

An Iterative Security Game for Computing Robust and Adaptive Network Flows

Supriyo Ghosh[†] and Patrick Jaillet[‡]

[†]IBM Research AI, Singapore 018983*

[‡]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

Email: supriyog@ibm.com[†], jaillet@mit.edu[‡]

November 26, 2019

Abstract

The recent advancement in cyberphysical systems (e.g., transportation, water management, energy distribution system, etc.) has led to an exponential growth in the use of automated devices which in turn has created new security challenges. By manipulating cyberphysical components, a potential cyber/physical attacker can modify the capacities of multiple edges so as to disrupt the network of interest and reduce the amount of flow going through the network. In this paper, we study the robust and adaptive maximum flow problem in a network so as to proactively counter such adversarial failures. Existing robust network flow models typically assume that the entire flow of an attacked edge gets lost. However, in many practical systems, the flow of an attacked edge could potentially be rerouted through adjacent edges with residual capacity. In order to address this feature, we propose a robust and sustainable network flow model to effectively counter possible 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 response (out of some feasible attacking scenarios) 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 in order to generate a new conservative flow strategy that is robust (maximin) against those attacks. This iterative game continues until the objectives of the adversary and the administrator 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

*Most of this work was done while the author was with Singapore MIT Alliance for Research and Technology (SMART) Center for Future Mobility (FM).

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 lost flow over four state-of-the-art benchmark approaches.

1 Introduction

Cyberphysical systems (CPSs) are deployed extensively in modern critical infrastructure systems including urban transportation, water management, energy distribution, and telecommunication systems. For example, a wide range of sensors and automated devices are installed at the road intersections of urban transportation for real-time traffic monitoring and intelligent traffic light management. While the proliferation of automated devices in CPSs 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 cyberphysical attacks such as tampering of traffic monitoring sensors (Zetter 2012, Reilly et al. 2015, Cerrudo 2014), manipulation of signal controllers (Ghena et al. 2014, Jacobs 2014) have been reported in recent past. By manipulating cyberphysical components of an edge, a malicious attacker can either break the edge completely or modify its capacity. In the context of transportation, an attacker can either set the traffic light to red or reduce the duration of green light (which is equivalent to capacity drop) by manipulating the sensors embedded in streets that feed data to traffic control systems.

There can be two principle ways to counter these strategic malicious attacks and to develop resilient and sustainable cyberphysical systems: (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 traffic flow to maximize the amount of flow that can be pushed to the destination node under worst-case adversarial attack. The malicious attacks can also be countered using a combination of both the reactive and proactive approaches. This paper focuses on proactive and robust planning for resilient control of cyberphysical systems. In other words, we consider the network flow problem in the framework of robust optimization (Ben-Tal and Nemirovski 1998, Bertsimas and Sim 2003) as the network structure is itself uncertain due to the possibility of (adversarial) edge failures (Bertsimas et al. 2013).

Our goal is to solve the robust network flow problem where the network administrator proactively plans to route a maximum amount of flow through the network by considering all the possible attacking behaviors of an adversary operating under a budget constraint. The ro-

bust network flow problem is motivated by the classical network interdiction problem (Wood 1993, Cormican et al. 1998, Sullivan and Cole Smith 2014) where a felonious entity carries illegal goods through the network and an interdictor places resources (e.g., security personnel) to inspect a subset of edges so as to detect and prevent the illegal activity. Many sequential and simultaneous game-theoretical models have been proposed to solve network interdiction problems (Washburn and Wood 1995, Bertsimas et al. 2016, Dahan and Amin 2015). Guo et al. (2016) propose a repeated game to solve the network interdiction problem and provide a column generation method to solve the repeated Stackelberg game with minimax objective. Another similar problem domain which has been solved effectively using game-theoretic models is strategic network design for critical infrastructure systems where the goal is to optimize a certain utility function by considering the possibility of edge failure (Laporte et al. 2010, Dziubiński and Goyal 2013). However, different from the objectives in network interdiction problems or strategic network design problems for critical infrastructure, our goal is to find a robust and adaptive flow strategy whereby an administrator is able to maximize the effective flow that can be pushed through the network under worst-case adversarial attacks possibly through rerouting.

Another relevant research theme focuses on identifying critical and vulnerable edges in a network (Wu and Amin 2018, Wu et al. 2018) and designing simultaneous attacker-defender game. Dahan et al. (2018) propose a simultaneous attacker-defender network flow game and show that the simultaneous game has a mixed strategy Nash equilibrium if both the adversary and administrator are restricted to a fixed budget constraint. From the resulting mixed strategy, they further derive the properties of other Nash equilibria for that game. However, they did not consider the option to reroute flows from an attacked edge. The robust and adaptive network flow problem was introduced by Bertsimas et al. (2013) to tackle the situation where the network structure (i.e., nodes and edges) is itself uncertain. For computing an adaptive maximum flow solution, they assume that the flow can be adjusted after the arc failure occurred, but the adjusted flow is always bounded by the initial flow assigned to an edge. They propose a linear optimization model to approximately solve the 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. In addition, to make the problem more realistic, we assume that the administrator faces a fixed cost for routing one unit of flow 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 two-stage robust model. 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. For the administrator, a feasible flow strategy should satisfy the flow conservation constraint at the nodes and the capacity constraint at the edges. As the strategy space of the administrator is extremely large and the attacker's strategy space grows exponentially with the budget, it is impossible to compute an equilibrium solution by considering all the pure strategies. To tackle such large strategy space, a common practice is to use an incremental strategy generation approach for choosing a small set of pure strategies which can be used to identify an equilibrium solution. Jain et al. (2011) propose a double oracle algorithm where both the players use best response against each other to incrementally generate their strategies and show that this iterative approach converges to a Stackelberg equilibrium. The final solution at the convergence is a mixed strategy (i.e., probabilistic distribution over the incrementally generated pure strategies) for both the player, which can readily be applied in a physical security scheduling (e.g., security patrolling in airport or physical road network) domain as both attacker and defender execute their actions simultaneously. In contrast, we assume that the adversary is more powerful and has a perfect knowledge of the pure flow strategy chosen by the administrator. Therefore, our goal is to identify a reliable pure strategy for the administrator that maximizes the worst-case adaptive flow value.

In order to solve the iterative two-player game and to identify a reliable pure strategy for the administrator, we propose a novel incremental strategy generation approach. 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 players 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 solution approaches for traditional Stackelberg games (e.g., double oracle algorithm) that converge to a mixed strategy equilibrium, our solution converges to a pure and reliable robust flow strategy for the administrator.

As the administrator model adds additional constraints related to current attack on top of decision model from previous iteration, the objective 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 of the adversary increases. These two objective values are guaranteed to converge at some point. We show that the game converges to a maximin optimal flow solution for the administrator and therefore, the objective of the administrator will be lower bounded by the value at which the game has converged for any realization of the feasible attacking scenarios.

The assumption of budget constraint for the adversary is valid in many cyberphysical attacking scenarios 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 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 learnt 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).

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 to minimize the traffic flow reaching the destination 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 solve both the 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 uses a myopic one-step reasoning against the adversary’s behavior to choose a flow strategy; (iii) A robust maximum flow solution from Bertsimas et al. (2013) where the administrator proactively computes a flow solution to improve the worst-case performance; and (iv) An approximate adaptive maximum flow solution adopted 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 urban 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 as Traffic Network Flow Game (**TNFG**).

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 the administrator in the current iteration and generates an attacking strategy (out of the feasible attacking scenarios under a given budget constraint) to optimally disrupt the 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 candidate edges in the network and then we solve 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 the benchmark approaches.

Structure of the paper. In §2, we elaborate on the relevant research. In §3, we formally describe our problem by allowing the flow of an attacked edge to be rerouted through adjacent edges with residual capacity. In §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 §4.1 and §4.2, respectively. In §4.3, we provide the key iterative steps of our overall two-player game. In §5, we present experimental setup and results to verify the utility of our proposed solution approach on small-scale problem instances. In §6, we describe two heuristic approaches to quickly solve the adversary’s decision problem. In §6.1, we explain the accelerated greedy heuristic and in §6.2, we describe the network partitioning based heuristic. In §7, we demonstrate the experimental results on large-scale problem instances by leveraging the computational effectiveness of the proposed heuristics. Finally, we provide concluding remarks in §8.

2 Related Work

Given the practical importance of ensuring cyberphysical security in critical urban infrastructure systems and evaluating the resilience and sustainability of such systems, several game-theoretic models have been proposed to model the attacker-defender interactions (Manshaei et al. 2013). To situate these contributions, we summarize along the following four threads of

research: (a) Network interdiction and strategic network design; (b) Security games in critical cyberphysical infrastructure systems; (c) Stackelberg games for ensuring physical security; and (d) Robust optimization.

2.1 Network Interdiction and Strategic Network Design

The robust network flow problem is motivated by the network interdiction problem where a malicious entity carries illegal goods through the network and a security agency deploy 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) or stochastic (Cormican et al. 1998) network interdiction problems. Several research (Wollmer 1964, Ratliff et al. 1975, Ball et al. 1989) propose methodologies for identifying the most important edges in the network within a specific budget constraint which can recommend a deterministic interdiction strategy. For solving the network interdiction problem in Euclidean space, Sullivan and Cole Smith (2014) provide a sequence of integer programs to iteratively identify the upper and lower bounds on solution quality and show that these bounds are convergent. Many sequential and simultaneous game-theoretical models have been proposed recently to solve network interdiction problem. Washburn and Wood (1995) propose a two-player zero-sum game to compute a probabilistic edge selection strategy for the interdictors. Bertsimas et al. (2016) introduce an arc-based and a path-based formulation to solve the sequential network interdiction game where the interdictor iteratively chooses a pure strategy and then the other player routes a feasible flow through the network. Game-theoretic models are widely adopted to solve other relevant class of problems that exhibit similar characteristics to the network interdiction problem such as strategic network design (Laporte et al. 2010, Dziubiński and Goyal 2013) for 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. On the contrary to network interdiction or strategic network design problem, we focus on identifying a robust flow strategy to improve operational efficiency of cyberphysical systems where the network is vulnerable to strategic attacks. Bertsimas et al. (2013) propose a robust and adaptive network flow models to proactively counter edge failures by assuming that the flow can be adjusted after the edge failure occurred, but the adjusted flow is bounded by the initial flow assigned to an edge. In contrast, we propose a two-player iterative game to solve a generic problem by considering the fact that the flow of an attacked edge might be rerouted through adjacent edges with residual capacity.

2.2 Security Games in Critical Cyberphysical Infrastructure

Our work closely resembles the broader research theme of network security games in critical infrastructure systems. Security games have been employed to defend vulnerable nodes and

edges in urban applications including transportation (Baykal-Guersoy et al. 2014), shipment of hazardous material (Szeto 2013) and communication networks (Gueye et al. 2012). Gueye and Marbukh (2012) use a game-theoretic approach to compute the right trade-off between the vulnerability and the security cost for supply-demand networks. They model an operator who transmits a feasible flow through the network and an attacker, who faces a fixed cost for attacking an edge, seeks to maximize the amount of lost flow by disrupting an edge. They show that the game converges to a mixed-strategy Nash equilibrium. In contrast, we allow the adversary to disrupt multiple edges simultaneously within a fixed budget constraint and propose an iterative game to identify a robust flow strategy for the operators. Prior research papers (Assadi et al. 2014, Dwivedi and Yu 2013) have designed solution approaches that are built upon the *max-flow* and *min-cut* theorem for the analysis of vulnerability in complex urban networks. Several recent research works (Dahan et al. 2018, He et al. 2012, Ma et al. 2011) have also designed simultaneous attacker-defender network flow games and provide methodologies for identifying critical and vulnerable edges in the network (Wu and Amin 2018, Wu et al. 2018). Dahan and Amin (2015) and Dahan et al. (2018) employ the *max-flow* and *min-cut* theorem within a simultaneous game framework to model the attacker-defender interaction. The simultaneous game efficiently captures the strategic uncertainty on the adversary’s behavior, and the game has a mixed strategy Nash equilibrium if both the adversary and administrator are restricted to a fixed budget constraint. However, they did not consider the option to reroute flows through the adjacent edges with residual capacity. Therefore, we develop an efficient technique using an iterative game to identify a robust flow strategy for an urban network where the flow of an attacked edge can be rerouted through other paths.

