall k \in \mathcal{K} \tag{10b}$$

$$\hat{z}^k_e \geq \hat{y}^k_e - \hat{x}_e \qquad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{10c}$$

$$\sum_{e\in\delta^+_v} \hat{x}_e - \sum_{e\in\delta^-_v} \hat{x}_e = 0 \qquad \forall v \in \mathcal{V} \setminus \{s\} \tag{10d}$$

$$\hat{x}_e \leq U_e \qquad \forall e \in \mathcal{E} \tag{10e}$$

$$\sum_{e\in\delta^+_v} \hat{y}^k_e - \sum_{e\in\delta^-_v} \hat{y}^k_e \geq 0 \qquad \forall k \in \mathcal{K}, v \in \mathcal{V} \setminus \{s\} \tag{10f}$$

$$\hat{y}^k_e \leq (1 - \hat{\mu}^k_e) U_e + \hat{\mu}^k_e \cdot m_e \qquad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{10g}$$

$$\hat{y}_e^k \le (1-\hat{\pi}_e^k)\hat{x}_e + \hat{\pi}_e^k \cdot U_e \qquad \forall k \in \mathcal{K}, e \in \mathcal{E} \qquad (10h)$$

$$\lambda \ge 0; \hat{x}_e \ge 0; \hat{y}_e^k \ge 0; \hat{z}_e^k \ge 0 \qquad \forall k \in \mathcal{K}, e \in \mathcal{E} \qquad (10i)$$

### 4.3. Overall solution approach

In this section, we put together the optimization problems described in Section 4.1 and Section 4.2. To better understand the overall framework of our proposed solution approach for solving the NFG problem, we explain the key iterative steps in Algorithm 1. We essentially formulate it as a leader–follower game, where the administrator is the leader and the adversary acts as the follower. In that sense, the adversary is more powerful and takes decisions after observing the decision of the administrator. As a result, the administrator needs to take robust decisions by considering all possible attacking behaviors of the adversary.

**Algorithm 1:** SolveNFG($G, U, p, \Gamma$)

**Initialize:** $\boldsymbol{X} \leftarrow \{\}, \boldsymbol{\mu} \leftarrow \{\}, \mu_{old} \leftarrow \mathbf{0}, V^L \leftarrow 0, V^U \leftarrow \infty$ ;
**while** $(V^U \neq V^L)$ **do**
  $\tilde{\boldsymbol{x}}, V^U \leftarrow \mathcal{P}_{\text{ADM}}(G, U, p, \boldsymbol{\mu})$ ; ▷ Solve administrator's problem (10a)-(10i) for attack set $\boldsymbol{\mu}$
  $\boldsymbol{x}_p \leftarrow \tilde{\boldsymbol{x}} \cap \boldsymbol{X}$ ; ▷ $\boldsymbol{x}_p$ contains previously executed flows
  $\boldsymbol{x}_c \leftarrow \tilde{\boldsymbol{x}} \setminus \boldsymbol{x}_p$ ; ▷ $\boldsymbol{x}_c$ contains flows which are not executed yet
  **if** $|\boldsymbol{x}_c| = 0$ **then**
    **return** $\tilde{\boldsymbol{x}}, \boldsymbol{\mu}$ ; ▷ Process converges
  **else**
    $x^* \leftarrow Random(\boldsymbol{x}_c)$ ; ▷ Randomly execute one flow from $\boldsymbol{x}_c$
    $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup x^*$ ; ▷ Add flow $x^*$ into flow strategy pool $\boldsymbol{X}$
  $\tilde{\mu}, V^L \leftarrow \mathcal{D}_{\text{ADV}}(G, U, p, x^*, \Gamma)$ ; ▷ Solve adversary's problem (7a)-(7g) for disrupting flow $x^*$
  **if** $\tilde{\mu} \neq \mu_{old}$ **then**
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} \cup \tilde{\mu}$ ; ▷ Add $\tilde{\mu}$ to attack pool $\boldsymbol{\mu}$
    $\mu_{old} \leftarrow \tilde{\mu}$ ; ▷ Update the old attack
  **else**
    **return** $x^*, \boldsymbol{\mu}$ ; ▷ Process converges
**return** $x^*, \boldsymbol{\mu}$

Let $\boldsymbol{X}$ and $\boldsymbol{\mu}$ store all the previously generated flow strategies and attacks respectively, and both are initialized as empty set. Let $\mu_{old}$ denote the attack from the last iteration, and it is initialized to no attacks. Let $V^L$ and $V^U$ denote the lower and the upper bound of the game which are initialized to 0 and $\infty$, respectively. In each iteration of the game, the administrator solves the optimization problem, $\mathcal{P}_{\text{ADM}}$ to compute the optimal robust flow strategy against the previously generated attack pool $\boldsymbol{\mu}$. Note that the administrator's optimization problem (10a)–(10i) can have multiple optimal solutions. For example, in the first iteration, the administrator essentially solves a *max-flow min-cost* problem which can have multiple flow solutions with optimal objective value. Let $\tilde{\boldsymbol{x}}$ denote the set of optimal robust flow solutions computed by the administrator in the current iteration[3]. In addition, the upper bound of the game, $V^U$ is updated to the administrator's objective value. From these flow solutions, we identify the flows, $\boldsymbol{x}_c$ which are not executed before. If all the solutions of $\tilde{\boldsymbol{x}}$ are executed before, then the process terminates as any solution from $\tilde{\boldsymbol{x}}$ would be a robust flow against any plausible attack from $\Psi$. Otherwise, we randomly pick one flow $x^*$ from $\boldsymbol{x}_c$ and execute it. We also add this new flow $x^*$ into the pool of flow solutions $\boldsymbol{X}$.

Once the administrator generates a flow $x^*$ and executes it, the adversary computes a best myopic attack $\tilde{\mu}$ to optimally disrupt the flow $x^*$ by solving the optimization model $\mathcal{D}_{\text{ADV}}$. In addition, the lower bound of the game, $V^L$ is updated to the adversary's objective value. If the adversary does not repeat the attack from the previous iteration (i.e., $\mu_{old}$), then we add the new attack $\tilde{\mu}$ into the attack pool $\boldsymbol{\mu}$, and update $\mu_{old}$ to the new attack $\tilde{\mu}$. On the contrary, if the adversary repeats the same attack from the previous iteration, then the process converges as $x^*$ is the robust flow strategy against $\tilde{\mu}$. Specifically, the objective values of both the players would converge at this point, and we can guarantee that if the administrator executes the flow $x^*$, then the lower bound on her objective value would be the value at which the objectives of the players converged. It should be noted that as the administrator's optimization problem can have uncountable number of solutions with optimal objective value, a situation might arise where although the objectives of both the players converge to the same value, they can come up with new strategies in subsequent iterations that lead to the same objective value. However, as the game value does not change from this point onward, we terminate the process if the lower and upper bound of the game converge. On the other hand, if the optimization problems of the players are solved sub-optimally (e.g., see Section 5 for large-scale problems), then the lower and upper bound of the game might not converge even though the players start repeating their previous strategies. To tackle these situations, we enforce both the termination conditions in Algorithm 1.

**Proposition 3.** *The proposed iterative game converges to a maximin equilibrium and produces a robust and adaptive maximum flow strategy for the administrator.*

**Proof.** As the adversary has a finite number of attacks (i.e., $|\Psi| = \binom{|\mathcal{E}|}{\Gamma}$), the proposed iterative game is guaranteed to converge. We begin the proof by showing that if the players start repeating their previous strategies and their respective decision problems are solved optimally, then the lower and upper bound of the game would converge. Let us suppose the game converges after $K$ iterations, and the set of $K-1$ attacks computed by the adversary in previous iterations is denoted by $\boldsymbol{\mu}$. Let $\tilde{x}$ represent the flow strategy of the administrator and $\tilde{\mu} \in \boldsymbol{\mu}$ denote the attack from the final iteration. As the adversary repeats one of her previous attacks in the final iteration, the objective of the adversary for the final iteration, $V^L = M(\tilde{x}, \tilde{\mu}) = \min_{\mu \in \boldsymbol{\mu}} M(\tilde{x}, \mu)$, where $M(\tilde{x}, \tilde{\mu})$ represents the adaptive value of the flow $\tilde{x}$ under attack $\tilde{\mu}$, which can be computed using the LO model (3a)–(3f). Similarly, as the administrator maximizes the minimum adaptive flow value over all the attacks in $\boldsymbol{\mu}$, the objective value of the administrator for the final iteration is $V^U = \min_{\mu \in \boldsymbol{\mu}} M(\tilde{x}, \mu) = V^L$.

Let $V^L = V^U = V$. From the adversary's optimized solution, since $\forall \mu \in \Psi, M(\tilde{x}, \mu) \ge V$, we know $\min_{\mu \in \Psi} M(\tilde{x}, \mu) \ge V$. On the other hand, from the administrator's optimized solution, since $\forall x \in \mathcal{X}, \min_{\mu \in \boldsymbol{\mu}} M(x, \mu) \le V$, we know $\min_{\mu \in \Psi} M(x, \mu) \le V$. By combining these two inequalities, we have $\forall x \in \mathcal{X}, \min_{\mu \in \Psi} M(\tilde{x}, \mu) \ge \min_{\mu \in \Psi} M(x, \mu)$, which concludes the proof that the administrator's final flow strategy $\tilde{x}$ is maximin optimal. ■

If Algorithm 1 ends up considering all the attacks (i.e., $K = |\Psi|$), then the proposed iterative approach will be slower than solving the original problem of identifying a flow strategy that maximizes the minimum adaptive flow value by considering all the attacks in $\Psi$. However, it is reasonable to expect that only some attacking choices are better to execute in practice than others. For example, if all the incoming edges to a node is attacked, then none of the outgoing edges from that node will be selected, or if the edge with maximum capacity in a source to destination path is attacked, then other edges in that path are less likely to be attacked. Our proposed approach provides a principled way to identify those crucial attacks. In fact, for all the problem instances in our experiment, we observe that the proposed iterative game converges within 20 iterations.

[3] For implementation purposes, we employ the solution pool feature in *CPLEX* which generates multiple solutions. In our experiments, we limit the solver to a maximum of 20 optimal solutions for each solve.

**Table 1**
IdentifyFlow$(G, U, p, \bar{x}, \mu)$

$$
\begin{aligned}
\max \quad & y_{(t,s)} - \sum_{e \in \mathcal{E}} p_e \cdot z_e & \\
\text{s.t.} \quad & \sum_{e \in \delta_v^+} y_e - \sum_{e \in \delta_v^-} y_e \geq 0 & \forall v \in \mathcal{V} \setminus s \\
& y_e \leq (1 - \mu_e) U_e + \mu_e m_e & \forall e \in \mathcal{E} \\
& y_e \leq (1 - \pi_e) \bar{x}_e + \pi_e \cdot U_e & \forall e \in \mathcal{E} \\
& z_e \geq y_e - \bar{x}_e & \forall e \in \mathcal{E} \\
& y_e \geq 0; z_e \geq 0 & \forall e \in \mathcal{E}
\end{aligned}
$$

## 5. Heuristics for large-scale networks

The runtime complexity of our iterative game approach increases with the network size and the adversary's budget value (refer to Section 6.1.3). The key reason behind this behavior is that we need to solve a non-convex quadratic program (7a)–(7g) for the adversary's decision problem in each iteration of the game. The adversary's decision problem seeks to identify $\Gamma$ best edges to attack so as to minimize the resulting objective value of the administrator for a given flow strategy. In case of "s-t planar graphs", this problem can be solved in polynomial time (Wollmer, 1964) if the entire flow of an attacked edge is assumed to be lost. However, if the flow of an attacked edge is allowed to reroute through other paths, then the resulting flow at an edge after attack is not upper bounded by the initial flow value and therefore, the existing polynomial time algorithms cannot be employed to solve our problem. Having said that, for a given attack, we can precompute the set of edges through which the additional flow might be rerouted and compute an upper bound on the resulting flow assigned to the edges. Therefore, the utility of a given attack can be computed by solving a *max-flow min-cost* problem in polynomial time, on a network whose edge capacities are set to the upper bounds. However, to compute an optimal attack, we need to solve such polynomial time algorithms for $\binom{|\mathcal{E}|}{\Gamma}$ times using an exhaustive search, which is practically intractable for large networks.

In this section, we provide two novel heuristic approaches to efficiently solve the complex optimization problem of the adversary. The first heuristic is developed using an accelerated greedy approach, and the second heuristic is a network partitioning based optimization approach. We now describe the details of these heuristic approaches for solving the adversary's decision problem.

### 5.1. Greedy approach

The greedy approach incrementally identifies the set of edges to be attacked by the adversary to disrupt the network for a given flow strategy. We begin with an alternative formulation of the optimization model (3a)–(3f) to evaluate the utility of an attack $\mu$ for a given flow scenario $\bar{x}$. The alternative LO model is compactly shown in Table 1. The value of $\pi$ is computed using Eq. (8), and is given as an input to the LO model. The only differentiating constraints from the LO model (3a)–(3f) are the third set of constraints, which use $\pi$ values to ensure the upper bound on the flow variables $y$. It should be noted that the solution of the LO model from Table 1 can be obtained in polynomial time by solving a *max-flow min-cost* problem on a modified network where the capacity of an edge $e$ is set to 0 if $\mu_e = 1$, $\bar{x}_e$ if $\pi_e = 0$, and otherwise it remains $U_e$.

