Defense Against Shortest Path Attacks

Benjamin A. Miller* Zohair Shafi* Wheeler Ruml† Yevgeniy Vorobeychik‡
Tina Eliassi-Rad* Scott Alfeld§

Abstract

Identifying shortest paths between nodes in a network is an important task in many applications. Recent work has shown that a malicious actor can manipulate a graph to make traffic between two nodes of interest follow their target path. In this paper, we develop a defense against such attacks by modifying the edge weights that users observe. The defender must balance inhibiting the attacker against any negative effects on benign users. Specifically, the defender's goals are: (a) recommend the shortest paths to users, (b) make the lengths of the shortest paths in the published graph close to those of the same paths in the true graph, and (c) minimize the probability of an attack. We formulate the defense as a Stackelberg game in which the defender is the leader and the attacker is the follower. We also consider a zero-sum version of the game in which the defender's goal is to minimize cost while achieving the minimum possible attack probability. We show that the defense problem is NP-hard and propose heuristic solutions for both the zero-sum and non-zero-sum settings. By relaxing some constraints of the original problem, we formulate a linear program for local optimization around a feasible point. We present defense results with both synthetic and real networks and show that our methods often reach the lower bound of the defender's cost.

1 Introduction

In numerous applications involving the routing of resources through a network, finding the shortest path between two nodes is an important problem. A malicious actor with the capacity to modify the graph could entice users to follow a particular path that could put them at risk. In cybersecurity, for example, an attacker could convince users to use compromised routers to intercept traffic and possibly steal resources [15]. To counter adversarial activity, it is important to consider defensive measures against such behavior.

Recent work has proposed an algorithm to manipulate the shortest path when the attacker is able to remove edges [20]. In this paper, taking inspiration from differential privacy, we propose a defense technique based on perturbing edge weights. Users are presented an altered set of edge weights that aims to provide the shortest paths possible while raising the attacker's cost. The contributions of this paper are as follows: (1) We define a defender cost based on the impact on user experience and probability of attack. (2) We formulate a Stackelberg game to optimize the defender's expected cost. (3) In a zero-sum setting, we show that this optimization is NP-hard. (4) We propose `PATHDEFENSE`, a heuristic algorithm that greedily increments edge weights until the user's cost is sufficiently low. (5) We present results on simulated and real networks demonstrating the cost improvement `PATHDEFENSE` provides.

2 Problem Definition

In our problem setting, a graph G has weights w, and an attacker intends to remove edges to make a particular target path be the shortest between its endpoints. The defender's goal is to publish approximate weights that provide users with short paths to their destinations while also increasing the burden on the adversary, making an attack less likely. This method is inspired by a differential privacy technique for approximating shortest paths without revealing true weights [22], though here we consider the weight perturbations in an optimization context. A simple example of this scenario is shown in Figure 1, which demonstrates that the defender can raise the attacker's required budget and the risk that there may be more disruption to the graph if the attack still occurs. We refer to the problem of minimizing the defender's cost in this context as the *Cut Defense* problem. The analysis over the remainder of the paper makes the following assumptions: (1) The attacker has a single target path p^* (not necessarily known to the defender) and uses a method known to the defender to optimize the attack. (2) If the optimization method identifies an attack within the attacker's bud-

*Northeastern University, Boston, MA, USA ({miller.be, shafi.z, t.eliassirad}@northeastern.edu)
†University of New Hampshire, Durham, NH, USA (ruml@cs.unh.edu)
‡Washington University in St. Louis, St. Louis, MO, USA (yvorobeychik@wustl.edu)
§Amherst College, Amherst, MA, USA (salfeld@amherst.edu)

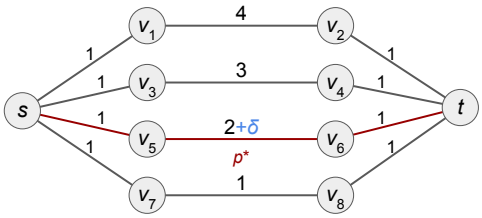


Figure 1: A simple example of the defense method. With no perturbation ($\delta = 0$), p^* can become the shortest path from s to t if only 1 edge is cut, whereas for $\delta \geq 2$, 3 edges must be removed. If the attacker has a budget of at least 3, however, the attack would cause more disruption, and the resulting cost to the defender would be higher. For example, if $\{s, v_1\}$, $\{s, v_3\}$, and $\{s, v_7\}$ are cut, all traffic between s and v_1 or v_3 will take a much longer path than it would have when $\delta = 0$.

get b (not necessarily known to the defender), the attack will occur. (3) True edge weights and removal costs are known to the attacker.

2.1 Notation We consider a graph $G = (V, E)$, which may be directed or undirected. Each edge has a nonnegative weight $w : E \to \mathbb{R}_{\geq 0}$. The weights denote true traversal distances. The defender publishes weights $w' : E \to \mathbb{R}_{\geq 0}$, which may be different than w. For a given source–destination pair $s, t \in V$, let $p(G, \hat{w}, s, t)$ be the shortest path in G from s to t using weights $\hat{w}$. For a path p between two nodes, let $\ell(G, \hat{w}, p)$ be the length of p in G using weights $\hat{w}$. We denote by p^* and b the attacker's target path and budget, respectively.

When determining the impact on users, we consider the distribution of source–destination pairs, $\mathcal{D}$, as this will help determine how often paths are disrupted. In addition, we assume the defender has uncertainty about p^* and b. The defender considers a distribution $\mathcal{P}$ of possible target paths and a distribution $\mathcal{B}$ of possible budgets. These distributions result in a distribution of user-observed graphs, $\mathcal{G}$, which we describe in the next section. The defender's cost (loss) function is denoted by L. A notation table is provided in our longer manuscript [19].

