Being Corrupt Requires Being Clever, But Detecting Corruption Doesn't

Yan Jin[1]
MIT, 77 Massachusetts Ave, MA, USA
yjin1@mit.edu

Elchanan Mossel[2]
MIT, 77 Massachusetts Ave, MA, USA
elmos@mit.edu

Govind Ramnarayan[3]
MIT, 77 Massachusetts Ave, MA, USA
govind@mit.edu

Abstract

We consider a variation of the problem of corruption detection on networks posed by Alon, Mossel, and Pemantle '15. In this model, each vertex of a graph can be either truthful or corrupt. Each vertex reports about the types (truthful or corrupt) of all its neighbors to a central agency, where truthful nodes report the true types they see and corrupt nodes report adversarially. The central agency aggregates these reports and attempts to find a single truthful node. Inspired by real auditing networks, we pose our problem for arbitrary graphs and consider corruption through a computational lens. We identify a key combinatorial parameter of the graph $m(G)$, which is the minimal number of corrupted agents needed to prevent the central agency from identifying a single corrupt node. We give an efficient (in fact, linear time) algorithm for the central agency to identify a truthful node that is successful whenever the number of corrupt nodes is less than $m(G)/2$. On the other hand, we prove that for any constant $\alpha > 1$, it is NP-hard to find a subset of nodes S in G such that corrupting S prevents the central agency from finding one truthful node and $|S| \leq \alpha m(G)$, assuming the Small Set Expansion Hypothesis (Raghavendra and Steurer, STOC '10). We conclude that being corrupt requires being clever, while detecting corruption does not.

Our main technical insight is a relation between the minimum number of corrupt nodes required to hide all truthful nodes and a certain notion of vertex separability for the underlying graph. Additionally, this insight lets us design an efficient algorithm for a corrupt party to decide which graphs require the fewest corrupted nodes, up to a multiplicative factor of $O(\log n)$.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Corruption detection, PMC Model, Small Set Expansion, Hardness of Approximation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.45

Related Version A full version of the paper with all proofs can be found at [8], `https://arxiv.org/abs/1809.10325`.

[1] Partially supported by Institute for Data, Systems and Society Fellowship.
[2] Partially supported by awards ONR N00014-16-1-2227, NSF CCF1665252 and DMS-1737944.
[3] Partially supported by awards NSF CCF 1665252 and DMS-1737944.

Acknowledgements We would like to thank Paxton Turner, Vishesh Jain, and Pasin Manurangsi for useful discussions and pointers to resources.

1 Introduction

1.1 Corruption Detection and Problem Set-up

We study the problem of identifying truthful nodes in networks, in the model of *corruption detection on networks* posed by Alon, Mossel, and Pemantle [1]. In this model, we have a network represented by a (possibly directed) graph. Nodes can be *truthful* or *corrupt*. Each node audits its outgoing neighbors to see whether they are truthful or corrupt, and sends reports of their identities to a central agency. The central agent, who is not part of the graph, aggregates the reports and uses them to identify truthful and corrupt nodes. Truthful nodes report truthfully (and correctly) on their neighbors, while corrupt nodes have no such restriction: they can assign arbitrary reports to their neighbors, regardless of whether their neighbors are truthful or corrupt, and coordinate their efforts with each other to prevent the central agency from gathering useful information.

In [1], the authors consider the problem of recovering the identities of almost all nodes in a network in the presence of many corrupt nodes; specifically, when the fraction of corrupt nodes can be very close to $1/2$. They call this the *corruption detection* problem. They show that the central agency can recover the identity of most nodes correctly even in certain bounded-degree graphs, as long as the underlying graph is a sufficiently good expander. The required expansion properties are known to hold for a random graph or Ramanujan graph of sufficiently large (but constant) degree, which yields undirected graphs that are amenable to corruption detection. Furthermore, they show that some level of expansion is necessary for identifying truthful nodes, by demonstrating that the corrupt nodes can stop the central agency from identifying any truthful node when the graph is a very bad expander (e.g. a cycle), even if the corrupt nodes only make up 0.01 fraction of the network.

This establishes that very good expanders are very good for corruption detection, and very bad expanders can be very bad for corruption detection. We note that this begs the question of how effective graphs that do not fall in either of these categories are for corruption detection. In the setting of [1], we could ask the following: given an *arbitrary* undirected graph, what is the smallest number of corrupt nodes that can prevent the identification of almost all nodes? When there are fewer than this number, can the central agency *efficiently* identify almost all nodes correctly? Alon, Mossel, and Pemantle study these questions for the special cases of highly expanding graphs and poorly expanding graphs, but do not address general graphs.

