Defender Stackelberg Game with Inverse Geodesic Length as Utility Metric

Haris Aziz
Data61 and UNSW Sydney, Australia
haris.aziz@data61.csiro.au

Serge Gaspers
Data61 and UNSW Sydney, Australia
sergeg@cse.unsw.edu.au

Edward J. Lee
Data61 and UNSW Sydney, Australia
e.lee@unsw.edu.au

Kamran Najeebullah
Data61 and UNSW Sydney, Australia
kamran.najeebullah@data61.csiro.au

ABSTRACT

The inverse geodesic length (IGL) is a well-known and widely used measure of network performance. It equals the sum of the inverse distances of all pairs of vertices in the network. A Stackelberg game is a strategic game in which one player commits to a strategy while taking into account that other players will respond accordingly. We propose a natural defender-attacker Stackelberg game on a network in which the defender wants to maximize the IGL level of the network and commits to protecting parts of the network while having knowledge of the strength of an attacker that wants to weaken the network. We present several algorithmic and complexity results concerning the problem of finding the optimal commitment for the defender. Some of our computational hardness results also answer open problems posed in prior work on IGL.

KEYWORDS

Network Analysis; Multi-agent Systems; Fixed Parameter Tractability; Network Vulnerability

ACM Reference Format:
Haris Aziz, Serge Gaspers, Edward J. Lee, and Kamran Najeebullah. 2018. Defender Stackelberg Game with Inverse Geodesic Length as Utility Metric. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Networks are ubiquitous, most critical infrastructure systems can be modeled as a network. Examples of such systems include transportation, power, water (drinking and irrigation), sanitation, communication and the Internet. Security of these infrastructure is of great importance requiring them to be robust against random failures or intentional attacks [1]. Unfortunately, security resources are often limited, dictating the need to optimize their use.

Strategic aspect of network analysis has emerged as an important area of research in many fields including AI (see e.g [2, 27]). The focus here is to identify the nodes/links that are most critical for a high performance of a network [2, 5, 26, 41]. The applications are not only limited to security [2] but also reach as far as epidemiology, sociology, physics and logistics (see e.g., [17, 21, 26, 41]).

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

One important issue in network analysis is how to quantify the network performance. Some of the frequently used measures of network performance include *component order connectivity* (size of the largest connected component) [10, 13], *clustering coefficient* (probability that two nodes are connected given that both are connected to a common third node) [40] and *inverse geodesic length (IGL;* sum of the inverse distances between all pairs of vertices). We opt to quantify the network performance by IGL. Formally, $IGL(G) = \sum_{\{u,v\} \subseteq V, u \neq v} \frac{1}{d(u,v)}$. Our choice is influenced by the frequent use of IGL as a measure of network performance across various fields including AI (e.g., [2]), network security (e.g., [17]), social network (e.g., [28]) and game theory (e.g., [17, 25, 32]). Moreover, Latora and Marchiori [23] found IGL to be effective on small-world graphs and studied several networks systems to show that it is the underlying general principle of construction for several real-world networks including transportation, communication and neural networks.

Game-theory provides a suitable setting to analyze the security of a network, facilitating adversarial reasoning [15]. In particular, *security games* based on the *Stackelberg leadership model (Stackelberg games)* (e.g., [19, 35]) have been of significant interest. In fact, modeling such adversarial security scenarios by Stackelberg games is a well-established and widely-used approach in the theory and practice of security games [33]. In such games a defender commits to an optimal surveillance or protection strategy so as to defend some infrastructure such as an airport [33]. One particular setting in these games is where an infrastructure is defined as a network and the players perform their set of operations on the nodes and edges of the network (see e.g., [16, 39]). The objective here mostly is to minimize damage to the network by computing an optimal allocation of security resources [31, 36].

In this paper, we take IGL as a well-established global measure of network performance and consider a strategic scenario based on Stackelberg leadership model. More precisely, we define a Stackelberg game involving two players—d (defender) and x (attacker). The attacker x wants to weaken the network G by reducing its IGL and the defender d wants to minimize the effect of the attack. We suppose that the defender and attacker have budgets k_d and k_x. The defender's set of actions is to protect a number of vertices k_d while the attacker's set of actions is to remove a given number of vertices k_x. A vertex can only be removed if it is not protected. If the defender d was to protect an optimal set of vertices S_d, this set would be of size k_d and S_d would be such that whichever set S_x of k_x vertices that the attacker x deletes, the

IGL value of the remaining graph would still be at least a certain threshold value T. We model the computation of the optimal set of vertices that the defender protects by the following computational problem Defender-Stackelberg Game (DeStackIGL).

Defender Stackelberg Game (DeStackIGL)	
Input:	A graph G, integers k_d, k_x and a target inverse geodesic length T.
Question:	Does there exist a set of vertices S_d with $\lvert S_d\rvert = k_d$ such that $IGL(G-S_x) \geq T$ for all $S_x \subseteq V(G) \setminus S_d$ with $\lvert S_x\rvert = k_x$?

This set S_d gives a strategy for the defender against any potential future attack on k_x vertices. Now, given that the defender protects some vertices, we consider the computational problem of the attacker, called RMinIGL. In RMinIGL, the defender's choice is provided as an input and the attacker's choice needs to be computed.

Restricted MinIGL (RMinIGL)	
Input:	A graph G, an integer k, a set of vertices S_d and a target inverse geodesic length T.
Question:	Does there exist a set of vertices $X \subseteq V(G) \setminus S_d$ with $\lvert X\rvert = k$ such that $IGL(G - X) \leq T$?

This problem is akin to the situation where the defender has already decided which vertices to protect and it is now the attacker's turn to play. For some of our algorithms, we first design an algorithm for RMinIGL and use it as a subroutine to solve DeStackIGL. We also note that RMinIGL generalizes MinIGL, which is the special case where $S_d = \emptyset$ and was studied in [2].

Contributions. We conduct a comprehensive (parameterized) complexity analysis of DeStackIGL. First, we observe that DeStackIGL is co-NP-hard even when restricted to bipartite and split graphs by a straightforward reduction from MinIGL. In terms of parameterized complexity, we give several contrasting results: DeStackIGL is co-W[1]-hard for parameter k_x, even when k_d is a constant, but it is FPT for parameter k_d when k_x is a constant; DeStackIGL is W[1]-hard for parameter $k_d + T$ but FPT for parameter $k_x + T$. As for structural parameters, DeStackIGL is FPT when the parameter is k_d plus either the vertex cover number or the neigborhood diversity of the input graph, but it is W[1]-hard when the parameter is the treewidth of the input graph and $k_d = 0$. Our results for the vertex cover number and the neigborhood diversity use a recent FPT result for *integer quadratic programming*. On the course to obtain these results we also address some of the open problems in [2]. In particular, we show that MinIGL is FPT for parameter neighborhood diversity while it is W[1]-hard for parameter $tw(G)$.