2.2 Stackelberg Game We frame the attacker–defender interaction as a Stackelberg game in which the defender is the leader and the attacker is the follower. The defender has full knowledge of the attacker's action set, and tries to choose the optimal defense given the attacker's assumed response.

Attacker The attacker will observe a graph $G = (V, E)$ with weights w' published by the defender, and may also know the true weights w. Each edge $e \in E$ has a removal cost $c(e) > 0$ that is known to the attacker. The attacker has a target path p^* from source s to destination t, and a budget b specifying the greatest edge removal cost the attacker can incur. The attacker runs an algorithm, which is known to the defender, that solves the Force Path Cut problem [20]: Find a set of edges E' where $c(E') := \sum_{e \in E'} c(e) \leq b$ and p^* is the shortest path from s to t in $G' = (V, E \setminus E')$ (using the published weights w'). If the attack algorithm yields a solution with cost greater than b, the attack is not worth the cost to the attacker, so $G' = G$.

Defender The defender publishes a modified set of weights w'. While the defender knows the method that the attacker will use, we assume there is uncertainty with respect to the attacker's target path p^* and budget b. The defender has a distribution over both of these variables, as defined above. The distributions $\mathcal{P}$ and $\mathcal{B}$ combine with the published weights w' to create a distribution over graphs $\mathcal{G}$ as follows. For a given p^* in $\mathcal{P}$, let E' be the solution given by the attack algorithm using the published weights, and assume it is a unique solution across all target paths. (If multiple target paths have the same solution, the probability of the resulting graph integrates across those paths.) Then the probability that users observe graph $G' = (V, E \setminus E')$ is

$$\Pr_{\mathcal{G}}[G'] = \Pr_{P \sim \mathcal{P}}[P = p^*] \cdot \Pr_{B \sim \mathcal{B}}[c(E') \leq B]. \tag{2.1}$$

The defender's goal is to publish a set of weights that has minimal expected cost, i.e.,

$$\hat{w}' = \arg\min_{w'} \mathbb{E}\left[L(G, w, w', \mathcal{D}, \mathcal{P}, \mathcal{B})\right]. \tag{2.2}$$

There are several considerations when defining the defender's cost, which we discuss in detail next.

2.3 Defender's Cost Function The attacker's cost function is simple: The goal is to execute the attack, so after computing the set of edges to remove, if it is within the attacker's budget, the cost is 0 (attack occurs), and otherwise the cost is 1 (attack does not occur). When determining the best course of action, the defender has three considerations. The first is the cost incurred by users of the network: the distance they must travel to get from their origin points to their destinations. If the users must travel longer distances, the cost to the defender is higher. Note that this is the actual distance traveled: The user selects a path p based on the perturbed weights w', but the distance is computed based the original weights w. There is also a cost associated with users traveling a different distance than advertised. If the length of p is ℓ^{true}, but the user is told the length is ℓ^{obs}, this may negatively affect the user's experience. If $\ell^{\text{obs}} < \ell^{\text{true}}$, the user will likely

be dissatisfied with traversing a longer distance than advertised. The case where $\ell^{\text{obs}} > \ell^{\text{true}}$ is less clear. If the advertised distance is only slightly greater than the true distance, the user may be happy to experience a shorter distance than advertised. If, on the other hand, the advertised distance is drastically larger, this may induce an additional burden on users, and thus additional cost for the defender.

Finally, there may be situations where there is some additional cost to the defender if the adversary is successful. This would be a cost *in addition* to the cost due to longer distances experienced by users after the attack. If, for example, the new traffic route allows the adversary to gain a competitive advantage over the defender, this would have a broader negative consequence than the specific issue of users experiencing longer distances. If this is an issue for the defender, there will be another component to the cost function to account for the expected cost of attacker success.

To formalize the cost function, we consider the three costs described above:

1. L_d: The average *distance* traveled by users
2. L_e: The average cost of the *error* between advertised and true path distances
3. L_s: The expected cost of attacker *success*

Cost 1 takes the expected value across source–destination pairs $u, v \sim \mathcal{D}$. While the path p from u to v is determined using the observed weights w', the distance experienced by users is based on the true weights w. Thus, for a user traveling from u to v, we use the path $p(G', w', u, v)$, which has length $\ell(G', w, p(G', w', u, v))$. Aggregating across all pairs, cost 1 is expressed as

$$L_d(G, w, w', \mathcal{B}, \mathcal{D}, \mathcal{P}) = \mathbb{E}\left[\ell(G', w, p(G', w', u, v))\right], \tag{2.3}$$

where the expectation is taken over

$$s, t \sim \mathcal{D} \text{ and } G' \sim \mathcal{G}(G, w', \mathcal{B}, \mathcal{P}). \tag{2.4}$$

Cost 2 considers the same path as cost 1, but rather than the distance traveled, the defender considers a function c_{err} of the error between the advertised and true path lengths. Then cost 2 is given by

$$L_e(G, w, w', \mathcal{B}, \mathcal{D}, \mathcal{P}) = \mathbb{E}\left[c_{\text{err}}(\ell^{\text{true}}, \ell^{\text{obs}})\right], \tag{2.5}$$

where $\ell^{\text{true}} = \ell(G', w, p(G', w', u, v))$ and $\ell^{\text{obs}} = \ell(G', w', p(G', w', u, v))$, and the expectation is once again taken over (2.4). The shape of c_{err} will vary based on the defender's belief about users' degree of dissatisfaction with errors in reported path lengths. Here we use the following c_{err} function, where $f_+, f_- > 0$ denote different marginal costs for over- or under-stating the path length, respectively:

$$c_{\text{err}}(\ell^{\text{true}}, \ell^{\text{obs}}) = \begin{cases} f_+(\ell^{\text{obs}} - \ell^{\text{true}}) & \text{if } \ell^{\text{obs}} \geq \ell^{\text{true}} \\ f_-(\ell^{\text{true}} - \ell^{\text{obs}}) & \text{if } \ell^{\text{obs}} < \ell^{\text{true}} \end{cases} \tag{2.6}$$