Additionally, [1] considers corruption detection when the corrupt agencies can choose their locations and collude arbitrarily, with no bound on their computational complexity. This is perhaps overly pessimistic: after all, it is highly unlikely that corrupt agencies can solve NP-hard problems efficiently and if they can, thwarting their covert operations is unlikely to stop their world domination. We suggest a model that takes into account computational considerations, by factoring in the computation time required to select the nodes in a graph that a corrupt party chooses to control. This yields the following question from the viewpoint of a corrupt party: given a graph, can a corrupt party compute the smallest set of nodes it needs to corrupt *in polynomial time*?

In addition to being natural from a mathematical standpoint, these questions are also well-motivated socially. It would be naïve to assert that we can weed out corruption in the real world by simply designing auditing networks that are expanders. Rather, these

networks may already be formed, and infeasible to change in a drastic way. Given this, we are less concerned with finding certain graphs that are good for corruption detection, but rather discerning how good *existing* graphs are; specifically, how many corrupt nodes they can tolerate. In particular, since the network structure could be out of the control of the central agency, algorithms for the central agency to detect corruption on arbitrary graphs seem particularly important.

It is also useful for the *corrupt* agency to have an algorithm with guarantees for any graph. Consider the following example of a corruption detection problem from the viewpoint of a corrupt organization. Country A wants to influence policy in country B, and wants to figure out the most efficient way to place corrupted nodes within country B to make this happen. However, if the central government of B can confidently identify truthful nodes, they can weight those nodes' opinions more highly, and thwart country A's plans. Hence, the question country A wants to solve is the following: given the graph of country B, can country A compute the optimal placement of corrupt nodes to prevent country B from finding truthful nodes? We note that in this question, too, the graph of country B is fixed, and hence, country A would like to have an algorithm that takes as input *any* graph and computes the optimal way to place corrupt nodes in order to hide all the truthful nodes.

We study the questions above for a variant of the corruption detection problem in [1], in which the goal of the central agency is to find a single truthful node. While this goal is less ambitious than the goal of identifying almost all the nodes, we think it is a very natural question in the context of corruption. For one, if the central agency can find a single truthful node, they can use the trusted reports from that node to identify more truthful and corrupt nodes that it might be connected to. The central agency may additionally weight the opinions of the truthful nodes more when making policy decisions (as alluded to in the example above), and can also incentivize truthfulness by rewarding truthful nodes that it finds and giving them more influence in future networks if possible (by increasing their out-degrees). Moreover, our proofs and results extend to finding larger number of truthful nodes as we discuss below.

Our results stem from a tie between the problem of finding a single truthful node in a graph and a measure of vertex separability of the graph. This tie not only yields an efficient and relatively effective algorithm for the central agency to find a truthful node, but also allows us to relate corrupt party's strategy to the problem of finding a good vertex separator for the graph. Hence, by analyzing the purely graph-theoretic problem of finding a good vertex separator, we can characterize the difficulty of finding a good set of nodes to corrupt. Similar notions of vertex separability have been studied previously (e.g. [13, 17, 3]), and we prove NP-hardness for the notion relevant to us assuming the Small Set Expansion Hypothesis (SSEH). The *Small Set Expansion Hypothesis* is a hypothesis posed by Raghavendra and Steurer [19] that is closely related to the famous Unique Games Conjecture of Khot [11]. In fact, [19] shows that the SSEH implies the Unique Games Conjecture. The SSEH yields hardness results that are not known to follow directly from the UGC, especially for graph problems like sparsest cut and treewidth ([20] and [2] respectively), among others.

1.2 Our Results

We now outline our results more formally. We analyze the variant of corruption detection where the central agency's goal is to find a single truthful node. First, we study how effectively the central agency can identify a truthful node on an arbitrary graph, given a set of reports.

Given an undirected graph[4] G, we let $m(G)$ denote the minimal number of corrupted nodes required to stop the central agency from finding a truthful node, where the minimum is taken over all strategies of the corrupt party (not just computationally bounded ones). We informally call $m(G)$ the "critical" number of corrupt nodes for a graph G. Then, we show the following:

▶ **Theorem 1.** *Fix a graph G and suppose that the corrupt party has a budget $b \leq m(G)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without knowledge of either $m(G)$ or b. Furthermore, the central agency's algorithm runs in linear time (in the number of edges in the graph G).*

Next, we consider the question from the viewpoint of the corrupt party: can the corrupt party efficiently compute the most economical way to allocate nodes to prevent the central agency from finding a truthful node? Concretely, we focus on a natural decision version of the question: given a graph G and a upper bound on the number of possible corrupted nodes k, can the corrupt party prevent the central agency from finding a truthful node?

