Weakening Covert Networks by Minimizing Inverse Geodesic Length

Haris Aziz, Serge Gaspers and **Kamran Najeebullah**
Data61, CSIRO, Australia
UNSW Sydney, Australia
{haris.aziz, kamran.najeebullah}@data61.csiro.au, sergeg@cse.unsw.edu.au

Abstract

We consider the problem of deleting nodes in a covert network to minimize its performance. 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 MINIGL problem the input is a graph G, a budget k, and a target IGL T, and the question is whether there exists a subset of vertices X with $|X| = k$, such that the IGL of $G - X$ is at most T. In network analysis, the IGL is often used to evaluate how well heuristics perform in strengthening or weakening a network. In this paper, we undertake a study of the classical and parameterized complexity of the MINIGL problem. The problem is NP-complete even if $T = 0$ and remains both NP-complete and $W[1]$-hard for parameter k on bipartite and on split graphs. On the positive side, we design several multivariate algorithms for the problem. Our main result is an algorithm for MINIGL parameterized by twin cover number.

1 Introduction

Strategic aspects of network analysis is an important research area in many fields including AI (see, e.g., [Michalak *et al.*, 2017]). Within this area, pinpointing the most important nodes or edges is a fundamental problem [Michalak *et al.*, 2013; Chen *et al.*, 2012; Zheng *et al.*, 2011]. Applications include disrupting an epidemic network (see, e.g., [Kovács and Barabási, 2015]), weakening a terrorist network (see, e.g., [Michalak *et al.*, 2013]), finding the most critical or vulnerable nodes (see, e.g., [Holme *et al.*, 2002]) and reducing reachability on a graph for network security (see, e.g., [Zheng *et al.*, 2011]). The problem is relevant to various fields and sectors such as epidemiology, sociology, physics, security and logistics. We focus on the problem of weakening a covert network by identifying and eliminating the individuals who are most critical for a high performance of these networks. The problem has been considered both in social network analysis and artificial intelligence (see, e.g., [Carley *et al.*, 2003; Lindelauf *et al.*, 2013; Michalak *et al.*, 2013]). A common practice among these studies is to use a heuristic to rank the vertices by their importance (see, e.g, [Michalak *et al.*, 2013; Michalak *et al.*, 2015; Szczepanski *et al.*, 2015]). In order to measure the quality of their pick they either use expert domain knowledge on the importance of vertices in the network or turn to widely used network performance measures like *component order connectivity* (size of the largest connected component) [Gross *et al.*, 2013; Drange *et al.*, 2016] and *inverse geodesic length (IGL;* sum of the inverse distances between all pairs of vertices). Formally, $IGL(G) = \sum_{\{u,v\} \subseteq V, u \neq v} \frac{1}{d(u,v)}$. The *average inverse geodesic length (AIGL)* of G is the inverse geodesic length of G normalized by the number of vertex pairs. Formally, $AIGL(G) = \frac{2}{n(n-1)} \cdot IGL(G)$. We opt to quantify network performance by IGL; the motivation for that is two-fold. Firstly, AIGL has been frequently used in the literature as a global measure of the connectivity or robustness of a network. It has been used to examine network vulnerability [Dagon *et al.*, 2008; Holme *et al.*, 2002] and the effect of critical nodes [Barabási and Albert, 1999] in the network security domain. It has also been used to identify influential nodes in a social network [Morone and Makse, 2015]. Game-theoretic values and centrality measures, such as the Shapley value and betweenness centrality of the nodes, have been used as heuristics to delete nodes with large impact on network performance [Holme *et al.*, 2002; Amer and Giménez, 2004; Michalak *et al.*, 2015; Szczepanski *et al.*, 2015]. In particular, Szczepanski *et al.* [2015] delete the nodes with the highest Shapley value as a heuristic to decrease the IGL. Secondly, performance measures like component order connectivity and diameter prove ineffective on dense graphs. On the contrary, IGL remains effective in identifying critical nodes irrespective of the input graph structure. Surprisingly, despite its widespread use as a network performance measure, the optimization problem with respect to IGL has not been examined previously. We consider the vertex deletion optimization problem corresponding to IGL:

MINIMIZE IGL (MINIGL)	
Input:	A graph G, an integer k, and a target inverse geodesic length T.
Question:	Does there exist $X \subseteq V(G)$, such that $\|X\| \leq k$ and $IGL(G - X) \leq T$?

The problem is NP-complete and we show that it remains

Table 1: Covert networks with their number of vertices (n), edges (m), vertex cover number (δ), twin cover number (γ), and neighborhood diversity (ω).

Dataset	n	m	δ	γ	ω
FIFA	450	5022	422	38	449
Drug Net	293	145	70	36	142
Suffragettes	112	1342	84	52	107
Caviar	110	163	25	17	60
Noordin	79	200	37	32	70

NP-complete on several graph classes that naturally occur in application domains [Belik, 2016; Borgatti, 2012]. Our main focus is a parameterized complexity analysis of the problem.