Algorithm 2 provides the details of the greedy algorithm. We start with an empty attacked edge set $\mu$. Let $E$ denote an edge set that initially contains all the edges in $\mathcal{E}$. We first compute the objective value $O$ for the given flow strategy $\bar{x}$ without executing any attacks in the network. In each iteration, we calculate the utility $g_e$ for adding an edge $e \in E$ in the current attack set $\mu$ by employing the LO model from Table 1, and compute the marginal gain in the adversary's objective value for attacking the edge $e$. Then we add the best edge $e^*$ (that provides maximum marginal gain) into $\mu$ and remove it from the candidate edge set $E$. This process continues until the budget for the adversary is exhausted.

**Algorithm 2:** Greedy$(G =< \mathcal{V}, \mathcal{E} >, U, p, \bar{x}, \Gamma)$

**Initialize:** $\mu \leftarrow \{\}; it \leftarrow 0; E \leftarrow \mathcal{E}$ ;
$O \leftarrow$ IdentifyFlow$(G, U, p, \bar{x}, \mu)$ ; ▷ Compute objective for the given flow $\bar{x}$
**repeat**
  $it \leftarrow it + 1$ ;
  $g_e, \hat{x} \leftarrow$ IdentifyFlow$(G, U, p, \bar{x}, \mu \cup \{e\})$ $\forall e \in E$;
  $g_e \leftarrow O - g_e$ $\forall e \in E$ ; ▷ Compute marginal gain for edge $e$
  $e^* \leftarrow \arg\max_{e \in E} g_e$ ; ▷ Choose edge $e^*$ with highest marginal gain
  $O \leftarrow O + g_{e^*}$ ; ▷ Update objective value for current attack $\mu$
  $\mu \leftarrow \mu \cup \{e^*\}$ ; ▷ Update current attack $\mu$
  $E \leftarrow E - \{e^*\}$ ; ▷ Update candidate edge set $E$
**until** $(|\mu| \geq \Gamma)$;
**return** $\mu$

Although the LO model from Table 1 can be solved in polynomial time, the greedy approach needs to solve it for $(\Gamma \times |\mathcal{E}|)$ times and therefore, it is not suitable for problems with a large number of edges. In case of a sub-modular objective function, lazy greedy algorithms (Minoux, 1978) can be employed to accelerate the solution process. However, the following two remarks show that the adversary's decision problem is neither sub-modular nor super-modular.

**Remark 1** (*The Adversary's Decision Problem is not Sub-modular*)**.** A function is monotone sub-modular if the marginal gain for adding an element into the subset is always higher than adding the same element into its superset. Fig. 2(a) illustrates that the adversary's decision problem does not exhibit this property. The network has 8 nodes and 11 edges, and the pair of numbers in each edge represents the flow and capacity of the corresponding edge. Let us assume that the unit cost for routing the flow is 0 for all the edges. If we add edge $e_{37}$ in an attacked edge set that only contains edge $e_{26}$, the marginal gain in the adversary's objective value remains 0, as the entire disrupted flow can be rerouted through the augmented path $\{e_{36}, e_{68}\}$. In contrast, if we add edge $e_{37}$ in the superset which contains edge $e_{26}$ and $e_{45}$, then the marginal gain in the objective value is 2, as the residual capacity of edge $e_{68}$ is now shared by the rerouted flow from both edge $e_{37}$ and $e_{45}$. As the marginal gain for adding edge $e_{37}$ into the superset is higher, the adversary's decision problem is not sub-modular.

**Remark 2** (*The Adversary's Decision Problem is not Super-modular*)**.** A function is monotone super-modular if the marginal gain for adding an element into the subset is always lower than adding the same element into its superset. Fig. 2(b) illustrates that the adversary's decision problem is not super-modular. We employ the same network with 8 nodes and 11 edges. In this example, if we add edge $e_{37}$ in the attacked edge set as the first element, then the entire flow is lost and therefore, the marginal gain in the adversary's objective value is 3. In contrast, if we add edge $e_{37}$ in the superset which contains edge $e_{26}$, the entire disrupted flow can be rerouted to the terminal node and the marginal gain remains 0. Hence, the adversary's decision problem is clearly not super-modular, as the marginal gain for adding edge $e_{37}$ into the superset is lower.

Even though the adversary's decision problem is neither sub-modular nor super-modular, the following remark shows that the upper bound on the marginal gain for attacking an edge can be precomputed, which provides us the basis to develop an accelerated greedy algorithm.

**Remark 3.** The maximum amount of lost flow for attacking an edge $e$ is bounded by the flow assigned to that edge, $\bar{x}_e$ if the adversary's budget value is 1 (i.e., $\Gamma = 1$).

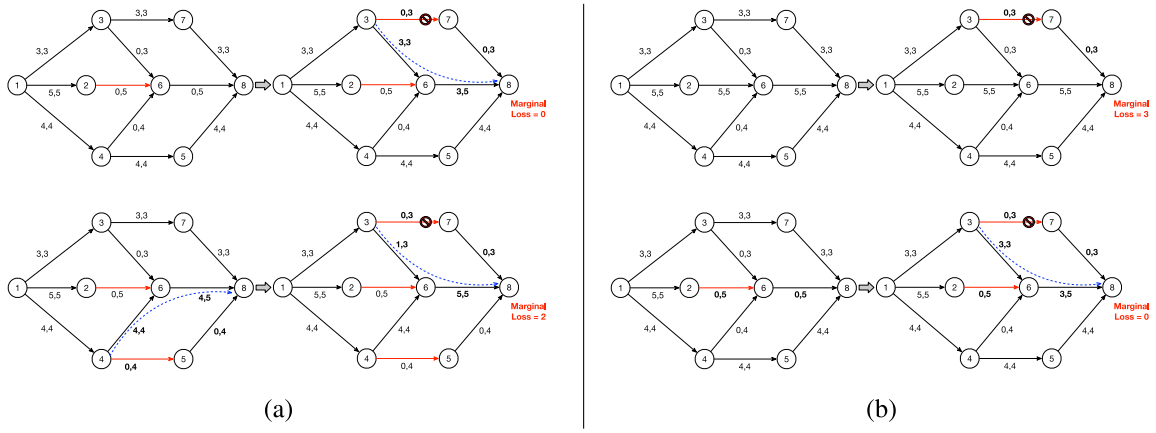


**Fig. 2.** Adversary's decision problem is (a) neither sub-modular; (b) nor super-modular.

As we incrementally elect one edge at a time for the greedy approach, we exploit the observation from Remark 3 to develop an accelerated greedy algorithm which is shown in Algorithm 3. Let $E$ denote the set of candidate edges which initially contains all the edges in $\mathcal{E}$. Let $\mu$ denote the set of attacked edges which is initialized as an empty set. We first compute the objective value $g^0$ for the given flow strategy $\bar{x}$ without considering any attacks. In each iteration, we keep an upper bound $B_e$ on the marginal gain for edge $e$, which is initialized to the given flow value $\bar{x}_e$. We also introduce a set $\hat{E}$ (initialized as an empty set) that stores the edges for which the marginal gain has already been computed in the current iteration. We iteratively select the edge $e^*$ with maximum upper bound and add it to the edge set $\hat{E}$. Then we employ the LO model from Table 1 to compute the objective value and marginal gain for adding the edge $e^*$ into the current attacked edge set $\mu$ and update the upper bound $B_{e^*}$ with the marginal gain value. In addition, as the flow values for the edges change due to modified attacks, we store the updated flow strategy $\hat{x}$. If the marginal gain for edge $e^*$ is at least the upper bound for all the unexplored edges (i.e., $e \in E \setminus \hat{E}$), then the best edge for the current iteration is identified. We then select the edge $e^{**}$ with the highest marginal gain, insert it to the set of attacked edges $\mu$ and remove it from the candidate edge set $E$. Finally, we update the flow strategy $\bar{x}$ with the modified flow strategy $\hat{x}$. Therefore, in each iteration, the accelerated greedy algorithm is able to identify the best edge without executing the LO model for $|E \setminus \hat{E}|$ times in comparison to the greedy algorithm, which significantly reduces the runtime. This iterative edge selection process continues until the adversary's budget is exhausted.

**Algorithm 3:** AcceleratedGreedy($G =< \mathcal{V}, \mathcal{E} >, U, p, \bar{x}, \Gamma$)

**Initialize:** $\mu \leftarrow \{\}; it \leftarrow 0; E \leftarrow \mathcal{E}$ ;
$g^0, \hat{x} \leftarrow$ IdentifyFlow$(G, U, p, \bar{x}, \mu)$ ; ▷ Compute objective for the given flow $\bar{x}$
**repeat**
  $it \leftarrow it + 1$ ;
  $B_e \leftarrow \bar{x}_e \forall e \in E$ ; ▷ Update upper bounds with the new flow $\bar{x}$ from last iteration
  $\hat{E} \leftarrow \{\}$ ;
  **repeat**
    $e^* \leftarrow \arg\max_{e \in E} B_e$ ; ▷ Find edge $e^*$ with highest upper bound
    $\hat{E} \leftarrow \hat{E} \cup \{e^*\}$ ; ▷ Update the list of visited edges
    $g^{it}, \hat{x} \leftarrow$ IdentifyFlow$(G, U, p, \bar{x}, \mu \cup \{e^*\})$;
    $B_{e^*} \leftarrow g^{it-1} - g^{it}$ ; ▷ Update the upper bound for edge $e^*$
    **if** $B_{e^*} \geq B_e, \forall e \in E \setminus \hat{E}$ **then**
      $e^{**} \leftarrow \arg\max_{e \in \hat{E}} B_e$ ; ▷ Choose the edge $e^{**}$ with highest marginal gain
      $\mu \leftarrow \mu \cup \{e^{**}\}$ ; ▷ Update current attacking edge set
      $E \leftarrow E - \{e^{**}\}$ ; ▷ Update the candidate edge set $E$
      $\bar{x} \leftarrow \hat{x}$ ; ▷ Update flow $\bar{x}$ according to new attack $\mu$
      **Break**;
  **until** *True*;
**until** $(|\mu| \geq \Gamma)$;
**return** $\mu$

### 5.2. Network partitioning based optimization approach

As the adversary's decision problem grows exponentially with the number of edges in the network, our second heuristic identifies a small subset of candidate edges that are highly likely to be present in an optimal solution, and solves the adversary's optimization problem, $D_{\text{ADV}}$ by only considering the subset of candidate edges. To identify the candidate edges, we propose a network partitioning based iterative approach. In each iteration, the network is partitioned into disjoint sub-networks and those sub-problems are solved independently to elect $\Gamma$ best edges to attack.

We begin the discussion with a random network partitioning method that is compactly shown in Algorithm 4. Let us consider a network with $N := |\mathcal{V}|$ nodes. We first randomly sample $(\frac{N}{2} - 1)$ nodes that excludes the source and the terminal node[4]. All the randomly sampled nodes along with the source node are kept in the first sub-network and the remaining nodes are kept in the second sub-network. In addition, an artificial terminal node $\hat{t}$ for the first sub-network and an artificial source node $\hat{s}$ for the second sub-network are created. For each directed edge $e := (u, v)$ in the network, we carry out the following operations:

- If both $u$ and $v$ lie in the first sub-network, we create a directed edge from node $u$ to node $v$.
- If both $u$ and $v$ lie in the second sub-network, we create a directed edge from node $u$ to node $v$.

[4] In case of multiple source nodes, we create an artificial source node and connect it to all the original source nodes. Similarly, in case of multiple terminal nodes, we create an artificial terminal node and connect all the original terminal nodes to the artificial terminal node. During the network partitioning process, we ensure that all the original source nodes are kept in the first sub-network and all the original terminal nodes are kept in the second sub-network.

- If $u$ lies in the first sub-network and $v$ lies in the second sub-network, then we create two directed edges—one from node $u$ to node $\hat{t}$ in the first sub-network and another one from node $\hat{s}$ to node $v$ in the second sub-network. The flow and the capacity values for both the newly introduced edges are directly taken from edge $e$[5].
- If $u$ lies in the second sub-network and $v$ lies in the first sub-network, then we remove edge $e$ from the sub-problems.

**Example 5.1.** Fig. 3 illustrates our random network partitioning approach on a small synthetic network. The network has 8 nodes and 16 edges. The numbers associated with each directed edge represent the corresponding flow and capacity values. The blue colored nodes represent the chosen nodes for the first sub-network, and other nodes are kept in the second sub-network. We create an artificial terminal node '$d$' for the first sub-network and an artificial source node '$s$' for the second sub-network. Finally, we replace each edge that connects the two sub-networks with two artificial edges (shown in red color), one sinks to the artificial terminal node and other one originates from the artificial source node. If multiple edges share the same source and destination node, we replace them with a single artificial edge with cumulative flow and capacity values.

Algorithm 5 describes the key steps of our network partitioning based heuristic approach. Let $\mu^{\mathcal{E}}$ denote the set of potential candidate edges, which is initialized as an empty set. In each iteration, we randomly partition the network into two sub-networks (i.e., $G^s, G^t$), compute the modified initial flows for two sub-networks (i.e., $\bar{x}^s, \bar{x}^t$), and solve the adversary's decision problem for both the sub-networks independently with a budget of $\frac{\Gamma}{2}$ for each sub-problem. Then we insert the resulting attacked edges from both the sub-problems to the candidate edge set $\mu^{\mathcal{E}}$. This iterative process continues until a predetermined number of iterations is completed or the cardinality of the candidate edge set reaches a given threshold value $\Delta$. Finally, we solve the adversary's optimization problem (7a)–(7g) to identify $\Gamma$ best edges to attack from the candidate edges set $\mu^{\mathcal{E}}$. Specifically, we manually set the value of $\mu_e$ to 0 if the edge $e$ does not belong to the candidate edge set (i.e., $e \notin \mu^{\mathcal{E}}$). This problem is computationally less expensive as the search space reduces from $|\mathcal{E}|$ to $|\mu^{\mathcal{E}}|(\ll |\mathcal{E}|)$.