Finally, cost 3 occurs if the attack is successful. The defender has a parameter $\lambda \geq 0$ that denotes the cost of attacker success. The cost to the defender is as follows, where $p^* = p(G', w', s_{p^*}, t_{p^*})$ only if p^* is the *unique* shortest path from s_{p^*} to t_{p^*}:

$$L_s = \lambda \Pr[p^* = p(G', w', s_{p^*}, t_{p^*})]. \tag{2.7}$$

If the only cost of an attack is the direct disruption to users accounted for in L_d and L_e, then the defender sets $\lambda = 0$. A pseudocode description of an algorithm for computing the cost is in our longer manuscript [19].

3 Optimization

We begin by formulating the optimization to solve Cut Defense. We then define a zero-sum version in which the defender's goal is to reduce cost given that the probability of attack is minimized. We propose a heuristic method that results in a feasible solution for a single target path, then extend its usage to multiple target paths. We finally derive a linear program for local optimization around a feasible point.

3.1 Non-Convex Optimization Formulation We optimize cost while varying perturbed weights. Let $\mathbf{w} \in \mathbb{R}^{|E|}_{\geq 0}$ be the vector of original edge weights, where each edge is given an arbitrary index corresponding to its vector entry. The vector $\mathbf{w}'$ contains the perturbed weights, $\mathbf{c}$ contains edge removal costs, and $\mathbf{x}_p$ is a binary indicator vector for path p, i.e., if the ith edge is in path p, the ith entry in $\mathbf{x}_p$ is 1, otherwise it is 0. Let $P(u, v)$ be the set of all paths from u to v, $q_{uv} = \Pr_{D \sim \mathcal{D}}[D = (u, v)]$ be the probability that a randomly selected user travels from u to v, $P_t = \text{support}(\mathcal{P})$ be all paths with nonzero probability of being the target, W be a large value to denote edge removal, and $\mathcal{X}(G, \mathbf{w}, p^*)$ be the set of attacks against graph G with weights $\mathbf{w}$ to make p^* be the shortest path between its terminal nodes. We solve Cut Defense by optimizing as follows:

$$\hat{\mathbf{w}}' = \arg\min_{\mathbf{w}'} \lambda(1 - z_\emptyset) + \sum_{u,v \in V} L_d(u, v) + L_e(u, v) \tag{3.8}$$

$$\text{s.t. } L_d(u, v) = q_{uv} \cdot \sum_{p^* \in P_t \cup \{\emptyset\}} z_{p^*} \cdot \ell^{\text{true}}_{uv, p^*} \quad \forall u, v \in V \tag{3.9}$$

$$L_e(u, v) = q_{uv} \cdot \sum_{p^* \in P_t \cup \{\emptyset\}} z_{p^*} \cdot L_e(u, v, p^*) \quad \forall u, v \in V \tag{3.10}$$

$$L_e(u,v,p^*) = \left(f_+ \cdot d^{\text{pos}}_{uv,p^*} + f_- \cdot d^{\text{neg}}_{uv,p^*}\right) \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\} \tag{3.11}$$

$$\Delta_{p^*} \in \mathcal{X}(G, \mathbf{w}', p^*) \quad \forall p^* \in P_t \tag{3.12}$$

$$\Delta_\emptyset = \mathbf{0} \tag{3.13}$$

$$\mathbf{c}^\top \Delta_{p^*} \leq \mathbf{c}^\top \Delta \quad \forall \Delta \in \mathcal{X}(G, \mathbf{w}', p^*), p^* \in P_t \tag{3.14}$$

$$z_{p^*} = \Pr(p^*) \sum_{i \geq \mathbf{c}^\top \Delta_{p^*}} \Pr_{B \sim \mathcal{B}}[B = i] \quad \forall p^* \in P_t \tag{3.15}$$

$$z_\emptyset = 1 - \sum_{p^* \in P_t} z_{p^*} \tag{3.16}$$

$$z_{p^*} \geq 0 \quad \forall p^* \in P_t \cup \{\emptyset\} \tag{3.17}$$

$$p_{uv,p^*} = \arg\min_{p \in P(u,v)} \mathbf{x}_p^\top (\mathbf{w}' + W\Delta_{p^*}) \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\} \tag{3.18}$$

$$\ell^{\text{true}}_{uv,p^*} = \mathbf{x}^\top_{p_{uv,p^*}} \mathbf{w} \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\} \tag{3.19}$$

$$\ell^{\text{obs}}_{uv,p^*} = \mathbf{x}^\top_{p_{uv,p^*}} \mathbf{w}' \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\} \tag{3.20}$$

$$d^{\text{pos}}_{uv,p^*}, d^{\text{neg}}_{uv,p^*} \geq 0 \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\} \tag{3.21}$$

$$d^{\text{pos}}_{uv,p^*} - d^{\text{neg}}_{uv,p^*} = \ell^{\text{obs}}_{uv,p^*} - \ell^{\text{true}}_{uv,p^*} \quad \forall u,v \in V, p^* \in P_t \cup \{\emptyset\}. \tag{3.22}$$

Note that p_{uv,p^*}, defined in (3.18), is the shortest path from u to v according to the published weights after the attacker attacks when the target path is p^*. An explanation of each constraint is provided in our longer manuscript [19].