Our Results. We observe that, for $T = 0$, MINIGL is equivalent to VERTEX COVER and it is therefore para-NP-hard for parameter T. For parameter k, we give reductions from CLIQUE on regular graphs to show that MINIGL is $W[1]$-hard and NP-complete, even when restricted to bipartite and split graphs. For parameter $k + T$, we give a kernel of size $O(k^2 + T)$, which establishes that the problem is FPT for this combined parameter. Next, we consider structural parameters. The vertex cover number is among the most widely studied structural parameters [Fellows *et al.*, 2008; Fomin *et al.*, 2014], neighborhood diversity [Lampis, 2012; Ganian, 2012] is a parameter that generalizes vertex cover to dense graphs, and Ganian [2015] recently proposed an alternative generalization, the twin cover number, that still preserves some of the nice properties of the vertex cover number. Our main result is that MINIGL is FPT parameterized by twin cover number. Since a vertex cover is a twin cover, this implies that MINIGL is also FPT parameterized by vertex cover number. However, we give a faster algorithm for this parameter. We also provide an FPT algorithm parameterized by neighborhood diversity and the deletion budget combined.

Our choice of parameters is motivated by real-world datasets. Table 1 shows five representative datasets of covert networks from [UCINET Software, 2017] with some parameter values. Even on this small sample, we can already see the importance of considering multiple parameters. While, for the FIFA network, both the vertex cover number and the neighborhood diversity are large, the smallest twin cover contains fewer than 10% of the vertices. For instances where all parameters are large, our current best approach will turn out to be a simple FPT algorithm parameterized by $\omega+k$.

Our key take-home message is that minimizing IGL is computationally hard even for very restricted graph classes and that a parametrized complexity approach by identifying and exploiting suitable parameters promises to be fruitful.

2 Preliminaries

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 [Diestel, 2010]. 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. A *d-regular graph* is a graph where each vertex has d neighbors. A *bipartite graph* is a graph whose vertex set can be partitioned into two independent sets. A *split graph* is a graph whose vertex set can be partitioned into a clique $\mathcal{K}$ and an independent set I. Such a partition $(\mathcal{K}, I)$ is known as a split partition. Throughout this article we assume that $\mathcal{K}$ is a maximal clique in a given split partition $(\mathcal{K}, I)$.

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 set of vertices $C \subseteq V(G)$ is a *twin cover* of G, if for every edge $xy \in E(G)$, either xy is incident to a vertex in C or x and y are true twins [Ganian, 2015]. A graph G has neighborhood diversity ω, if there exists a partition of $V(G)$ into at most ω 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 [Lampis, 2012].

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 $f(k) \cdot |I|^{O(1)}$. $W[1]$ is a class of parameterized decision problems closed under parameterized reductions. $W[1]$-hard problems are considered unlikely to be in FPT [Downey and Fellows, 2013]. A para-NP-hard problem is NP-hard even for constant values of the parameter. We refer to [Cygan *et al.*, 2015] for a detailed exposition of parameterized complexity.

3 Intractability for Parameter T

In this section we consider the parameterized complexity of MINIGL when parameterized by the target inverse geodesic length T. We show that MINIGL is NP-complete, even when $T = 0$, by observing that it is equivalent to the well-known NP-complete VERTEX COVER problem [Karp, 1972], where

for a given graph G and integer k, the question is whether k vertices can be deleted from G to make the graph edgeless. Since MINIGL remains NP-complete for a constant value of T, MINIGL is para-NP-hard when parameterized by T.

Theorem 3.1. MINIGL is NP-complete and para-NP-hard for parameter T.

4 Intractability for Parameter k

Given the previous hardness result for general graphs, we consider MINIGL on special classes of graphs for which VERTEX COVER can be solved in polynomial time, namely split graphs and bipartite graphs. We also consider another natural parameter, namely the size of the solution k. We begin with a proof that for a split graph with vertex partition $(\mathcal{K}, I)$, there is an optimal solution for MINIGL containing only vertices from $\mathcal{K}$. It can be proved by an exchange argument replacing a vertex from I by a vertex from $\mathcal{K}$ and analyzing the impact on the IGL.

Lemma 4.1. For each subset of vertices $X \subseteq V(G)$ in a split graph $G = (\mathcal{K}, I, E)$, there is a subset of vertices $Y \subseteq \mathcal{K}$ with $|Y| \leq |X|$, such that $IGL(G - Y) \leq IGL(G - X)$.

We now consider MINIGL on split graphs and call this problem SPLIT-MINIGL. We show that SPLIT-MINIGL is NP-complete and $W[1]$-hard for parameter k. We provide a parameterized reduction from the CLIQUE problem on regular graphs, where, given a regular graph G and an integer k, the question is whether G has a clique of size k. The restriction of CLIQUE to regular graphs is known to be NP-complete and $W[1]$-hard for parameter k [Cai, 2008].

Theorem 4.2. SPLIT-MINIGL is NP-complete and $W[1]$-hard for parameter k.