Related Work. Our paper is related to two strands of research: (A) *network analysis* in particular considering the connectivity or robustness of a network [3], and (B) *security games* in which a defender executes an optimal surveillance or protection strategy so as to defend some infrastructure [33]. Whereas (A) is studied in physics, computer science, and social sciences, (B) is an area of applied game theory, AI, and operations research. There are several works that either consider (1) algorithmic aspects of network connectivity [29], or they focus on (2) security games including those in which defenders, attackers, and infrastructure facilities are on a network [16, 18, 20, 31, 35, 36, 39]. However, ours is the first work on Stackelberg strategies for defending a network with IGL as the global measure of the network. We note here that our algorithmic work using a game-theoretic approach may be of independent interest to finding sets of influential nodes in a network. In the case of our model, this set would the one that the defender protects.

Some works that are closely related are ones where infrastructures are defined as a network on which defender and attacker(s) can perform their set of actions (see e.g., [16, 39]). In such games the objective is to minimize damage to a network by computing a schedule of security resources based on a mixed strategy, eliminating the possibility of human errors [31, 36]. The utility the players achieved is based on the value they have for the certain resources in the network. In our work, we take IGL as the value that the players want to affect.

In several projects on both the theory and applications in security games, both the defender and attacker are allowed to use mixed strategies. In this paper, we focus on pure actions as a stepping stone to understand the algorithmic and complexity aspects of the underlying problems. Pure strategies are also natural in this setting where a defender might need to implement her strategies before a potential attacker decides where to attack.

2 PRELIMINARIES

Graph Theory. Let $G = (V, E)$ be an undirected, simple graph. We denote the set of vertices and edges in G as $V(G)$ and $E(G)$, respectively, with $n = |V|$ and $m = |E|$. For graph terminologies not defined here we refer to [8]. Let $u, v \in V$ with $u \neq v$. An edge xy is *incident* to v if $v \in \{x, y\}$. We say that u and v are *adjacent* or *neighbors* if $uv \in E(G)$. We denote the *neighborhood* of v, i.e., the set of vertices adjacent to v, as $N(v)$. The *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. The closed neighborhood of a vertex set S is $N[S] = \bigcup_{v \in S} N[v]$, and its open neighborhood is $N(S) = N[S] \setminus S$. A *path* between u and v is an alternating sequence of vertices and edges that starts with u and ends with v, with each edge in the path being adjacent to the preceding and succeeding vertex, and each vertex occurring at most once in the sequence. The *length* of a path is its number of edges. The *distance* between u and v, denoted by $d(u, v)$, is the length of a shortest path between u and v. The *ith neighborhood* of a vertex v is the set of vertices at distance i from v and is denoted by $N^i(v)$. If there is no path between u and v then their distance is infinite. A pair of vertices at finite distance is *connected*. Let $S \subseteq V$. A graph induced on S is denoted as $G[S]$, i.e., $V(G[S]) = S$ and $E(G[S]) = \{uv \in E : u, v \in S\}$. Similarly, we denote by $G - S$ the graph $G[V \setminus S]$. A *connected component* of G is a maximal subgraph of G where each vertex pair is connected.

Two vertices u and v are *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. Vertices u, v are *true twins* if they are twins and $uv \in E(G)$. A graph G has neighborhood diversity η, if there exists a partition of $V(G)$ into at most η sets, such that all the vertices in each set are twins; such a partition is called the neighborhood partition of G and can be computed in polynomial time [22]. A *vertex cover* of a graph G is a set of vertices such that each edge is incident to at least one of these vertices. The *vertex cover number* of a graph is the smallest size of a vertex cover of the graph.

A *tree decomposition* of a graph is a set of bags connected in a tree-like fashion, where each bag is a subset of vertices of the graph, each edge of the graph is contained in at least one bag and the bags containing a specific vertex of the graph form a (connected), non-empty subtree of the tree.

Game Theory. A 2-player game is defined by a tuple (N, A, u). The set $N = \{1, 2\}$ denotes the set of players. The term $A = A_1 \times A_2$ denotes the set of action profiles of the players where A_1 and A_2 are the action sets of player 1 and 2 respectively. Finally, $u = (u_1, u_2)$ is the utility function profile of the players where $u_j : A \to \mathbb{R}$ is an utility function for player $j \in N$. A Stackelberg game is a strategic game in which one player is defined as the leader who can make a decision and commit to a strategy before other players who are defined as followers. We focus on a two-player zero sum game. In such games, for all action profiles $a \in A$, $u_1(a) + u_2(a) = 0$. We focus on pure strategies for both the defender and attacker. As is standard in Stackelberg games modelling security scenarios, we will typically consider player 1 as the defender (d) and player 2 as the attacker(x).

Parameterized Complexity. A parameterized decision problem Π is in *FPT (Fixed Parameter Tractable)*, if there is an algorithm solving any instance x with parameter k in time $f(k) \cdot |x|^c$, where $f(k)$ is a computable function of k and c is a constant. A parameterized reduction from a parameterized decision problem Π_1 to a parameterized decision problem Π_2 is an algorithm, which, for any instance I of Π_1 with parameter k produces an equivalent instance I' of Π_2 with parameter k' such that there exists a computable function f such that $k' \leq f(k)$ and the running time of the algorithm is $f(k) \cdot |I|^{O(1)}$. The complexity class $W[1]$ is a class of parameterized decision problems closed under parameterized reductions. It is unlikely for a $W[1]$-hard problem to be in FPT [9]. A para-NP-hard problem is NP-hard even for constant values of the parameter. We refer to [7, 9] for a detailed exposition of parameterized complexity. A *kernel*, or *kernelization* algorithm for a parameterized problem is a polynomial time algorithm producing an equivalent instance of the same parameterized problem such that the size of the resulting instance is upper bounded by a function of the input parameter.

3 ALGORITHMS AND COMPLEXITY

For any instance (G, k, T) of MinIGL an instance (G, k_d, k_x, T') of DeStackIGL can be obtained, by setting $k_d = 0$, $k_x = k$ and $T' = T + \epsilon$, where $0 < \epsilon \leq \frac{1}{2}$, that is equivalent to the complement of MinIGL. This implies that the complexity results known for MinIGL also apply to DeStackIGL.

Theorem 3.1. DeStackIGL is co-NP-hard even when restricted to bipartite or split graphs.

We provide a parameterized reduction from Maximum Partial Vertex Cover (MaxPVC) problem to show that DeStackIGL is co-W[1]-hard for parameters k_d and k_x combined. In MaxPVC we are given a graph G and integers k and t, and the question is whether there exists a set $S \subseteq V(G)$ such that $|S| \leq k$ and at least t edges are incident to at least one vertex in S. MaxPVC is know to be W[1]-hard when parameterized by k [14]. The following theorem shows that DeStackIGL is co-W[1]-hard for parameter k_x even when k_d is a constant.

Theorem 3.2. DeStackIGL is co-W[1]-hard for parameter k_x, even when $k_d = 1$.