2.3 Stackelberg Games in Physical Security

Another relevant area of research in security game has focused on identifying optimal Stackelberg strategies (Tsai et al. 2010, Kar et al. 2017). These leader-follower based sequential games have been applied successfully in many real-world 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) to cyber security (Sinha et al. 2015). The fundamental concept behind these problems resembles network interdiction problem where security personnel need to be allocated to defend adversarial events. However, solving these large normal-form Stackelberg games are practically challenging and therefore, sophisticated methods are needed to speed up the solution process by exploiting the specific domain structure. To identify a randomized patrolling strategy 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 and iteratively generates tighter bounds using fast and efficient algorithms. In the similar direction, Guo et al. (2016) develop 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 network security game. On the contrary to enumerating the entire exponential sized strategy space for the players, the proposed double oracle algorithm generates the oracle for the attacker and the defender by adding their pure strategy best response in each iteration and finally compute an equilibrium solution for the restricted game consisting of pure strategy oracles of both the players. The final solution at the convergence is a mixed strategy for both the player, which can be applied in a physical security scheduling domain as both the players execute their actions simultaneously. However, these approaches cannot be readily employed to solve our problem as we have exponential 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. In addition, we assume that the adversary is more powerful and has a perfect knowledge of the flow strategy chosen by the administrator and therefore, we need to identify a reliable and robust pure strategy for the administrator.

2.4 Robust Optimization

The last thread of research which is complementary to our work presented in this paper is on data-driven robust optimization. Robust optimization is a broad research area and the solution methodology varies for different problem settings (Bertsimas et al. 2011, 2018). Several research efforts have been made to design robust solutions 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 robust optimization solution also varies with different types of uncertainty sets such as ellipsoidal (Bertsimas and Sim 2004) and interval (Li and Azarm 2008) uncertainty sets. Our proposed iterative game and incremental strategy generation approach can be adopted for data-driven robust optimization with worst-case objective and interval uncertainty set.

3 Problem Formulation

We begin with a formal definition of the Traffic Network Flow Game (**TNFG**). Let $G = \langle \mathcal{V}, \mathcal{E} \rangle$ denote an urban network, where \mathcal{V} symbolizes the set of nodes and $s, t \in \mathcal{V}$ represent the source and terminal node, respectively. \mathcal{E} denotes 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 traffic 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. In addition, we introduce the following notations:

- δ_v^+ : The set of incoming edges to node v . For the source node s , δ_s^+ includes the artificial edge e_{ts} .

- δ_v^- : The set of outgoing edges from node v . For the terminal node t , δ_t^- includes the artificial edge e_{ts} .
- Λ_v : The set of all possible simple paths (i.e., without any internal cycles) from node v to terminal node t .
- $\Lambda_\sigma = \bigcup_{v \in \sigma} \Lambda_v$: The set of all possible simple paths from one of the node of set σ to terminal node, t .
- Γ : Budget for the adversary. That is to say, a maximum of Γ edges can be attacked.

A flow scenario x is a function $x : \mathcal{E} \rightarrow \mathbb{R}_{\geq 0}$ 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:

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

Let \mathcal{X} denote the set of all possible flow scenarios that satisfy the flow conservation constraints (1). An attacking scenario μ is a function $\mu : \mathcal{E} \rightarrow \{0, 1\}$, where μ_e is set to 1 if the edge $e \in \mathcal{E}$ is attacked by the adversary and 0 otherwise. The set of possible attacking scenarios that satisfy the budget constraint of the adversary is denoted by Ψ .

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

We now describe the traffic network flow game in the context of urban networks where the flow of an attacked edge might be rerouted through adjacent edges with residual capacity. In a fully secured network, the optimal strategy for the network administrator is to adopt one of the *Max-Flow* solutions (Ford and Fulkerson 1956). However, in case of an adversarial attack, the *Max-Flow* solution is not necessarily an optimal strategy. Therefore, our goal is to find a robust flow strategy for the network administrator against the worst-case attacking scenario from Ψ .

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 fixed cost for routing or rerouting one unit of flow through a particular edge. Let p_e denote the cost for routing or rerouting one unit of flow through edge $e \in \mathcal{E}$ and W denote the reward for successfully getting one unit of flow to the destination node. If the sum of initial routing cost and rerouting cost (due to attacks) for pushing one unit of flow to the destination node exceeds W , then an optimal strategy is to send zero flow through the network. To avoid such situation, we assume that the value of p_e is always upper bounded by $\frac{W}{2L}$, 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 attacking scenario μ , 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 (3)-(8), where y_e denotes the resulting amount of flow going through edge $e \in \mathcal{E}$ and z_e represents the amount of additional flow that is being rerouted through edge e due to the disruption from the attacking scenario μ . The objective function (3) 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 edges¹. It should be noted that, herein and throughout the rest of the paper we fix the value of W to 1 (i.e., $p_e \leq \frac{1}{2L}$).

$$M(x, \mu) = \max \left\{ W y_{(t,s)} - \sum_e p_e z_e \right\} - \sum_e p_e x_e \quad (3)$$

$$\text{s.t. } \sum_{e \in \delta_v^+} y_e - \sum_{e \in \delta_v^-} y_e \geq 0 \quad \forall v \in \mathcal{V} \setminus s \quad (4)$$

$$y_e = x_e \quad \forall e \notin \Lambda_{\sigma_\mu} \quad (5)$$

$$y_e \leq (1 - \mu_e) U_e + \mu_e m_e \quad \forall e \in \Lambda_{\sigma_\mu} \quad (6)$$

$$z_e \geq y_e - x_e \quad \forall e \in \Lambda_{\sigma_\mu} \quad (7)$$

$$y_e \geq 0; z_e \geq 0 \quad \forall e \in \mathcal{E} \quad (8)$$

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 (4) to avoid infeasibility issues. Let σ_μ denote the set of source nodes of the attacked edges for attacking scenario μ . As indicated earlier, Λ_{σ_μ} represents the set of all possible paths from any of the nodes of set σ_μ to the terminal node. If an edge e does not belong to any of these paths, then the flow through that edge will not be impacted and therefore, using constraints (5), the LO model ensures that the value of y_e is equal to the input flow value x_e . For all the other edges which belong to one of the paths in Λ_{σ_μ} , 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 is not attacked, then we can send maximum U_e amount of traffic flow through edge e . We combine these two possibilities and represent them using constraints (6). Finally, constraints (7)-(8) insure the amount of rerouted flow for edge e , $z_e = \max(0, y_e - x_e)$.

¹As the input flow scenario x is fixed, the initial flow routing cost (i.e., $\sum_e p_e x_e$) is constant in the objective function (3).

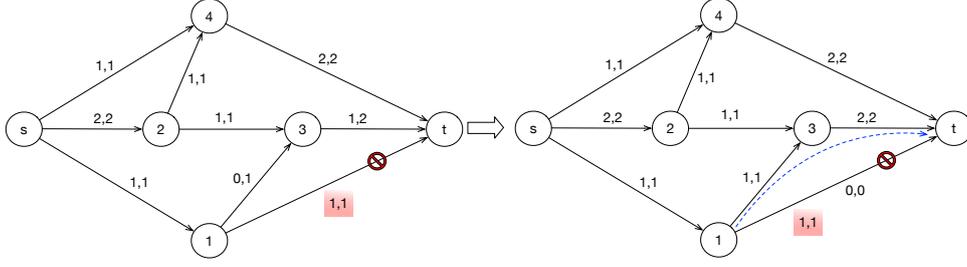


Figure 1: Illustration of resulting flow after an attack in a network.

Example 3.1 We provide an illustrative example in Figure 1 to better explain the constraints in the LO model. In this example, we have six nodes and nine edges. We show an initial flow solution x in the first network, where each pair of numbers represents the traffic flow and the capacity for that particular edge. Assume that the adversary can attack a maximum of one edge in the network. Assume the adversary attacks the 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 edge e_{13} and e_{3t} , the resulting flows for all the other edges remain the same. The resulting flow after the attack 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 attacking scenario that optimally disrupts the flow x . Therefore, for a given flow x , an optimal attacking scenario $\mu(x)$ and the objective of the adversary (also referred as the adaptive value of x) $AV(x)$ is defined as follows:

$$\begin{aligned} \mu(x) &\in \arg \min_{\mu \in \Psi} M(x, \mu) \\ AV(x) &= \min_{\mu \in \Psi} M(x, \mu) \end{aligned} \tag{9}$$

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

$$\max_{x \in \mathcal{X}} AV(x) = \max_{x \in \mathcal{X}} M(x, \mu(x)) \tag{10}$$

Lemma 1 *The problem of computing a robust and adaptive maximum flow strategy to be solved by the administrator from expression (10) is strongly NP-hard.*

Proof. The adaptive maximum flow problem introduced by Bertsimas et al. (2013) is a related problem where 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 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 §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 the amount of incoming flow to the destination node. Next, the adversary finds an optimal attacking scenario to disrupt the flow solution computed by the administrator. In the subsequent iterations, the administrator considers all the attacking scenarios revealed by the adversary in previous iterations and finds a robust (maximin) strategy against those attacks. The adversary always computes the 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. We now describe the details of the 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 (9) to generate an attacking scenario that optimally disrupt the flow \bar{x} by considering the fact that the administrator can reroute the flow of an attacked edge through other paths with residual capacity. We now provide an alternative formulation of the problem (9) so as to mathematically represent the notion of rerouting paths. The set of expressions (11)-(18) illustrates the alternative decision problem for the adversary. The objective function (11) corresponds to finding the attacking scenario μ that minimizes the resulting objective of the administrator, represented by the inner maximization problem. Constraints (12) ensure a weaker notion of the flow conservation at each node and constraints (13) enforce the capacity constraint on the flow at each edge. Let ρ_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 π_e denote a variable which is set to 0 if none of the edges in the set ρ_e is attacked. Then, constraints (14) enforce that the flow passing through an edge e is bounded by the given flow \bar{x}_e if none of the edges in ρ_e is attacked (i.e., $\pi_e = 0$). Constraints (15) and (17) assure that the value of π_e is set to 0 if none of edges in set ρ_e is attacked, and otherwise the value of π_e is set to 1. Finally, the additional flow rerouted through edge e , z_e can be computed

as $\max(0, y_e - \bar{x}_e)$. However, as the inner objective function minimizes the rerouting cost, constraints (16) are sufficient alone to accurately compute the value of z_e .

$$\min_{\mu \in \Psi} \left\{ \max_{e \in \mathcal{E}} y_{(t,s)} - \sum_{e \in \mathcal{E}} p_e \cdot z_e \right\} \quad (11)$$

$$\text{s.t. } \sum_{e \in \delta_v^+} y_e - \sum_{e \in \delta_v^-} y_e \geq 0 \quad \forall v \in \mathcal{V} \setminus s \quad (12)$$

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

$$y_e \leq (1 - \pi_e)\bar{x}_e + \pi_e \cdot U_e \quad \forall e \in \mathcal{E} \quad (14)$$

$$\pi_e \leq \sum_{e' \in \rho_e} \mu_{e'} \quad \forall e \in \mathcal{E} \quad (15)$$

$$z_e \geq y_e - \bar{x}_e \quad \forall e \in \mathcal{E} \quad (16)$$

$$\pi_e \in \{0, 1\} \quad \forall e \in \mathcal{E} \quad (17)$$

$$y_e \geq 0; z_e \geq 0 \quad \forall e \in \mathcal{E} \quad (18)$$

Observation 1 *The integrality constraints (17) can be relaxed to $0 \leq \pi_e \leq 1$, without compromising the feasibility or optimality of the optimization problem (11)-(18).*

Proof: As μ variables are binary, constraints (15) enforce that the value of π_e is set to 0 if none of the edges in set ρ_e is attacked. On the other hand, if some of the edges in set ρ_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 π_e is used by constraints (14) to only enforce an upper bound on the flow variable y_e , even with continuous relaxation, π_e will take a value of 1 to maximize the inner objective of expression (11). ■

Unfortunately, the adversary's optimization model cannot be solved directly as a linear program due to the minimax function in objective (11). However, as all the variables of the inner maximization problem are continuous, we can convert the entire problem into a minimization problem by taking dual of the inner problem. To compute the dual, we first obtain the Lagrangian function, L by relaxing constraints (12)-(17) using price variables $\alpha, \beta, \gamma, \delta, \omega$ and η , respectively.

$$\begin{aligned} L(\alpha, \beta, \gamma, \delta, \eta, \omega) = & -y_{(t,s)} + \sum_{e \in \mathcal{E}} p_e z_e + \sum_{v \in N \setminus s} \alpha_v \left[\sum_{e \in \delta_v^-} y_e - \sum_{e \in \delta_v^+} y_e \right] + \sum_e \delta_e \left[\pi_e - \sum_{e' \in \rho_e} \mu_{e'} \right] \\ & + \sum_{e \in \mathcal{E}} \gamma_e \left[y_e - \bar{x}_e - \pi_e (U_e - \bar{x}_e) \right] + \sum_e \eta_e \left[\pi_e - 1 \right] + \sum_e \omega_e \left[y_e - z_e - \bar{x}_e \right] \\ & + \sum_{e \in \mathcal{E}} \beta_e \left[y_e - U_e + \mu_e (U_e - m_e) \right] \end{aligned} \quad (19)$$

We can now write the optimization problem for the adversary by using the dual problem from the Lagrangian function L . Specifically, we set μ as variable in the dual problem and incorporate the domain constraints (24) of μ to represent the optimization problem of the adversary. The optimization problem, which is quadratic nature, is compactly shown in Table 1.