It should be noted that, in case of extensively large networks, even the sub-problems might become intractable if we partition the network into two sub-networks. In such scenarios, our approach can be extended to recursively partition the sub-networks into even smaller networks, and then the decision problem can be solved independently for all the small networks to compute the set of candidate edges. In addition, as the network size increases, the value of the threshold parameter $\Delta$ needs to be calibrated accordingly for efficiently solving the final decision problem of the adversary over the candidate edges.

**Algorithm 4:** NetworkPartitioning($G =< \mathcal{V}, \mathcal{E} >$)

$\mathcal{V}^s \leftarrow$ RandomSample $(\mathcal{V} \setminus \{s, t\}, |\mathcal{V}|/2)$ ; ▷ Randomly sample half of the nodes
$\mathcal{V}^s \leftarrow \mathcal{V}^s \cup \{s\} \cup \{\hat{t}\}$ ; ▷ $\mathcal{V}^s$ represents the set of nodes in the first sub-network
$\mathcal{V}^t \leftarrow \mathcal{V} \setminus \{\mathcal{V}^s \cup \{s\}\} \cup \{\hat{s}\}$ ; ▷ $\mathcal{V}^t$ represents the set of nodes in the second sub-network
$\mathcal{E}^s \leftarrow \{\}$ ; ▷ $\mathcal{E}^s$ is the set of edges in the first sub-network
$\mathcal{E}^t \leftarrow \{\}$ ; ▷ $\mathcal{E}^t$ is the set of edges in the second sub-network
**for** $e := (u, v) \in \mathcal{E}$ **do**
  **if** $u, v \in \mathcal{V}^s$ **then**
    $\mathcal{E}^s \leftarrow \mathcal{E}^s \cup e$ ; ▷ Add edge $e$ directly if both $u$ and $v$ lie in first sub-network
  **if** $u, v \in \mathcal{V}^t$ **then**
    $\mathcal{E}^t \leftarrow \mathcal{E}^t \cup e$ ; ▷ Add edge $e$ directly if both $u$ and $v$ lie in second sub-network
  **if** $(u \in \mathcal{V}^s) \wedge (v \in \mathcal{V}^t)$ **then**
    $\mathcal{E}^s \leftarrow \mathcal{E}^s \cup \{u, \hat{t}\}$ ; ▷ Create an edge between $u$ and $\hat{t}$ in first sub-network
    $\mathcal{E}^t \leftarrow \mathcal{E}^t \cup \{\hat{s}, v\}$ ; ▷ Create an edge between $\hat{s}$ and $v$ in second sub-network
**return** $G^s =< \mathcal{V}^s, \mathcal{E}^s >, G^t =< \mathcal{V}^t, \mathcal{E}^t >$

**Algorithm 5:** PartitioningHeuristic($G =< \mathcal{V}, \mathcal{E} >, U, p, \bar{x}, \Gamma$)

**Initialize:** $\mu^{\mathcal{E}} \leftarrow \{\}; it \leftarrow 0$;
**repeat**
  $it \leftarrow it + 1$ ;
  $\{G^s, G^t\} \leftarrow$ NetworkPartitioning$(G)$ ; ▷ Random partitioning
  $\mu^s \leftarrow \mathcal{D}_{\text{ADV}}(G^s, U, p, \bar{x}^s, \frac{\Gamma}{2})$ ; ▷ Solve for first sub-network
  $\mu^t \leftarrow \mathcal{D}_{\text{ADV}}(G^t, U, p, \bar{x}^t, \frac{\Gamma}{2})$ ; ▷ Solve for second sub-network
  $\mu^{\mathcal{E}} \leftarrow \mu^{\mathcal{E}} \cup \mu^s \cup \mu^t$ ; ▷ Add the new set of potential edges to $\mu^{\mathcal{E}}$
**until** $(|\mu^{\mathcal{E}}| \geq \Delta) \vee (it \geq M)$;
$\mu \leftarrow \mathcal{D}_{\text{ADV}}$ $(\{G, \mu^{\mathcal{E}}\}, U, p, \bar{x}, \Gamma)$ ; ▷ Solve the problem over $\mu^{\mathcal{E}}$ edges
**return** $\mu$

## 6. Empirical results

In this section, we demonstrate the performance of our two-player iterative game approach on a set of synthetic and real-world data sets. We perform experiments on a 3.4 GHz Intel Core i7 machine with 16GB DDR3 RAM and all the linear optimization models are solved using IBM ILOG CPLEX optimization Studio V12.7.1. Section 6.1 provides the performance analysis of our optimal solution strategy from Section 4 on a wide range of small-scale problem instances. Section 6.2 presents the performance of our approach on a set of large-scale problem instances, where the adversary's decision problem in each iteration of the game is solved using heuristics from Section 5 for computational efficiency. We compare the performance of our approach against the following four well-known state-of-the-art benchmark approaches:

1. Max-flow min-cost solution (**MF**): In this approach, we assume that the administrator is not aware of any attacks and sends the optimal (i.e., a *max-flow min-cost* solution) flow through the network.
2. One-step planning (**OSP**): This is a myopic game model where the administrator first computes a *max-flow min-cost* solution, then the adversary identifies an optimal attack to disrupt that solution and finally, the administrator finds an optimal flow solution for the network which is damaged according to the attack revealed by the adversary. This one-stage game model is applicable to the setting where the administrator takes a myopic view and assumes that the attacker cannot observe the flow sent by the administrator. Therefore, from the administrator's perspective, the best policy for the adversary is to attack the initial *max-flow min-cost* solution which can be computed if the network structure and edge capacities are known to her.
3. Robust flow solution (**RF**): In this approach, we compute a robust flow solution where the entire flow of an attacked edge is assumed to be lost. Bertsimas et al. (2013) propose an optimization model for computing a robust flow solution by assuming that a maximum of $\Gamma_v$ incoming edges to node $v$ can fail. We modify

[5] It should be noted that due to our edge reconstruction, there might be multiple parallel edges from one node of the first sub-network to the artificial terminal node or from the artificial source node to another node of the second sub-network. In that case, we replace all the parallel edges having same source and destination node with a single edge whose flow and capacity values are computed as the sum of flows and capacities of all the parallel edges.

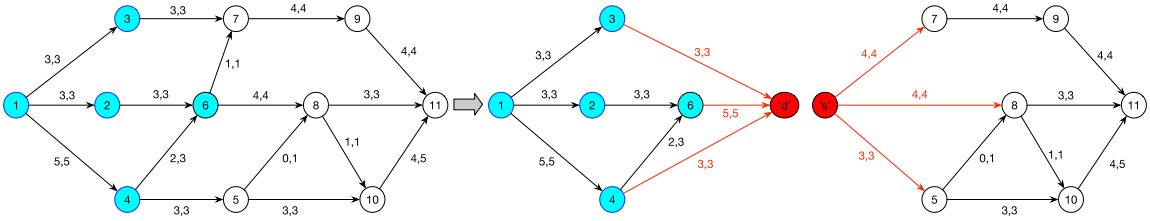


**Fig. 3.** Illustration of partitioning approach.

the optimization model to ensure that a maximum of $\Gamma$ edges in the network can be attacked. The details of the modified optimization model is provided in A.

4. Approximate adaptive maximum flow solution (**AAMF**): In this approach, our goal is to compute an adaptive maximum flow solution where the flow can be adjusted after the edge failure occurred. We employ a linear optimization model from Bertsimas et al. (2013) which provides an approximate solution to the adaptive maximum flow problem. The details of the approximate adaptive maximum flow solution is provided in B.

Finally, we refer to our iterative game approach as robust and adaptive maximum flow (**RAMF**) solution. To ensure fairness in comparison, we assume that the adversary always acts as a follower and optimally disrupts the resulting flow solution for all five approaches[6]. Since our goal is to optimize the network flow solution under adversarial conditions, we consider two crucial and complementary metrics for performance comparison:

- **Objective** – The objective value of the administrator, which we want to maximize, is computed as the difference between the amount of flow pushed to the terminal node and the total cost of routing and rerouting the flow through the network; and
- **Lost flow** – The amount of lost flow is computed as the difference between the amount of flow sent from the source node and the amount of flow reached to the terminal node. The amount of lost flow is an important indicator in many practical applications, e.g., these can correspond to congestion in case of urban transportation or these can introduce additional pressure in pipes in case of crude oil distribution application.

The trade-off between the objective value and the amount of lost flow can be captured using the routing cost parameter. If the routing cost is very high, then the optimal solution is to send zero flow through the network. On the other hand, if the routing cost is 0, then the optimal solution can have a large amount of lost flow. Therefore, we generate the routing costs randomly from a given range and report both the objective value and the amount of lost flow as performance metrics.

### 6.1. Empirical results on small-scale data sets

In this section, we provide the following key comparison results of our *RAMF* approach against four benchmark approaches on small problem instances:

1. Sensitivity results with respect to the amount of lost flow and the administrator's objective value on a set of synthetic networks by varying three tunable input parameters: (a) the number of nodes; (b) the edge density which controls the number of edges; and (c) the budget value of the adversary, $\Gamma$.
2. Runtime analysis and convergence results of our *RAMF* approach.
3. Performance of our proposed heuristic approaches against the optimal solution.
4. Behavioral insights from the strategies generated by the administrator and by the adversary.
5. Performance analysis on a real-world benchmark data set called SNDlib (Orlowski et al., 2010).

#### 6.1.1. Sensitivity results over different settings of input parameters

We now demonstrate the performance comparison on a set of synthetic problem instances. We generate random connected directed networks by varying the number of nodes and the edge density. The integer-valued edge capacities are drawn randomly from the range of 1 to 20 and the unit costs for transporting flows, $p$ are generated randomly from the range of 0.01 to 0.1. In the default setting of experiments, we use networks with 20 nodes and 0.8 edge density, and the adversary's budget value is set to 5. For each input setting, we generate 5 random problem instances and report the average amount of lost flow and the average percentage gains in the administrator's objective value. Let $U^{max}$ denote the maximum capacity of an edge (i.e., $U^{max} = \max_e U_e$)[7]. Then, the maximum loss in the objective value due to adversarial attacks is upper bounded by $(U^{max} \times \Gamma)$. So, the percentage gain of our approach against a benchmark approach (e.g., *MF*) is computed as the ratio between the difference in objectives and the upper bound on the marginal gain.

$$\%\text{Gain over } MF = \frac{(\text{Obj of } RAMF \text{ - Obj of } MF)\times 100}{U^{max} \times \Gamma} \tag{11}$$

Fig. 4(a) shows the net amount of lost flow due to attacks, where we vary the number of nodes in the $X$-axis. As expected, the amount of lost flow is significantly high for the *MF* approach, as it ignores the adversary's attacking behavior. The amount of lost flow for *RF* and *AAMF* are also significantly higher than our *RAMF* approach. The *OSP* approach is more conservative and therefore, the amount of lost flow for the *OSP* is relatively lower. The amount of lost flow for our *RAMF* approach is almost always lower than all four benchmark approaches. Fig. 4(b) delineates the percentage gain in the objective value of the administrator in a logarithmic scale. Our approach always outperforms all the benchmark approaches. The average percentage gains in the objective value for our approach over *MF*, *OSP*, *RF* and *AAMF* are 4.8%, 25.4%, 6.3% and 4.2%, respectively.

Fig. 4(c) shows the net amount of lost flow for all the approaches, where we vary the edge density from 0.4 to 0.8 in the $X$-axis. We observe a consistent pattern that the amount of lost flow for *MF*, *RF*, *AAMF* approaches are at least two times higher than the *RAMF* approach. While the amount of lost flow for the *OSP* approach is almost similar for edge density 0.4 to 0.6, our *RAMF* approach performs better when the number of edges increases. Fig. 4(d) delineates the percentage gain in the objective value in a logarithmic scale. The gap between objectives and the gain for our approach remain consistent in all the settings. As expected, *AAMF* always outperforms *RF* approach due to the flow adjustment. On an average, the percentage gains in objective

[6] For all the experiments, we set the modified capacity of an edge $e$, $m_e$ to 0 if the edge is attacked by the adversary.

[7] As the integer-valued edge capacities are drawn randomly from the range of 1 to 20, we set the value of $U^{max}$ to 20.