Proof. Let (G, k, t) be an instance of MaxPVC with $n = |V(G)|$ vertices and $m = |E(G)|$ edges. We construct an instance (G', k_d, k_x, T) for DeStackIGL, where $V(G') = V(G) \cup \{u\}$, $E(G') = E(G) \cup \{uv_i | v_i \in V(G)\}$, $k_d = 1$, $k_x = k$ and $T = n + m + \epsilon - (k + t) + \frac{1}{2}\left(\frac{(n-k)(n-k-1)}{2} - (m - t)\right)$ for some ϵ with $0 < \epsilon \leq \frac{1}{2}$. We show that (G, k, t) is a Yes-instance if and only if (G', k_d, k_x, T) is a No-instance. Let us assume that (G, k, t) is a Yes-instance and has a solution $S \subseteq V(G)$ with $|S| = k$. Clearly, in G' the best choice for d is to protect u, as $|N(u)| \geq |N(v_i)|$ for all $v_i \in V(G') \setminus \{u\}$ and u connects each pair in $V(G') \setminus \{u\}$. Thus, $S_d = \{u\}$. As u cannot be attacked, $\mathrm{dist}(v_i, v_j) \leq 2$ for every two vertices $v_i, v_j \in V(G')$, irrespective of the vertices attacked by x. This means that the optimal choice for x is to maximize the number of pairs of vertices at distance 2. Say x attacks the set of vertices $S_x = S$. But this means $IGL(G' - S_x) = n + m - (k + t) + \frac{1}{2}\left(\frac{(n-k)(n-k-1)}{2} - (m - t)\right) < T$. Thus (G', k_d, k_x, T) is a No-instance.

Conversely, let us assume that (G', k_d, k_x, T) is a No-instance. Note that u is the optimal choice for d, hence x must choose from $V(G') \setminus \{u\} = V(G)$. But this means that we can cover at least t edges by selecting a set of vertices S of size k in G otherwise (G', k_d, k_x, T) is a Yes-instance. Hence (G, k, t) is Yes-instance. □

We now show that DeStackIGL becomes FPT for parameter k_d when k_x is bounded by a constant. Given an instance (G, k_d, k_x, T) of DeStackIGL, let $\mathcal{S}$ be the set of all vertex subsets of size k_x in G. Notice that in time $O(n^{k_x})$ we can find all candidate sets $S_{x_i} \in \mathcal{S}$ and compute their IGL impact I where $I(S_{x_i}) = IGL(G) - IGL(G - S_{x_i})$. We sort $S_{x_i} \in \mathcal{S}$ in decreasing order of their IGL impact.

Next, we iteratively define a series of k_x-hitting set problem instances with universe U_i and collection R_i. In the Minimum d-Hitting Set problem the input is a collection of subsets R, each of size at most d, of a finite universe U and an integer k, and the question is whether there exists a set $H \subseteq U$ with $|H| \leq k$ such H contains at least one element from each subset in R. In each iteration i, we perform the following two steps;

(1) We define an instance of k_x-hitting set by setting $R_i = \{S_1, S_2, \cdots, S_{i-1}, S_i\}$ and setting $U_i = \bigcup_{S_i \in R_i} V(S_i)$, where $S_1, S_2, \cdots, S_i$ are chosen in order from $\mathcal{S}$.
(2) We compute a hitting set of size k_d for the instance, if one exists, using an FPT algorithm parameterized by k_d.

We iterate until a No-instance is found. It is well known that Minimum d-Hitting Set is FPT for parameter k when d is a constant [30], and the fastest known algorithms [12, 38] run in time $(d - 0.9245)^k |U|^{O(1)}$. Consequently, we have an algorithm with a running time of $n^{k_x} \cdot (k_x - 0.9245)^{k_d} \cdot n^{O(1)}$ for DeStackIGL.

Theorem 3.3. DeStackIGL is FPT for parameter k_d when $k_x \in O(1)$ and can be solved in time $n^{O(1)} \cdot (k_x - 0.9245)^{k_d}$.

Proof. Since $k_x \in O(1)$, all subsets of size k_x in G can be found in polynomial time. Similarly, using an all-pair shortest paths algorithm [34], the IGL impact of a subset can be computed in polynomial time. Clearly, since the number of sets in $\mathcal{S}$ is polynomial, they can be sorted in decreasing order of their IGL impact in polynomial time. It remains to show that once we have a sorted list of candidate sets $\mathcal{S}$, iteratively solving k_x-hitting set instances provides an optimal solution to DeStackIGL. First, consider the optimal choice of the attacker x. The attacker will choose a candidate set from $\mathcal{S}$ with maximum IGL impact that contains no unprotected vertex. This is because once the defender protects a vertex from a candidate set, the attacker cannot choose this exact same candidate set (note that the attacker can still choose other vertices from the candidate set, but all such choices are also separately contained in $\mathcal{S}$). Therefore, the defender's optimal choice is to protect at least one vertex from as many candidate sets with highest IGL impact as possible. This leads naturally to the iterative k_x-hitting set solution described above. □

We now consider the target inverse geodesic length T as parameter and show that DeStackIGL is W[1]-hard for parameters k_d and T combined. We provide a parameterized reduction from the Clique problem where, given a graph G and an integer k, the question is whether G has a clique of size k. Clique is one of the classic NP-complete problems and and is also know to be $W[1]$-hard when parameterized by k [9].

Theorem 3.4. DeStackIGL is W[1]-hard for parameter $k_d + T$.

Proof. Let (G, k) be an instance of Clique. We construct an instance (G, k_d, k_x, T) of DeStackIGL by setting $k_d = k$, $k_x = n - k$ and $T = \frac{k(k-1)}{2}$. We show that (G, k) is a Yes-instance if and only if (G, k_d, k_x, T) is a Yes-instance. Suppose that (G, k_d, k_x, T) is a Yes-instance. Since x has enough budget to delete all but the protected vertices in G and $k_d = k$, (G, k_d, k_x, T) can only be a Yes-instance if d defends vertices that induces a graph with inverse geodesic length at least $\frac{k(k-1)}{2}$. But for a graph on k vertices to have inverse geodesic length $\frac{k(k-1)}{2}$, all vertex pairs need to be at distance 1, and so d defends a complete graph on k vertices. Thus (G, k) is a Yes-instance. Conversely, suppose that (G, k) is a Yes-instance and there exists a solution S of size k. But this means d can defend a clique S_d of size k_d with $IGL(G[S_d]) = \frac{k(k-1)}{2}$. Hence (G, k_d, k_x, T) is a Yes-instance. □

Given the hardness result for the combined parameter $k_d + T$, we now consider the parameter T in combination with k_x and show that DeStackIGL is FPT for parameter $k_x + T$ by designing a kernel of size $O(k_x^2 + T)$. We define the following reduction rules that are applied in the same order they are defined here.

Reduction Rule 3.5 (Isolated Vertices)**.** If there exists a vertex $x \in V(G)$ such that $|N(x)| = 0$, then delete x.

For the next two reduction rules, let $q := 2\sqrt{\frac{9}{16} + T} - \frac{3}{2}$, which is the positive solution for the equation $T = q + \frac{1}{2}\binom{q}{2}$.

Reduction Rule 3.6 (High Degree Vertices)**.** If there exists a vertex $x \in V(G)$ such that $|N(x)| \geq q + k_x$ and $k_d \geq 1$, then return Yes.

Reduction Rule 3.7 (Bounded Edge Set)**.** If $|E(G)| > T + k_x(q + k_x - 1)$ and $k_d \geq 1$, then return Yes.

The correctness proof for these rules is straightforward and we skip it here due to space constraints.

Theorem 3.8. DeStackIGL has a kernel of size $O(k_x^2 + T)$.