Proof. For a CLIQUE instance $(G = (V, E), k)$ with $n = |V|$ and $m = |E|$, where G is d-regular with $d > 2$, we obtain a SPLIT-MINIGL instance (G', k, T) as follows. The vertex set of G' consists of the clique $\mathcal{K}' = V$ and the independent set $I' = E$. Set $E(G') = \{x_1x_2 : x_1, x_2 \in V\} \cup \{xe : x \in e \in E\}$. See Fig. 1b. We retain the value of k and set $T = T\left(\frac{k(k-1)}{2},\ kd - k(k-1),\ m + \frac{k(k-1)}{2} - kd\right)$, where

$$\begin{aligned} T(\beta_0, \beta_1, \beta_2) &= \frac{(n-k)\cdot(n-k-1)}{2} + 2m - k\cdot d \\ &+ \frac{1}{2}\Big(\beta_2\cdot(n-k-2) + \beta_1\cdot(n-k-1) \\ &\quad + (n-k)\cdot\frac{d\cdot(d-1)}{2}\Big) \\ &+ \frac{1}{3}\left(\frac{(m-\beta_0)\cdot(m-\beta_0-1)}{2} - (n-k)\cdot\frac{d\cdot(d-1)}{2}\right) \end{aligned} \tag{1}$$

is the IGL of a graph obtained from G' by deleting k vertices from $\mathcal{K}'$ so that I' has β_i vertices of degree i. We now show that (G', k, T) is a Yes-instance for SPLIT-MINIGL iff (G, k) is a Yes-instance for CLIQUE.

Suppose (G', k, T) is a Yes-instance for SPLIT-MINIGL and there is a solution $X' \subseteq V(G')$ with $|X'| = k$. By Lemma 4.1 we may assume that $X' \subseteq \mathcal{K}'$. Let $G'' = G' - X'$. Let α_i denote the number of vertices in I' that have degree i in G''. We have, $IGL(G'') = T(\alpha_0, \alpha_1, \alpha_2)$. Let us now determine the values of α_1 and α_2 in terms of α_0:

$$\alpha_1 = kd - 2\alpha_0 \qquad \text{and} \qquad \alpha_2 = m + \alpha_0 - kd. \tag{2}$$

Notice that by (1) and (2), the value of $IGL(G'') = T(\alpha_0, \alpha_1, \alpha_2)$ decreases with increasing α_0. In other words, the inverse geodesic length is minimized when the number of isolated vertices is maximized. We observe that $IGL(G'') \leq T$ only if $\alpha_0 = \frac{k(k-1)}{2}$. But this is the maximum number of vertices that can be isolated by deleting k vertices from G', meaning that X' forms a clique of size k in G. Hence, (G, k) is a Yes-instance for CLIQUE.

Conversely, suppose that (G, k) is a Yes-instance for CLIQUE and there exists a solution $X \subseteq V(G)$, with $|X| = k$, then, deleting X from G' isolates $\frac{k(k-1)}{2}$ vertices in I'. Therefore, $IGL(G' - X) \leq T$. Thus, (G', k, T) is a Yes-instance for SPLIT-MINIGL. □

We further consider MINIGL on bipartite graphs and call this problem BIPARTITE-MINIGL. We show that BIPARTITE-MINIGL remains NP-complete and $W[1]$-hard when parameterized by k. Again, we provide a parameterized reduction from CLIQUE on regular graphs.

Theorem 4.3. BIPARTITE-MINIGL is NP-complete and $W[1]$-hard for parameter k.

Proof. Consider an instance (G, k) of CLIQUE, where G is a d-regular graph, $d > 2$, with n vertices and m edges. We construct an instance (G', k, T) for BIPARTITE-MINIGL. We construct the vertex set $V(G')$ with a bipartition (A, B), where $A = V(G)$ and $B = B_1 \cup B_2$ with $B_i = \{e_i : e \in E(G)\}$. The edge set of G' is $E(G') = \{xe_1 : e_1 \in B_1, x \in e\} \cup \{xe_2 : x \in A, e_2 \in B_2\}$. In this way each vertex in A has degree $d + m$, each vertex in B_1 has degree 2, and each vertex in B_2 has degree n (see Fig. 1c). We retain the value of k and set $T = T\left(\frac{k\cdot(k-1)}{2}\right)$, where

$$\begin{aligned} T(\beta_0) &= m\cdot(n-k) + 2m - k\cdot d \\ &+ \frac{1}{2}\Big(m\cdot(m-\beta_0) + \frac{(n-k)\cdot(n-k-1)}{2} \\ &\quad + (n-k)\cdot\frac{d\cdot(d-1)}{2} + \frac{m\cdot(m-1)}{2}\Big) \\ &+ \frac{1}{3}\left((n-k)\cdot(m-\beta_0) - (2m - k\cdot d)\right) \\ &+ \frac{1}{4}\left(\frac{(m-\beta_0)(m-\beta_0-1)}{2} - (n-k)\cdot\frac{d\cdot(d-1)}{2}\right). \end{aligned} \tag{3}$$

Since vertices in A have much higher degrees than vertices in B and they also occur on paths between vertices in B, one can show that for each subset $S \subseteq V(G')$, there is a subset $S' \subseteq A$, such that $IGL(G' - S') \leq IGL(G' - S)$. We now show that (G', k, T) is a Yes-instance for BIPARTITE-MINIGL iff (G, k) is a Yes-instance for CLIQUE.

Suppose that (G', k, T) is a Yes-instance for BIPARTITE-MINIGL and there is a solution $X' \subseteq A$ with $|X'| = k$. Let $G'' = G' - X'$ and α_0 be the number of isolated vertices in G''. Then, $IGL(G'') = T(\alpha_0)$. Note that by (3), the value of $IGL(G'')$ decreases with increasing α_0.