One potential concern when calculating the expected cost across pairs of nodes is the possibility that the graph could become disconnected, leaving some inter-node distances infinite. However, due to the following theorem, we do not need to be concerned about this possibility. A proof and a discussion of practical issues are provided in our longer manuscript [19].

THEOREM 3.1. *The optimal Δ_{p^*} in (3.14) will not disconnect G.*

3.2 Zero-Sum Formulation In the prior section, we assumed a non-zero-sum game in which the optima for the attacker and defender may coincide. We gain additional insight into the problem by considering the zero-sum version of the problem, in which the defender's primary goal is ensuring the attack does not occur. In this case, we are given the same information as in Cut Defense except the cost of attack success λ. Instead, the defender manipulates the weights w' to minimize the probability of attack, i.e.,

$$z_{\min} = \min_{w'} \sum_{p^* \in \mathcal{P}} \Big(\Pr_{P \sim \mathcal{P}}[P = p^*] \cdot \Pr_{B \sim \mathcal{B}}[c(E'(G, w', p^*)) \leq B] \Big). \tag{3.23}$$

Within the minimized attack probability, however, the defender wants the cost to be as low as possible. Thus,

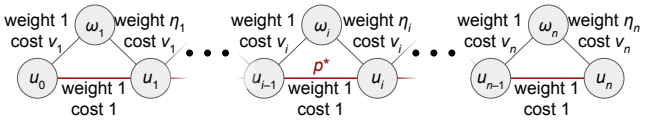


Figure 2: Reduction from Knapsack to Zero-Sum Cut Defense. The ith item in the set corresponds to a triangle $\{u_{i-1}, \omega_i, u_i\}$. All target paths go from $s = u_0$ to $t = u_n$. The target path p^* traverses the bottom edges on the figure, highlighted in red. Keeping defender cost low while ensuring the probability of attack is 0 is equivalent to keeping the knapsack's weight low while ensuring the value is sufficient.

the *Zero-Sum Cut Defense* problem is given by

$$\hat{w}' = \arg\min_{w'} L_d(G, w, w', \mathcal{B}, \mathcal{D}, \mathcal{P}) + L_e(G, w, w', \mathcal{B}, \mathcal{D}, \mathcal{P}) \tag{3.24}$$

$$\text{s.t. } z_{\min} = \sum_{p^* \in \mathcal{P}} \Big(\Pr_{P \sim \mathcal{P}}[P = p^*] \cdot \Pr_{B \sim \mathcal{B}}[c(E'(G, w', p^*)) \leq B] \Big). \tag{3.25}$$

Note that L_s is not considered in the objective in this formulation, since the attack probability is fixed at its minimum possible value. We show that this version of the problem is NP-hard.

THEOREM 3.2. *Zero-Sum Cut Defense is NP-hard.*

Proof Sketch. We prove NP hardness via reduction from the Knapsack problem. Given a set of n items with values $\nu_i \in \mathbb{Z}_+$ and weights $\eta_i \in \mathbb{Z}_+$, $1 \leq i \leq n$, and two thresholds U and H, the Knapsack problem is to determine whether there is a subset of items with total value at least U with weight no more than H. For each item, we create a triangle in a graph, where consecutive triangles share a node as shown in Fig. 2. The ith triangle consists of the nodes u_{i-1}, u_i, and ω_i. Let $s = u_0$ and $t = u_n$. We create a Zero-Sum Cut Defense instance in which the support of $\mathcal{P}$ consists of the single path from s to t that passes through no nodes ω_i for any $1 \leq i \leq n$. For all i, edge $\{u_{i-1}, u_i\}$ has weight 1 and removal cost 1, $\{u_{i-1}, \omega_i\}$ has weight 1 and cost v_i, and edge $\{\omega_i, u_i\}$ has weight w_i and cost v_i. The adversary's budget is $U - 1$ with probability 1. The defender's cost only considers traffic going from s to t, i.e., $\Pr_{(x,y) \sim \mathcal{D}}[(x,y) = (s,t)] = 1$. Let $f_+ = 1$ and $f_- = H' = \sum_{i=1}^n w_i$.

Since the adversary's budget is $U - 1$, in order to minimize the attack probability (in this case, make it 0), the defender must force some subset of edges along p^* to have length at least the same as the two-hop

paths running parallel to them. The removal costs on the parallel paths of these edges must sum to at least U: the defender's "value" is an increased cost for the attacker. The increase in distance traveled by the user will be commensurate with the weights of the items associated with the perturbed edges. This provides a direct mapping between solving Knapsack and solving Zero-Sum Cut Defense on the generated graph. A detailed proof is provided in our longer manuscript [19]. ∎

Although the problem cannot be efficiently solved in general, we find feasible points fairly easily by increasing the length of p^* until the cost of the attack is sufficiently high. Starting with weights w' initialized to the true weights w, the procedure is as follows:

1. $E' \leftarrow \text{attack}(G, w', p^*)$
2. $p \leftarrow$ 2nd shortest path between the terminals of p^*, if it exists
3. pick an edge e from $E_{p^*} \setminus E_p$
4. increase $w'(e)$ by $\delta = \ell(G, w', p) - \ell(G, w', p^*)$

Here E_p is the set of edges on path p. This procedure continues until either p^* is the longest path between its terminals or $c(E')$ exceeds the largest possible attack budget. This procedure yields a feasible point given the attacker's algorithm. If we continue until p^* is the longest path, we must be at a feasible point: all other paths that connect p^*'s endpoints need to be cut. This observation yields the following theorem.