$\min_{\alpha, \beta, \gamma, \delta, \eta, \omega, \mu} \sum_e \beta_e U_e - \sum_e \beta_e \mu_e (U_e - m_e) + \sum_e \gamma_e \bar{x}_e + \sum_e \delta_e \sum_{e' \in \rho_e} \mu_{e'} + \sum_e \eta_e + \sum_e \omega_e \bar{x}_e$	(20)
$\text{s.t. } \beta_e + \gamma_e + \omega_e + \alpha_v - \alpha_w \geq 0$	$\forall e = (v, w)$ (21)
$\delta_e + \eta_e - \gamma_e (U_e - \bar{x}_e) \geq 0$	$\forall e \in \mathcal{E}$ (22)
$\omega_e \leq p_e$	$\forall e \in \mathcal{E}$ (23)
$\sum_e \mu_e \leq \Gamma$	(24)
$\alpha_t = 1; \alpha_s = 0$	(25)
$\alpha_v, \beta_e, \gamma_e, \delta_e, \eta_e, \omega_e \geq 0; \mu_e \in \{0, 1\}$	(26)

Table 1: ADVERSARYPROBLEM($G, U, \mathbf{p}, \mathbf{x}, \Gamma$)

The objective function (20) computes a point-wise minimum of linear functions of μ and therefore, concave in μ (see Boyd and Vandenberghe 2004). So, the optimization problem delineated in Table 1 is a concave quadratic program. 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 μ to continuous ones (see e.g., the proof of Lemma 6 of Bertsimas et al. 2013). However, if μ variables are binary, then the bilinear terms in the objective function (20) can be represented as explicit implication constraints and these constraints are efficiently implemented by existing mixed-integer optimization solvers like CPLEX or Gurobi.

4.2 Optimization Problem for the Administrator

The administrator's goal is to identify a robust flow strategy that maximizes the objective value in the worst-case attacking scenario as indicated in problem (10). However, as the strategy space of the adversary can be significantly large, we use an incremental approach and consider the attacking scenarios revealed by the adversary in previous iterations. 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.

$\max \lambda \tag{29}$	
$s.t. \lambda \leq \hat{y}_{(t,s)}^k - \sum_e p_e [\hat{x}_e + \hat{z}_e^k] \quad \forall k \in \mathcal{K} \tag{30}$	
$\hat{z}_e^k \geq \hat{y}_e^k - \hat{x}_e \quad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{31}$	
$\sum_{e \in \delta_v^+} \hat{x}_e - \sum_{e \in \delta_v^-} \hat{x}_e = 0 \quad \forall v \in \mathcal{V} \setminus s \tag{32}$	
$\hat{x}_e \leq U_e \quad \forall e \in \mathcal{E} \tag{33}$	
$\sum_{e \in \delta_v^+} \hat{y}_e^k - \sum_{e \in \delta_v^-} \hat{y}_e^k \geq 0 \quad \forall k \in \mathcal{K}, v \in \mathcal{V} \setminus s \tag{34}$	
$\hat{y}_e^k \leq (1 - \mu_e^k)U_e + \mu_e^k \cdot m_e \quad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{35}$	
$\hat{y}_e^k \leq (1 - \hat{\pi}_e^k)\hat{x}_e + \hat{\pi}_e^k \cdot U_e \quad \forall k \in \mathcal{K}, e \in \mathcal{E} \tag{36}$	
$\lambda \geq 0; \hat{x}_e \geq 0; \hat{y}_e^k \geq 0 \tag{37}$	

Table 2: ADMINISTRATORPROBLEM($G, U, \mathbf{p}, \boldsymbol{\mu}$)

Given a set of K attacking scenarios revealed by the adversary in previous K iterations, the administrator generates a flow strategy that maximizes the minimum amount of traffic flow reaching the destination over K attacking scenarios. Let $\boldsymbol{\mu}^k$ denote the attacking scenario k and $\hat{\mathbf{x}}$ denote the robust flow strategy against all the K attacking scenarios. Even though the initial flow decision $\hat{\mathbf{x}}$ is chosen, it may lead to different outcomes under different attacking scenarios and the actual amount of flow that reaches the destination will vary depending on the attacking scenario. So, we introduce $\hat{\mathbf{y}}^k$ variables to represent the actual flow and $\hat{\mathbf{z}}^k$ variables to denote the additional rerouted flow passing through the edges under the attacking scenario k .

As indicated in expression (3), the reward of the administrator is computed as the difference between the actual flow pushed to the destination and the accumulated cost for routing the initial flow and the rerouted flow. Therefore, the objective of the administrator is stated as follows:

$$\max_k \min \hat{y}_{(t,s)}^k - \sum_e p_e [\hat{x}_e + \hat{z}_e^k] \tag{27}$$

The objective function (27) can easily be linearized using expression (28), where we maximize the proxy variable λ that represents the worst-case objective value over K attacking scenarios.

$$\begin{aligned} &\max \lambda \\ &\mathbf{s.t.} \lambda \leq \hat{y}_{(t,s)}^k - \sum_e p_e [\hat{x}_e + \hat{z}_e^k] \quad \forall k \in K \end{aligned} \tag{28}$$

We show the entire linear optimization problem for the administrator compactly in Table 2. Constraints (31) ensure that the amount of flow being rerouted through edge e for attacking

scenario k , \hat{z}_e^k is equivalent to $\max(0, \hat{y}_e^k - \hat{x}_e)$. Constraints (32)-(33) enforce the flow preservation and capacity constraints on the flow variables \hat{x} . Constraints (34)-(35) assure the flow preservation and capacity constraints on the flow variables \hat{y}^k for attacking scenario k . $\hat{\pi}_e^k$ is a given input which is set to 0 if none of the edges in edge set ρ_e (from which the flow can be rerouted to edge e) is attacked under attacking scenario k and otherwise, it is fixed to 1:

$$\hat{\pi}_e^k = \min(1, \sum_{e' \in \rho_e} \mu_{e'}^k) \quad \forall k \in K, e \in \mathcal{E} \quad (38)$$

Therefore, constraints (36) enforce that the actual flow going through edge e under attacking scenario k is bounded by the initial flow \hat{x}_e if none of the edges in set ρ_e is attacked and otherwise, the upper bound is set to the capacity of the edge, U_e .

4.3 Overall Solution Approach

In this section, we put together the optimization problems delineated in §4.1 and §4.2. To better understand the overall framework of our proposed solution approach for solving the TNFG, 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 is the follower. In that sense, the adversary is more powerful and can take 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.

Let \mathbf{X} and μ store all the previously generated flow and attacking scenarios respectively, and both are initialized as empty set. μ_{old} keeps track of the attacking scenario from the last iteration and it is initialized to no attacking scenario. Let V^L and V^U denote the lower and upper bound of the game which are initialized to 0 and ∞ , respectively. In each iteration of the game, the administrator solves the optimization problem from Table 2 to compute the optimal robust flow strategy against the previously generated attacking scenario pool μ . Note that the administrator's optimization problem from Table 2 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{x} denote the set of optimal robust flow solutions computed by the administrator in the current iteration. We also update the upper bound of the game V^U to the objective value of the administrator. From these flow solutions, we identify the flows, x_c which are not executed before. If all the solutions of \tilde{x} are executed before, then the process terminates as any solution from \tilde{x} is a robust flow against any possible attack from Ψ . Otherwise, we randomly pick one flow x^* from x_c and execute it. We also add this new flow x^* into the pool of flow solutions \mathbf{X} .

Once the administrator generates a flow x^* and executes it, the adversary computes the best attacking scenario $\tilde{\mu}$ to optimally disrupt the flow x^* by solving the optimization model from Table 1. In addition, the lower bound of the game V^L is updated to the adversary's objective

Algorithm 1: SOLVETNFG(G, U, \mathbf{p}, Γ)

```
1 Initialize:  $\mathbf{X} \leftarrow \{\}, \boldsymbol{\mu} \leftarrow \{\}, \mu_{old} \leftarrow \mathbf{0}, V^L \leftarrow 0, V^U \leftarrow \infty$  ;
2 while ( $V^U \neq V^L$ ) do
3    $\tilde{\mathbf{x}}, V^U \leftarrow \text{ADMINSTRATORPROBLEM}(G, U, \mathbf{p}, \boldsymbol{\mu})$  ;  $\triangleright$  Robust flow solutions against
    $\boldsymbol{\mu}$ 
4    $\mathbf{x}_p \leftarrow \tilde{\mathbf{x}} \cap \mathbf{X}$  ;  $\triangleright \mathbf{x}_p$  contains previously executed flows
5    $\mathbf{x}_c \leftarrow \tilde{\mathbf{x}} \setminus \mathbf{x}_p$  ;  $\triangleright \mathbf{x}_c$  contains flows which are not executed yet
6   if  $|\mathbf{x}_c| = 0$  then
7     break ;  $\triangleright$  Process converges
8   else
9      $x^* \leftarrow \text{Random}(\mathbf{x}_c)$  ;  $\triangleright$  Randomly pick one flow from  $\mathbf{x}_c$  and execute it
10     $\mathbf{X} \leftarrow \mathbf{X} \cup x^*$  ;  $\triangleright$  Add flow  $x^*$  into flow strategy pool  $\mathbf{X}$ 
11     $\tilde{\boldsymbol{\mu}}, V^L \leftarrow \text{ADVERSARYPROBLEM}(G, U, \mathbf{p}, x^*, \Gamma)$  ;  $\triangleright$  Generate best attack against  $x^*$ 
12    if  $\tilde{\boldsymbol{\mu}} \neq \mu_{old}$  then
13       $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} \cup \tilde{\boldsymbol{\mu}}$  ;  $\triangleright$  Adversary executes  $\tilde{\boldsymbol{\mu}}$  and add it to attacking solution pool  $\boldsymbol{\mu}$ 
14       $\mu_{old} \leftarrow \tilde{\boldsymbol{\mu}}$  ;  $\triangleright$  Update the old attacking scenario
15    else
16      break ;  $\triangleright$  Process converges
17 return  $\tilde{\mathbf{x}}, \boldsymbol{\mu}$ 
```

value. If the adversary does not repeat the attacking scenario from previous iteration (i.e., μ_{old}), then we add the new attacking scenario $\tilde{\boldsymbol{\mu}}$ into the attacking scenario pool $\boldsymbol{\mu}$ and update μ_{old} to the new attacking scenario $\tilde{\boldsymbol{\mu}}$. On the contrary, if the adversary repeats the same attack from previous iteration, then the process converges as x^* is the robust flow scenario against $\tilde{\boldsymbol{\mu}}$. 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, they can come up with new strategies in subsequent iterations that lead to same objective value. To avoid such situation, we terminate the process if the lower and upper bound of the game converges. On the other hand, if the optimization problems of the players are solved sub-optimally (e.g., see §6 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 1 *The proposed iterative game converges to a maximin equilibrium and produces*

a robust maximum flow strategy for the administrator.

Proof. As the adversary has a finite number of attacking scenarios (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 $K - 1$ attacking scenarios computed by the adversary in previous iterations is denoted by μ . Let \tilde{x} represent the flow strategy of the administrator and $\tilde{\mu} \in \mu$ denote the attacking scenario of the adversary in the last iteration. As the adversary repeats one of her previous attacks in the final iteration, the objective of the adversary for the final iteration, $V^L = M(\tilde{x}, \tilde{\mu}) = \min_{\mu \in \mu} M(\tilde{x}, \mu)$, where $M(\tilde{x}, \tilde{\mu})$ represents the adaptive value of the flow \tilde{x} under attacking scenario $\tilde{\mu}$ which can be computed using the LO model (3)-(8). Similarly, as the administrator maximizes the minimum adaptive flow value over all the attacks in μ , the objective value for the final iteration is $V^U = \min_{\mu \in \mu} M(\tilde{x}, \mu) = V^L$.

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

If Algorithm 1 ends up considering all the attacking scenarios (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 attacking scenarios in Ψ . 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 principle way to identify those crucial attacking scenarios. In fact, for all the problem instances in our experiment, we observe that the proposed iterative game converges within 20 iterations.