In other words, minimizing IGL is equivalent to maximizing the number of isolated vertices. We have that $IGL(G'') \leq T$ only if $\alpha_0 = \frac{k\cdot(k-1)}{2}$. But this is the maximum number of vertices that can be isolated by deleting k vertices from G'. Thus, there are $\frac{k\cdot(k-1)}{2}$ isolated vertices in B_1 and X' forms a clique in G. Hence, (G, k) is a Yes-instance for CLIQUE.

Conversely, suppose that (G, k) is a Yes-instance of the CLIQUE problem and there exists a solution $X \subseteq V(G)$, with $|X| = k$. But then deleting X from A isolates $\frac{k\cdot(k-1)}{2}$ vertices in B_1. Therefore, $IGL(G' - X) \leq T$. Hence (G', k, T) is a Yes-instance for BIPARTITE-MINIGL. □

5 Polynomial Kernel for Parameter $k + T$

Here we prove that MINIGL is FPT for parameter $k + T$ by providing a polynomial-time preprocessing algorithm which produces an equivalent instance of MINIGL of size $O(k^2 + T)$. In the language of parameterized complexity, we design a kernel of size $O(k^2 + T)$. The following reduction rules are applied in the order we state them here.

Reduction Rule 5.1 (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 $T = q + \frac{q(q-1)}{4}$.

Reduction Rule 5.2 (High Degree Vertices)**.** If there exists a vertex $x \in V(G)$ such that $|N(x)| > q + k$, then remove x from G and set $k := k - 1$.

Reduction Rule 5.3 (Bounded Edge Set)**.** If $|E(G)| > T + k(q + k)$, then return NO.

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

Theorem 5.4. MinIGL has a kernel of size $O(k^2 + T)$.

Proof. Given an instance (G, k, T) of MINIGL by applying reduction rules 5.1–5.3 exhaustively, we obtain an instance (G', k', T) where number of edges in G' is at most $T + k(q + k) = O(T + k(\sqrt{T} + k)) = O(k^2 + T)$. Since degree of each vertex is at least 1, G' has $O(k^2 + T)$ vertices. □

6 FPT for Parameter Twin Cover Number

In this section we consider the twin cover number of the graph as a parameter. Although, the TWIN COVER problem (given a graph G and an integer k, does G have a twin cover of size at most k?) is NP-complete [Ganian, 2015], a smallest twin cover can be computed in time $O(|E| + l \cdot |V| + 1.2738^l)$, where l is the twin cover number [Ganian, 2015]. So, we may assume that the input contains a smallest twin cover of G.

6.1 Description of the Algorithm

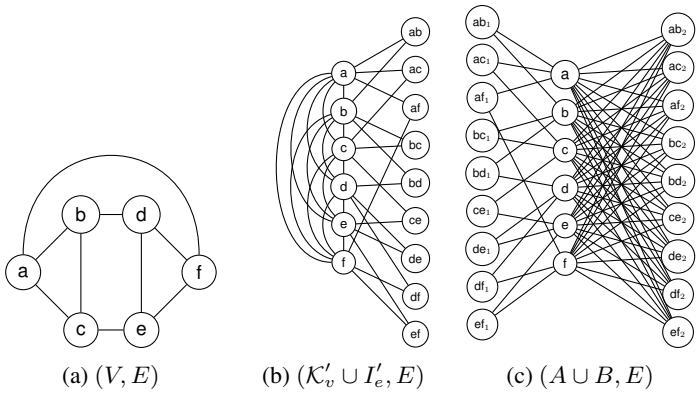
Let (G, k, T) be an instance of MINIGL and C be a smallest twin cover of G with $|C| = l$. Let $\mathcal{P}_c$ be a partition of $V(G) \setminus C$ into a set of equivalence classes $\{P_1, P_2, \ldots, P_q\}$, such that, for any two vertices u and v, if $N(u) \cap C = N(v) \cap C$, then u and v belong to the same equivalence class. This implies that $|\mathcal{P}_c| \leq 2^l$, as there is at most one equivalence class for each subset of C. Ganian [2015] observed that $G[V(G) \setminus C]$ is a disjoint union of cliques. In particular, for each $P_i \in \mathcal{P}_c$, $G[P_i]$ is a disjoint union of cliques. Let the largest clique in P_i be denoted as $Z_{max}(G[P_i])$. Let $\{\mathcal{P}_c^*, \mathcal{P}_s, \mathcal{P}_l\}$ be a partitioning of the equivalence classes in $\mathcal{P}_c$ as follows: an equivalence class $P_a \in \mathcal{P}_c$ belongs to $\mathcal{P}_c^*$ if there is no path from a vertex of P_a to a vertex in another equivalence class from $\mathcal{P}_c$; an equivalence class $P_b \in \mathcal{P}_c \setminus \mathcal{P}_c^*$ belongs to $\mathcal{P}_s$, if $|P_b| \leq 2l^2 + l$; lastly, $\mathcal{P}_l = \mathcal{P}_c \setminus (\mathcal{P}_c^* \cup \mathcal{P}_s)$. Using these notations, we outline Algorithm 1.