Proof. Given an instance (G, k_d, k_x, T) of DeStackIGL by applying Reduction Rules 3.5–3.7 exhaustively, we obtain an instance (G', k_d, k_x, T), where the number of edges in G' is at most $T + k_x(q + k_x - 1) = O(T + k_x(\sqrt{T} + k_x)) = O(k_x^2 + T)$. Since degree of each vertex is at least 1, G' has $O(k_x^2 + T)$ vertices.

Observe that the above reduction rules does not apply to the case when $k_d = 0$. Notice that, if $k_d = 0$, DeStackIGL is equivalent to the complement of MinIGL. It is known that MinIGL has a kernel of size $O(k^2 + T)$ [2], where k denotes the attacker's budget. Thus, DeStackIGL has a kernel of size $O(k_x^2 + T)$. □

We now turn to structural parameters and first consider tree-width, which is one of the most widely studied structural parameters. We will show that DeStackIGL is W[1]-hard for parameter tree-width even when $k_d = 0$. We provide a parameterized reduction from the Equitable Coloring (EC) problem. In EC, we are given a graph G and an integer r, and the question is whether there exists a partition $\mathcal{V} = (V_1, V_2, \ldots, V_r)$ of $V(G)$ such that each part is an independent set and the numbers of vertices in any two parts V_i, V_j differ by at most one? EC is known to be W[1]-hard for parameter tree-width and number of partitions r combined [11].

Theorem 3.9. DeStackIGL is co-W[1]-hard for parameter $tw(G)$ even when $k_d = 0$.

Proof. Let (G, r) be an instance of EC where G has tree-width tw. The parameter is $r + tw$. Assume without loss of generality that $l = \frac{n}{r}$, where $n = |V(G)|$ and l is an integer. We construct an instance $(G', k_d = 0, k_x = n(r-1), T)$ of DeStackIGL where G' is defined as follows;

- create a vertex set $C = \{c_1, c_2, \ldots, c_r\}$ where each $c_i \in C$ corresponds to a color class,
- corresponding to each vertex $v_i \in V(G)$ create a set of r vertices $V'_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,r}\}$ and denote $V' = V'_1 \cup V'_2 \cup \cdots \cup V'_n$,
- create a set of r graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_r\}$ where each $G_i \in \mathcal{G}$ is a copy of G,
- for each i, $1 \leq i \leq r$, connect each $v \in V(G_i)$ by a path of length $L = 2(n(n^4 + 1) + r(n + 1))$ to $c_i \in C$,
- create a set of vertices D_v of size $R = L$ for each vertex $v \in G_i \in \mathcal{G}$ and make the vertices in D_v adjacent to v and call v a heavy vertex (see Figure 1),
- connect each $v_{i,j} \in V'$ to the ith vertex $v \in V(G_i)$ in each $G_i \in \mathcal{G}$.

In order to define value for T, we construct a graph G_t with r connected components where each connected component is constructed as follows; create a set $\mathcal{H}_v$ of l heavy vertices, create a vertex c and connect each $l_i \in \mathcal{H}_v$ to c by paths of length L; add $n - l$ paths of length $L - 1$ that have c as one endpoint; create r vertices corresponding to each $l_i \in \mathcal{H}_v$ and make them adjacent to l_i. We set $T = IGL(G_t) + \epsilon$ for some ϵ with $0 \leq \epsilon \leq \frac{1}{2}$. This completes our construction, see Figure 1 for a basic example of the construction.

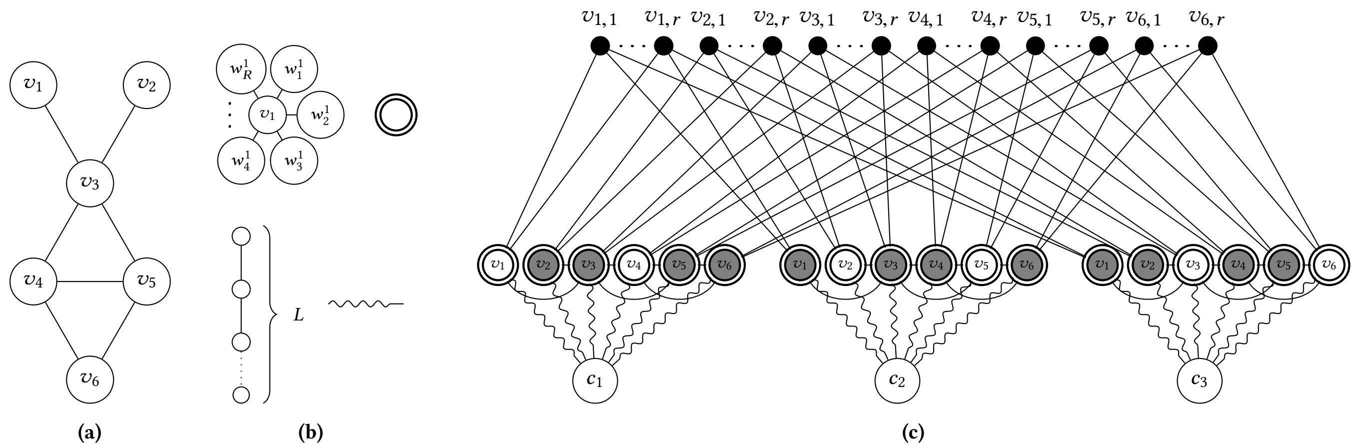

Figure 1: Depiction of (a) input graph G (b) vertices with R leaves and paths of length L (c) graph G' obtained by transforming G.

Let us first show that $tw(G') \in O(r \cdot tw)$. Consider a tree decomposition of G of minimum width. We take the same tree decomposition for each copy G_i of G, $1 \leq i \leq r$, and we merge copies of the same bag. Each bag now has size at most $r \cdot (tw + 1)$. For each vertex $v_j \in V(G)$, we add the vertices $V'_j = \{v_{j,1}, v_{j,2}, \ldots, v_{j,r}\}$ to all the bags containing copies of v_j. Since each bag contains copies of at most $tw + 1$ vertices in $V(G)$, this adds at most $r \cdot (tw + 1)$ vertices to each bag. Next, add $C = \{c_1, c_2, \ldots, c_r\}$ to all the bags. Each bag now has width at most $r + 2 \cdot r \cdot (tw+1)$. For each path P connecting a vertex $v \in V(G_j)$ to a color vertex c_i, select a bag containing v and append a path of bags, one for each edge $xy \in P$, containing $\{x, y, c_i\}$. Finally, for each heavy vertex v and each vertex $u \in D_v$, create a bag containing $\{u, v\}$ and make it adjacent to a bag containing v. This creates a tree decomposition of G' of width $O(r \cdot tw)$.

Now let us show that (G, r) is a Yes-instance if and only if (G', k_d, k_x, T) is a No-instance. Suppose (G, r) is a Yes-instance and there exists an equitable coloring $\mathcal{V} = \{V_1, V_2, \ldots, V_r\}$ of G with r colors where for each $V_i \in \mathcal{V}$, $|V_i| = l$ and for any pair of parts $V_i, V_j \in \mathcal{V}$, $V_i \cap V_j = \emptyset$. Let $\mathcal{V}_C = \{V_{C_1}, V_{C_2}, \ldots, V_{C_r}\}$ where each $V_{C_i} = V(G) - V_i$ is a vertex cover of size $n - l$. Since $r \cdot (n - l) = n(r - 1)$, we can obtain a graph isomorphic to G_t by deleting the heavy vertices corresponding to each vertex cover V_{C_i} from the corresponding graph copy $G_i \in \mathcal{G}$. But $IGL(G_t) = T - \epsilon$. Hence (G', k_d, k_x, T) is a No-instance.