THEOREM 3.3. *When $\mathcal{P}$ consists of a single path p^*, the procedure above yields a feasible point for Zero-Sum Cut Defense.*

While having multiple possible target paths complicates the problem, we use a similar principle to reduce the probability of attack. For each target path, we increase the edge weights as described above. We then apply the attack and use the number of edges removed to calculate the attack probability. Then, starting from the original graph, we consider the target paths in order of increasing attack probability. We then increase the edge weights on each target path again, accumulating the new weights each time. This prioritizes the path at the end of the sequence, which has the highest probability of resulting in an attack.

3.3 Heuristic Method From the zero-sum case, we see that increasing the weights on target paths is an effective strategy. Taking this as inspiration, we propose a heuristic algorithm that iteratively chooses an edge e from some target path p^* and increments its weight to add another path to P_{p^*}. We call this algorithm `PATHDEFENSE` (see Algorithm 2). At each iteration, the algorithm considers edges on which the smallest possible weight increase will provide one target path with a new competing path. For a given target path p^*, these edges are identified by applying the attack and finding the second-shortest path p between the source and destination of p^*, if such a path exists. The edges in p^* that are not part of p may be incremented to add p as a competing path that must be cut to make p^* shortest (see Algorithm 1). The attack probability is evaluated after considering each possible perturbation, and whichever perturbation results in the smallest attack probability is kept. If multiple perturbations result in the same attack probability, the edge is chosen that maximizes the average length of p^*. This procedure continues until (1) all target paths are the longest between their terminals, or (2) a threshold is reached in terms of cost, attack probability, or number of iterations.

Algorithm 1 `get_edge_increments`

Input: graph $G = (V, E)$, perturbed weights w', p^* dist. $\mathcal{P}$
Output: weight increment set R

```
R ← ∅
for all p* ∈ P_t do
    E_temp ← attack(G, w', p*)
    G' ← (V, E \ E_temp)
    if p* is not the only path from s to t in G' then
        p ← 2nd shortest path from s to t in G' using w'
        δ ← ℓ(G', w', p) − ℓ(G', w', p*)
        for all e ∈ E_p \ E_p* do
            R ← R ∪ {(e, δ)}
        end for
    end if
end for
return R
```

3.4 Local Optimization Around a Feasible Point Once a feasible point is identified, we relax the hardest constraints to formulate a linear program for local optimization. In this case, we fix the attack that occurs for each p^*, and ensure that the observed shortest path between each pair of nodes remains the same as the weights are varied. By fixing the attack, we are given a value for Δ_{p^*} and z_{p^*}, thus removing constraints (3.12)–(3.17) from the nonconvex optimization in Section 3.1. By fixing the shortest path, we are given a value for p_{uv,p^*}, replacing constraint (3.18) with

$$\ell^{\text{obs}}_{uv,p^*} \leq \mathbf{x}_p^\top (\mathbf{w}' + W\Delta_{p^*}) \quad \forall u, v \in V, p^* \in P_t \cup \{\emptyset\}, p \in P(u, v). \tag{3.26}$$

Algorithm 2 PATHDEFENSE Heuristic Defense Algorithm

Input: graph $G = (V, E)$, true weights w, budget dist. $\mathcal{B}$, p^* dist. $\mathcal{P}$, (u, v) pair dist. $\mathcal{D}$, attack cost λ, cost threshold ϵ_c, attack prob. threshold ϵ_a, max. iterations $i_{\max}$
Output: perturbed weights w'

$w' \leftarrow w$
repeat
  $R \leftarrow$ `get_edge_increments`$(G, w', \mathcal{P})$
  $(e_{\min}, \delta_{\min}, z_{\min}, \ell_{\min}) \leftarrow (\emptyset, 0, 2, 0)$
  for all $(e, \delta) \in R$ **do**
    $w'(e) \leftarrow w'(e) + \delta$
    $z \leftarrow 0$
    for all $p^* \in P_t$ **do**
      $E_{\text{temp}} \leftarrow \text{attack}(G, w', p^*)$
      $z \leftarrow z + \Pr_{P\sim\mathcal{P}}[P = p^*] \Pr_{B\sim\mathcal{B}}[B \geq c(E_{\text{temp}})]$
      if $z < z_{\min}$ **or** $(z = z_{\min}$ **and** $\mathbb{E}_{P\sim\mathcal{P}}[\ell(G, w', P)] > \ell_{\min})$ **then**
        $e_{\min} \leftarrow e$
        $\delta_{\min} \leftarrow \delta$
        $z_{\min} \leftarrow z$
        $\ell_{\min} \leftarrow \mathbb{E}_{P\sim\mathcal{P}}[\ell(G, w', P)]$
      end if
    end for
    $w'(e) \leftarrow w'(e) - \delta$
  end for
  $w'(e_{\min}) \leftarrow w'(e_{\min}) + \delta_{\min}$
  $L_d, L_e, L_s \leftarrow \text{cost}(G, w, w', \mathcal{B}, \mathcal{P}, \mathcal{D}, \lambda)$
  $i \leftarrow i + 1$
until $L_d + L_e + L_s < \epsilon_c$ **or** $z_{\min} < \epsilon_a$ **or** $|R| = 0$ **or** $i \geq i_{\max}$
return w'

All remaining constraints in the nonconvex program are linear. This is not, however, sufficient to locally optimize: The attack Δ_{p^*} must be both necessary (not cut superfluous edges) and sufficient (cut all paths that compete with p^*). To optimize within this context, we add the constraints

$$\mathbf{x}_{p^*}^\top \mathbf{w}' \leq \mathbf{x}_p^\top (\mathbf{w}' + W\Delta_{p^*}) - \epsilon_{p^*} \quad \forall p^* \in P_t, p \in P(s_{p^*}, t_{p^*}) \tag{3.27}$$

$$\mathbf{x}_{p^*}^\top \mathbf{w}' \geq \mathbf{x}_p^\top \mathbf{w}' \quad p^* \in P_t, p \in P_{p^*} \tag{3.28}$$

Here P_{p^*} is the set of paths competing with p^* to be the shortest path from s_{p^*} to t_{p^*}. To ensure sufficiency, (3.27) constrains all paths between the terminals of a target path p^* to be strictly longer than p^*. The additional variables ϵ_{p^*} may be measured based on the difference in lengths between p^* and the second-shortest path after the attack. Constraint (3.28) ensures necessity by making all paths that competed with p^* at the feasible point remain competitive. Since all constraints are linear, we use constraint generation to explicitly state only a subset of the necessary constraints [4].