Input : A graph G, an integer k, a target IGL T, and a twin cover C of G
Parameter: $l = |C|$
Output : "yes" if (G, k, T) is a Yes-instance for MINIGL, and "no" otherwise

1 Guess which subset $C' \subseteq C$ to delete;
2 Define $\mathcal{P}_c$ for $V(G) \setminus C$;
3 Guess which equivalence classes to delete entirely;
4 Guess the number of vertices $k' \leq k$ to delete from equivalence classes in $\mathcal{P}_c^*$;
5 Greedily delete k' vertices from $\mathcal{P}_c^*$;
6 Guess which vertices from the equivalence classes in $\mathcal{P}_s$ to delete;
7 **if** $k \leq l$ **then**
8 Guess the number of vertices to delete from each equivalence class in $\mathcal{P}_l$ and greedily delete these vertices;
9 **else return** no;
10 **if** $IGL(G) \leq T$ **then return** yes;
11 **return** no;

Algorithm 1: MINIGL parameterized by the size of a twin cover of the input graph.

In the description of the algorithm, when we state that the algorithm *guesses* an object, this means that the algorithm branches into all possible values that this object can take. In Step 1, the algorithm guesses which subset $C' \subseteq C$ to delete, meaning that it branches into one subproblem for each $C' \subseteq C$ and in this subproblem, all deletion sets containing C' but no other vertex from C are considered. Then, C' is included in the solution and k is decreased by $|C'|$. From this point onward, we set $C := C \setminus C'$ and $l := |C|$. In Step 2, we define a set of equivalence classes $\mathcal{P}_c$ on $V(G) \setminus C$ as described earlier. In Step 3, the algorithm guesses which equivalence classes to delete entirely. In Step 4, the algorithm guesses the number of vertices k' that are deleted from $\mathcal{P}_c^*$. In Step 5, the algorithm performs k' iterations. In each iteration, for each equivalence class $P_a \in \mathcal{P}_c^*$, it picks an arbitrary vertex $v \in Z_{max}(G[P_a])$ and computes $IGL(G - v)$. It then greedily deletes one of those vertices whose deletion decreases the IGL the most. In Step 6, for each vertex in every equivalence class in $\mathcal{P}_s$, the algorithm guesses whether to include it in the solution. In Step 7, if the deletion budget exceeds the size of the remaining twin cover C then the algorithm returns "no", otherwise it performs Step 8. In Step 8, for each equivalence class $P_i \in \mathcal{P}_l$, the algorithm guesses the number of vertices to be deleted from P_i. For



(a) (V, E) (b) $(\mathcal{K}'_v \cup I'_e, E)$ (c) $(A \cup B, E)$

Figure 1: Figures (b) and (c) depict a graph obtained by transforming the 3-regular graph in (a).

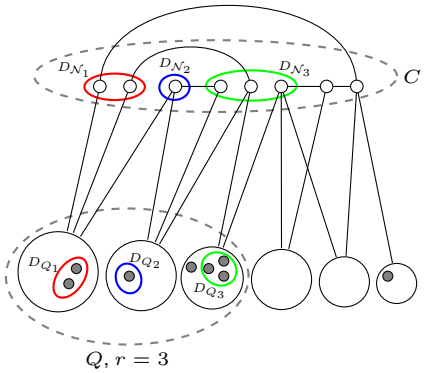


Figure 2: Connected component G' considered in the correctness proof of Step 9 of Algorithm 1.

each such guess, the corresponding number of vertices is iteratively and greedily deleted from the largest remaining clique $Z_{max}(G[P_i])$. In Step 10, the algorithm checks if the graph G obtained by deleting k vertices has the desired IGL. If so, it returns "yes" otherwise it returns "no".

6.2 Correctness

In this section we prove that the algorithm described in the previous section correctly solves MINIGL. Clearly, Steps 1-3 and 6 are exhaustive while Steps 4, 7, 10, and 11 are trivial. In Step 5 and 8, when the algorithm deletes a vertex from an equivalence class P_i, it greedily chooses a vertex from the largest clique $Z_{max}(G[P_i])$ in P_i. This greedy choice can be shown to be optimal by using the properties of twin covers.

Lemma 6.1. For a graph G, a twin cover C of G, an equivalence class $P_x \in \mathcal{P}_c$ defined over $V(G) \setminus C$ (as defined in Section 6.1) and a pair of vertices x, y, where $x \in P_x$ and $y \in Z_{max}(G[P_x])$, we have that $IGL(G-y) \leq IGL(G-x)$.

Apart from the greedy selection of vertices from each equivalence class in $\mathcal{P}_l$, the rest of Step 8 is exhaustive. It remains to show that Step 9 is sound, i.e., if we reach this step with $k > l$, and there is a set of vertices S such that $|S| \leq k$, $S \cap C = \emptyset$ and $IGL(G - S) \leq T$, then there is another set S^* such that $|S^*| \leq k$ and $IGL(G - S^*) \leq T$, but $S^* \cap C \neq \emptyset$. Therefore, the current choice for C' can be ignored. Before presenting the formal argument we make an observation that will help us prove the correctness of Step 9.

Observation 6.2. In a connected graph with twin cover number l, any two vertices are at distance at most $2l$.

Claim 6.3. Let G be the graph obtained by performing Steps 1 to 6 in Algorithm 1, with twin cover C and a set of equivalence classes $\mathcal{P}_l$. If the algorithm reaches Step 9 with $k > l$, and there is a set of vertices S such that $|S| = k$, $S \cap C = \emptyset$ and $IGL(G - S) \leq T$, then there is a set S^* such that $|S^*| = k$ and $IGL(G - S^*) \leq T$, but $S^* \cap C \neq \emptyset$.