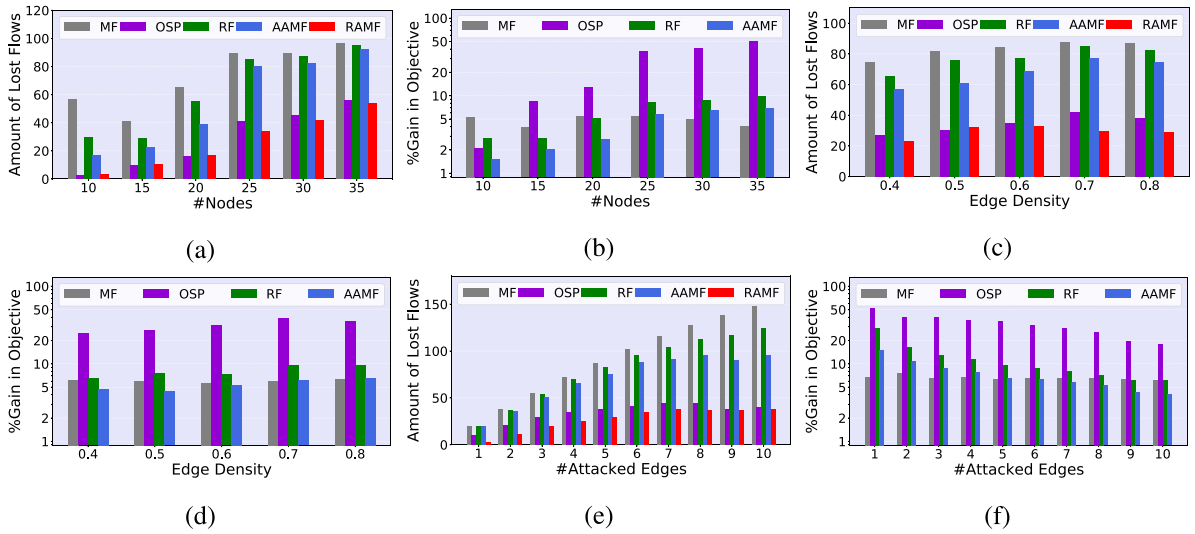


**Fig. 4.** Effect of number of nodes on (a) The amount of lost flow and (b) The objective value; Effect of number of edges on (c) The amount of lost flow and (d) The objective value; Effect of the value of $\Gamma$ on (e) The amount of lost flow and (f) The objective value.

for our approach over *MF*, *OSP*, *RF* and *AAMF* approaches are 6%, 31.8%, 8.1% and 5.4%, respectively.

Fig. 4(e) exhibits the net amount of lost flow for all the approaches, where we vary the adversary's budget value from 1 to 10 in the $X$-axis. The amount of lost flow for the *MF* approach increases monotonically from 20 to 150 as we increase the adversary's budget value, whereas the amount of lost flow for the *RAMF* approach is always bounded by 40. Moreover, we observe that the amount of lost flow for our *RAMF* approach remains consistent when the value of $\Gamma$ goes beyond 7. Such sensitivity analysis results can be used for initial estimation of the adversary's budget value. Fig. 4(f) demonstrates the percentage gains in the objective value for our *RAMF* approach. As the upper bound on marginal gain (i.e., $U^{max} \times \Gamma$) increases with the adversary's budget value, the percentage gains in objective value over *OSP*, *RF* and *AAMF* approaches reduce monotonically with the value of $\Gamma$. Although the percentage gain for our approach over the *MF* approach remains consistent for different values $\Gamma$, the net difference between the objective values increases monotonically. Therefore, these results clearly indicate that the performance of our approach improves gradually if the adversary becomes stronger.

### 6.1.2. Convergence results

Fig. 5(a) shows the convergence of our proposed iterative game on a problem with 35 nodes and edge density 0.4, where the adversary's budget is set to 5. The $X$-axis represents the iteration number of the game, and the $Y$-axis denotes the objective value obtained by both the players. As expected, the objective value for the administrator reduces monotonically over the iterations. As the administrator generates a conservative solution over the iterations, the adversary's ability to disrupt the flow strategy reduces. The objective values converge to 349 after 15 iterations. So, we can claim that the administrator's objective will at least be 349 (for any attack from $\Psi$) if the resulting robust flow strategy is executed. Fig. 5(b) delineates the objective values of the players in each iteration of the game on another problem instance with 20 nodes, edge density 0.8 and the adversary's budget value as 10, which converges after 14 iterations. We experimentally observe that the game converges within 20 iterations for all the other problem instances.

### 6.1.3. Runtime performance

As all the benchmark approaches provide solution from a single-step optimization model, the runtimes for them are always lower than our iterative *RAMF* approach. Therefore, we only demonstrate the runtime performance of our *RAMF* approach for different settings of network and the adversary's budget value. Fig. 5(c) presents the runtime for the *RAMF* approach in seconds in a logarithmic scale. As shown clearly, the runtime increases monotonically with the network size (in terms of both the number of nodes and the number of edges). Furthermore, the quadratic optimization problem for the adversary, $D_{\text{ADV}}$ becomes computationally expensive as we increase the value of $\Gamma$ and therefore, the runtime increases monotonically as the adversary becomes stronger.

### 6.1.4. Performance of heuristic approaches

To assess the performance of our two proposed heuristic approaches against the optimal solution on small-scale problem instances, we provide two sets of experimental results: (a) Optimality gap in the administrator's final flow solution quality if the adversary's decision in the intermediate stages of the game is solved using heuristic approaches; and (b) Optimality gap in the adversary's solution quality in each iteration of the game if her decision problem is solved with heuristic approaches as opposed to solving it optimally.

To understand the optimality gap in the administrator's final solution quality, we generate two flow strategies for the administrator—one by solving the adversary's decision problem using heuristic approaches in each iteration of the game and another by solving the adversary's decision problem optimally in each iteration. Then, we employ the optimal attack from Section 4.1 to disrupt both these final flow strategies and compare their performance. Fig. 6 demonstrates the net objective value and the lost flow comparison. On an average, the administrator's final objective value for our heuristic based solution is 6.8%, 9.7% and 9.4% far from the optimal solution for different settings of nodes, edge density and the adversary's budget value, respectively.

To assess the optimality gap in the adversary's solution quality, in each iteration of the game, we solve the adversary's decision problem both optimally as well as using heuristic approaches, and execute her optimal attack for the game play. Fig. 7 exhibits the average solution quality comparison between the attack generated using the heuristic approaches and the optimal attack in each iteration of the game. It should be noted that a lower objective value and higher amount of lost flow are better for the adversary. On an average, the adversary's objective values for the heuristic approaches are 3.2%, 4.5% and 5.1% far from the optimal solution for different settings of nodes, edge density and the adversary's budget value, respectively.

### 6.1.5. Behavioral insights from the strategies of the administrator and the adversary

We now provide behavioral insights generated from the strategies of the adversary and the administrator for different approaches. Fig. 8(a)

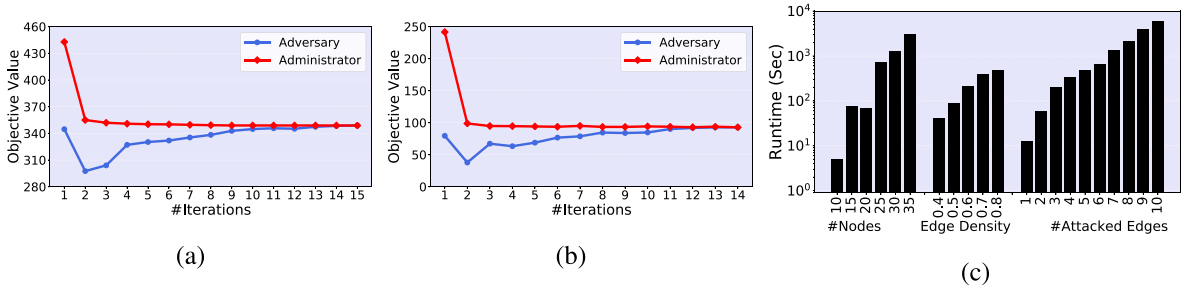


**Fig. 5.** Convergence of the proposed iterative game on problems with (a) 35 nodes and (b) $\Gamma$=10; (c) Runtime results of our *RAMF* approach for varying number of nodes, edges and the adversary's budget value.

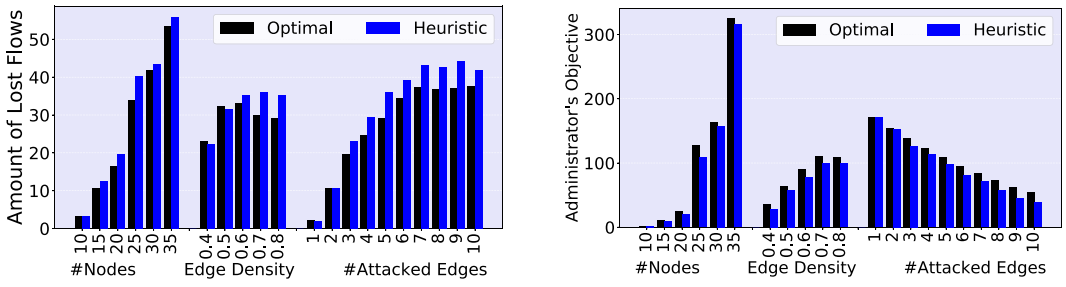


**Fig. 6.** Performance comparison between the proposed heuristic approaches and optimal solution on (a) The amount of lost flow; and (b) The administrator's objective value.

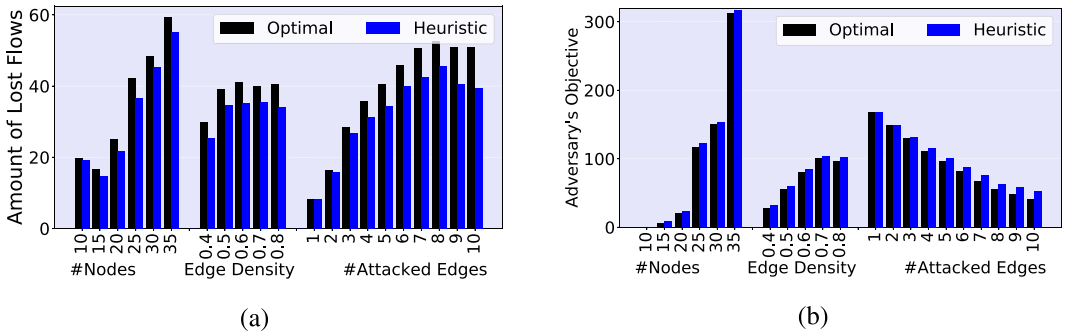


**Fig. 7.** Optimality gap for the heuristic approaches on (a) The amount of lost flow; and (b) The adversary's objective value.

demonstrates the average amount of initial flow assigned to edges that are being attacked by the adversary's strategy. We observe that *MF*, *RF* and *AAMF* solutions transmit higher amount of flow through critical high capacity edges, and the adversary can gain significantly by attacking those critical edges. The *OSP* solution assigns a lower amount of flow in all the edges that leads to a significantly lower objective value. Our *RAMF* solution is able to find intelligent flow strategies that transmit high amount of flow through the edges whose flow can be rerouted if attacked, and therefore, the adversary has more incentive in attacking edges with less flow than those important edges with higher flow. On an average, the amount of flow assigned to attacked edges by our *RAMF* approach is 60%, 8%, 56% and 50% less than the *MF*, *OSP*, *RF* and *AAMF* approaches, respectively. To further investigate the nature of the flow solutions generated by the administrator, we compute the residual capacity for the intermediate edges (i.e., except for the edges connected to source or terminal node) in which the administrator has assigned a portion of flow. Fig. 8(b) depicts the average residual capacity left by the administrator's initial flow solution on the intermediate edges. In contrast to other benchmark approaches, our *RAMF* solutions diversify the flow assigned to intermediate edges to maximize the flow reaching the terminal node under nominal condition and leave enough residual capacity to reroute flows in case of adversarial attacks.

### 6.1.6. Performance analysis on SNDlib data set

In this section, we provide performance comparison results on instances from Survivable Network Design Library [SNDLib] (Orlowski et al., 2010). SNDLib database consists of 26 problem instances. The capacities of edges, $U$ are directly taken from the database[8]. The unit edge costs for routing flows, $p$ are randomly drawn from the range of 0.01 to 0.1. In the default setting of experiments, we set the adversary's budget value to 5. However, due to small number of edges in some instances, we observe that the flow reaching to the terminal node is always 0 for all the approaches, if we allow the adversary to attack 5 edges. So, we reduce the adversary's budget value accordingly for those instances.

[8] For some instances, the capacities for all the edges are stated as 0 in the SNDlib database. For those instances, we set the capacities to randomly drawn integer values between 500 and 1000.

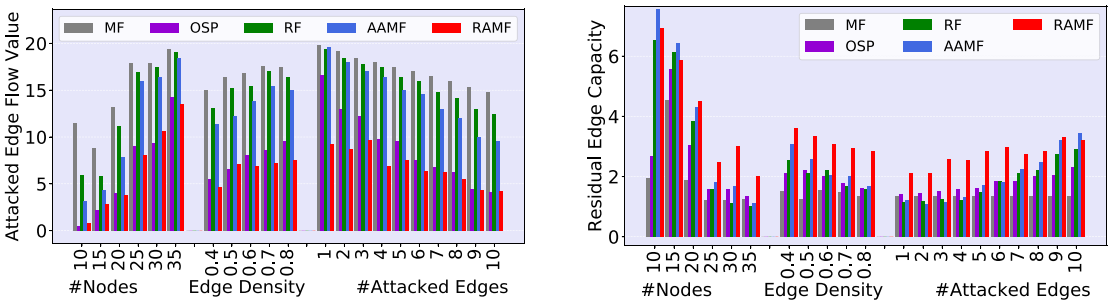


**Fig. 8.** Behavioral insights from solutions: (a) Amount of flow assigned to edges attacked by the adversary; and (b) Amount of residual capacity allocated to intermediate edges by the administrator's flow strategy.