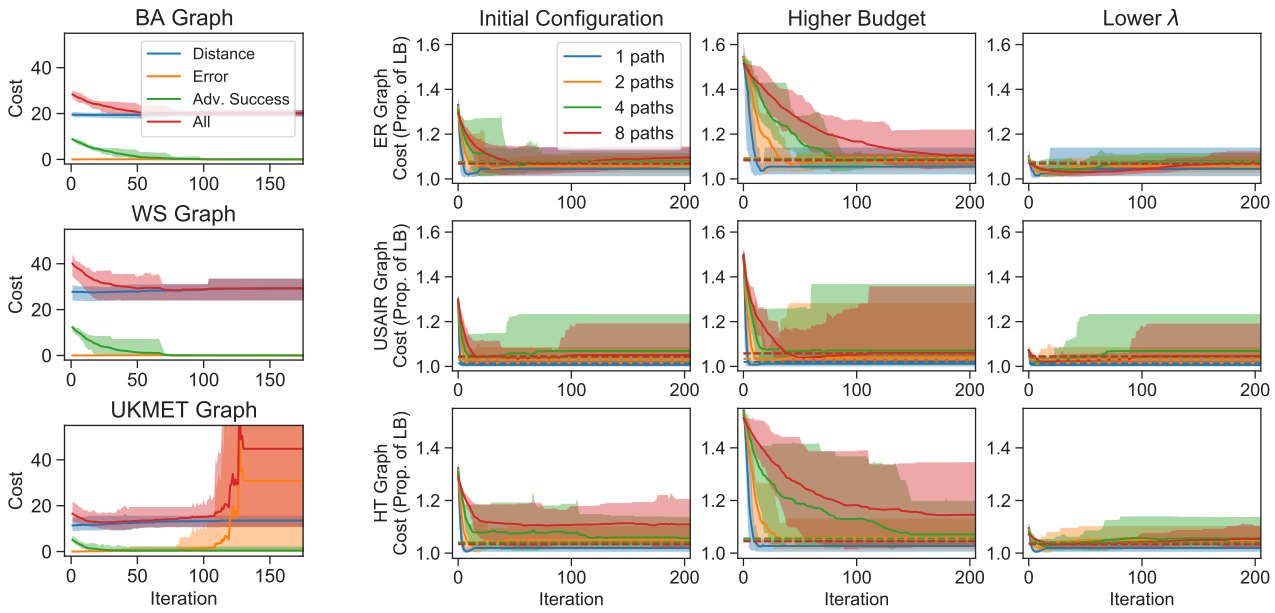
4 Experiments

We demonstrate the optimization procedure with 4 synthetic network generators and 4 real networks. All synthetic networks have 250 nodes and an average degree of approximately 12. We use Erdős–Rényi (ER) random graphs, Barabási–Albert (BA) preferential attachment graphs, Watts–Strogatz (WS) small-world graphs, and stochastic blockmodel (SBM) graphs where nodes are separated into communities of size 200 and 50. To provide some variation in edge weights, all edges are given weights drawn from a Poisson distribution with rate parameter 20. Removal costs are set to 1 for all edges.

The real network datasets include 2 transportation networks, 1 social network, and 1 computer network. The transportation networks are United States airports (USAIR), where edge weights are the number of seats on flights between airports [9], and United Kingdom metro stops (UKMET), where weights are travel times between stops in minutes [13]. The social network is attendees at the 2009 ACM Hypertext Conference (HT), where weights are the number of face-to-face interactions during the conference [16]. The computer network is an autonomous system (AS) graph [17], with weights from a Poisson distribution as in the synthetic networks. Weights for USAIR and HT are inverted to create distances rather than similarities. Statistics and links to the datasets are in our longer manuscript [19].

4.1 Experimental Setup For each experiment on a given dataset, to evaluate the effects of varying the number of target paths, we choose 1, 2, 4, or 8 target paths. Source–destination pairs are chosen uniformly at random and the target paths include the 5th shortest and every second path thereafter until the desired number of targets is reached. In some cases, all target paths have the same terminal nodes; in others, we choose independently for each path. For SBM and AS graphs, we also consider the case where the two terminal nodes are from one community of nodes, but the target path traverses nodes in another one. (We call this an *extra-community* path.) This emulates a scenario where an outside attacker wants the traffic to take a relatively unnatural path, e.g., computer traffic unnecessarily crossing national boundaries.