Proof. Let $S \subseteq (V(G) \setminus C)$ such that $|S| = k$ and $IGL(G - S) \leq T$. Let G' be a connected component of G with twin cover $C' \subseteq C$ and a set of equivalence classes $\mathcal{P}'_l \subseteq \mathcal{P}_l$ such that $|S \cap V(G')| \geq |\bigcup_{P_i \in \mathcal{P}'_l} N(P_i)|$. It is easy to see that such a connected component G' always exists, as $k > l$. Let $\mathcal{P}'_l = \{Q_1, Q_2, \ldots, Q_q\}$, where $q \leq 2^l$ and $|Q_i| \geq |Q_{i+1}|$. Let $S' = V(G') \cap S$. We show that if $k > l$, there exist a set S^* with $|S^*| = |S'|$, such that $IGL(G' - S^*) \leq IGL(G' - S')$ and $S^* \cap C' \neq \emptyset$. Let r be the smallest index such that for $Q = Q_1 \cup Q_2 \cup Q_3 \ldots \cup Q_r$, $|Q \cap S'| \geq |\bigcup_{i=1}^{r} N(Q_i)|$. Since $|S'| \geq |N(Q)|$, we can always find such an index r. Let $\mathcal{P}_r = \{Q_1, Q_2, \ldots, Q_r\}$. We order the vertices in $N(Q)$ in descending order of the size of their neighboring equivalence class (if a vertex is adjacent to more than one equivalence class we order it with respect to the largest equivalence class it is adjacent to, and ties are broken arbitrarily). Denote this ordering by π. For $1 \leq i \leq r$, denote $D_{Q_i} = Q_i \cap S'$. For $1 \leq i < r$, let $k_i = |D_{Q_i}|$ while $k_r = |N(Q)| - \sum_{i=1}^{r-1} k_i$. To obtain the set S^*, we start from S' and then replace each set $D_{Q_i} \subseteq S'$ with a set of vertices $D_{\mathcal{N}_i} \subseteq N(Q)$ with $|D_{\mathcal{N}_i}| = k_i$. For each i from 1 to r, the vertices composing $D_{\mathcal{N}_i}$ are picked sequentially from $N(Q)$ according to π. Accordingly, $D_{\mathcal{N}_i} \subseteq (N(Q_1) \cup N(Q_2) \cup \ldots \cup N(Q_i))$ and each vertex $u \in N(Q)$ belongs to exactly one such set $D_{\mathcal{N}_i}$. Figure 2 depicts G' and highlights each D_{Q_i} along with its corresponding $D_{\mathcal{N}_i}$. When deleting a vertex set X from G', we say that the contribution of a vertex $v \in X$ is $\sum_{u \in X, u \neq v} \frac{1}{2 \cdot d(u,v)} + \sum_{u \in V(G') \setminus X} \frac{1}{d(u,v)}$. In other words, for a vertex pair (u, v), we attribute $\frac{1}{d(u,v)}$ if $u \notin X$, but only half of that quantity if $u \in X$. Thus, the contributions of all vertices in X equals the decrease in IGL when X is deleted.

Let us now compare the mutually exclusive impact of deleting D_{Q_i} and $D_{\mathcal{N}_i}$. Note that no remaining equivalence class is deleted entirely, due to Step 3. In particular, this means that all vertices in C remain at the same distance.

First let us consider the impact of deleting the set of vertices D_{Q_i}. We delete $(|Q_i| - 1) + (|Q_i| - 2) + \ldots + (|Q_i| - k_i)$ paths of length one that connect the vertices in D_{Q_i} to each other and to the remaining vertices in Q_i. Note that $\sum_{j=1}^{k_i} (|Q_i| - j) \leq |D_{Q_i}| \cdot |Q_i|$. Moreover, we know that $|N(D_{Q_i}) \cap C| \leq |N(Q)|$. Therefore, by deleting D_{Q_i} we delete at most $|D_{Q_i}| \cdot |N(Q)|$ paths of length one that connect vertices in $N(Q)$ to the vertices in D_{Q_i}. Overall, the exclusive impact of deleting D_{Q_i} is at most $|D_{Q_i}| \cdot |Q_i| +$