**Table 2**
Empirical results on SNDlib data set.

| Problem instance | | | | Amount of lost flow | | | | | %Gain in objective of RAMF over | | | | Runtime (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\|\mathcal{V}\|$ | $\|\mathcal{E}\|$ | $\Gamma$ | MF | OSP | RF | AAMF | RAMF | MF | OSP | RF | AAMF | RAMF |
| abilene | 14 | 22 | 2 | 1828 | 1602 | 1796 | 1692 | 1602 | 2.2 | 65.4 | 1.9 | 0.9 | 0.2 |
| atlanta | 17 | 31 | 3 | 28711 | 5000 | 12000 | 12000 | 7966 | 3.9 | 5.8 | 1.8 | 1.3 | 3.6 |
| brain | 163 | 399 | 5 | 4512 | 4142 | 4512 | 4220 | 4142 | 1.3 | 67.6 | 9.7 | 2.9 | 65.7 |
| cost266 | 39 | 80 | 5 | 4507 | 3545 | 4314 | 4050 | 3859 | 2.9 | 55.5 | 2.2 | 2.4 | 11.0 |
| dfn-bwin | 12 | 50 | 5 | 3657 | 1155 | 3626 | 3060 | 118 | 8.4 | 19.8 | 10.2 | 8.5 | 2.6 |
| dfn-gwin | 13 | 54 | 5 | 3013 | 1361 | 2750 | 2795 | 979 | 8.8 | 37.7 | 10 | 8.4 | 2.0 |
| di-yuan | 13 | 49 | 5 | 4653 | 1869 | 4377 | 3850 | 1803 | 9.7 | 33.2 | 9.7 | 7.1 | 1.2 |
| france | 27 | 60 | 5 | 4502 | 2912 | 4236 | 4340 | 3755 | 2.5 | 42.7 | 2.5 | 3.1 | 1.3 |
| geant | 24 | 48 | 5 | 4633 | 3840 | 4467 | 4435 | 3984 | 1.8 | 64.3 | 3.5 | 2.7 | 2.3 |
| germany50 | 52 | 109 | 5 | 4338 | 3989 | 4148 | 4130 | 4046 | 2.9 | 66.1 | 8.6 | 6.7 | 6.6 |
| giul39 | 41 | 189 | 5 | 4710 | 1488 | 4447 | 4190 | 2118 | 10.5 | 24.3 | 33.7 | 13.0 | 3183 |
| india35 | 37 | 97 | 5 | 4489 | 3421 | 4290 | 4195 | 3737 | 2.4 | 57.1 | 5.5 | 4.1 | 5.4 |
| janos-us-ca | 41 | 139 | 5 | 4647 | 3093 | 4219 | 4310 | 3100 | 9.3 | 44.7 | 16.1 | 14.7 | 2443 |
| janos-us | 28 | 97 | 5 | 3891 | 1784 | 3205 | 3205 | 3057 | 3.7 | 26.1 | 9.8 | 3.4 | 7.8 |
| newyork | 18 | 58 | 5 | 4700 | 3106 | 4607 | 4470 | 3604 | 3.9 | 46.5 | 3.3 | 4.1 | 4.1 |
| nobel-eu | 30 | 55 | 5 | 4403 | 3367 | 4020 | 3690 | 3582 | 3.9 | 52.4 | 2.7 | 1.5 | 0.8 |
| nobel-germany | 19 | 39 | 5 | 4691 | 2612 | 4400 | 4400 | 3441 | 2.6 | 39.3 | 1.5 | 1.5 | 1.8 |
| nobel-us | 16 | 33 | 5 | 4828 | 3293 | 4726 | 4665 | 4621 | 0.8 | 50.9 | 0.5 | 0.3 | 1.3 |
| norway | 29 | 64 | 5 | 4100 | 2321 | 3817 | 3215 | 3028 | 4.2 | 37.6 | 7.7 | 1.9 | 5.6 |
| pdh | 13 | 41 | 5 | 4283 | 1422 | 4039 | 3645 | 2062 | 8.6 | 21.0 | 9.5 | 8.9 | 0.8 |
| pioro40 | 42 | 106 | 5 | 4283 | 1422 | 4039 | 3645 | 2062 | 1.5 | 57.2 | 6.4 | 2.2 | 15.1 |
| polska | 14 | 26 | 3 | 2695 | 1908 | 2672 | 2460 | 1837 | 3.9 | 54.1 | 7.1 | 6.7 | 0.3 |
| sun | 29 | 115 | 5 | 4552 | 2918 | 4324 | 3930 | 3692 | 4.0 | 45.3 | 17.6 | 5.8 | 53.2 |
| ta1 | 26 | 66 | 5 | 4552 | 2918 | 4324 | 3930 | 3692 | 10.6 | 33.9 | 14.5 | 11.7 | 80.7 |
| ta2 | 67 | 138 | 5 | 38464 | 15776 | 25200 | 15120 | 14023 | 9.3 | 17.1 | 5.4 | 1.3 | 8.0 |
| zib54 | 56 | 108 | 5 | 4786 | 4378 | 4671 | 4523 | 4483 | 1.3 | 70.3 | 3.5 | 2.9 | 11.7 |

Table 2 elaborates the comparison results on all instances of SNDlib data set. For each instance, we show the network details (i.e., the number of nodes and edges, and the adversary's budget value), the amount of lost flow for all five approaches and the runtime for our *RAMF* approach. We also provide the percentage gains in objective value for our approach over four benchmarks. For all the instances, the amount of lost flow for the *MF* approach is significantly higher than the *RAMF* approach. The *RAMF* approach always outperforms all benchmark approaches with respect to maximizing the administrator's objective value. On an average, our approach improves the objective value by 4.8%, 43.7%, 7.9% and 4.9% over *MF*, *OSP*, *RF* and *AAMF*, respectively. In addition, we observe that our approach is computationally attractive for these structured benchmark networks. The runtime for our approach is always bounded by 90 s except for two instances ('giul39' and 'janos-us-ca') for which the edge capacities are generated randomly as they are stated as 0 in the SNDlib database.

### 6.2. Empirical results on large-scale data sets

In this section, we present empirical results on a set of synthetic and real-world large-scale problem instances. For these large-scale problem instances, we employ heuristics from Section 5 to efficiently solve the adversary's decision problem. We demonstrate the following key performance results on large-scale data sets:

1. Solution quality comparison between our two proposed heuristics.
2. Sensitivity results on synthetic networks by varying three tunable input parameters.
3. Runtime analysis of our *RAMF* approach.
4. Performance analysis on a benchmark data set from RMFGEN (Goldfarb and Grigoriadis, 1988) networks.

#### 6.2.1. Performance comparison between our two proposed heuristic approaches

We begin by showing performance comparison between the accelerated greedy approach from Section 5.1 and the network partitioning based heuristic approach from Section 5.2. We empirically observe that the partitioning based heuristic mostly performs better than the greedy approach, especially at the later stage of the game. However, as both the approaches provide sub-optimal solutions, the greedy approach outperforms the partitioning based heuristic in some cases. To illustrate this behavior, we show the amount of lost flow, the net objective value and the runtime for both the heuristic approaches in each iteration of the game on a problem instance with 50 nodes, 0.4 edge density

**Table 3**
Solution quality and runtime comparison between two proposed heuristic approaches.

| Iteration | Amount of lost flow | | Objective value | | | Runtime (s) | |
|---|---|---|---|---|---|---|---|
| | Greedy | Partitioning | Greedy | Partitioning | Difference | Greedy | Partitioning |
| 1 | 466 | 447 | 1171.99 | 1190.91 | **−18.92** | 2.05 | 106.68 |
| 2 | 143 | 213 | 1071.76 | 1011.77 | 59.99 | 29.92 | 224.19 |
| 3 | 124 | 185 | 1096.4 | 1040.24 | 56.16 | 31.24 | 132.21 |
| 4 | 135 | 185 | 1118.11 | 1077.06 | 41.05 | 30.86 | 183.59 |
| 5 | 178 | 197 | 1117.71 | 1104.38 | 13.33 | 24.18 | 265.3 |
| 6 | 168 | 151 | 1157.89 | 1181.58 | **−23.69** | 25.66 | 684.01 |
| 7 | 187 | 182 | 1152.21 | 1158.04 | **−5.83** | 26.14 | 505.1 |
| 8 | 135 | 165 | 1206.67 | 1187.68 | 18.99 | 31.19 | 278.07 |
| 9 | 144 | 173 | 1201.79 | 1179.36 | 22.43 | 32.65 | 240.05 |
| 10 | 127 | 138 | 1214.84 | 1211.69 | 3.15 | 32.49 | 822.93 |
| 11 | 127 | 135 | 1214.81 | 1214.27 | 0.54 | 32.53 | 523.11 |

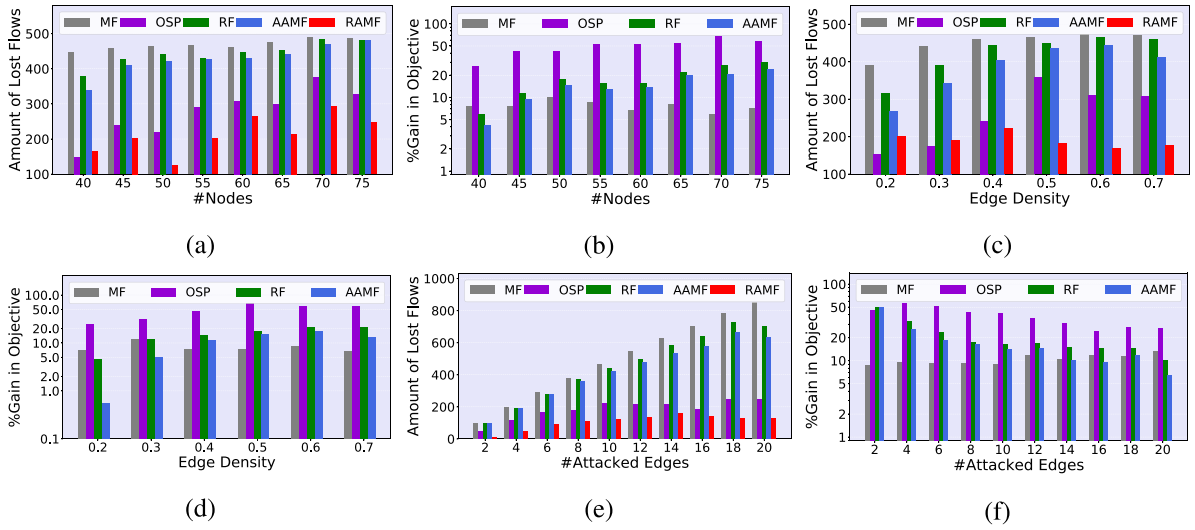


**Fig. 9.** Effect of number of nodes on (a) The amount of lost flow and (b) The objective value; Effect of number of edges on (c) The amount of lost flow and (d) The objective value; Effect of the value of $\Gamma$ on (e) The amount of lost flow and (f) The objective value.

and the adversary's budget as 10. As the adversary seeks to minimize the administrator's objective value, a better quality solution should provide lower objective value and higher amount of lost flow. As shown in Table 3, while the partitioning based heuristic mostly outperforms the greedy approach, the solution quality of the greedy approach is better in some iterations (e.g., 1, 6 and 7). As the partitioning based heuristic does not always dominate the greedy approach, we solve the adversary's decision problem using both heuristics in each iteration of the game and choose the attack with better solution quality. In terms of the computational complexity, as expected, the greedy approach is proven to be much faster than the network partitioning based heuristic.

### 6.2.2. *Sensitivity results with different settings of input parameters*

We now present empirical results on a set of large-scale synthetic problem instances. We use the same setting used in Section 6.1.1 to generate a set of large directed connected synthetic networks. For all the networks, we randomly draw integer-valued edge capacities from the range of 1 to 50. The unit costs for routing flows through edges are drawn randomly from the range of 0.01 to 0.1. In the default setting of experiments, we use networks with 50 nodes and 0.4 edge density, and the adversary's budget value is set to 10.

Fig. 9(a) demonstrates the net amount of lost flow for different approaches, where we vary the number of nodes from 40 to 75 in the $X$-axis. The amount of lost flow for the *MF* approach is significantly high in all the settings. The *RF* and *AAMF* approaches perform equally poorly in reducing the amount of lost flow. Except for relatively small problem instances with 40 nodes, our *RAMF* approach always outperforms all four benchmark approaches in terms of reducing the amount of lost flow. Fig. 9(b) delineates the percentage gains in the objective value for our *RAMF* approach against four benchmarks in a logarithmic scale. As clearly shown, the percentage gains in objective for our approach over all the benchmarks are always positive. The average percentage gains in the objective value for our approach over *MF*, *OSP*, *RF* and *AAMF* approaches are 7.7%, 50.1%, 18.2% and 14.97%, respectively.

Fig. 9(c) shows the net amount of lost flow for all five approaches, where we vary the edge density from 0.2 to 0.7 in the $X$-axis. The amount of lost flow for *MF*, *RF* and *AAMF* approaches are always significantly high. Except for edge density 0.2 and 0.3, our *RAMF* approach outperforms the *OSP* approach in terms of reducing the amount of lost flow. Fig. 9(d) delineates that the *AAMF* approach always provides a better quality solution over the *RF* approach. The *MF* approach outperforms the *RF* and *AAMF* approaches on larger problem instances with edge density 0.4 and beyond. As expected, the *OSP* approach performs poorly in maximizing the administrator's objective value in all the cases. On an average, the percentage gains in objective value for our *RAMF* approach over *MF*, *OSP*, *RF* and *AAMF* approaches are 8.2%, 47.3%, 15.2% and 10.5%, respectively.