We actually focus on an easier question: can the corrupt party accurately compute $m(G)$, the minimum number of nodes that they need to control to prevent the central agency from finding a truthful node? Not only do we give evidence that computing $m(G)$ exactly is computationally hard, but we also provide evidence that $m(G)$ is hard to approximate. Specifically, we show that approximating $m(G)$ to any constant factor is NP-hard under the Small Set Expansion Hypothesis (SSEH); or in other words, that it is SSE-hard.

▶ **Theorem 2.** *For every $\beta > 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V, E)$, it is SSE-hard to distinguish between the case where $m(G) \leq \epsilon \cdot |V|$ and $m(G) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes for a graph to within any constant factor is SSE-hard.*

This Theorem immediately implies the following Corollary 25.

▶ Corollary 25 (restated). Assume the SSE Hypothesis and that P $\neq$ NP. Fix any $\beta > 1$. There does not exist a polynomial-time algorithm that takes as input an arbitrary graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\beta \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.

We note that in Corollary 25, the bad party's input is only the graph G: specifically, they do not have knowledge about the value of $m(G)$.

Our proof for Theorem 2 is similar to the proof of Austrin, Pitassi, and Wu [2] for the SSE-hardness of approximating treewidth. This is not a coincidence: in fact, the "soundness" in their reduction involves proving that their graph does not have a good 1/2 vertex separator, where the notion of vertex separability (from [4]) is very related to the version we use to categorize the problem of hiding a truthful vertex. We give the proof of Theorem 2 in Section 3.2.

However, if one allows for an approximation factor of $O(\log |V|)$, then $m(G)$ can be approximated efficiently. Furthermore, this yields an approximation algorithm that the corrupt party can use to find a placement that hinders detection of a truthful node.

▶ **Theorem 3.** *There is a polynomial-time algorithm that takes as input a graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\log |V| \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

[4] Unless explicitly specified, all graphs are undirected by default.

The proof of Theorem 3, given in Section 3.2, uses a bi-criterion approximation algorithm for the k-vertex separator problem given by [13]. As alluded to in Section 1.1, Theorems 2 and 3 both rely on an approximate characterization of $m(G)$ in terms of a measure of vertex separability of the graph G, which we give in Section 3.

Additionally, we note that we can adapt Theorems 1 and 2 (as well as Corollary 25) to a more general setting, where the central agency wants to recover some arbitrary number of truthful nodes, where the number of nodes can be proportional to the size of the graph. We describe how to modify our proofs to match this more general setting in Section 5 in full-length version [8].

Together, Theorems 1 and 2 uncover a surprisingly positive result for corruption detection: it is computationally easy for the central agency to find a truthful node when the number of corrupted nodes is only somewhat smaller than the "critical" number for the underlying graph, but it is in general computationally hard for the corrupt party to hide the truthful nodes even when they have a budget that far exceeds the "critical" number for the graph.

1.2.1 Results for Directed Graphs

As noted in [1], it is unlikely that real-world auditing networks are undirected. For example, it is likely that the FBI has the authority to audit the Cambridge police department, but it is also likely that the reverse is untrue. Therefore, we would like the central agency to be able to find truthful nodes in directed graphs in addition to undirected graphs. We notice that the algorithm we give in Theorem 1 extends naturally to directed graphs.

▶ **Theorem 4.** *Fix a directed graph D and suppose that the corrupt party has a budget $b \leq m(D)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without the knowledge of either $m(D)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

The proof of Theorem 4 is similar to the proof of Theorem 1, and effectively relates the problem of finding a truthful node on directed graphs to a similar notion of vertex separability, suitably generalized to directed graphs.

1.2.2 Results for Finding An Arbitrary Number of Good Nodes

In fact, the problem of finding one good node is just a special case of finding an arbitrary number of good nodes, g, on the graph G. We define $m(G, g)$ as the minimal number of bad nodes required to prevent the identification of g good nodes on the graph G. We relate it to an analogous vertex separation notion, and prove the following two theorems, which are extensions of Theorems 1 and 2 to this setting.