$|D_{Q_i}| \cdot (|N(Q)|)$. Alternatively, we consider deleting the set of vertices $D_{\mathcal{N}_i} \subseteq N(Q)$. Since each vertex in $D_{\mathcal{N}_i}$ has at least $|Q_i|$ neighbors in Q (given that $D_{\mathcal{N}_i} \subseteq (N(Q_1) \cup N(Q_2) \ldots \cup N(Q_i))$ and $|Q_i| \geq |Q_{i+1}|$), by deleting the set of vertices $D_{\mathcal{N}_i}$, we delete at least $|D_{\mathcal{N}_i}| \cdot |Q_i|$ paths of length one. By Observation 6.2 we know that for any pair of vertices $u, v \in N(Q)$, $d(u,v) \leq 2l$. Therefore, for each vertex $x \in D_{\mathcal{N}_i}$, we attribute at least half of $\frac{1}{2l}$ to x for each vertex $y \in N(Q) \setminus \{x\}$. Therefore, the contribution of $D_{\mathcal{N}_i}$ is at least $\frac{|D_{\mathcal{N}_i}| \cdot (|N(Q)|-1)}{4l}$ for their distances to the vertices in $N(Q)$. So far, for the deletion of $D_{\mathcal{N}_i}$ we have counted an exclusive impact of at least $|D_{\mathcal{N}_i}| \cdot |Q_i| + \frac{|D_{\mathcal{N}_i}| \cdot (|N(Q)|-1)}{4l}$. Observe that G' contains at least two equivalence classes (connected components with one equivalence class are handled in Step 5 of the algorithm). Since we replace each D_{Q_i} by $D_{\mathcal{N}_i}$ and $N(Q) = \bigcup_{i=1}^{r} D_{\mathcal{N}_i}$, we disconnect all the equivalence classes in $\mathcal{P}_r$ from each other and from $V(G') \setminus Q$. We know that the distance between any two equivalence classes is at most $2l$. Therefore, for a pair of equivalence classes (Q_i, Q_j) where $Q_j \in (\mathcal{P}_l \setminus \{Q_i\})$, we delete $|Q_i| \cdot |Q_j|$ paths of length at most $2l$. However $|D_{Q_i}| \cdot |Q_j| + |D_{Q_j}| \cdot |Q_i| - |D_{Q_i}| \cdot |D_{Q_j}|$ of these paths are also deleted when we delete $D_{Q_i} \cup D_{Q_j}$. Hence, the exclusive impact of deleting vertices in $D_{\mathcal{N}_i}$ is at least $\frac{(|Q_i|-|D_{Q_i}|) \cdot (|Q_j|-|D_{Q_j}|)}{4l}$ for these paths between equivalence classes. Therefore, in order to show that $IGL(G' - S^*) \leq IGL(G' - S')$, it suffices to show that

$$\frac{1}{4l}(|Q_i| - |D_{Q_i}|) \cdot (|Q_j| - |D_{Q_j}|) \geq |D_{Q_i}| \cdot (|N(Q)|) - \frac{1}{4l}|D_{\mathcal{N}_i}| \cdot (|N(Q)| - 1). \quad (4)$$

We know that size of both equivalence classes $|Q_i|, |Q_j|$ is at least $2l^2 + l$. Also, we know that $|D_{Q_i}| = |D_{\mathcal{N}_i}| \leq l$ and $|D_{Q_j}| \leq l$. Similarly, we know that $|N(Q)| \leq l$. Thus, replacing the values in equation (4) we get that $l^3 \geq \frac{4l^2-l+1}{4}$., which holds for any $l \geq 1$. This proves that $IGL(G' - S^*) \leq IGL(G' - S')$. Moreover, $|S^*| = |S'|$ and $S^* \cap C' \neq \emptyset$. □

6.3 Running Time

By considering the number of choices in each guessing step, and computing the IGL of a graph using a polynomial-time algorithm solving ALL-PAIRS SHORTEST PATH [Thorup, 1999], one can upper bound the running time of the algorithm by $2^l \cdot 2^{2^l} \cdot n^{O(1)} \cdot 2^{2^l \cdot (2l^2+l)} \cdot (l+1)^{2^l}$. More concisely, its running time is $2^{O(2^l \cdot l^2)} \cdot n^{O(1)}$.

Theorem 6.4. MINIGL is FPT when parameterized by the twin cover number.

7 Faster Algorithm for Parameter Vertex Cover Number

The previous result implies that MINIGL is FPT when parameterized by the vertex cover number of the input graph. Here, we give a faster algorithm for this parameter.

Theorem 7.1. MINIGL can be solved in time $2^{O(l^2)} \cdot n^{O(1)}$, where l is the vertex cover number of the input graph.

Proof. Let (G, k, T) be an input for MINIGL. A smallest vertex cover S for G can be computed in $1.2738^l \cdot n^{O(1)}$ time [Chen *et al.*, 2010]. Let $I = V(G) \setminus S$ and note that I is an independent set. Let $\mathcal{I}_c = \{I_1, I_2, ..., I_q\}$ be a set of equivalence classes partitioning I such that two vertices from I belong to the same equivalence class if they have the same neighborhood in C. Thus, $q \leq 2^l$. Note that if $k \geq l$, we can choose $X = S$ as $IGL(G - S) = 0$. Now we consider the case when $k < l$. First observe that for all $u, v \in I_i$, $IGL(G - u) = IGL(G - v)$. Using this fact, we define a branching algorithm that branches into at most $(2^l + l)$ subinstances to choose a vertex from S or an equivalence class. In a subinstance, the algorithm deletes the chosen vertex from S or an arbitrary vertex from the chosen equivalence class, and k decreases by one. The resulting search tree has size at most $(2^l + l)^k \leq (2^l + l)^l$. □

8 FPT for Parameter $\omega + k$

We now outline an algorithm for MINIGL parameterized by the neighborhood diversity and deletion budget combined. Although simple, it is of practical interest as it is never asymptotically slower than a brute-force $n^{k+O(1)}$ approach.

Theorem 8.1. MINIGL can be solved in time $\omega^k \cdot n^{O(1)}$, where ω is the neighborhood diversity of the input graph.