In each experiment, $\mathcal{B}$ is a Poisson distribution whose rate parameter is the average number of edges removed by the attack across all target paths. For the attack, we use the version of PATHATTACK that solves a relaxed linear program and randomly rounds to get an integer solution [21]. The source–destination distribution emphasizes the portions of the graph with target paths: with probability 0.5, we draw two nodes both either on a target path or on the true shortest



(a) (b)

Figure 3: Cost of PATHDEFENSE when all target paths share terminals. Lower cost is better for the defender. Plots include the average cost (solid line) and the cost range across trials (shaded area). (a) Cost broken down by component for BA, WS, and UKMET graphs with 4 target paths. There is a substantial reduction in cost due to the probability of adversary success being reduced, and the cost due to errors in published distances is minimal for BA and WS, whereas L_e increases a substantial amount in the UKMET data, as it is difficult to avoid traversing perturbed edges. (b) Defender costs, normalized by a lower bound, for ER (top), USAIR (middle), and HT (bottom) graphs. Results are shown for the original budget and λ (left), when the attacker budget is doubled (center), and when λ is reduced by five times (right). The average zero-sum result is also shown (dashed line). As expected, increasing the adversary's budget results in slower convergence, and decreasing the attack success cost reduces the improvement provided by PATHDEFENSE.

path between its endpoints, and with probability 0.5 we do not. (Pairs are uniformly distributed within each category.) For each setting, results are aggregated across 10 trials. The cost of attacker success λ is set to one half of the cost based on distances alone when there is no attack. We used a CentOS Linux cluster with 32 cores per machine for our experiments. Each job was allocated 10 GB of memory. We used Python 3.8.1 with Gurobi 9.5.1 (`https://www.gurobi.com`) for optimization and NetworkX 2.4 (`https://networkx.org`) for graph analysis.

4.2 Results We first consider how the three components of the defender's cost vary when running PATHDEFENSE. Representative results are shown in Fig. 3(a). Cost is typically dominated by the true distance traveled by users. While the traffic is primarily on the portion of the graph affected by the attack, the impact of errors is negligible in comparison. One reason for this phenomenon is that increasing edge weights discourages their use: when a path looks longer, fewer users will take it and it will not be considered in the cost. In the UKMET case, however, this changes after about 100 iterations, when the cost from errors drastically increases. The metro graph is somewhat tree-like, making it difficult to avoid traversing perturbed edges. In all cases, the overall reduction in cost comes from a large reduction in the probability of attack and a small increase in the average distance traveled.

The cost of PATHDEFENSE for three additional datasets is shown in Fig. 3(b). The plots include cases where the rate parameter of the budget distribution is doubled and where the cost of adversary success is reduced by a factor of five. We report all costs as a proportion of a lower bound: the cost when there is no attack and no perturbation. When the attacker's budget is doubled, the initial defender cost is much larger, but the cost eventually obtained by PATHDEFENSE is very similar. It is typical for PATHDEFENSE to outperform the zero-sum case at an early iteration when there

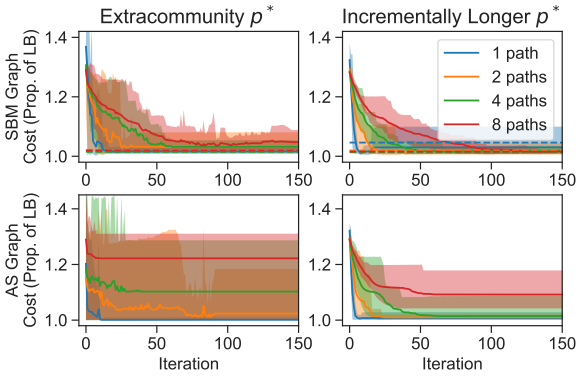


Figure 4: Results when the attacker targets an extra-community path (left), in comparison to when targets are incrementally longer short paths (right), in SBM (top) and AS (bottom) graphs. Plots include the average cost (solid line) and the cost range across trials (shaded area), as well as the average zero-sum result (dash line). (The zero-sum procedure is only reported for SBM; it did not complete within 24 hours for AS.) PATHDEFENSE yields lower relative cost in the case where the target path is incrementally longer than the true shortest path than in the extra-community case.

are few target paths, though this does not always happen. When there are 8 target paths in the HT graph, the zero-sum procedure produces a better result than PATHDEFENSE. This may be due to the clustering that exists in the social network: it may promote oscillation between competing paths, whereas the zero-sum method focuses on one path at a time. The results on all datasets and the results that compare with a simple baseline defense using alternative attack methods are provided in our longer manuscript [19].

Fig. 4 shows results where the attacker targets a path that exits and re-enters a community. The lowest relative cost is higher in this case than when paths are chosen by enumerating consecutive shortest paths, which is consistent with intuition. In the SBM graph with extra-community target paths, we again see that the zero-sum method yields lower cost than PATHDEFENSE, suggesting that optimizing each target path in sequence is effective in this case as well.

5 Related Work

The problem of releasing a graph that can be useful while not revealing sensitive information has received much attention since the discovery of the problem of deanonymization [3]. Much of this research has been concerned with privacy-preserving release of social network data, where nodes are anonymized w.r.t. topological features such as degree [18], neighborhood [25], or cluster [5]. Sharing of sensitive graph data has been studied in the context of differential privacy [10]. Sealfon [22] applied differential privacy to graph weights in the context of computing shortest paths. Other methods have considered differential privacy for unweighted graphs. For example, a data-driven low-dimensional projection [2] and random low-dimensional projections [6] have been applied for cut queries, i.e., calculating the number of edges that must be removed to disconnect two sets of vertices. Other work does not necessarily preserve distances between pairs of nodes, but maintains the distribution of distances [8].

Outside of differential privacy, there is work on reliably finding shortest paths when a graph is located on an untrusted server [14]. In other work, an actor wants to "buy" a path from s to t, with the prices only known to current owners [1]. In this setting, a buyer can be forced to overpay for the path [11], which is similar to the Cut Defense goal of forcing an attacker to expend extra resources, though in a different data access mechanism. Recent work in adversarial machine learning has expanded to attacking graph structure to alter vertex classification outcomes [26] or vertex embeddings [7]. Other work has aimed to defend against such attacks by, for example, using a low-rank representation of the graph [12] or filtering based on node attribute similarity [23].