5 Experimental Results on Small-scale Data Sets

To evaluate the performance of our two-player iterative game approach², we create a set of synthetic networks. We show the performance analysis by varying three tunable input parameters: (a) Number of nodes; (b) The edge density which controls the number of edges; and (c) Budget (i.e., maximum number of edges that can be attacked) of the adversary, Γ . Finally, we show the

²All the linear optimization models are solved using IBM ILOG CPLEX optimization Studio V12.7.1 on a 3.4 GHz Intel Core i7 machine with 16GB DDR3 RAM.

runtime performance of our approach for all the instances. We consider two key performance metrics for comparison:

1. **Lost flow** – The difference between the flow sent from the source node and the amount of flow reached to the terminal node; and
2. **Objective** – The objective value for 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.

We compare the performance of our approach against the following four well-known state-of-the-art benchmark approaches:

1. **Max-Flow solution (MF)**: In this approach, we assume that the administrator is not aware of any attacks and sends the optimal (i.e., *Max-Flow* solution) flow through the network.
2. **One step planning (OSP)**: This is a one-stage game model where the administrator first computes a *Max-Flow* solution, then the adversary identifies an optimal attack to disrupt the *Max-Flow* solution and finally, the administrator finds an optimal flow solution for the network which is damaged according to the attacking scenario revealed by the adversary. This one-stage game model is applicable to the settings 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* solution which can be computed if the network structure and edge capacities are known to her.
3. **Robust flow solution (RF)**: For 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 Γ_v incoming edges to node v can fail. We modify the optimization model to ensure that a maximum of Γ edges in the network can be attacked. The details of the modified optimization model is provided in Appendix A.
4. **Approximate adaptive maximum flow solution (AAMF)**: For this approach, our goal is to compute an adaptive maximum flow solution where the flow can be adjusted after the edge failure occurred. We adopt an optimization model from Bertsimas et al. (2013) which provides an approximate solution to the adaptive maximum flow problem. The details of this approach is provided in Appendix 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 the five approaches.

5.1 Sensitivity Analysis by Varying the Number of Nodes

We compare our approach against the benchmark approaches by varying the number of nodes in the network. Figure 2 demonstrates the comparison results, where we vary the number of nodes in the X-axis. The edge density for all the randomly generated directed graphs is fixed to 0.4 and the adversary budget is set to 5. The capacities of the edges are randomly drawn from the range of 1 to 20 and the unit edge costs for routing flows, p are generated randomly from the range of 0.01 to 0.1. For each problem category with a fixed number of nodes, we create 5 problem instances and report the average statistic over all the 5 instances. Figure 2(a) shows the net amount of lost flow due to attacks. As expected, the amount of lost flow is significantly high for the *MF* approach, as it ignores the adversary’s attacking behavior. The number of lost flow for *RF* and *AAMF* is also significantly higher than our *RAMF* approach. The *OSP* approach is more conservative and therefore, the lost flow for the *OSP* is always lower than other benchmarks. The lost flow for our *RAMF* approach is almost always lower than all the four benchmarks.

Figure 2(b) delineates the percentage gain in the objective value of the administrator in a logarithmic scale. Let U^{max} denote the maximum capacity of an edge (i.e., $U^{max} = \max_e U_e$)³. Then, the maximum loss in the objective value due to adversarial attacks is bounded by $(U^{max} \times \Gamma)$, where Γ is the budget for the adversary. So, the 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.

$$\% \text{Gain over } MF = \frac{(\text{Obj of } RAMF - \text{Obj of } MF) \times 100}{U^{max} \times \Gamma} \quad (39)$$

As shown in Figure 2(b), our approach always outperforms all the four 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.

5.2 Sensitivity Analysis by Varying the Number of Edges

We now demonstrate a sensitivity analysis by varying the number of edges in the network. Figure 3 presents the comparison results, where we vary the edge density from 0.4 to 0.8 in the X-axis. For these experiments, we consider a network with 20 nodes and the adversary budget is set to 5. We create 5 random instances for each of the settings and report the average statistic. Figure 3(a) shows the net amount of lost flow for all the approaches. We observe a consistent pattern that the lost flows for the *MF*, *RF*, *AAMF* approaches are at least two times higher than our approach in all the cases. While the lost flow for the *OSP* approach is almost

³As the capacities are drawn randomly from the range of 1 to 20, we set the value of U^{max} to 20.

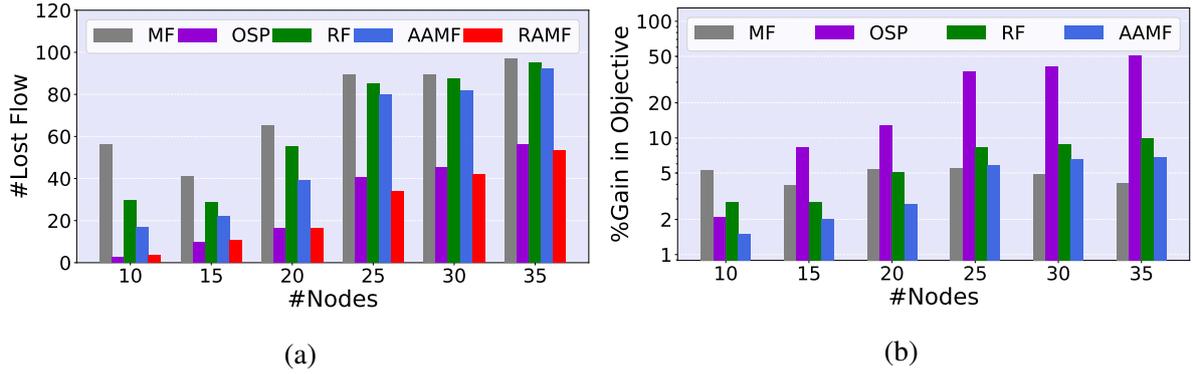


Figure 2: Effect of number of nodes on (a) Lost flow; (b) Objective.

similar for edge density 0.4 to 0.6, our *RAMF* approach performs better when the number of edges increases.

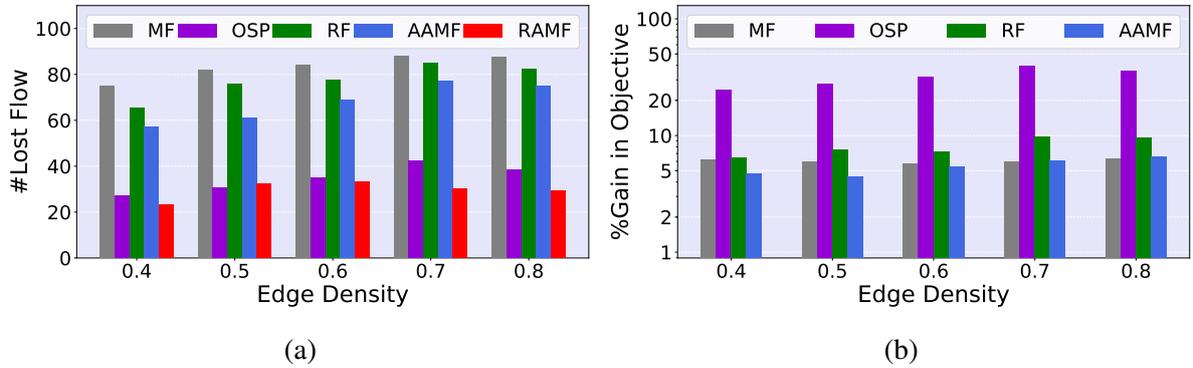


Figure 3: Effect of number of edges on (a) Lost flow; (b) Objective.

Figure 3(b) 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. The percentage gains in the objective value for our approach against all the benchmark approaches are always positive. On an average, the percentage gains in objective for our approach over the *MF*, *OSP*, *RF* and *AAMF* approaches are 6%, 31.8%, 8.1% and 5.4%, respectively.

5.3 Sensitivity Analysis by Varying the Adversary's Budget

In this thread of results, we present the effect of adversary's budget on both the performance metrics. Figure 4 demonstrates the comparison results, where we vary the adversary's budget from 1 to 10 in the X-axis. For these experiments, we generate five random instances of networks with 20 nodes and edge density 0.8. Figure 4(a) exhibits net amount of lost flow for all the approaches. The amount of lost flow for *MF* approach increases monotonically from 20 to 150 as we increase the adversary's budget, whereas the lost flow for the *RAMF* approach is

always bounded by 40. Moreover, we observe that the number of lost flow for our *RAMF* approach remains consistent when the value of Γ goes beyond 7. Such sensitivity analysis results can be used for initial estimation of the adversary’s budget value.

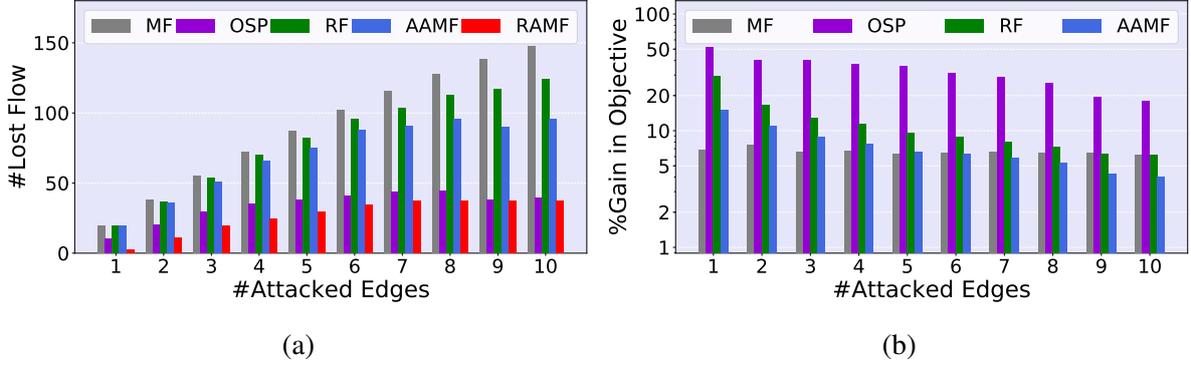


Figure 4: Effect of adversary’s budget on (a) Lost flow; (b) Objective.

Figure 4(b) demonstrates the percentage gains in the objective value for our approach over all the benchmarks in a logarithmic scale. As the upper bound on marginal gain (i.e., $U^{max} \times \Gamma$) increases with adversary’s budget, the percentage gain in objective over the *OSP*, *RF* and *AAMF* approaches reduces monotonically with the value of Γ . Although the gain of our approach over the *MF* approach remains consistent for different values Γ , the net difference between the objectives increases monotonically. Therefore, these results clearly indicate that the performance of our approach improves gradually if the adversary becomes stronger.

In a nutshell, we observe that the *RAMF* approach reduces the lost flow significantly over *MF*, *RF* and *AAMF* approaches. On the other hand, the percentage gains in the objective value for the *RAMF* approach over all the benchmarks are always positive and the gain is significantly high over the *OSP* approach. 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.

5.4 Convergence Results

Figure 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 the players. As expected, the objective for the administrator reduces monotonically over the iterations. As the administrator generates more conservative solution over the iterations, the adversary’s ability to disrupt the flow strategy reduces. The objectives converge to 349 after 15 iterations. So, we can claim that the administrator’s objective will at least be 349 (for any attacking scenario from Ψ) if the resulting robust flow strategy is adopted. Figure 5(b) delineates objectives of the players in each iteration of the game on another problem instance with 20 node, edge density

0.8 and adversary’s budget as 10, which converges after 14 iterations. For all the other problem instances in our experiment, we observe that the game converges within 20 iterations.

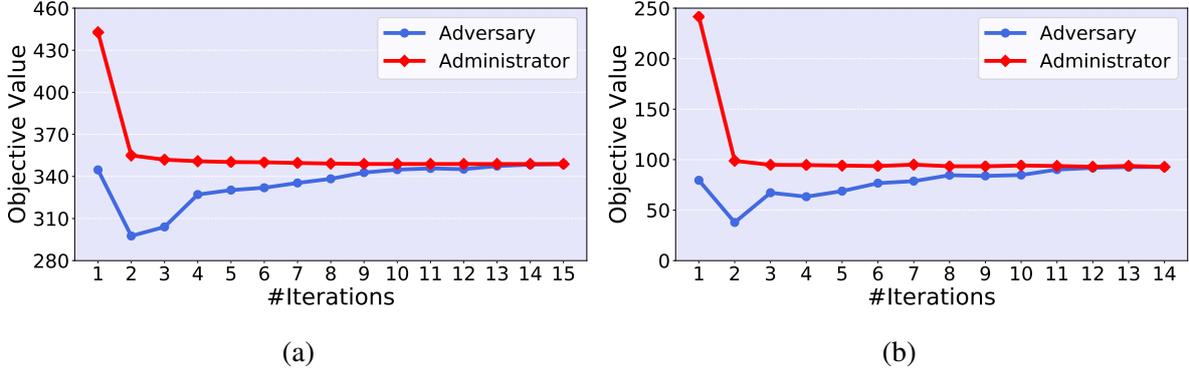


Figure 5: Convergence of the proposed iterative game on problems with (a) 35 nodes; and (b) $\Gamma=10$.