Proof. Let (G, k, T) be an instance of MINIGL and ω be the neighborhood diversity of G. Let $V_d = \{V_1, V_2, \cdots, V_\omega\}$ be the neighborhood partition of G. Let $u, v \in V_i$, by definition of neighborhood partition, $N(u) = N(v)$, hence $IGL(G - u) = IGL(G - v)$. Considering this observation, we define a branching algorithm that branches on each part $V_i \in V_d$ and deletes a vertex arbitrarily from the selected part. Since we delete k vertices the branching tree has depth k. Consequently, the algorithm runs in time $\omega^k \cdot n^{O(1)}$. □

9 Future Directions

Our results point to natural future directions. Besides vertex deletion as destructive action, other actions are of interest, e.g., removing edges or decreasing the weights of edges. Structural parameters such as treewidth, cliquewidth, and feedback vertex set number are also of interest, as well as approximation algorithms. But the most pressing question that we leave open is the complexity of MINIGL on trees.

Acknowledgements

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). Aziz is supported by a Julius Career Award.

References

[Amer and Giménez, 2004] Rafael Amer and José M. Giménez. A connectivity game for graphs. *Math. Method. of Oper. Res.*, 60:453–470, 2004.

[Barabási and Albert, 1999] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[Belik, 2016] Ivan Belik. The analysis of split graphs in social networks based on the k-cardinality assignment problem. *IJNS*, 1(1):53–62, 2016.

[Borgatti, 2012] Stephen P. Borgatti. Two-mode concepts in social network analysis. *Computational Complexity: Theory, Techniques, and Applications*, pages 2912–2924, 2012.

[Cai, 2008] Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008.

[Carley *et al.*, 2003] Kathleen M. Carley, Jeffrey Reminga, and Natasha Kamneva. Destabilizing terrorist networks. In *Proc. of NAACSOS*. Wilcox Steven P, 2003.

[Chen *et al.*, 2010] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci*, 411(40-42):3736–3756, 2010.

[Chen *et al.*, 2012] Duanbing Chen, Linyuan L, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica A*, 391(4):1777–1787, 2012.

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

[Dagon *et al.*, 2008] David Dagon, Guofei Gu, and Christopher P. Lee. *A Taxonomy of Botnet Structures*, pages 143–164. Springer, Boston, MA, 2008.

[Diestel, 2010] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2010.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Drange *et al.*, 2016] Pål G. Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

[Fellows *et al.*, 2008] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In *Proc. of ISAAC*, pages 294–305. Springer, 2008.

[Fomin *et al.*, 2014] Fedor V. Fomin, Mathieu Liedloff, Pedro Montealegre-Barba, and Ioan Todinca. Algorithms parameterized by vertex cover and modular width, through potential maximal cliques. In *Proc. of SWAT*, pages 182–193, 2014.

[Ganian, 2012] Robert Ganian. Using neighborhood diversity to solve hard problems. *CoRR*, abs/1201.3091, 2012.

[Ganian, 2015] Robert Ganian. Improving vertex cover as a graph parameter. *Discrete Math. Theor. Comput. Sci.*, 17(2):77–100, 2015.

[Gross *et al.*, 2013] Daniel Gross, Monika Heinig, Lakshmi Iswara, L. William Kazmierczak, Kristi Luttrell, John T. Saccoman, and Charles Suffel. A survey of component order connectivity models of graph theoretic networks. *WSEAS Transaction Math*, 12:895–910, 2013.

[Holme *et al.*, 2002] Petter Holme, Beom J. Kim, Chang N. Yoon, and Seung K. Han. Attack vulnerability of complex networks. *Phys. Rev. E*, 65:056109, 2002.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Proc. of Symp. on Complexity of Computer Computations*, pages 85–103, 1972.

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

[Lampis, 2012] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[Lindelauf *et al.*, 2013] Roy Lindelauf, Herbert Hamers, and Bart Husslage. Cooperative game theoretic centrality analysis of terrorist networks: The cases of jemaah islamiyah and al qaeda. *Eur. J. Oper. Res*, 229(1):230–238, 2013.

[Michalak *et al.*, 2013] Tomasz P. Michalak, Talal Rahwan, Piotr L. Szczepanski, Oskar Skibski, Ramasuri Narayanam, Nicholas R. Jennings, and Michael J. Wooldridge. Computational analysis of connectivity games with applications to the investigation of terrorist networks. In *Proc. of IJCAI*, pages 293–301, 2013.

[Michalak *et al.*, 2015] Tomasz P. Michalak, Talal Rahwan, Oskar Skibski, and Michael J. Wooldridge. Defeating terrorist networks with game theory. *IEEE Intell. Syst.*, 30(1):53–61, 2015.

[Michalak *et al.*, 2017] Tomasz P. Michalak, Talal Rahwan, and Michael J. Wooldridge. Strategic social network analysis. In *Proc. of AAAI*, 2017.

[Morone and Makse, 2015] Flaviano Morone and Hernán A. Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524:6568, 2015.

[Szczepanski *et al.*, 2015] Piotr L. Szczepanski, Tomasz P. Michalak, and Talal Rahwan. Efficient algorithms for game-theoretic betweenness centrality. *Artif. Intell.*, 231:39–63, 2015.

[Thorup, 1999] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM*, 46(3):362–394, 1999.

[UCINET Software, 2017] UCINET Software. Datasets: Covert networks, 2017. Retrieved from `https://sites.google.com/site/ucinetsoftware/datasets/covert-networks` on 2017-02-17.

[Zheng *et al.*, 2011] Alice X. Zheng, John Dunagan, and Ashish Kapoor. Active graph reachability reduction for network security and software engineering. In *Proc. of IJCAI*, 2011.