Fig. 9(e) exhibits the net amount of lost flow for all the approaches, where we vary the budget of the adversary from 2 to 20 in the $X$-axis. The amount of lost flow for the *MF* approach increases monotonically from 100 to almost 900 as we increase the adversary's budget, whereas the amount of lost flow for our *RAMF* approach is always bounded by 160. Moreover, we observe that the amount of lost flow for our *RAMF* approach remains steady when the adversary's budget value goes beyond 14. Although Fig. 9(f) demonstrates that the percentage gains

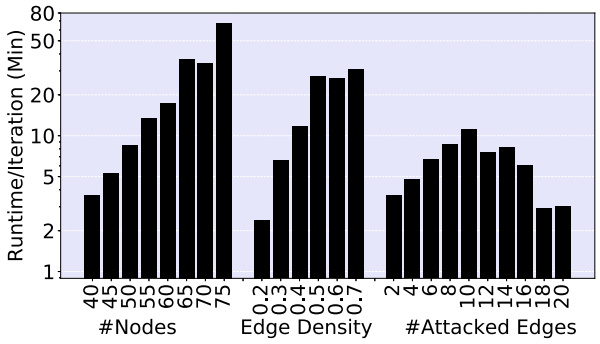


**Fig. 10.** Runtime results for large problem instances.

in objective value for our approach over the benchmark approaches almost always reduce monotonically with the increasing value of $\Gamma$, the net difference between the objective values increases monotonically with the value of $\Gamma$, which indicates that the performance of our approach improves gradually if the adversary becomes stronger. On an average, the percentage gains in the objective value for our *RAMF* approach over *MF*, *OSP*, *RF* and *AAMF* approaches are 10.4%, 38.1%, 21.1% and 17.6%, respectively.

In a nutshell, we observe a consistent pattern that our *RAMF* approach reduces the amount of lost flow significantly over *MF*, *RF* and *AAMF* approaches. On the other hand, the percentage gain in the objective value for the *RAMF* approach over the *OSP* approach is always significantly high. Therefore, we can conclude that among five approaches, only our *RAMF* approach is able to maintain the right trade-off between the two performance metrics. Moreover, the empirical results on large-scale problem instances replicate the similar trend observed for the optimal results on the small problem instances as presented in Section 6.1.1.

#### 6.2.3. Runtime performance

We now demonstrate the runtime performance of our *RAMF* approach for different settings of network size and the adversary's budget value. As the number of iterations required to converge the game may vary randomly for different problems, we show the average runtime in each iteration of the game.

Fig. 10 presents the runtime for the *RAMF* approach in minutes in a logarithmic scale. The runtime almost always increases monotonically with the network size (both in terms of the number of nodes and edges). The runtime increases monotonically with the adversary's budget value until $\Gamma = 10$. However, as the sub-problems of the network partitioning based heuristic approach have relatively small number of edges and the optimization problem tries to identify $\frac{\Gamma}{2}$ potential edges from a small set of edges, the combinatorial space of the sub-problems reduces once the budget value becomes large. Therefore, the runtime starts to decrease as the value of $\Gamma$ goes beyond 10.

#### 6.2.4. Performance analysis on RMFGEN data set

In the last thread of results, we provide performance comparison between different approaches on instances from RMFGEN networks (Goldfarb and Grigoriadis, 1988). RMFGEN networks are widely used for validating large-scale network flow solutions. We employ 7 moderately large RMFGEN networks for the experiments[9]. As some of the problem instances are undirected networks, we convert them into directed networks by randomly adding edges until it evolves into a connected (i.e., every node has at least one incoming and one outgoing edge) network. As the capacities of edges are mentioned as continuous values in the database, we generate random integer-valued edge capacities from the range of 10 to 50. The unit routing costs for the edges are randomly drawn from the range of 0.01 to 0.1. Finally, we set the adversary's budget value to 10 for all the problem instances.

Table 4 elaborates the performance comparison results on the instances of RMFGEN networks. For each instance, we show the network details (i.e., the number of nodes and edges, and the adversary's budget value), the amount of lost flow and the objective values for all five approaches, and the average runtime (in minutes) per iteration for the *RAMF* approach. The amount of lost flow is always significantly higher for *MF*, *RF* and *AAMF* approaches in comparison to our *RAMF* approach. For all the problem instances, the *RAMF* approach also provides higher objective value over all the benchmark approaches. On an average, our approach improves the objective value by 8%, 50.1%, 47.9% and 39.6% over *MF*, *OSP*, *RF* and *AAMF* approaches, respectively. Most importantly, we observe that our proposed heuristic approaches can scale gracefully to solve these large problem instances while providing a significant performance gain over the benchmark approaches.

## 7. Model extension: Full flow rerouting in the third stage

In this paper, we propose a solution methodology for the network flow games by assuming that the administrator can only reroute flows from an attacked edge through a forward walk. That is to say, the flow of an attacked edge $e := (u, v)$ can only be rerouted through edges that lie in one of the walks from the node $u$ to the terminal node. However, in some practical flow applications, the administrator can have the power to reroute additional flows through any active edges in the network once the attack has been realized. In this section, we demonstrate that our proposed solution methodology can easily be extended to tackle such situations where the administrator is allowed to reroute flows through all the active edges in the third stage of the game.

The main impediment in allowing full rerouting in the third stage of the game comes from the fact that one can imagine a trivial optimal solution to such a three-stage model that sends zero flow in the first stage, then the attacker reveals no attack (or a random poor attack) plan in the second stage and in the third stage, the administrator sends a *max-flow min-cost* solution. However, such a trivial solution does not serve our purpose, as we need to generate a first stage solution that performs well under nominal condition. Therefore, in the third stage solution, we need to enforce that flow arriving in the network from the source node should be upper bounded by the initial flow (i.e., first stage flow solution) sent from the source node.

To allow full rerouting of flows in the third stage, we begin by redefining the $\pi$-variables (refer to Eq. (8)), that control the upper bounds on the amount of allocated flows in the third stage, using Eq. (12). We set the value of $\pi_e$ to 0 only if the originating node $u$ of the edge $e$ is either the source node of the network, or it belongs to the set of source nodes $S$, in case the network has multiple source nodes. Specifically, if an edge is originated from the source node, then the third stage adjusted flow in that edge is upper bounded by the initial amount of allocated flow, otherwise the upper bound is set to the capacity of the edge $U_e$, by fixing the value of $\pi_e$ to 1.

$$\pi_e = \begin{cases} 0 & \text{if } e := (u, v) | u \in S \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

With this modified definition of the $\pi$-variables, we now present the decision problems of the administrator and the adversary, where the administrator can reroute flows through all the edges in the third stage of the game. To solve the administrator's decision problem, we can directly employ the optimization model (10a)–(10i), where the input $\pi$-variables are computed using Eq. (12), instead of Eq. (8). From the adversary's perspective, the full rerouting of flows further simplifies her decision problem. The optimization model (13a)–(13f), denoted by

[9] The data set is collected from http://elib.zib.de/pub/mp-testdata/maxflow/index.html

**Table 4**
Empirical results on RMFGEN data set.

| Problem instance | | | | Amount of lost flow | | | | | Objective value | | | | | Runtime/Iteration |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\lvert\mathcal{V}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\Gamma$ | MF | OSP | RF | AAMF | RAMF | MF | OSP | RF | AAMF | RAMF | RAMF |
| elist96 | 96 | 348 | 10 | 357 | 169 | 276 | 269 | 145 | 155.5 | 61.2 | 170.9 | 163.8 | 216.7 | 18.30 |
| elist96d | 96 | 539 | 10 | 439 | 343 | 378 | 376 | 213 | 452.9 | 307.0 | 323.6 | 365.4 | 508.2 | 11.26 |
| elist160 | 160 | 586 | 10 | 438 | 266 | 370 | 370 | 310 | 408.7 | 255.7 | 345.3 | 345.4 | 449.1 | 28.07 |
| elist160d | 160 | 933 | 10 | 481 | 377 | 456 | 454 | 345 | 1602.1 | 1380.1 | 1356.1 | 1467.8 | 1647.7 | 54.14 |
| elist200 | 200 | 818 | 10 | 463 | 344 | 408 | 405 | 352 | 969.3 | 737.2 | 840.8 | 904.8 | 1010.9 | 25.37 |
| elist200d | 200 | 1370 | 10 | 476 | 413 | 458 | 458 | 452 | 2255.4 | 1938.3 | 1823.5 | 1750.6 | 2272.8 | 35.64 |
| elist500 | 500 | 2042 | 10 | 477 | 422 | 453 | 440 | 430 | 3106.8 | 2799.6 | 2694.4 | 2848.2 | 3126.1 | 56.11 |

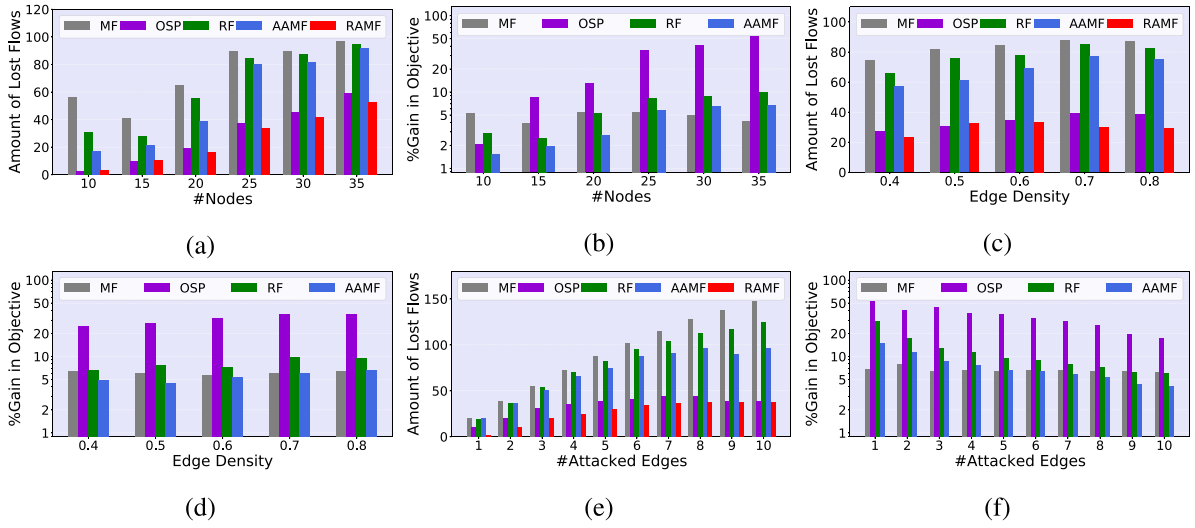


**Fig. 11.** Empirical results on small-scale data sets with full rerouting in the third stage of the game: Effect of number of nodes on (a) The amount of lost flow and (b) The objective value; Effect of number of edges on (c) The amount of lost flow and (d) The objective value; Effect of the value of $\Gamma$ on (e) The amount of lost flow and (f) The objective value.

$\mathcal{P}_{\text{ADV-F}}$, demonstrates the modified decision problem of the adversary. The only differentiating factor from the optimization model (6a)–(6h), presented in Section 4.1, is that instead of representing the $\pi$-variables as a function of the attack variables $\mu$, the $\pi$-variables are now given as inputs (computed using Eq. (12)) to the optimization model $\mathcal{P}_{\text{ADV-F}}$.

$$\mathcal{P}_{\text{ADV-F}} = \min_{\mu\in\Psi}\Big\{\max\ y_{(t,s)} - \sum_{e\in\mathcal{E}} p_e\cdot z_e\Big\} \tag{13a}$$

$$\text{s.t.} \sum_{e\in\delta_v^+} y_e - \sum_{e\in\delta_v^-} y_e \geq 0 \qquad \forall v\in\mathcal{V}\setminus\{s\} \tag{13b}$$

$$y_e \leq (1-\mu_e)U_e + \mu_e m_e \qquad \forall e\in\mathcal{E} \tag{13c}$$

$$z_e \geq y_e - \bar{x}_e \qquad \forall e\in\mathcal{E} \tag{13d}$$

$$y_e \leq (1-\pi_e)\bar{x}_e + \pi_e U_e \qquad \forall e\in\mathcal{E} \tag{13e}$$

$$y_e \geq 0; z_e \geq 0 \qquad \forall e\in\mathcal{E} \tag{13f}$$

As the inner maximization problem is a linear program, we employ the same dualization technique from Section 4.1 to convert the entire problem into a minimization problem. The quadratic dual problem, denoted by $\mathcal{D}_{\text{ADV-F}}$, is compactly shown using the optimization model (14a)–(14f), where $\alpha, \beta, \omega$ and $\gamma$ represent nonnegative dual price variables for constraints (13b)–(13e), respectively. Finally, we can employ the same heuristic approaches from Section 5 to solve the adversary's decision problem for large-scale networks. The only change required is to solve the third stage evaluation model in Table 1 using the modified definition of $\pi$-variables from Eq. (12).

$$\mathcal{D}_{\text{ADV-F}} = \min_{\alpha,\beta,\gamma,\omega,\mu} \sum_{e\in\mathcal{E}} \beta_e U_e - \sum_{e\in\mathcal{E}} \beta_e\mu_e(U_e - m_e) + \sum_{e\in\mathcal{E}} \omega_e\bar{x}_e + \sum_{e\in\mathcal{E}} \gamma_e\left[\bar{x}_e + \pi_e(U_e - \bar{x}_e)\right] \tag{14a}$$

$$\text{s.t. } \beta_e + \gamma_e + \omega_e + \alpha_v - \alpha_w \geq 0 \qquad \forall e := (v,w)\in\mathcal{E} \tag{14b}$$

$$\omega_e \leq p_e \qquad \forall e\in\mathcal{E} \tag{14c}$$

$$\sum_{e\in\mathcal{E}} \mu_e \leq \Gamma \tag{14d}$$

$$\alpha_t = 1; \alpha_s = 0 \tag{14e}$$

$$\alpha_v, \beta_e, \gamma_e, \omega_e \geq 0; \mu_e \in \{0,1\} \tag{14f}$$

To evaluate the performance of our proposed two-player game-based solution methodology in the setting where the administrator is allowed to reroute flows through all the edges, we carry out the same set of experiments from Section 6.1.1 on synthetic small-scale data sets. Fig. 11 demonstrates the performance of our proposed *RAMF* approach against four benchmark approaches with respect to the amount of lost flow and the administrator's objective value, for different settings of the number of nodes, edge density and the adversary's budget value. We observe a similar and consistent pattern as shown in Fig. 4 with the forward flow rerouting assumption. Our *RAMF* approach significantly reduces the amount of lost flow over all the benchmark approaches (albeit by a small amount over the *OSP* approach). In terms of the administrator's objective value, on an average over all the settings, the *RAMF* approach provides 5.8%, 30%, 8.7% and 5.7% gains over *MF*, *OSP*, *RF* and *AAMF* approaches, respectively. Furthermore, we observe that the full rerouting of flows has a limited impact on the percentage gains in the administrator's objective value in comparison to the experimental results presented in Section 6.1.1. Over all the different settings of network size and the adversary's budget value, the average percentage gains in the administrator's objective value for the *RAMF* approach with full rerouting deteriorate only by 0.032%, −0.033%, 0.05% and 0.029% over *MF*, *OSP*, *RF* and *AAMF* approaches, in comparison to the percentage gains with forward flow rerouting.