6 Conclusion and Future Work

We present a framework and algorithms for defending against shortest path attacks. We formulate the defense as a Stackelberg game in which the defender alters the weights of the graph before the attacker removes edges to make the target path shortest. The defender's cost includes components to limit the average distance traveled by users, the error in the published distances, and the probability of attacker success. We show that the zero-sum version of this problem is NP-hard and provide a greedy edge weight increment procedure to find a feasible point. Using this same procedure, we propose the PATHDEFENSE algorithm and apply it to several real and synthetic datasets. Across a wide set of experiments, we observe that PATHDEFENSE reduces the attack probability to a negligible level (typically less than 10^{-6}) while only slightly increasing the cost borne by users (by less than 5% in over 87% of cases).

There is currently no performance guarantee for PATHDEFENSE. Future work in this area should focus on obtaining such a guarantee, perhaps by narrowing the scope of the problem (e.g., single target path, weights only increased). Lastly, using a parallel method that exploits sparseness [24] could be key to applying PATHDEFENSE at scale.

Acknowledgement

BAM was supported by the United States Air Force under Contract No. FA8702-15-D-0001. TER was supported in part by the Combat Capabilities Development Command Army Research Laboratory (under Cooperative Agreement No. W911NF-13-2-0045) and by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. YV was supported by grants from the Office of Naval Research (N00014-24-1-2663) and National Science Foundation (CAREER Award IIS-1905558, IIS-2214141, and CNS-2310470). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

References

[1] A. Archer and É. Tardos, *Frugal path mechanisms*, ACM Transactions on Algorithms, 3 (2007), pp. 1–22.

[2] R. Arora and J. Upadhyay, *On differentially private graph sparsification and applications*, NeurIPS, (2019).

[3] L. Backstrom, C. Dwork, and J. Kleinberg, *Wherefore art thou R3579X? Anonymized social networks, hidden patterns, and structural steganography*, in WWW, 2007, pp. 181–190.

[4] W. Ben-Ameur and J. Neto, *A constraint generation algorithm for large scale linear programs using multiple-points separation*, Math. Program., 107 (2006), pp. 517–537.

[5] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, *Class-based graph anonymization for social network data*, in VLDB, 2009, pp. 766–777.

[6] J. Blocki, A. Blum, A. Datta, and O. Sheffet, *The Johnson-Lindenstrauss transform itself preserves differential privacy*, in FOCS, 2012, pp. 410–419.

[7] A. Bojchevski and S. Günnemann, *Adversarial attacks on node embeddings via graph poisoning*, in ICML, 2019, pp. 695–704.

[8] R. Chen, B. Fung, P. S. Yu, and B. C. Desai, *Correlated network data publication via differential privacy*, The VLDB Journal, 23 (2014), pp. 653–676.

[9] V. Colizza, R. Pastor-Satorras, and A. Vespignani, *Reaction–diffusion processes and metapopulation models in heterogeneous networks*, Nature Physics, 3 (2007), pp. 276–282.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, in TCC, 2006, pp. 265–284.

[11] E. Elkind, A. Sahai, and K. Steiglitz, *Frugality in path auctions*, in SODA, 2004, pp. 701–709.

[12] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, *All you need is low (rank): Defending against adversarial attacks on graphs*, in WSDM, 2020, pp. 169–177.

[13] R. Gallotti and M. Barthelemy, *The multilayer temporal network of public transport in Great Britain*, Scientific Data, 2 (2015), pp. 1–8.

[14] J. Gao, J. X. Yu, R. Jin, J. Zhou, T. Wang, and D. Yang, *Neighborhood-privacy protected shortest distance computing in cloud*, in SIGMOD, 2011, pp. 409–420.

[15] D. Goodin, *How 3 hours of inaction from Amazon cost cryptocurrency holders $235,000*, Ars Technica, (2022).

[16] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, *What's in a crowd? analysis of face-to-face behavioral networks*, Journal of Theoretical Biology, 271 (2011), pp. 166–180.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graphs over time: Densification laws, shrinking diameters and possible explanations*, in KDD, 2005, pp. 177–187.

[18] K. Liu and E. Terzi, *Towards identity anonymization on graphs*, in SIGMOD, 2008, pp. 93–106.

[19] B. A. Miller et al., *Defense against shortest path attacks*, arXiv preprint arXiv:2305.19083, (2023).

[20] B. A. Miller, Z. Shafi, W. Ruml, Y. Vorobeychik, T. Eliassi-Rad, and S. Alfeld, *PATHATTACK: Attacking shortest paths in complex networks*, in ECML PKDD, 2021, pp. 532–547.

[21] ———, *Attacking shortest paths by cutting edges*, ACM Transactions on Knowledge Discovery from Data, 18 (2023).

[22] A. Sealfon, *Shortest paths and distances with differential privacy*, in PODS, 2016, pp. 29–41.

[23] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, *Adversarial examples for graph data: Deep insights into attack and defense*, in IJCAI, 2019, pp. 4816–4823.

[24] S. Yang, X. Liu, Y. Wang, X. He, and G. Tan, *Fast all-pairs shortest paths algorithm in large sparse graph*, in ICS, 2023, p. 277–288.

[25] B. Zhou and J. Pei, *Preserving privacy in social networks against neighborhood attacks*, in ICDE, 2008, pp. 506–515.

[26] D. Zügner, A. Akbarnejad, and S. Günnemann, *Adversarial attacks on neural networks for graph data*, in KDD, 2018, pp. 2847–2856.