5.5 Runtime Performance

As all the benchmark approaches provide solution from single-step optimization model, the runtimes for them are always lower than our iterative *RAMF* approach. Therefore, in this thread of results, we only demonstrate the runtime performance of our *RAMF* approach for different settings of network and adversary’s budget. Figure 6 presents the runtime for the *RAMF* in seconds in a logarithmic scale. As shown clearly, the runtime increases monotonically as the size of the network (in terms of both the number of nodes and edges) increases. Furthermore, as expected, the quadratic optimization problem for the adversary from Table 1 becomes more computationally expensive as we increase the value of Γ and therefore, the runtime increases monotonically as the adversary becomes stronger. As the computational complexity of our approach increases gradually with all the three input tunable parameters, we only show the performance on small-scale problems in this section.

5.6 Results on SNDlib Data Set

In this section, we provide comparison results of our *RAMF* approach against four benchmarks on instances from the Survivable Network Design Library [SNDLib] (Orlowski et al. 2010). SNDLib database consists of 26 problem instances. The capacities of edges, U are directly adopted from the database⁴. The unit edge costs for routing flows, p are randomly drawn from the range of 0.01 to 0.1. In the default settings of our experiments, we set the adversary’s budget to 5. However, due to small number of edges in some instances, we observe that the

⁴For some instances, the capacities for all the edges are stated as 0 in the SNDlib database. For those instances, we set the capacities to a randomly drawn value between 500 to 1000.

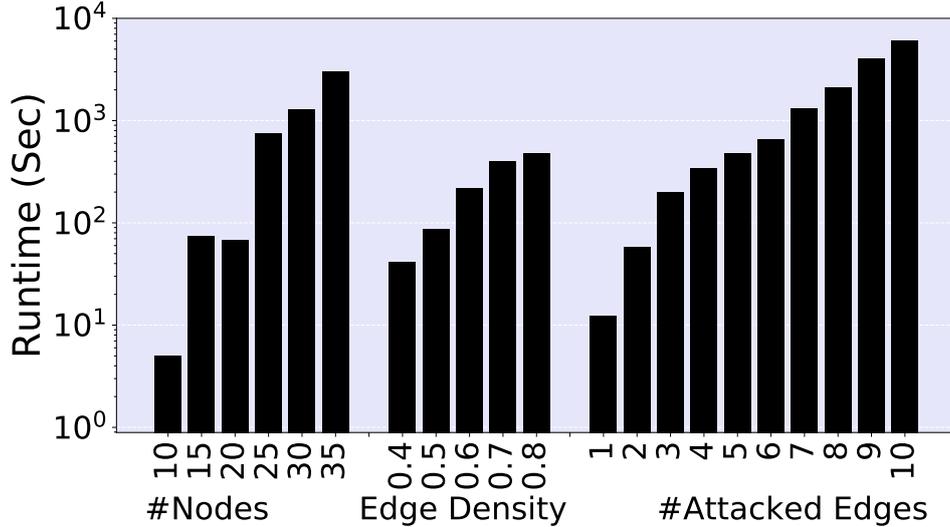


Figure 6: Runtime results for varying number of nodes, edges and adversary’s budget.

flow reaching the terminal node is always 0 for all the approaches, if we allow the adversary to manipulate 5 edges. So, we reduce the adversary’s budget accordingly for those instances.

Table 3 elaborates the comparison results on all the 26 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), the amount of lost flow for all the five approaches and the runtime for our *RAMF* approach. We also provide the percentage gains in objective value for our approach over the four benchmarks, which are computed using the maximum capacity and the adversary’s budget value. For all the instances, the amount of lost flow for the *MF* approach is significantly higher than the *RAMF* approach. The percentage gains in objective of the *RAMF* approach against the four benchmarks are always positive. 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 seconds except for two instances (i.e., ‘giul39’ and ‘janos-us-ca’) for which the edge capacities are generated randomly as they are stated as 0 in the SNDlib database.

6 Heuristics for Large-scale Networks

As indicated in §5.5, the runtime complexity of our *RAMF* approach increases with the network size and adversary’s budget value. The key reason behind this behavior is that we need to solve a complex quadratic program from Table 1 for the adversary’s decision problem in each iteration of the game. The adversary’s decision problem seeks to identify Γ best edges to attack so as to minimize the resulting objective 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

Problem Instance				Number of Lost Flow					%Gain in Objective of RAMF over				Runtime (sec)
Name	$ \mathcal{V} $	$ \mathcal{E} $	Γ	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 3: Empirical results on SNDlib data set.

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. However, for a given attacking scenario, 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 *min cost max flow* problem (in polynomial time) on a network whose edge capacities are set to the upper bounds. However, to compute an optimal attacking scenario, 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 the two heuristic approaches for solving the adversary's decision problem.

6.1 Greedy Approach

The greedy approach incrementally identifies the set of edges to be attacked by the adversary so as to disrupt the network for a given flow strategy. We begin with an alternative formulation of the optimization model (3)-(8) to evaluate the utility of an attacking scenario μ for a given flow scenario \bar{x} . The alternative linear optimization (LO) model is compactly shown in Table 4. The objective is to route maximum amount of flow to the terminal node while minimizing the total amount of rerouting cost. First two sets of constraints enforce the flow preservation and capacity constraints. The value of π is computed using equation (38) and is given as an input to the LO model. So, the third set of constraints ensure that if an edge e does not belong to any of the rerouted paths from the attacked edges, then the flow through the edge is bounded by the given flow value \bar{x}_e . The last set of constraints compute the amount of rerouted flow z_e through the edge e . It should be noted that the solution of the LO model can be obtained in polynomial time by solving a *min cost max flow* 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 .

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

Table 4: IDENTIFYFLOW($G, U, \mathbf{p}, \bar{\mathbf{x}}, \mu$)

Algorithm (2) provides the details of the greedy algorithm. We start with an empty attacked edge set μ . Let E denote an edge set that initially contains all the edges. We first compute the objective value O for the given administrator's 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 μ by employing the LO model from Table 4 and compute the marginal gain in the adversary's objective for attacking the edge e . Then we add the best edge e^* (that provides maximum marginal gain) into μ and remove it from potential edge set E . This process continues until the budget for the adversary is exhausted.

Algorithm 2: GREEDY($G = \langle \mathcal{V}, \mathcal{E} \rangle, U, \mathbf{p}, \bar{x}, \Gamma$)

```

1 Initialize:  $\mu \leftarrow \{\}; it \leftarrow 0; E \leftarrow \mathcal{E};$ 
2  $O \leftarrow \text{IDENTIFYFLOW}(G, U, \mathbf{p}, \bar{x}, \mu);$  ▷ Compute objective for the given flow  $\bar{x}$ 
3 repeat
4    $it \leftarrow it + 1;$ 
5    $g_e, \hat{x} \leftarrow \text{IDENTIFYFLOW}(G, U, \mathbf{p}, \bar{x}, \mu \cup \{e\}) \quad \forall e \in E;$ 
6    $g_e \leftarrow O - g_e \quad \forall e \in E;$  ▷ Compute marginal gain for edge  $e$ 
7    $e^* \leftarrow \arg \max_{e \in E} g_e;$  ▷ Choose edge  $e^*$  with highest marginal gain
8    $O \leftarrow O + g_{e^*};$  ▷ Update objective value for current attack  $\mu$ 
9    $\mu \leftarrow \mu \cup \{e^*\};$  ▷ Update current attacking scenario  $\mu$ 
10   $E \leftarrow E - \{e^*\};$  ▷ Update potential edge set  $E$ 
11 until ( $|\mu| \geq \Gamma$ );
12 return  $\mu$ 

```

Although the LO model from Table 4 can be solved in polynomial time, the greedy approach needs to solve it ($\Gamma \times |\mathcal{E}|$) times and therefore, it is not suitable for problems with large number of edges. In case of sub-modular objective function, lazy greedy algorithms (Minoux 1978) can be employed to accelerate the solution process. However, the following two examples show that the adversary's decision problem is neither sub-modular nor super-modular.

Example 6.1 *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. Figure 7(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 the edge e_{37} in the attacked edge set which only contains edge e_{26} , the marginal gain in objective remains 0 as the entire flow can be rerouted through the augmented path $\{e_{36}, e_{68}\}$. In contrast, if we add the edge e_{37} in the superset which contains e_{26} and e_{45} , then the marginal gain in objective is 2, as the residual capacity of e_{68} is now shared by the rerouted flow from both e_{37} and e_{45} . As the

marginal gain for adding edge e_{37} into superset is higher, the adversary's decision problem is not sub-modular.

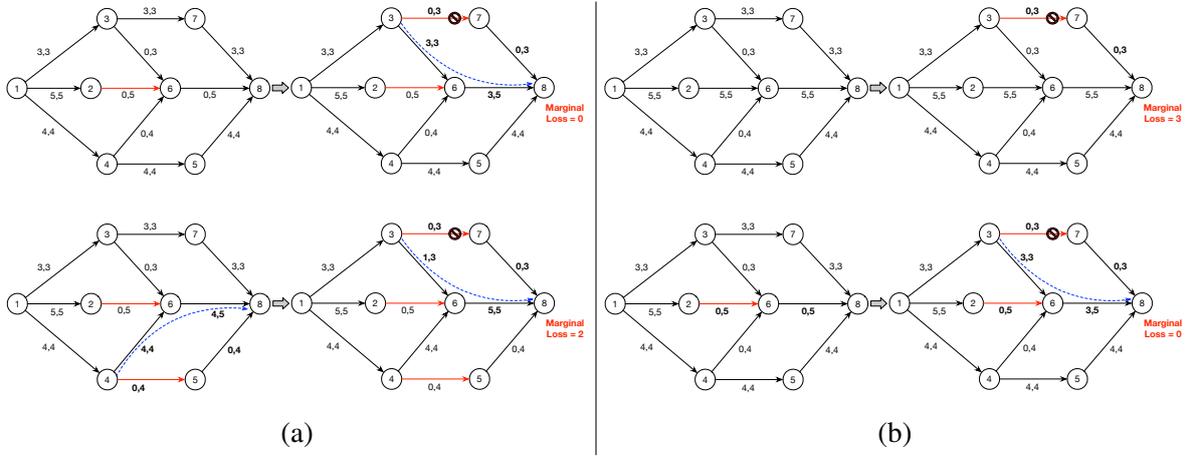


Figure 7: Adversary's decision problem is (a) neither sub-modular; (b) nor super-modular.

Example 6.2 *The adversary's decision problem is not super-modular:* A function is referred as 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. Figure 7(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 the edge e_{37} in the attacked edge set as the first element, then the entire flow is lost and therefore, the gain in the adversary's objective is 3. In contrast, if we add the edge e_{37} in the superset which contains edge e_{26} , the entire flow can be rerouted to the terminal node and the marginal gain in objective is 0. Hence, the adversary's decision problem is clearly not super-modular, as the marginal gain for adding edge e_{37} into superset is lower.

Even though the adversary's decision problem is neither sub-modular nor super-modular, the following observation 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.