▶ **Theorem 5.** *Fix a graph G and the number of good nodes to recover, g. Suppose that the corrupt party has a budget $b \leq m(G, g)/2$. If $g < |V| - 2b$, then the central agency can identify g truthful nodes, regardless of the strategy of the corrupt party, and without knowledge either of $m(G, g)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

▶ **Theorem 6.** *For every $\beta > 1$ and every $0 < \delta < 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V, E)$, it is SSE-hard to distinguish between the case where $m(G, \delta|V|) \leq \epsilon \cdot |V|$ and $m(G, \delta|V|) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes such that it is impossible to find $\delta|V|$ good nodes within any constant factor is SSE-hard.*

The proof of Theorem 6 is similar to the proof of Theorem 1, and the hardness of approximation proof also relies on the same graph reduction and SSE conjecture. Proofs are presented in Section 5 in our full-length paper [8].

1.3 Related Work

The model of corruptions posed by [1] is identical to a model first suggested by Preparata, Metze, and Chien [18], who introduced the model in the context of detecting failed components in digital systems. This work (as well as many follow-ups, e.g. [9, 12]) looked at the problem of characterizing which networks can detect a certain number of corrupted nodes. Xu and Huang [22] give necessary and sufficient conditions for identifying a single corrupted node in a graph, although their characterization is not algorithmically efficient. There are many other works on variants of this problem (e.g. [21, 5]), including recovering node identities with one-sided or two-sided error probabilities in the local reports [14] and adaptively finding truthful nodes [7].

We note that our model of a computationally bounded corrupt party and our stipulation that the graph is fixed ahead of time rather than designed by the central agency, which are our main contributions to the model, seem more naturally motivated in the setting of corruptions than in the setting of designing digital systems. Even the question of identifying a single truthful node could be viewed as more naturally motivated in the setting of corruptions than in the setting of diagnosing systems. We believe there are likely more interesting theoretical questions to be discovered by approaching the PMC model through a corruptions lens.

The identifiability of a single node in the corruptions setting was studied in a recent paper of Mukwembi and Mukwembi [15]. They give a linear time greedy algorithm to recover the identify of a single node in many graphs, *provided that corrupt nodes always report other corrupt nodes as truthful*. Furthermore, this assumption allows them to reduce identifying all nodes to identifying a single node. They argue that such an assumption is natural in the context of corruptions, where corrupt nodes are selfishly incentivized not to out each other. However, in our setting, corrupt nodes can not only betray each other, but are in fact incentivized to do so for the good of the overarching goal of the corrupt party (to prevent the central agency from identifying a truthful node). Given [15], it is not a surprise that the near-optimal strategies we describe for the corrupt party in this paper crucially rely on the fact that the nodes can report each other as corrupt.

Our problem of choosing the best subset of nodes to corrupt bears intriguing similarities to the problem of influence maximization studied by [10], where the goal is to find an optimal set of nodes to target in order to maximize the adoption of a certain technology or product. It is an interesting question to see if there are further similarities between these two areas. Additionally, social scientists have studied corruption extensively (e.g.[6], [16]), though to the best of our knowledge they have not studied it in the graph-theoretic way that we do in this paper.

1.4 Comparison to Corruption in Practice

Finally, we must address the elephant in the room. Despite our theoretical results, corruption *is* prevalent in many real-world networks, and yet in many scenarios it is not easy to pinpoint even a single truthful node. One reason for that is that some of assumptions do not seem to hold in some real world networks. For example, we assume that audits from the truthful nodes are not only non-malicious, but also perfectly reliable. In practice this assumption is unlikely to be true: many truthful nodes could be non-malicious but simply unable to audit

their neighbors accurately. Further assumptions that may not hold in some scenarios include the notion of a central agency that is both uncorrupted and has access to reports from every agency, and possibly even the assumption that the number of corrupt nodes is less than $|V|/2$. In addition, networks G may have very low critical numbers $m(G)$ in practice. For example, there could be a small set of three nodes (named, "President", "Congress" and "Houses") that is all corrupt, and all audits in the graph are performed by one of these three nodes. It is thus plausible that a corrupt party could use the structure of realistic auditing networks for their corruption strategy to overcome our worst-case hardness result.

While this points to some shortcomings of our model, it also points out ways to change policy that would potentially bring the real world closer to our idealistic scenario, where a corrupt party has a much more difficult computational task than the central agency. For example, we can speculate that perhaps information should be gathered by a transparent centralized agency, that significant resources should go into ensuring that the centralized agency is not corrupt, and that networks ought to have good auditing structure (without important agencies that can be audited by very few nodes).