Conversely, suppose (G', k_d, k_x, T) is a No-instance. We will prove that there is a set $S_x \subseteq V(G')$ with $|S_x| = k_x$ such that $IGL(G' - S_x) < T$ if and only if $G' - S_x$ is isomorphic to G_t. Without loss of generality, assume that S_x has no vertex that has degree 1 in G', otherwise replace it by its neighbor. We will now show that we may assume that S_x contains no vertex from V'. If S_x contains a vertex $v_{i,j} \in V'_i$, we consider several cases. If S_x also contains all neighbors of $v_{i,j}$, then $IGL(G' - S_x) = IGL(G' - (S_x \setminus \{v_i, j\}))$. If $V'_i \subseteq S_x$, then $IGL(G' - S_x) \geq IGL(G' - ((S_x \setminus V_i) \cup N_{G'}(V_i)))$, i.e., the attacker had better attack the r neighbors of V'_i rather than V'_i since the neighbors lie on all paths containing a vertex from V'_i. If $V'_i \not\subseteq S_x$, then the deletion of $v_{i,j}$ only affects the IGL for pairs of vertices that contain $v_{i,j}$. But then, the IGL can be decreased more by adding a neighbor of $v_{i,j}$ to S_x instead of $v_{i,j}$ since heavy vertices have (more than) R vertices at distance 1. Therefore, assume from now on that S_x contains no vertex from V'.

We will now show that $S_x \subseteq \bigcup_{G_i \in \mathcal{G}} V(G_i)$ (i.e. S_x only contains heavy vertices). If this is not the case then there is a vertex in V' that is adjacent to at least two heavy vertices in $G' - S_x$. Thus, we have two heavy vertices that are at distance at most 2 in $G' - S_x$. The paths from the degree-1 neighbors of one of them to the degree-1 neighbors of the other one contribute at least $\frac{1}{4}R^2$ to the IGL. Deleting one of these two heavy vertices would decrease the IGL by at least $\frac{1}{4}R^2 + R$. We will compare this against the decrease in IGL by the deletion of a vertex $c_i \in C$ (and note that the deletion of a vertex on the path from c_i to a vertex in G_i has lesser impact on the IGL). To upper bound the impact on the IGL of the deletion of c_i, we consider pairs of vertices that have distance at most L in G' and pairs that have distance more than L in G'. For pairs of vertices at distance at most L whose distance increases by the deletion of c_i, both vertices are on the paths from c_i to $V(G_i)$. The number of such pairs at distance ρ is $n + \binom{n}{2}(\rho - 1)$, and so these pairs contribute at most $\sum_{\rho=1}^{L} \frac{1}{\rho}\left(n + \binom{n}{2}(\rho - 1)\right) \leq n^2 \cdot L$ to the IGL. For the number of pairs at distance more than L, observe that G' has $r \cdot n \cdot (R + 2 + L) + r$ vertices. So, pairs at distance at least L contribute at most $\frac{1}{L}r^2n^2 \cdot (R + 2 + L)^2$ to the IGL. Since $L = R$, deleting c_i decreases the IGL by at most $n^2 \cdot R + r^2n^2 \cdot (4R + 9)$. Since $R = 2(n(n^4 + 1) + r(n + 1))$, this is smaller than $\frac{1}{4}R^2 + R$.

Now that we have established that x always prefers to delete heavy vertices, it remains to show that S_x where $G - S_x = G_t$ is her optimal choice. Notice that in order for $G - S_x$ to be isomorphic to G_t, S_x must contain a vertex cover for each copy of graph $G_i \in \mathcal{G}$, as no two heavy vertices are connected through an edge in G_t. Similarly, S_x must be picked such that no more than l heavy vertices reside in the same connected component. We note that (G', k_d, k_x, T) is a Yes-instance if either of the above two cases is not satisfied. For the first case, if $S_x \cap V(G_i)$ is not a vertex cover of G_i, then the paths between the degree-1 neighbors of two adjacent heavy vertices contribute an additional $\frac{1}{3}R^2$ to the IGL. For the second case, observe that the IGL is minimized if the connected components of $G' - S_x$ are of the same size. If there exists such a set S_x of $n(r-1)$ vertices then the heavy vertices in each connected component of G_t form an independent set of size l. Also, no two connected components contain a heavy vertex that correspond to the same vertex in G otherwise they would have been connected through the copy of the vertex in V'. Thus the heavy vertices in each connected component of G_t corresponds to a vertices of a color class in G. Hence, (G, r) is a Yes-instance. □

Note that the above proof constructs an instance of DeStackIGL by setting $k_d = 0$. This implies that we obtain an instance of a problem that is the complement of MinIGL. Thus, the above result also applies to MinIGL.

Corollary 3.10. MinIGL is W[1]-hard for parameter tree-width.

An Integer Quadratic Program (IQP) is an optimization problem whose input is an $\xi \times \xi$ integer matrix Q, $\mu \times \xi$ integer matrices A and C and μ-dimensional integer vectors b and d. The task is to solve the following optimization problem:

$$\begin{aligned} \textbf{minimize} \quad & y^{tr} Q y \\ \textbf{subject to} \quad & Ay \le b \\ & Cy = d \\ & y \in \mathbb{Z}^{\xi}. \end{aligned}$$

Let α be the maximum absolute value in A and Q. We then have the following useful proposition from [24].

Proposition 3.1. *There exists an algorithm that given an instance I of IQP, runs in time $f(\xi, \alpha) \cdot |I|^{O(1)}$ and outputs a vector $y \in Z^{\xi}$. If the input IQP has a feasible solution then y is feasible, and if the input IQP is not unbounded, then y is an optimal solution.*

Theorem 3.11. RMinIGL is FPT for parameter vertex cover number.

Proof. Let (G, k, S_d, T) be an instance of RMinIGL with $G = (V, E)$, and let ν be the smallest size of a vertex cover in G. We will construct an FPT algorithm with respect to ν.

The algorithm first computes a smallest vertex cover $W \subseteq V$ in time $O(1.2738^{\nu} + \nu n)$ using an algorithm from [6].

We now need to consider which k vertices we would like to delete from G. Consider the set of possibly deleted vertices from the vertex cover by enumerating all $U \subseteq W$ such that $|U| \le k$. Set $W := W \setminus U$, $G := G - U$, and $k := k - |U|$.

Since W is a vertex cover, we have that $V \backslash W$ is an independent set. Next, we define equivalence classes on the vertices of $V \backslash W$ by having two vertices $u, v \in V \backslash W$ in the same equivalence class if $N(u) = N(v)$ in G. We thus have a function which maps vertices to equivalence classes $P : V \setminus W \to [\phi]$, where $\phi \le 2^{\nu}$ is the number of equivalence classes in $V \backslash W$ and we use the shorthand $[\phi] = \{1, 2, ..., \phi\}$. The function P is computable in polynomial time. For simplicity, we will refer to an equivalence class i by P_i for $i \in [\phi]$.