Observation 2 *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 is 1 (i.e., $\Gamma = 1$).*

As we incrementally elect one edge at a time for the greedy approach, we exploit the Observation (2) to develop an accelerated greedy algorithm which is shown in Algorithm (3). Let E denote the set of candidate edges and μ be 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 4 to compute the objective value and marginal gain for adding the edge e^* in the current attacking set μ 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 equal or greater than 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 highest marginal gain, insert it to the set of attacked edges μ 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($G = \langle \mathcal{V}, \mathcal{E} \rangle, U, \mathbf{p}, \bar{\mathbf{x}}, \Gamma$)

```

1 Initialize:  $\mu \leftarrow \{\}; it \leftarrow 0; E \leftarrow \mathcal{E};$ 
2  $g^0, \hat{\mathbf{x}} \leftarrow \text{IDENTIFYFLOW}(G, U, \mathbf{p}, \bar{\mathbf{x}}, \mu);$   $\triangleright$  Compute objective for the given flow  $\bar{\mathbf{x}}$ 
3 repeat
4    $it \leftarrow it + 1;$ 
5    $B_e \leftarrow \bar{x}_e \forall e \in E;$   $\triangleright$  Update upper bounds with the new flow  $\bar{\mathbf{x}}$  from last iteration
6    $\hat{E} \leftarrow \{\};$ 
7   repeat
8      $e^* \leftarrow \arg \max_{e \in E} B_e;$   $\triangleright$  Choose the edge  $e^*$  with highest upper bound
9      $\hat{E} \leftarrow \hat{E} \cup \{e^*\};$   $\triangleright$  Update the list of visited edges
10     $g^{it}, \hat{\mathbf{x}} \leftarrow \text{IDENTIFYFLOW}(G, U, \mathbf{p}, \bar{\mathbf{x}}, \mu \cup \{e^*\});$ 
11     $B_{e^*} \leftarrow g^{it-1} - g^{it};$   $\triangleright$  Update the upper bound for edge  $e^*$ 
12    if  $B_{e^*} \geq B_e, \forall e \in E \setminus \hat{E}$  then
13       $e^{**} \leftarrow \arg \max_{e \in \hat{E}} B_e;$   $\triangleright$  Choose the edge  $e^{**}$  with highest marginal gain
14       $\mu \leftarrow \mu \cup \{e^{**}\};$   $\triangleright$  Update the current attacking edge set
15       $E \leftarrow E - \{e^{**}\};$   $\triangleright$  Update the candidate edge set  $E$ 
16       $\bar{\mathbf{x}} \leftarrow \hat{\mathbf{x}};$   $\triangleright$  Update the flow scenario  $\bar{\mathbf{x}}$  according to new attack  $\mu$ 
17      Break;
18    until True;
19 until  $(|\mu| \geq \Gamma);$ 
20 return  $\mu$ 

```

6.2 Network Partitioning Based Optimization

As indicated in §5.5, the complexity of the adversary’s decision problem grows exponentially with the number of edges in the network. So, for our second heuristic, we first identify a small subset of candidate edges that are highly likely to be present in an optimal solution and then the adversary’s optimization problem from Table 1 is solved by only considering the small set 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 Γ best edges to attack.

We begin the discussion with a random network partitioning approach that is compactly shown in Algorithm (4). Let, we have a network with $N(= |\mathcal{V}|)$ nodes. We first randomly sample $(\frac{N}{2} - 1)$ nodes that excludes the source and terminal node⁵. 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 first sub-network and an artificial source node \hat{s} for the second sub-network are created. For each 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 between them.
- If both u and v lie in the second sub-network, we create a directed edge between them.
- If u lies in first sub-network and v lies in the second sub-network, then we create two 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 capacity values for both the newly introduced edges are directly taken from edge e ⁶.
- If u lies in the second sub-network and v lies in first sub-network, then we remove edge e from the sub-problems.

Example 6.3 *Figure (8) 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*

⁵In case of multiple source nodes, we create an artificial source node and connect it to all the original source nodes. Similarly, in case multiple terminal nodes, we create an artificial terminal node and connect all the original terminal nodes to the artificial 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.

⁶It 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 terminal node or from the source node of second sub-network to another node. In that case, we replace all the parallel edges with 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.

Algorithm 4: NETWORKPARTITIONING($G = \langle \mathcal{V}, \mathcal{E} \rangle$)

```

1  $\mathcal{V}^s \leftarrow \text{RANDOMSAMPLE}(\mathcal{V} \setminus \{s, t\}, |\mathcal{V}|/2)$ ;            $\triangleright$  Randomly sample half of the nodes
2  $\mathcal{V}^s \leftarrow \mathcal{V}^s \cup \{s\} \cup \{\hat{t}\}$ ;                        $\triangleright \mathcal{V}^s$  represents the set of nodes in the first sub-network
3  $\mathcal{V}^t \leftarrow \mathcal{V} \setminus \mathcal{V}^s \cup \{\hat{s}\}$ ;                  $\triangleright \mathcal{V}^t$  represents the set of nodes in the second sub-network
4  $\mathcal{E}^s \leftarrow \{\}$ ;                                            $\triangleright \mathcal{E}^s$  represents the set of edges in the first sub-network
5  $\mathcal{E}^t \leftarrow \{\}$ ;                                            $\triangleright \mathcal{E}^t$  represents the set of edges in the second sub-network
6 for  $e = \{u, v\} \in \mathcal{E}$  do
7   if  $u, v \in \mathcal{V}^s$  then
8      $\mathcal{E}^s \leftarrow \mathcal{E}^s \cup e$ ;    $\triangleright$  Add edge  $e$  directly if both  $u$  and  $v$  lie in first sub-network
9   if  $u, v \in \mathcal{V}^t$  then
10     $\mathcal{E}^t \leftarrow \mathcal{E}^t \cup e$ ;    $\triangleright$  Add edge  $e$  directly if both  $u$  and  $v$  lie in second sub-network
11  if  $(u \in \mathcal{V}^s) \wedge (v \in \mathcal{V}^t)$  then
12     $\mathcal{E}^s \leftarrow \mathcal{E}^s \cup \{u, \hat{t}\}$ ;    $\triangleright$  Create an edge between  $u$  and  $\hat{t}$  in first sub-network
13     $\mathcal{E}^t \leftarrow \mathcal{E}^t \cup \{\hat{s}, v\}$ ;    $\triangleright$  Create an edge between  $\hat{s}$  and  $v$  in second sub-network
14 return  $G^s = \langle \mathcal{V}^s, \mathcal{E}^s \rangle, G^t = \langle \mathcal{V}^t, \mathcal{E}^t \rangle$ 

```

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.

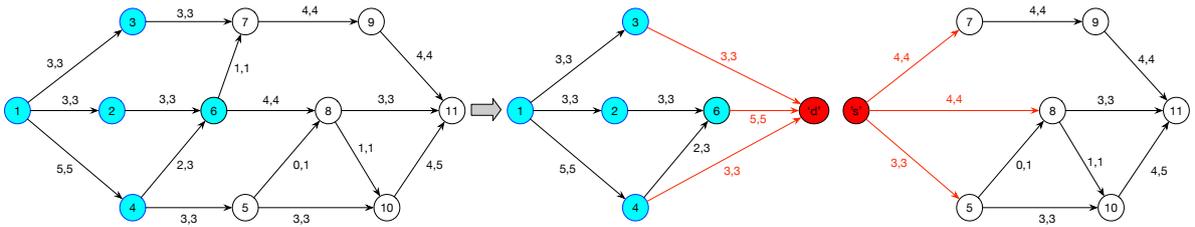


Figure 8: Illustration of partitioning approach.

Algorithm (5) describes the key steps for the 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) and solve the adversary’s decision model 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 Δ . Finally, we solve the adversary’s optimization problem from Table 1 to identify the best Γ edges to attack from the candidate edges $\mu^\mathcal{E}$. Specifically, we manually set the value of μ_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 Δ needs to be reduced accordingly for solving the final decision problem over the candidate edges.

Algorithm 5: PARTITIONINGHEURISTIC($G = \langle \mathcal{V}, \mathcal{E} \rangle, U, \mathbf{p}, \bar{\mathbf{x}}, \Gamma$)

```

1 Initialize:  $\mu^\mathcal{E} \leftarrow \{\}; it \leftarrow 0;$  ▷ Initialize candidate edge set  $\mu^\mathcal{E}$  as an empty set
2 repeat
3    $it \leftarrow it + 1;$ 
4    $\{G^s, G^t\} \leftarrow \text{NETWORKPARTITIONING}(G);$  ▷ Randomly partition the network
5    $\mu^s \leftarrow \text{ADVERSARYPROBLEM}(G^s, U, \mathbf{p}, \bar{\mathbf{x}}, \frac{\Gamma}{2});$  ▷ Solve first sub-network problem
6    $\mu^t \leftarrow \text{ADVERSARYPROBLEM}(G^t, U, \mathbf{p}, \bar{\mathbf{x}}, \frac{\Gamma}{2});$  ▷ Solve second sub-network problem
7    $\mu^\mathcal{E} \leftarrow \mu^\mathcal{E} \cup \mu^s \cup \mu^t;$  ▷ Add the new set of potential edges to  $\mu^\mathcal{E}$ 
8 until  $(|\mu^\mathcal{E}| \geq \Delta) \vee (it \geq M);$ 
9  $\mu \leftarrow \text{ADVERSARYPROBLEM}(\{G, \mu^\mathcal{E}\}, U, \mathbf{p}, \bar{\mathbf{x}}, \Gamma);$  ▷ Solve the problem over  $\mu^\mathcal{E}$  edges
10 return  $\mu$ 

```

7 Experimental Results on Large-scale Data Sets

In this section, we present empirical results on large-scale problem instances. We use the same setting used in §5 to generate a set of large synthetic networks. We demonstrate the sensitivity results by varying three input tunable parameters: (a) the number of nodes, (b) the edge density, and (c) the adversary’s budget. For all the networks, we randomly draw the capacities for the edges from the range of 1 to 50. Therefore, the value of maximum capacity U^{max} is set to 50. The unit costs for routing the flow through edges (i.e., \mathbf{p}) are drawn randomly from the range of 0.01 to 0.1. We compare our *RAMF* approach against the four benchmark approaches (i.e., *MF*, *OSP*, *RF* and *AAMF*) that are explained in §5.

For the large-scale problem instances, we employ the heuristics from §6 to efficiently solve the adversary’s decision problem. In each iteration of the game, both the accelerated greedy

approach from §6.1 and the network partitioning based heuristic approach from §6.2 are solved. We begin by showing the performance comparison between the two proposed heuristics. We empirically observe that the partition based heuristic mostly performs better than the greedy approach, specially 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 number of lost flows, overall objective value and runtime for both the heuristics in each iteration of the game on a problem instance with 50 nodes, 0.4 edge density 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 number of lost flow. As shown in Table 5, while the partitioning based heuristic mostly outperforms the greedy approach, the solution quality of the greedy approach is better for a few 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 the heuristics in each iteration of the game and choose the attacking scenario that provides a 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.

Iteration	Lost Flow		Objective Value			Runtime (Sec)	
	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

Table 5: Solution quality and runtime comparison between two proposed heuristics.

7.1 Sensitivity Analysis by Varying the Number of Nodes

Figure 9 demonstrates the performance comparison results between our *RAMF* approach and the four benchmark approaches, where we vary the number of nodes from 40 to 75 in the

X-axis. In all the settings, the edge density for the randomly generated directed graphs is fixed to 0.4 and the adversary’s budget is set to 10. Figure 9(a) shows the net amount of lost flow for all the five approaches. As expected, 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 number of lost flow. The lost flow for the *OSP* is always lower than other benchmark approaches. Except for the relatively small problem instances with 40 nodes, our *RAMF* approach always outperforms all the four benchmark approaches in terms of reducing the number of lost flow.

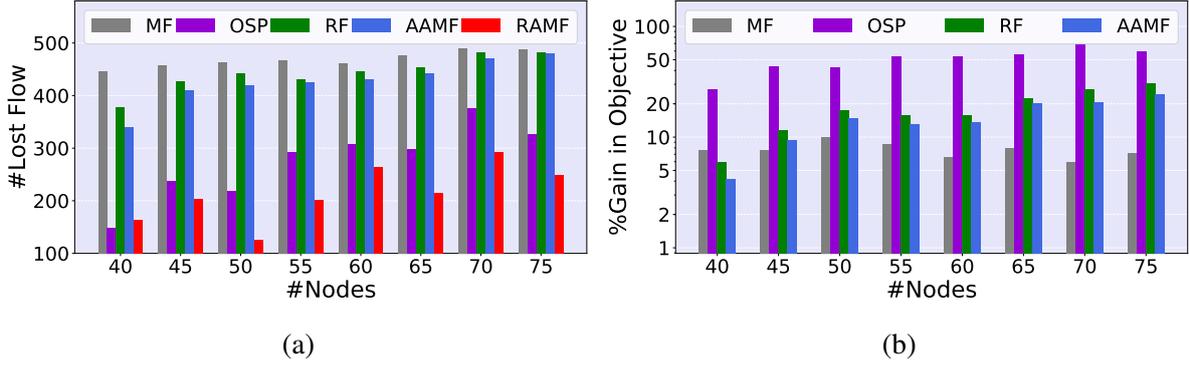


Figure 9: Effect of number of nodes on (a) Lost flow; (b) Objective.