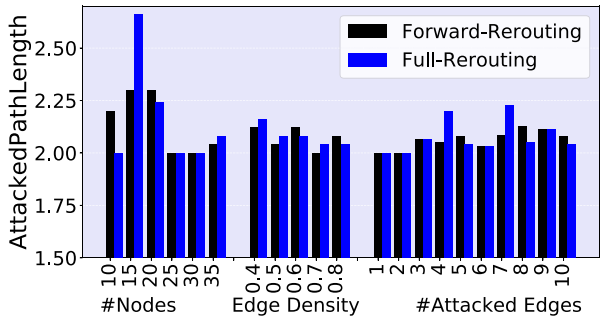


**Fig. 12.** Distances from the attacked edges to the terminal node—understanding the behavior of the adversary's solutions generated with full rerouting vs. forward path rerouting.

Lastly, we demonstrate the behavior of the solutions generated with full rerouting against the solutions presented in Section 6.1.1. If we allow rerouting of flows from an attacked edge in a forward path only, then intuitively the adversary has benefits in attacking edges that are closer to the terminal node. Therefore, in Fig. 12, we show the average shortest path length from the source node of an attacked edge to the terminal node, for both the forward path and full rerouting settings. The average shortest path length over all the problem instances for the forward path and full rerouting settings are 2.08 and 2.1, respectively. These results indicate that the adversary does not necessarily attack edges near to the terminal node in case of forward path rerouting, as the administrator might take the advantage by rerouting flows through shorter residual paths that incurs minimal rerouting cost. However, as expected, we observe that the adversary is prone to attack edges that disconnect the source node of an attacked edge to the terminal node in case of forward path rerouting, so as to disrupt the entire flow reaching to that source node. Over all the problem instances across different settings of nodes, edges and the adversary's budget value, the adversary attacks such edges 69 times in total for the forward path rerouting, as opposed to only 20 times in case of full rerouting.

## 8. Concluding remarks

To evaluate the resilience and sustainability of modern critical infrastructure networks, we propose a robust and adaptive network flow model by assuming that the network parameters are deterministic but the network structure (e.g., edges) is vulnerable to adversarial attacks or failures. To compute a robust and adaptive network flow strategy, we introduce a novel scenario generation approach based on a two-player iterative game between the network administrator and an adversary. In each iteration of the game, the adversary identifies an optimal attack to disrupt the flow strategy generated by the administrator in the current iteration, and the administrator computes a robust flow strategy by considering a set of attacks revealed by the adversary in previous iterations. As the computational complexity of the adversary's decision problem increases significantly with network size, we propose two novel heuristics – one leverages an accelerated greedy approach and the other employs a network partitioning based optimization approach – to speed up the solution process. The empirical results on multiple synthetic and real-world benchmark data sets demonstrate that our proposed approach scales gracefully to large-scale problem instances and improves the operational efficiency of the network by reducing the expected amount of lost flow.

In the future, this work can be extended in the following two directions: (a) Develop faster heuristics for solving the decision problem of both the adversary and the administrator, to scale up the solution process to massive real-world urban networks with tens of thousands of edges; and (b) Incorporate precise domain constraints in the optimization models of both the players to cater to specific real-world application domain. For example, in the context of urban transportation, a detailed traffic model with congestion effects would make the model more realistic. However, incorporating precise traffic details (e.g., representing the routing cost as a latency function of traffic to capture the congestion effects) would make our solution approach computationally intractable due to non-linearity in objective function and therefore, efficient approximation methods need to be designed by analyzing the properties of the specific application domain.

## CRediT authorship contribution statement

**Supriyo Ghosh:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Visualization. **Patrick Jaillet:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

## Acknowledgments

This work was partially supported by the Singapore National Research Foundation through the Singapore-MIT Alliance for Research and Technology (SMART) Centre for Future Urban Mobility (FM) and National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative. The authors thank the anonymous reviewers and the editor for their valuable comments and suggestions.

## Appendix A. Robust flow solution

In this section, we provide the details of the robust flow (RF) solution (Bertsimas et al., 2013) that is used as a benchmark approach. For this benchmark approach, our goal is to proactively compute a robust flow solution by assuming that the entire flow of an attacked edge is lost. For a fair comparison with other approaches, we modified the robust flow solution proposed by Bertsimas et al. (2013) to ensure that a maximum of $\Gamma$ edges in the network can be attacked. Let $x$ denote the resulting robust flow scenario and $\mathcal{L}_\Gamma$ denote the worst-case lost flow value if the adversary is restricted to attack a maximum of $\Gamma$ edges. The optimization model (A.1) compactly delineates the details of our robust flow solution, where the inner optimization model computes the value of $\mathcal{L}_\Gamma$ as the sum of the first $\Gamma$ biggest edge flow values.

$$\begin{aligned} &\max_{x} \; x_{(t,s)} - \mathcal{L}_\Gamma && \textbf{where}, \quad \mathcal{L}_\Gamma = \max_{\mu} \sum_e x_e \mu_e \\ &\text{s.t.} \sum_{e\in\delta_v^+} x_e - \sum_{e\in\delta_v^-} x_e = 0, \quad \forall v \in \mathcal{V}\setminus\{s\} && \text{s.t.} \sum_e \mu_e \le \Gamma \\ &\quad 0 \le x_e \le U_e, \quad \forall e \in \mathcal{E} && \quad 0 \le \mu_e \le 1, \quad \forall e \in \mathcal{E} \end{aligned} \tag{A.1}$$

The two components of the problem (A.1) can be combined by taking the dual of the inner problem $\mathcal{L}_\Gamma$. To achieve this goal, we first compute the *Lagrangian* function (A.2) by introducing the price variables $\theta$ and $\zeta$. From this *Lagrangian* function, we construct the dual problem (A.3).

$$\min_{\zeta,\theta} -\sum_e x_e\mu_e + \zeta\Big(\sum_e \mu_e - \Gamma\Big) + \sum_e \theta_e(\mu_e - 1) \tag{A.2}$$

$$\begin{aligned} \max_{\zeta,\theta} \quad & -\sum_e \theta_e - \zeta\Gamma \\ \text{s.t.} \quad & \theta_e + \zeta \ge x_e, \quad \forall e \in \mathcal{E} \\ & \theta_e \ge 0, \zeta \ge 0 \end{aligned} \tag{A.3}$$

Putting the optimization models (A.1) and (A.3) together, we construct the linear optimization model (A.4) that is used to solve the

robust maximum flow problem.

$$\begin{aligned}
\max_{x,\theta,\zeta} \quad & x_{(t,s)} - \sum_{e} \theta_e - \zeta \Gamma \\
\text{s.t.} \quad & \sum_{e\in\delta_v^+} x_e - \sum_{e\in\delta_v^-} x_e = 0, \qquad \forall v \in \mathcal{V}\setminus\{s\} \\
& \theta_e + \zeta \geq x_e, \qquad \forall e \in \mathcal{E} \\
& 0 \leq x_e \leq U_e, \qquad \forall e \in \mathcal{E} \\
& \theta_e \geq 0, \zeta \geq 0
\end{aligned} \tag{A.4}$$

## Appendix B. Approximate adaptive maximum flow solution

In this section, we provide the details of another benchmark approach that computes an approximate adaptive maximum flow (AAMF) solution by assuming that the flow can be adjusted after the edge failure occurred. Bertsimas et al. (2013) show that the adaptive maximum flow problem is strongly NP-hard. Therefore, they propose a scalable linear optimization model to approximately solve the problem. Let $x$ denote the resulting approximate adaptive maximum flow solution and $\theta$ denote the largest edge flow value. In addition, let us define an $s-t$ cut for the network as a subset $S \in \mathcal{V}$ of nodes with $s \in S$ and $t \in V \setminus S$. We say that a node $v$ is on the $s$-side if $v \in S$ and on the $t$-side if $v \in \mathcal{V} \setminus S$. Let $\delta^+(S)$ denote the set of directed outgoing edges $e := (v, w)$ from the $S$ side such that $v \in S$ and $w \in \mathcal{V} \setminus S$ and $\delta^-(S)$ represent the set of directed incoming edges $e := (v, w)$ to the $S$ side with $v \in \mathcal{V} \setminus S$ and $w \in S$. Then, the capacity of the $s-t$ cut is defined as: $Cap(S) = \sum_{e\in\delta^+(S)} U_e$. Let us assume that $S$ represents the $s-t$ cut with minimum capacity value for the network of interest (i.e., a min-cut solution). The linear optimization model (B.1) provides details of the approximate solution proposed in Bertsimas et al. (2013).

$$\begin{aligned}
\max_{x,\theta} \quad & \sum_{e\in\delta^+(S)} x_e - \sum_{e\in\delta^-(S)} x_e - \theta\Gamma \\
\text{s.t.} \quad & \sum_{e\in\delta_v^+} x_e - \sum_{e\in\delta_v^-} x_e = 0, \qquad \forall v \in \mathcal{V}\setminus\{s,t\} \\
& x_e \leq \theta, \qquad \forall e \in \mathcal{E} \\
& 0 \leq x_e \leq U_e, \qquad \forall e \in \mathcal{E}
\end{aligned} \tag{B.1}$$

Let $x^*$ be a flow with maximum robust flow value, such that $\beta \mathrm{Val}(x^*) \leq \mathrm{RVal}(x^*)$ for some $\beta \in (0, 1]$, where $\mathrm{Val}(x^*)$ represents the objective value for the administrator if no attack is executed in the network, and $\mathrm{RVal}(x^*)$ denotes the robust flow value of $x^*$. Further, suppose $\bar{x}, \bar{\theta}$ be the optimal solution for the optimization problem (B.1). Let $\rho$ denote the value of $(1-(1-\frac{1}{n})^n)$, where $n$ represents the number of nodes in the network. Then, Bertsimas et al. (2013) show that $\bar{x}$ is a $1-(((1-\rho)/\rho)((1-\beta)/\beta))$-approximation of $x^*$, i.e.,

$$\mathrm{RVal}(\bar{x}) \geq \Big(1 - \frac{1-\rho}{\rho}\frac{1-\beta}{\beta}\Big)\mathrm{RVal}(x^*)$$

In addition, Bertsimas et al. (2013) show that the resulting flow from problem (B.1) provides an $\alpha$-approximation to the optimal adaptive maximum flow solution. Let $x^*$ be an adaptive maximum flow such that $\beta \mathrm{Val}(x^*) \leq \mathrm{RVal}(x^*)$ for some $\beta \in (0, 1]$, and $\bar{x}, \bar{\theta}$ be the optimal solution for the optimization problem (B.1). Then, the adaptive value of $\bar{x}$, $\mathrm{AVal}(\bar{x})$ yields a $\beta(1-(((1-\rho)/\rho)((1-\beta)/\beta)))$-approximation for the adaptive value of $x^*$, i.e.,

$$\mathrm{AVal}(\bar{x}) \geq \beta\Big(1 - \frac{1-\rho}{\rho}\frac{1-\beta}{\beta}\Big)\mathrm{AVal}(x^*)$$

## References


Adulyasak, Yossiri, Jaillet, Patrick, 2015. Models and algorithms for stochastic and robust vehicle routing with deadlines. Transp. Sci. 50 (2), 608–626.

Ahmed, Asrar, Varakantham, Pradeep, Adulyasak, Yossiri, Jaillet, Patrick, 2013. Regret based robust solutions for uncertain Markov decision processes. In: Advances in Neural Information Processing Systems. pp. 881–889.

Ahuja, Ravindra K, Magnanti, Thomas L, Orlin, James B, 1993. Network Flows: Theory, Algorithms, and Applications. Prentice Hall.

Alderson, David L, Brown, Gerald G, Carlyle, W Matthew, 2015. Operational models of infrastructure resilience. Risk Anal. 35 (4), 562–586.