In order to get a handle on the connectivity in G, our algorithm will branch on which equivalence classes will have all their vertices deleted; we say that these equivalence classes *vanish*. For this, we consider the set of functions of the form $f : [\phi] \to \{0, 1\}$ where if $f(i) = 0$ then partition P_i vanishes and it is a requirement that we delete all vertices from the equivalence class P_i, i.e. $P_i \subseteq X$. If $f(i) = 1$ then the equivalence class P_i does not vanish and so at least one vertex from P_i is not deleted. The algorithm will construct and enumerate all such functions f that do not vanish any equivalence class containing a vertex from S_d, of which there are at most $2^{2^{\nu}}$ as $\phi \le 2^{\nu}$.

We can now proceed to the construction of an integer quadratic program (IQP) for computing the minimum IGL of a graph G with vertex cover W, a computed partition function P and a vanishing function f. Let variable x_i represent the number of vertices that remain in part P_i after deleting the vertices in X, in other words $x_i = |P_i \backslash X|$. Consider the following integer quadratic program.

$$\begin{aligned} \textbf{minimize} \quad & OBJ \\ \textbf{subject to} \quad & \sum_{i=1}^{\phi} x_i \ge |V \backslash W| - k \\ & 1 \le x_i && \forall\, 1 \le i \le \phi \text{ with } f(i) = 1 \\ & x_i = 0 && \forall\, 1 \le i \le \phi \text{ with } f(i) = 0 \\ & |S_d \cap P_i| \le x_i && \forall\, 1 \le i \le \phi \\ & x_i \le |P_i| && \forall\, 1 \le i \le \phi \\ & x_i \in \mathbb{Z}^{\phi} && \forall\, 1 \le i \le \phi, \end{aligned}$$

where OBJ is a function, defined below, which represents the IGL of G after deleting vertices in X as defined by the variables x_i. As a first step, let us upper bound the diameter of the connected components of the graph, that is, the maximum finite distance of two vertices in the graph.

Claim 3.12. In a graph with vertex cover number ν, each connected component has diameter at most 2ν.

Proof. Let graph $G = (V, E)$ have a vertex cover W of size $|W| = \nu$. Let $P = v_1, v_2, ..., v_l$ be a path of length l passing through distinct vertices $v_i \in V$ for $i \in [l]$. As $V \setminus W$ is an independent set, any path which passes through a vertex $v_j \in V \setminus W$ must either end at v_j, begin at v_j or have path vertices $v_{j-1}, v_{j+1} \in W$. In the worst case, P contains all ν vertices in W in its path, and at most $\nu + 1$ vertices in $V \setminus W$, giving a path with $2\nu + 1$ vertices on it, and hence a path length of at most 2ν. □

Since the function f determines which equivalence classes vanish, we can compute the distances between every pair of non-vanishing equivalence classes and vertex cover vertices. This is exactly the distance in the graph obtained from G by deleting all vanishing equivalence classes and merging each remaining equivalence class into a single vertex. We denote by $\delta(x, y)$ the distance between x and y in this graph, where x and y are vertices from W or equivalence classes from P. We have that $\delta(P_i, P_i) = 2$ if the vertices in P_i have a neighbor and $\delta(P_i, P_i) = \infty$ if not. We take the convention that any number divided by ∞ equals 0.

We now define OBJ', which is the inverse geodesic length of the graph obtained after deleting X:

$$OBJ' := \sum_{u \in W} \sum_{v \in W \setminus \{u\}} \frac{1}{\delta(u,v)} + \sum_{u \in W} \sum_{\substack{i \in [\phi] \\ f(i)=1}} \frac{x_i}{\delta(u, P_i)} + \sum_{\substack{i \in [\phi] \\ f(i)=1}} \frac{x_i \cdot (x_i - 1)}{2 \cdot \delta(P_i, P_i)} + \sum_{\substack{i \in [\phi] \\ f(i)=1}} \sum_{\substack{j \in [\phi] \setminus \{i\} \\ f(j)=1}} \frac{x_i \cdot x_j}{\delta(P_i, P_j)}$$

Since the coefficients of OBJ need to be integers, we defined OBJ by multiplying OBJ' by the least common multiple of the integers 1 to $2v$. We observe that all coefficients that multiply the variables x_i are bounded as a function of v. □

Theorem 3.13. RMinIGL is FPT for parameter neighborhood diversity.

Proof. The proof is similar to the proof of Theorem 3.11. Let (G, k, S_d, T) be an instance of RMinIGL with $G = (V, E)$ and let η be its neighborhood diversity. We construct a function $P : V \to [\eta]$ where P_i is an equivalence class defined by the rule that if $u, v \in P_i$ then $N(u) \setminus \{v\} = N(v) \setminus \{u\}$, and there are η such equivalence classes, as defined by the definition of neighborhood diversity. We note that any partition is either an independent set or a clique.

We once again consider the set of *vanishing functions* $f : [\eta] \to \{0, 1\}$, where $f(i) = 1$ requires all vertices of P_i to be deleted and $f(j) = 1$ requires that not all vertices of P_i are deleted.

Let variable x_i represent the number of vertices that remain in equivalence class P_i after deleting the vertices in X, in other words $x_i = |P_i \setminus X|$. Consider the following integer quadratic program.

$$\begin{array}{lll}
\textbf{minimize} & OBJ & \\
\textbf{subject to} & \sum_{i=1}^{\eta} x_i \geq |V| - k & \\
& 1 \leq x_i & \forall\, 1 \leq i \leq \eta \text{ with } f(i) = 1 \\
& x_i = 0 & \forall\, 1 \leq i \leq \eta \text{ with } f(i) = 0 \\
& |S_d \cap P_i| \leq x_i & \forall\, 1 \leq i \leq \eta \\
& x_i \leq |P_i| & \forall\, 1 \leq i \leq \eta \\
& x_i \in \mathbb{Z}^{\eta} & \forall\, 1 \leq i \leq \eta.
\end{array}$$

Given f, we can again determine the distances between equivalence classes in $G - X$. Denote by $\delta(P_i, P_j)$ the distance between a vertex in P_i and a vertex in P_j in $G - X$. We use the convention that $\delta(P_i, P_i)$ is 1 if P_i is a clique, 2 if P_i is an independent set with at least one neighbor in $G - X$, and ∞ if P_i consists of isolated vertices in $G - X$. We observe that the diameter of each connected component of the graph is at most η.

We define the inverse geodesic length of $G - X$ as

$$OBJ' := \sum_{\substack{i \in [\eta] \\ f(i)=1}} \frac{x_i}{\delta(P_i, P_i)} + \sum_{\substack{i \in [\eta] \\ f(i)=1}} \sum_{\substack{j \in [\eta] \setminus \{i\} \\ f(j)=1}} \frac{x_i \cdot x_j}{\delta(P_i, P_j)}$$

Since the coefficients of OBJ need to be integers, we defined OBJ by multiplying OBJ' by the least common multiple of the integers 1 to η. We observe that all coefficients that multiply the variables x_i are bounded as a function of η. There are at most 2^{η} vanishing functions, and the resulting IQPs have a number of variables bounded by the neighborhood diversity number η, and a maximum coefficient upper bounded by a function of η. By Proposition 3.1, solving the IQP is FPT for η, hence RMinIGL is FPT for parameter η. □

Theorem 3.14. DeStackIGL is FPT for parameter vertex cover number and defender budget k_d combined.