Figure 9(b) delineates the percentage gain in the objective value for our *RAMF* approach against the four benchmarks in a logarithmic scale. The percentage gain in the objective is computed as the ratio between the difference in objectives and upper bound on the marginal gain (refer to equation 39). 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 the *MF*, *OSP*, *RF* and *AAMF* approaches are 7.7%, 50.1%, 18.2% and 14.97%, respectively.

7.2 Sensitivity Analysis by Varying the Number of Edges

We now demonstrate a sensitivity analysis by varying the number of edges in the network. Figure 10 presents the performance comparison results where we vary the edge density from 0.2 to 0.7 in the X-axis. For these experiments, we consider networks with 50 nodes and the adversary’s budget is fixed to 10. Figure 10(a) shows the net amount of lost flow for all the five approaches. We observe a consistent pattern that the lost flows for the *MF*, *RF*, *AAMF* approaches are always significantly higher than our *RAMF* approach. While the lost flow for the *OSP* approach is lower than our approach for smaller networks with edge density 0.2 and 0.3, our *RAMF* approach significantly outperforms the *OSP* approach when the number of edges increases.

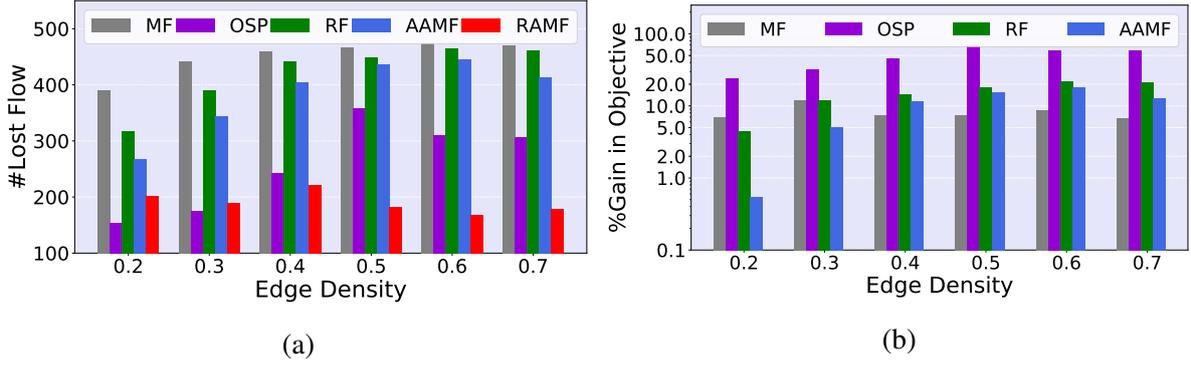


Figure 10: Effect of number of edges on (a) Lost flow; (b) Objective.

Figure 10(b) delineates the percentage gain in the objective value of the *RAMF* approach in a logarithmic scale. 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 objective value in all the cases. The percentage gains in the objective value for our approach against all the benchmark approaches are always positive. On an average, the percentage gains in objective for our *RAMF* approach over the *MF*, *OSP*, *RF* and *AAMF* approaches are 8.2%, 47.3%, 15.2% and 10.5%, respectively.

7.3 Sensitivity Analysis by Varying the Adversary's Budget

In this thread of results, we present the effect of the adversary's budget value on both the performance metrics. Figure 11 demonstrates the comparison results, where we vary the budget of the adversary from 2 to 20 in the X-axis. Figure 11(a) exhibits the net amount of lost flow for all the approaches. The amount of lost flow for *MF* approach increases monotonically from 100 to almost 900 as we increase the adversary's budget, whereas the number of lost flow for our *RAMF* approach is always lower than all the benchmarks and is always bounded by 160. Moreover, similar to the results presented in §5.3, we observe that the number of lost flow for our *RAMF* approach remains steady when the adversary's budget value goes beyond 14.

Figure 11(b) demonstrates the percentage gains in the objective value for our approach over the four benchmarks. Similar to the pattern observed in Figure 4(b), the gains in objective for our approach over the *OSP*, *RF* and *AAMF* approaches almost always reduce monotonically with the increasing value of Γ (although the pattern is not always coherent as the adversary's decision problem is solved sub-optimally). The net difference between the objectives of our *RAMF* approach and the *MF* approach increases monotonically with the value of Γ , which indicates that the performance of our approach improves gradually if the adversary becomes stronger. On an average, the percentage gains in the objective of our *RAMF* approach over the *MF*, *OSP*, *RF* and *AAMF* approaches are 10.4%, 38.1%, 21.1% and 17.6%, respectively.

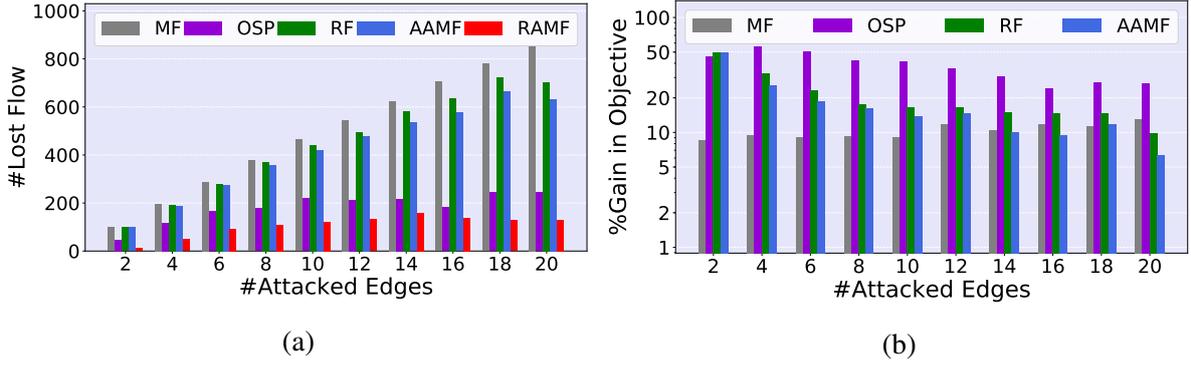


Figure 11: Effect of the adversary's budget value on (a) Lost flow; (b) Objective.

In summary, all the three threads of results provide a consistent pattern (similar pattern is observed with the small-scale results presented in §5) that our *RAMF* approach reduces the lost flow significantly over the *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, from these results we can conclude that even though the two proposed heuristics solve the adversary's decision problem sub-optimally, the empirical results on the large-scale problems replicate the similar trend observed for the optimal results on the small-scale problem instances.

7.4 Runtime Performance

We now demonstrate the runtime performance of our *RAMF* approach for different settings of network size and adversary's budget. As the number of iterations required to converge the game may vary randomly for different problems, we show the average runtime for one iteration of the game. Figure 12 presents the runtime for the *RAMF* approach in minutes in a logarithmic scale. In the X-axis, the number of nodes is varied from 40 to 75 (with edge density 0.4 and the adversary's budget is fixed to 10), the edge density is varied from 0.2 to 0.7 (in a network with 50 nodes and the adversary's budget is set to 10) and the adversary's budget is varied from 2 to 20 (in a network with 50 nodes and edge density 0.4). We observe that 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 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 and the complexity of the sub-problems reduces once the budget value becomes large. Therefore, the runtime starts to decrease as the value of Γ goes beyond 10.

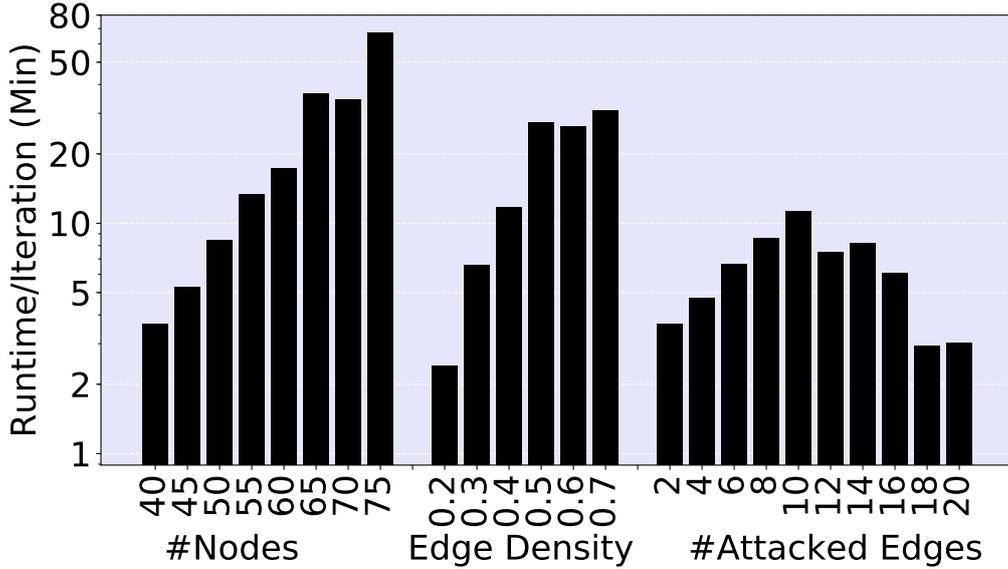


Figure 12: Runtime results for varying number of nodes, edges and adversary’s budget.

7.5 Results on RMFGEN Data Set

In the last thread of results, we provide the 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-scale RMFGEN networks for the experiment⁷. As some of the instances are undirected networks, we convert them into directed networks by randomly adding edges until the network becomes connected (i.e., every node has at least one incoming and one outgoing edge). As the capacities of edges, U are mentioned as continuous value in the database, we generate the capacity values randomly from the range of 10 to 50. The unit routing costs for the edges, p 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 6 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 the five approaches and the average runtime (in minutes) per iteration for the *RAMF* approach. As expected, the net number of lost flow for the *MF* approach is significantly higher than other approaches for all the instances. The percentage gains in objective value for the *RAMF* approach over all the benchmark approaches are always positive. On an average, our approach improves the objective value by 8%, 50.1%, 47.9%, 39.6% over the *MF*, *OSP*, *RF* and *AAMF* approaches, respectively. Most importantly, we observe that our proposed heuristics can scale gracefully to solve these large problem instances while providing a significant performance

⁷The data set is collected from <http://elib.zib.de/pub/mp-testdata/maxflow/index.html>

Problem Instance				Number of Lost Flow					Objective Value					Runtime/ Iteration
Name	$ \mathcal{V} $	$ \mathcal{E} $	Γ	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

Table 6: Empirical results on RMFGEN data set.

gain over the benchmark approaches.

8 Concluding Remarks

Modern critical cyberphysical infrastructure systems comprise of various automated devices which are vulnerable to cyber/physical attacks. To evaluate the resilience and sustainability of such urban systems, we propose a robust and adaptive network flow model 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 attacking scenario 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 attacking scenarios 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 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 the large-scale problem instances and improves the operational efficiency of the network by reducing the expected lost flow.

In 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, so as to scale up the solution process to massive real-world urban networks consisting of tens of thousands 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 flow to capture the congestion effects) will 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 problem domain so as to linearize the optimization models.

Acknowledgements

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.