Alderson, David L, Brown, Gerald G, Carlyle, W Matthew, Cox Jr., Louis Anthony, 2013. Sometimes there is no "most-vital" arc: Assessing and improving the operational resilience of systems. Mil. Oper. Res. 18 (1), 21–37.

Alderson, David L, Brown, Gerald G, Carlyle, W Matthew, Wood, R Kevin, 2011. Solving defender-attacker-defender models for infrastructure defense. Tech. rep.. Naval Postgraduate School Monterey CA Dept Of Operations Research.

Altner, Douglas S, Ergun, Özlem, Uhan, Nelson A, 2010. The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. Oper. Res. Lett. 38 (1), 33–38.

Assadi, Sepehr, Emamjomeh-Zadeh, Ehsan, Norouzi-Fard, Ashkan, Yazdanbod, Sadra, Zarrabi-Zadeh, Hamid, 2014. The minimum vulnerability problem. Algorithmica 70 (4), 718–731.

Atamtürk, Alper, Zhang, Muhong, 2007. Two-stage robust network flow and design under demand uncertainty. Oper. Res. 55 (4), 662–673.

Ball, Michael O, Golden, Bruce L, Vohra, Rakesh V, 1989. Finding the most vital arcs in a network. Oper. Res. Lett. 8 (2), 73–76.

Baykal-Guersoy, Melike, Duan, Zhe, Poor, H Vincent, Garnaev, Andrey, 2014. Infrastructure security games. European J. Oper. Res. 239 (2), 469–478.

Ben-Tal, Aharon, Goryashko, Alexander, Guslitzer, Elana, Nemirovski, Arkadi, 2004. Adjustable robust solutions of uncertain linear programs. Math. Program. 99 (2), 351–376.

Ben-Tal, Aharon, Nemirovski, Arkadi, 1998. Robust convex optimization. Math. Oper. Res. 23 (4), 769–805.

Ben-Tal, Aharon, Nemirovski, Arkadi, 2000. Robust solutions of linear programming problems contaminated with uncertain data. Math. Program. 88 (3), 411–424.

Bertsimas, Dimitris, Brown, David B, Caramanis, Constantine, 2011. Theory and applications of robust optimization. SIAM Rev. 53 (3), 464–501.

Bertsimas, Dimitris, Goyal, Vineet, 2010. On the power of robust solutions in two-stage stochastic and adaptive optimization problems. Math. Oper. Res. 35 (2), 284–305.

Bertsimas, Dimitris, Gupta, Vishal, Kallus, Nathan, 2018. Data-driven robust optimization. Math. Program. 167 (2), 235–292.

Bertsimas, Dimitris, Nasrabadi, Ebrahim, Orlin, James B, 2016. On the power of randomization in network interdiction. Oper. Res. Lett. 44 (1), 114–120.

Bertsimas, Dimitris, Nasrabadi, Ebrahim, Stiller, Sebastian, 2013. Robust and adaptive network flows. Oper. Res. 61 (5), 1218–1242.

Bertsimas, Dimitris, Sim, Melvyn, 2003. Robust discrete optimization and network flows. Math. Program. 98 (1–3), 49–71.

Bertsimas, Dimitris, Sim, Melvyn, 2004a. The price of robustness. Oper. Res. 52 (1), 35–53.

Bertsimas, Dimitris, Sim, Melvyn, 2004b. Robust discrete optimization under ellipsoidal uncertainty sets. Tech. rep.. Citeseer, MIT.

Brown, Gerald G, Carlyle, W Matthew, Salmerón, Javier, Wood, R Kevin, 2006. Defending critical infrastructure. Interfaces 36 (6), 530–544.

Brown, Matthew, Saisubramanian, Sandhya, Varakantham, Pradeep Reddy, Tambe, Milind, 2014. Streets: Game-theoretic traffic patrolling with exploration and exploitation. In: Innovative Applications in Artificial Intelligence (IAAI). AAAI Press, pp. 2966–2971.

Cappanera, Paola, Scaparra, Maria Paola, 2011. Optimal allocation of protective resources in shortest-path networks. Transp. Sci. 45 (1), 64–80.

Cerrudo, Cesar, 2014. Hacking US (and UK, Australia, France, etc.) traffic control systems. IOActive (Apr, 2014). https://Ioactive.Com/Hacking-Us-and-Uk-Australia-France-Etc/.

Church, Richard L, Scaparra, Maria Paola, 2007. Protecting critical assets: The r-interdiction median problem with fortification. Geogr. Anal. 39 (2), 129–146.

Cormican, Kelly J, Morton, David P, Wood, R Kevin, 1998. Stochastic network interdiction. Oper. Res. 46 (2), 184–197.

Dahan, Mathieu, Amin, Saurabh, 2015. Network flow routing under strategic link disruptions. In: 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, pp. 353–360.

Dahan, Mathieu, Amin, Saurabh, Jaillet, Patrick, 2018. Probability distributions on partially ordered sets and network security games. arXiv preprint arXiv:1811.08516.

Dwivedi, Ajendra, Yu, Xinghuo, 2013. A maximum-flow-based complex network approach for power system vulnerability analysis. IEEE Trans. Ind. Inf. 9 (1), 81–88.

Dziubiński, Marcin, Goyal, Sanjeev, 2013. Network design and defence. Games Econom. Behav. 79, 30–43.

El Ghaoui, Laurent, Oustry, Francois, Lebret, Hervé, 1998. Robust solutions to uncertain semidefinite programs. SIAM J. Optim. 9 (1), 33–52.

Fang, Fei, Nguyen, Thanh Hong, Pickles, Rob, Lam, Wai Y, Clements, Gopalasamy R, An, Bo, Singh, Amandeep, Tambe, Milind, Lemieux, Andrew, et al., 2016. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In: National Conference on Artificial Intelligence (AAAI). pp. 3966–3973.

Ghena, Branden, Beyer, William, Hillaker, Allen, Pevarnek, Jonathan, Halderman, J Alex, 2014. Green lights forever: Analyzing the security of traffic infrastructure. WOOT 14, 7–7.

Ghosh, Supriyo, Koh, Jing Yu, Jaillet, Patrick, 2019. Improving customer satisfaction in bike sharing systems through dynamic repositioning. In: International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, pp. 5864–5870.

Ghosh, Supriyo, Trick, Michael, Varakantham, Pradeep, 2016. Robust repositioning to counter unpredictable demand in bike sharing systems. In: International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, pp. 3096–3102.

Goldfarb, Donald, Grigoriadis, Michael D, 1988. A computational comparison of the dinic and network simplex methods for maximum flow. Ann. Oper. Res. 13 (1), 81–123.

Gueye, Assane, Marbukh, Vladimir, 2012. A game-theoretic framework for network security vulnerability assessment and mitigation. In: International Conference on Decision and Game Theory for Security. Springer, pp. 186–200.

Gueye, Assane, Marbukh, Vladimir, Walrand, Jean C, 2012. Towards a metric for communication network vulnerability to attacks: A game theoretic approach. In: International Conference on Game Theory for Networks. Springer, pp. 259–274.

Guo, Qingyu, An, Bo, Zick, Yair, Miao, Chunyan, 2016. Optimal interdiction of illegal network flow. In: International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, pp. 2507–2513.

He, Fei, Zhuang, Jun, Rao, Nageswara SV, 2012. Game-theoretic analysis of attack and defense in cyber-physical network infrastructures. In: IIE Annual Conference. Institute of Industrial and Systems Engineers (IISE). Citeseer, p. 1.

Jacobs, Suzanne, 2014. Researchers hack into Michigan's traffic lights. MIT Technology Review (Aug, 2014). https://Www.Technologyreview.Com/S/530216/Researchers-Hack-Into-Michigans-Traffic-Lights/.

Jain, Manish, Kardes, Erim, Kiekintveld, Christopher, Ordónez, Fernando, Tambe, Milind, 2010. Security games with arbitrary schedules: A branch and price approach. In: National Conference on Artificial Intelligence (AAAI). pp. 792–797.

Jain, Manish, Korzhyk, Dmytro, Vaněk, Ondřej, Conitzer, Vincent, Pěchouček, Michal, Tambe, Milind, 2011. A double oracle algorithm for zero–sum security games on graphs. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 327–334.

Kar, Debarun, Nguyen, Thanh H, Fang, Fei, Brown, Matthew, Sinha, Arunesh, Tambe, Milind, Jiang, Albert Xin, 2017. Trends and applications in stackelberg security games. Handb. Dyn. Game Theory 1–47.

Laporte, Gilbert, Mesa, Juan A, Perea, Federico, 2010. A game theoretic framework for the robust railway transit network design problem. Transp. Res. B 44 (4), 447–459.

Li, Mian, Azarm, Shapour, 2008. Multiobjective collaborative robust optimization with interval uncertainty and interdisciplinary uncertainty propagation. J. Mech. Des. 130 (8).

Lozano, Leonardo, Smith, J. Cole, 2017. A backward sampling framework for interdiction problems with fortification. INFORMS J. Comput. 29 (1), 123–139.

Ma, Chris YT, Rao, Nageswara SV, Yau, David KY, 2011. A game theoretic study of attack and defense in cyber-physical systems. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 708–713.

Manshaei, Mohammad Hossein, Zhu, Quanyan, Alpcan, Tansu, Bacşar, Tamer, Hubaux, Jean-Pierre, 2013. Game theory meets network security and privacy. ACM Comput. Surv. 45 (3), 25.

Mattia, Sara, 2013. The robust network loading problem with dynamic routing. Comput. Optim. Appl. 54 (3), 619–643.

McMahan, H Brendan, Gordon, Geoffrey J, Blum, Avrim, 2003. Planning in the presence of cost functions controlled by an adversary. In: International Conference on Machine Learning (ICML). pp. 536–543.

Minoux, Michel, 1978. Accelerated greedy algorithms for maximizing submodular set functions. In: Optimization Techniques. Springer, pp. 234–243.

Orlin, James B, 1997. A polynomial time primal network simplex algorithm for minimum cost flows. Math. Program. 78 (2), 109–129.

Orlowski, Sebastian, Wessäly, Roland, Pióro, Michal, Tomaszewski, Artur, 2010. Sndlib 1.0 – Survivable network design library. Networks 55 (3), 276–286.

Pita, James, Jain, Manish, Marecki, Janusz, Ordóñez, Fernando, Portway, Christopher, Tambe, Milind, Western, Craig, Paruchuri, Praveen, Kraus, Sarit, 2008. Deployed ARMOR protection: The application of a game theoretic model for security at the los angeles international airport. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 125–132.

Poss, Michael, Raack, Christian, 2013. Affine recourse for the robust network design problem: Between static and dynamic routing. Networks 61 (2), 180–198.

Ratliff, H Donald, Sicilia, G Thomas, Lubore, SH, 1975. Finding the n most vital links in flow networks. Manage. Sci. 21 (5), 531–539.

Reilly, Jack, Martin, Sébastien, Payer, Mathias, Bayen, Alexandre M, 2014. On cybersecurity of freeway control systems: Analysis of coordinated ramp metering attacks. Transp. Res., Part B.

Scaparra, Maria P, Church, Richard L, 2008a. A bilevel mixed-integer program for critical infrastructure protection planning. Comput. Oper. Res. 35 (6), 1905–1923.

Scaparra, Maria P, Church, Richard L, 2008b. An exact solution approach for the interdiction median problem with fortification. European J. Oper. Res. 189 (1), 76–92.

Smith, J. Cole, Lim, Churlzu, Sudargho, Fransisca, 2007. Survivable network design under optimal and heuristic interdiction scenarios. J. Global Optim. 38 (2), 181–199.

Sullivan, Kelly M, Smith, J. Cole, 2014. Exact algorithms for solving a euclidean maximum flow network interdiction problem. Networks 64 (2), 109–124.

Szeto, WY, 2013. Routing and scheduling hazardous material shipments: Nash game approach. Transportmetrica B: Transp. Dyn. 1 (3), 237–260.

Tsai, Jason, Yin, Zhengyu, Kwak, Jun-young, Kempe, David, Kiekintveld, Christopher, Tambe, Milind, 2010. Urban security: Game-theoretic resource allocation in networked physical domains. In: National Conference on Artificial Intelligence (AAAI). pp. 881–886.

Vorobyov, Sergiy A, Gershman, Alex B, Luo, Zhi-Quan, 2003. Robust adaptive beamforming using worst-case performance optimization: A solution to the signal mismatch problem. IEEE Trans. Signal Process. 51 (2), 313–324.

Washburn, Alan, Wood, R Kevin, 1995. Two-person zero-sum games for network interdiction. Oper. Res. 43 (2), 243–251.

Wollmer, Richard, 1964. Removing arcs from a network. Oper. Res. 12 (6), 934–940.

Wood, R Kevin, 1993. Deterministic network interdiction. Math. Comput. Modelling 17 (2), 1–18.

Wu, Manxi, Amin, Saurabh, 2018. Security of transportation networks: Modeling attacker-defender interaction. arXiv preprint arXiv:1804.00391.

Wu, Manxi, Jin, Li, Amin, Saurabh, Jaillet, Patrick, 2018. Signaling game-based misbehavior inspection in V2I-enabled highway operations. In: 57st IEEE Annual Conference on Decision and Control (CDC).

Zetter, Kim, 2012. Hackers breached railway network, disrupted service. https://Www.Wired.Com/2012/01/Railway-Hack/.

Zhang, Chao, Bucarey, Victor, Mukhopadhyay, Ayan, Sinha, Arunesh, Qian, Yundi, Vorobeychik, Yevgeniy, Tambe, Milind, 2016. Using abstractions to solve opportunistic crime security games at scale. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 196–204.