Proof. Let $(G = (V, E), k_d, k_x, T)$ be an instance of DeStackIGL with v being the size of a minimum vertex cover in G. Let W be a vertex cover of size v computed in time $O(1.2738^v + vn)$ [6].

This algorithm will compute a solution to DeStackIGL by first considering all valid defender strategies as subsets using the vertex cover W, and then calling RMinIGL as a subroutine in order to compute the minimum IGL that an attacker could compute given a valid defender subset.

We first enumerate subsets of the vertex cover W with at most size k_d in time 2^v. Let $W' \subseteq W$ be such a set. Then we take the graph $V \setminus W$ which we know to be an independent set and compute $\phi \leq 2^v$ equivalence classes on $V \setminus W$, where two vertices $u, v \in V$ are in the same independence class if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. We then do an ϕ way branching, picking an equivalence class, and then any vertex inside the equivalence class, $k_d - |W|$ times which in the worst case takes ϕ^{k_d} time. It suffices to consider branching only on equivalence classes as all vertices in a particular class contribute the same amount to reducing the IGL.

We then let S_d be the selected vertices from both a computed subset of W and the ϕ way branching, and create a (G, k_x, S_d, T) instance for Theorem 3.11.

As the running time of S_d selection is upper bounded by $2^v \cdot (2^v)^{k_d}$ and the IQP create from Theorem 3.11 is FPT for parameter v then it follows that DeStackIGL is FPT for parameter vertex cover and defender budget k_d. □

Theorem 3.15. DeStackIGL is FPT for parameter neighborhood diversity and defender budget k_d combined.

Proof. Let (G, k_d, k_x, T) be an instance of DeStackIGL with $G = (V, E)$ with η being the neighborhood diversity number in G. We will compute $P : V \to [\eta]$ the partition function for equivalence classes based on neighbors, and for simplicity denote the set $\{v : v \in V, P(v) = i\} = P_i$.

As the removal of any vertex in some partition P_i contributes to the same reduction in IGL, then we proceed to perform an η way branching, picking a partition each time, and a single vertex from the partition, to a depth of k_d for a total of k_d vertices.

We then let S_d be the selected vertices from this procedure, and create a (G, k_x, S_d, T) instance for Theorem 3.13. A solution to DeStackIGL will be the S_d that returns the largest value form RMinIGL

As the running time of S_d selection is upper bounded by η^{k_d} and the IQP created from Theorem 3.13 is FPT for parameter η then it follows that DeStackIGL is FPT for parameter neighborhood diversity and defender budget k_d. □

4 EXPERIMENTAL RESULTS

Empirical results were obtained for RMinIGL and DeStackIGL on real-world datasets [37] with running times shown in Tables 1 and 2. For comparison, a brute force algorithm for RMinIGL was implemented enumerating all $\binom{n}{k}$ possible vertex subsets to remove of an input graph G and looked for the smallest IGL among them. For the experiments in Table 1, we have set $S_d = \emptyset$. A brute force implementation of DeStackIGL was implemented by also enumerating $\binom{n}{k_d}$ possible protected subsets, before running the brute force RMinIGL algorithm described above with $k = k_x$, for a total running time within a polynomial factor of $\binom{n}{k_d} \cdot \binom{n}{k_x}$.

We see that in comparison to the brute force implementations, the FPT implementations are efficient. Especially, when the values of the parameters namely vertex cover number ν, neighborhood diversity η and defender budget are small compared to the number of vertices. The outlier is the 'Rhodes' network for which our methods do not give an improvement (except in one case). But this is not surprising given that our methods are tailored to instances with small parameter values whereas both the vertex cover number and neighborhood diversity of the 'Rhodes' network are quite large compared to the number of vertices.

Our implementations were tested on a Macbook Pro (A1707) with Intel 2.6 GHz Core i7 (I7-6700HQ) CPU . The implementations were written in Python, with extensive use of the scientific programming package NumPy and outsourcing the majority of the IQP problem to Gurobi. We make note that while Gurobi is efficient, it is a general purpose solver and does not implement Lokshtanov's FPT algorithm [24]. Nevertheless, for Integer Linear Programming, it has been established in some domains (see, e.g., [4]) that off-the-shelf solvers also perform well for problems that are FPT. It is possible though, that solvers exploiting a small number of variables and coefficients might have a large advantage over Gurobi.

Table 1: Running time (seconds) for RMinIGL

Data	n	ν	η	k	Algorithms		
					FPT ν	FPT η	Brute Force
Greece	18	6	11	3	1.58	2.81	4.30
				4	3.56	9.48	17.96
Cielnet	25	7	17	3	11.40	20.23	42.50
				4	43.85	73.23	212.62
Acero	25	7	18	3	14.23	27.04	44.07
				4	38.95	76.39	206.62
Cocaine	28	6	16	3	20.77	29.39	85.39
				4	45.28	68.70	464.00
Jake	38	8	19	3	36.17	65.35	534.30
				4	88.15	139.10	3372.96
Mambo	31	12	22	3	69.93	63.35	158.98
				4	223.46	234.97	714.41
Gangs	35	12	29	3	144.41	226.15	234.79
				4	847.99	1329.08	1788.30
Rhodes	22	13	18	3	19.91	15.17	13.18
				4	86.08	54.10	53.66
Siren	44	18	21	3	15.17	129.98	991.66
				4	54.10	505.42	9648.39
Togo	33	10	20	3	50.14	28.23	164.03
				4	223.74	96.57	1142.38

Table 2: Running time (seconds) for DeStackIGL

Data	n	ν	η	k_d, k_x	Algorithms		
					FPT $\nu + k_d$	FPT $\eta + k_d$	Brute Force
Greece	18	6	11	2, 2	20	32	101
				3, 3	155	164	1919
Cielnet	25	7	17	2, 2	79	521	1159
				3, 3	635	6942	49582
Acero	25	7	18	2, 2	343	485	1409
				3, 3	5975	8614	47544
Cocaine	28	6	16	2, 2	289	478	2520
Jake	38	8	19	2, 2	893	1642	26616
Mambo	31	12	22	2, 2	1946	2141	7381
Gangs	35	12	29	2, 2	6884	10418	13055
Rhodes	22	13	18	2, 2	548	460	543
Siren	44	18	21	2, 2	17085	4709	75918
Togo	33	10	20	2, 2	2387	1113	11659

5 CONCLUSIONS

We have analyzed the parameterized complexity of DeStackIGL for several parameters. We were mainly concerned about the border between tractability and intractability. Our results suggest that for the problem to become FPT, it is more important to bound the attacker budget than the defender budget, unless one can identify some nice structure in the input instances, such as bounded neighborhood diversity or vertex cover number.

There are several questions that we leave open. It might be possible to strengthen some of our hardness results. In particular, we conjecture that DeStackIGL is Σ_2^P-complete. The class of trees remains of interest for this problem, and it remains open from [2] whether MinIGL is polynomial-time solvable on trees.