Appendix

A Robust Flow Solution

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 Γ edges in the network can be attacked. Let \mathbf{x} denote the resulting robust flow scenario and \mathcal{L}_Γ denote the worst-case lost flow value if the adversary is restricted to attack a maximum of Γ edges. The optimization model (40) compactly delineates the details of our robust flow solution, where the inner optimization model computes the value of \mathcal{L}_Γ as the sum of the first Γ biggest edge flow values.

$$\begin{aligned}
\max_{\mathbf{x}} \quad & x_{(t,s)} - \mathcal{L}_\Gamma & \text{where,} \quad & \mathcal{L}_\Gamma = \max_{\boldsymbol{\mu}} \sum_e x_e \mu_e \\
\text{s.t.} \quad & \sum_{e \in \delta_v^+} x_e - \sum_{e \in \delta_v^-} x_e = 0, \quad \forall v \in \mathcal{V} \setminus \{s\} & & \text{s.t.} \quad \sum_e \mu_e \leq \Gamma \\
& 0 \leq x_e \leq U_e, \quad \forall e \in \mathcal{E} & & 0 \leq \mu_e \leq 1, \quad \forall e \in \mathcal{E}
\end{aligned} \tag{40}$$

The two components of problem (40) can be combined by taking the dual of the inner problem \mathcal{L}_Γ . To achieve this goal, we first compute the *Lagrangian* function (41) by introducing the price variables θ and δ . From this *Lagrangian* function, we construct the dual problem (42).

$$\min_{\delta, \theta} - \sum_e x_e \mu_e + \delta \left(\sum_e \mu_e - \Gamma \right) + \sum_e \theta_e (\mu_e - 1) \tag{41}$$

$$\begin{aligned}
& \max_{\delta, \theta} \quad - \sum_e \theta_e - \delta \Gamma \\
& \text{s.t.} \quad \theta_e + \delta \geq x_e, \quad \forall e \in \mathcal{E} \\
& \quad \quad \theta_e \geq 0, \delta \geq 0
\end{aligned} \tag{42}$$

Putting the optimization models (40) and (42) together, we construct the linear optimization model (43) that is used to solve the robust maximum flow problem.

$$\begin{aligned}
& \max_{x, \theta, \delta} \quad x_{(t,s)} - \sum_e \theta_e - \delta \Gamma \\
& \text{s.t.} \quad \sum_{e \in \delta_v^+} x_e - \sum_{e \in \delta_v^-} x_e = 0, \quad \forall v \in \mathcal{V} \setminus \{s\} \\
& \quad \quad \theta_e + \delta \geq x_e, \quad \forall e \in \mathcal{E} \\
& \quad \quad 0 \leq x_e \leq U_e, \quad \forall e \in \mathcal{E} \\
& \quad \quad \theta_e \geq 0, \delta \geq 0
\end{aligned} \tag{43}$$

B Approximate Adaptive Maximum Flow Solution

In this section, we provide the details of another benchmark approach that computes an approximate adaptive maximum flow 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 θ 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 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 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 denotes the min-cut solution for the network of interest, i.e., the $s - t$ cut S has the minimum capacity value. The linear optimization model (44) provides details of the approximate solution proposed in Bertsimas et al. (2013).

$$\begin{aligned}
& \max_{x, \theta} \quad \sum_{e \in \delta^+(S)} x_e - \sum_{e \in \delta^-(S)} x_e - \theta \Gamma \\
& \text{s.t.} \quad \sum_{e \in \delta_v^+} x_e - \sum_{e \in \delta_v^-} x_e = 0, \quad \forall v \in \mathcal{V} \setminus \{s, t\} \\
& \quad \quad x_e \leq \theta, \quad \forall e \in \mathcal{E} \\
& \quad \quad 0 \leq x_e \leq U_e, \quad \forall e \in \mathcal{E}
\end{aligned} \tag{44}$$

Let x^* be a flow with maximum robust flow value, such that $\beta\text{Val}(x^*) \leq \text{RVal}(x^*)$ for some $\beta \in (0, 1]$, where $\text{Val}(x^*)$ represents the objective value for the administrator if no attack is executed in the network, and $\text{RVal}(x^*)$ denotes the robust flow value of x^* . Further, suppose $\bar{x}, \bar{\theta}$ be the optimal solution for the optimization problem (44). Let ρ 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.,

$$\text{RVal}(\bar{x}) \geq \left(1 - \frac{1 - \rho}{\rho} \frac{1 - \beta}{\beta}\right) \text{RVal}(x^*)$$

In addition, Bertsimas et al. (2013) show that the resulting flow from problem (44) provides an α -approximation to the optimal adaptive maximum flow solution. Let x^* be an adaptive maximum flow such that $\beta\text{Val}(x^*) \leq \text{RVal}(x^*)$ for some $\beta \in (0, 1]$, and $\bar{x}, \bar{\theta}$ be the optimal solution for the optimization problem (44). Then, adaptive value of \bar{x} , $\text{AVal}(\bar{x})$ yields a $\beta(1 - (((1 - \rho)/\rho)((1 - \beta)/\beta)))$ -approximation for the adaptive value of x^* , i.e.,

$$\text{AVal}(\bar{x}) \geq \beta \left(1 - \frac{1 - \rho}{\rho} \frac{1 - \beta}{\beta}\right) \text{AVal}(x^*)$$

References

- Adulyasak, Yossiri, Patrick Jaillet. 2015. Models and algorithms for stochastic and robust vehicle routing with deadlines. *Transportation Science* **50**(2) 608–626.
- Ahmed, Asrar, Pradeep Varakantham, Yossiri Adulyasak, Patrick Jaillet. 2013. Regret based robust solutions for uncertain markov decision processes. *Advances in neural information processing systems*. 881–889.
- Altner, Douglas S, ÖZlem Ergun, Nelson A Uhan. 2010. The maximum flow network interdiction problem: valid inequalities, integrality gaps, and approximability. *Operations Research Letters* **38**(1) 33–38.
- Assadi, Sepehr, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, Hamid Zarrabi-Zadeh. 2014. The minimum vulnerability problem. *Algorithmica* **70**(4) 718–731.
- Ball, Michael O, Bruce L Golden, Rakesh V Vohra. 1989. Finding the most vital arcs in a network. *Operations Research Letters* **8**(2) 73–76.
- Baykal-Guersoy, Melike, Zhe Duan, H Vincent Poor, Andrey Garnaev. 2014. Infrastructure security games. *European Journal of Operational Research* **239**(2) 469–478.
- Ben-Tal, Aharon, Arkadi Nemirovski. 1998. Robust convex optimization. *Mathematics of operations research* **23**(4) 769–805.
- Bertsimas, Dimitris, David B Brown, Constantine Caramanis. 2011. Theory and applications of robust optimization. *SIAM review* **53**(3) 464–501.
- Bertsimas, Dimitris, Vishal Gupta, Nathan Kallus. 2018. Data-driven robust optimization. *Mathematical Programming* **167**(2) 235–292.

- Bertsimas, Dimitris, Ebrahim Nasrabadi, James B Orlin. 2016. On the power of randomization in network interdiction. *Operations Research Letters* **44**(1) 114–120.
- Bertsimas, Dimitris, Ebrahim Nasrabadi, Sebastian Stiller. 2013. Robust and adaptive network flows. *Operations Research* **61**(5) 1218–1242.
- Bertsimas, Dimitris, Melvyn Sim. 2003. Robust discrete optimization and network flows. *Mathematical programming* **98**(1-3) 49–71.
- Bertsimas, Dimitris, Melvyn Sim. 2004. Robust discrete optimization under ellipsoidal uncertainty sets. Tech. rep., MIT.
- Boyd, Stephen, Lieven Vandenbergh. 2004. *Convex optimization*. Cambridge university press.
- Brown, Matthew, Sandhya Saisubramanian, Pradeep Reddy Varakantham, Milind Tambe. 2014. Streets: game-theoretic traffic patrolling with exploration and exploitation. *Innovative Applications in Artificial Intelligence (IAAI)*. AAAI Press, 2966–2971.
- 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/> .
- Cormican, Kelly J, David P Morton, R Kevin Wood. 1998. Stochastic network interdiction. *Operations Research* **46**(2) 184–197.
- Dahan, Mathieu, Saurabh Amin. 2015. Network flow routing under strategic link disruptions. *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, 353–360.
- Dahan, Mathieu, Saurabh Amin, Patrick Jaillet. 2018. Probability distributions on partially ordered sets and network security games. *arXiv preprint arXiv:1811.08516* .
- Dwivedi, Ajendra, Xinghuo Yu. 2013. A maximum-flow-based complex network approach for power system vulnerability analysis. *IEEE Transactions on Industrial Informatics* **9**(1) 81–88.
- Dziubiński, Marcin, Sanjeev Goyal. 2013. Network design and defence. *Games and Economic Behavior* **79** 30–43.
- Fang, Fei, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, Andrew Lemieux, et al. 2016. Deploying paws: Field optimization of the protection assistant for wildlife security. *National Conference on Artificial Intelligence (AAAI)*. 3966–3973.
- Ford, Lester R, Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* **8**(3) 399–404.
- Ghena, Branden, William Beyer, Allen Hillaker, Jonathan Pevarnek, J Alex Halderman. 2014. Green lights forever: Analyzing the security of traffic infrastructure. *WOOT* **14** 7–7.
- Ghosh, Supriyo, Michael Trick, Pradeep Varakantham. 2016. Robust repositioning to counter unpredictable demand in bike sharing systems. *International Joint Conference on Artificial Intelligence (IJCAI)* .
- Goldfarb, Donald, Michael D Grigoriadis. 1988. A computational comparison of the dinic and network simplex methods for maximum flow. *Annals of Operations Research* **13**(1) 81–123.

- Gueye, Assane, Vladimir Marbukh. 2012. A game-theoretic framework for network security vulnerability assessment and mitigation. *International Conference on Decision and Game Theory for Security*. Springer, 186–200.
- Gueye, Assane, Vladimir Marbukh, Jean C Walrand. 2012. Towards a metric for communication network vulnerability to attacks: A game theoretic approach. *International Conference on Game Theory for Networks*. Springer, 259–274.
- Guo, Qingyu, Bo An, Yair Zick, Chunyan Miao. 2016. Optimal interdiction of illegal network flow. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2507–2513.
- He, Fei, Jun Zhuang, Nageswara SV Rao. 2012. Game-theoretic analysis of attack and defense in cyber-physical network infrastructures. *IIE Annual Conference. Institute of Industrial and Systems Engineers (IISE)*. Citeseer, 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, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe. 2010. Security games with arbitrary schedules: A branch and price approach. *National Conference on Artificial Intelligence (AAAI)*. 792–797.
- Jain, Manish, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, Milind Tambe. 2011. A double oracle algorithm for zero-sum security games on graphs. *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 327–334.
- Kar, Debarun, Thanh H Nguyen, Fei Fang, Matthew Brown, Arunesh Sinha, Milind Tambe, Albert Xin Jiang. 2017. Trends and applications in stackelberg security games. *Handbook of Dynamic Game Theory* 1–47.
- Laporte, Gilbert, Juan A Mesa, Federico Perea. 2010. A game theoretic framework for the robust railway transit network design problem. *Transportation Research Part B: Methodological* **44**(4) 447–459.
- Li, Mian, Shapour Azarm. 2008. Multiobjective collaborative robust optimization with interval uncertainty and interdisciplinary uncertainty propagation. *Journal of mechanical design* **130**(8) 081402.
- Ma, Chris YT, Nageswara SV Rao, David KY Yau. 2011. A game theoretic study of attack and defense in cyber-physical systems. *Computer Communications Workshops (INFOCOM WKSHPs), 2011 IEEE Conference on*. IEEE, 708–713.
- Manshaei, Mohammad Hossein, Quanyan Zhu, Tansu Alpcan, Tamer Başçar, Jean-Pierre Hubaux. 2013. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)* **45**(3) 25.
- Minoux, Michel. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*. Springer, 234–243.
- Orlowski, Sebastian, Roland Wessály, Michal Pióro, Artur Tomaszewski. 2010. Sndlib 1.0survivable network design library. *Networks: An International Journal* **55**(3) 276–286.
- Pita, James, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe,

- Craig Western, Praveen Paruchuri, Sarit Kraus. 2008. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*. International Foundation for Autonomous Agents and Multiagent Systems, 125–132.
- Ratliff, H Donald, G Thomas Sicilia, SH Lubore. 1975. Finding the n most vital links in flow networks. *Management Science* **21**(5) 531–539.
- Reilly, Jack, Sébastien Martin, Mathias Payer, Alexandre M Bayen. 2015. On cybersecurity of freeway control systems: Analysis of coordinated ramp metering attacks. *Transportation Research Board 94th Annual Meeting* .
- Sinha, Arunesh, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, Albert Xin Jiang. 2015. From physical security to cybersecurity. *Journal of Cybersecurity* **1**(1) 19–35.
- Sullivan, Kelly M, J Cole Smith. 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: transport dynamics* **1**(3) 237–260.
- Tsai, Jason, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, Milind Tambe. 2010. Urban security: Game-theoretic resource allocation in networked physical domains. *National Conference on Artificial Intelligence (AAAI)*. 881–886.
- Vorobyov, Sergiy A, Alex B Gershman, Zhi-Quan Luo. 2003. Robust adaptive beamforming using worst-case performance optimization: A solution to the signal mismatch problem. *IEEE transactions on signal processing* **51**(2) 313–324.
- Washburn, Alan, Kevin Wood. 1995. Two-person zero-sum games for network interdiction. *Operations research* **43**(2) 243–251.
- Wollmer, Richard. 1964. Removing arcs from a network. *Operations Research* **12**(6) 934–940.
- Wood, R Kevin. 1993. Deterministic network interdiction. *Mathematical and Computer Modelling* **17**(2) 1–18.
- Wu, Manxi, Saurabh Amin. 2018. Security of transportation networks: Modeling attacker-defender interaction. *arXiv preprint arXiv:1804.00391* .
- Wu, Manxi, Li Jin, Saurabh Amin, Patrick Jaillet. 2018. Signaling game-based misbehavior inspection in v2i-enabled highway operations. *IEEE 57st 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, Victor Bucarey, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, Milind Tambe. 2016. Using abstractions to solve opportunistic crime security games at scale. *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 196–204.