Finally, in situations where the attacker cannot wait to see which vertices the defender protects, it might make sense to consider mixed strategies. Repeated games would also be an interesting direction for future research.

ACKNOWLEDGEMENTS

Haris Aziz is supported by a Julius Career Award. Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134). Edward J. Lee and Kamran Najeebullah are supported by the Australian Government Research Training Program Scholarship.

REFERENCES

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *Nature* 406, 6794 (2000), 378–382.
[2] Haris Aziz, Serge Gaspers, and Kamran Najeebullah. 2017. Weakening Covert Networks by Minimizing Inverse Geodesic Length. In *Proc. of 26th IJCAI*. 779–785.
[3] Ulrik Brandes and Thomas Erlebach (Eds.). 2005. *Network Analysis*. Theoretical Computer Science and General Issues, Vol. 3418. Springer-Verlag.

[4] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. 2015. Elections with Few Candidates: Prices, Weights, and Covering Problems. In *Algorithmic Decision Theory (Lecture Notes in Computer Science)*, Vol. 9346. 414–431.

[5] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. 2012. Identifying influential nodes in complex networks. *Physica A: Statistical Mechanics and its Applications* 391, 4 (2012), 1777–1787.

[6] Jianer Chen, Iyad A. Kanj, and Ge Xia. 2010. Improved upper bounds for vertex cover. *Theoratical Computer Science* 411, 40-42 (2010), 3736–3756.

[7] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.

[8] Reinhard Diestel. 2010. *Graph Theory*. Springer.

[9] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer.

[10] Pål G. Drange, Markus S. Dregi, and Pim van 't Hof. 2016. On the Computational Complexity of Vertex Integrity and Component Order Connectivity. *Algorithmica* 76, 4 (2016), 1181–1202.

[11] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. 2011. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation* 209, 2 (2011), 143–153.

[12] Fedor V. Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. 2010. Iterative compression and exact algorithms. *Theoretical Computer Science* 411, 7-9 (2010), 1045–1053.

[13] Daniel Gross, Monika Heinig, Lakshmi Iswara, L. William Kazmierczak, Kristi Luttrell, John T. Saccoman, and Charles Suffel. 2013. A Survey of Component Order Connectivity Models of Graph Theoretic Networks. *WSEAS Transactions on Mathematics* 12 (2013), 895–910.

[14] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. 2007. Parameterized Complexity of Vertex Cover Variants. *Theory of Computing Systems* 41, 3 (2007), 501–520.

[15] Qingyu Guo, Bo An, Yevgeniy Vorobeychik, Long Tran-Thanh, Jiarui Gan, and Chunyan Miao. 2016. Coalitional Security Games. In *Proc. of 15th AAMAS*. 159–167.

[16] Qingyu Guo, Bo An, Yevgeniy Zick, and Chunyan Miao. 2016. Optimal interdiction of illegal network flow. In *Proc. of 25th IJCAI*. 2507–2513.

[17] Petter Holme, Beom J. Kim, Chang N. Yoon, and Seung K. Han. 2002. Attack vulnerability of complex networks. *Physical Review. E* 65, 5 (2002), 056109–056123.

[18] Manish Jain, Vincent Conitzer, and Milind Tambe. 2013. Security Scheduling for Real-world Networks. In *Proc. of 12th AAMAS*. 215–222.

[19] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. 2010. Security Games with Arbitrary Schedules: A Branch and Price Approach. In *Proc. of 24th AAAI*. 792–797.

[20] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. 2011. A double oracle algorithm for zero-sum security games on graphs. In *Proc. of 10th AAMAS*. 327–334.

[21] István A. Kovács and Albert-László Barabási. 2015. Network science: Destruction perfected. *Nature* 524, 7563 (2015), 38–39.

[22] Michael Lampis. 2012. Algorithmic Meta-theorems for Restrictions of Treewidth. *Algorithmica* 64, 1 (2012), 19–37.

[23] Vito Latora and Massimo Marchiori. 2001. Efficient behavior of small-world networks. *Physical Review Letters* 87, 19 (2001), 198701–198705.

[24] Daniel Lokshtanov. 2015. Parameterized Integer Quadratic Programming: Variables and Coefficients. *CoRR* abs/1511.00310 (2015), 15. arXiv:1511.00310

[25] Tomasz P. Michalak, Talal Rahwan, Oskar Skibski, and Michael J. Wooldridge. 2015. Defeating Terrorist Networks with Game Theory. *IEEE Intelligent Systems* 30, 1 (2015), 53–61.

[26] Tomasz P. Michalak, Talal Rahwan, Piotr L. Szczepanski, Oskar Skibski, Ramasuri Narayanam, Nicholas R. Jennings, and Michael J. Wooldridge. 2013. Computational Analysis of Connectivity Games with Applications to the Investigation of Terrorist Networks. In *Proc. of 23rd IJCAI*. 293–301.

[27] Tomasz P. Michalak, Talal Rahwan, and Michael Wooldridge. 2017. Strategic Social Network Analysis. In *Proc. of 31st AAAI*. 4841–4845.

[28] Flaviano Morone and Hernán A. Makse. 2015. Influence Maximization in Complex Networks Through Optimal Percolation. *Nature* 524 (2015), 65–68. Issue 7563.

[29] Hiroshi Nagamochi and Toshihide Ibaraki. 2008. *Algorithmic Aspects of Graph Connectivity*. Cambridge University Press.

[30] Rolf Niedermeier and Peter Rossmanith. 2003. An efficient fixed-parameter algorithm for 3-Hitting Set. *J. Discrete Algorithms* 1, 1 (2003), 89–102.

[31] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2008. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proc. of 7th AAMAS*. 125–132.

[32] Piotr L. Szczepanski, Tomasz P. Michalak, and Talal Rahwan. 2015. Efficient algorithms for game-theoretic betweenness centrality. *Artificial Intelligence* 231 (2015), 39–63.

[33] Milind Tambe. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

[34] Mikkel Thorup. 1999. Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of ACM* 46, 3 (1999), 362–394.

[35] Jason Tsai, Thanh Hong Nguyen, and Milind Tambe. 2012. Security Games for Controlling Contagion. In *Proc. of 26th AAAI*. 1464–1470.

[36] Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, and Milind Tambe. 2011. IRIS A Tool for Strategic Security Allocation in Transportation Networks. In *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 88106.

[37] UCINET Software. [n. d.]. Datasets: Covert Networks. ([n. d.]). Retrieved from https://sites.google.com/site/ucinetsoftware/datasets/covert-networks on 2017-02-17.

[38] Magnus Wahlström. 2007. *Algorithms, measures and upper bounds for satisfiability and related problems*. Ph.D. Dissertation. Linköping University, Sweden.

[39] Zhen Wang, Yue Yin, and Bo An. 2016. Computing optimal monitoring strategy for detecting terrorist plots. In *Proc. of 30th AAAI*. 637–643.

[40] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world'networks. *Nature* 393, 6684 (1998), 440–442.

[41] Alice X. Zheng, John Dunagan, and Ashish Kapoor. 2011. Active Graph Reachability Reduction for Network Security and Software Engineering. In *Proc. of 22nd IJCAI*. 1750–1756.