2 Preliminaries

2.1 General Preliminaries

We denote undirected graphs by $G = (V, E)$, where V is the vertex set of the graph and E is the edge set. We denote directed graphs by $D = (V, E_D)$. When the underlying graph is clear, we may drop the subscripts. Given a vertex u in an undirected graph G, we let $\mathcal{N}(u)$ denote the *neighborhood* (set of neighbors) of the vertex in G. Similarly, given a vertex u in a directed graph D, let $\mathcal{N}(u)$ denote the set of *outgoing* neighbors of u: that is, vertices $v \in V$ such that $(u, v) \in E_D$.

2.1.1 Vertex Separator

▶ **Definition 7** (k-vertex separator,[17],[3]). For any $k \geq 0$, we say a subset of vertices $U \subseteq V$ is k-vertex separator of a graph G, if after removing U and incident edges, the remaining graph forms a union of connected components, each of size at most k.

Furthermore, let

$$S_G(k) = \min\big(|U| : U \text{ is a } k\text{-vertex separator of } G\big)$$

denote the size of the minimal k-vertex separator of graph G.

2.1.2 Small Set Expansion Hypothesis

In this section we define the Small Set Expansion (SSE) Hypothesis introduced in [19]. Let $G = (V, E)$ be an undirected d-regular graph.

▶ **Definition 8** (Normalized edge expansion). For a set $S \subseteq V$ of vertices, denote $\Phi_G(S)$ as the normalized edge expansion of S,

$$\Phi_G(S) = \frac{|E(S, V \backslash S)|}{d|S|},$$

where $|E(S, V \backslash S)|$ is the number of edges between S and $V \backslash S$.

The Small Set Expansion Problem with parameters η and δ, denoted $\mathrm{SSE}(\eta, \delta)$, asks whether G has a small set S which does not expand or all small sets of G are highly expanding.

▶ **Definition 9** (SSE(η, δ))**.** Given a regular graph $G = (V, E)$, distinguish between the following two cases:

- **Yes** There is a set of vertices $S \subseteq V$ with $S = \delta|V|$ and $\Phi_G(S) \leq \eta$
- **No** For every set of vertices $S \subseteq V$ with $S = \delta|V|$ it holds that $\Phi_G(S) \geq 1 - \eta$

The *Small Set Expansion Hypothesis* is the conjecture that deciding SSE(η, δ) is NP-hard.

▶ **Conjecture 10** (Small Set Expansion Hypothesis [19])**.** *For every $\eta > 0$, there is a $\delta > 0$ such that SSE(η, δ) is NP-hard.*

We say that a problem is *SSE-hard* if it is at least as hard to solve as the SSE problem. The form of conjecture most relevant to our proof is the following "stronger" form of the SSE Hypothesis. [20] showed that the SSE-problem can be reduced to a quantitatively stronger form of itself. In order to state this version, we first need to define the *Gaussian noise stability*.

▶ **Definition 11** (Gaussian Noise Stability)**.** Let $\rho \in [-1, 1]$. Define $\Gamma_\rho : [0, 1] \mapsto [0, 1]$ by

$$\Gamma_\rho(\mu) = Pr[X \leq \Phi^{-1}(\mu) \wedge Y \leq \Phi^{-1}(\mu)]$$

where X and Y are jointly normal random variables with mean 0 and covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$.

The only fact that we will use for stating the stronger form of SSEH is the asymptotic behavior of $\Gamma_\rho(\mu)$ when ρ is close to 1 and μ bounded away from 0.

▶ **Fact 12.** *There is a constant $c > 0$ such that for all sufficiently small ϵ and all $\mu \in [1/10, 1/2]$,*[5]

$$\Gamma_{1-\epsilon}(\mu) \leq \mu(1 - c\sqrt{\epsilon}).$$

▶ **Conjecture 13** (SSE Hypothesis, Equivalent Formulation [20])**.** *For every integer $q > 0$ and $\epsilon, \gamma > 0$, it is NP-hard to distinguish between the following two cases for a given regular graph $G = (V, E)$:*

- ***Yes*** *There is a partition of V into q equi-sized sets $S_1, \cdots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$.*
- ***No*** *For every $S \subseteq V$, letting $\mu = |S|/|V|$, it holds that $\Phi_G(S) \geq 1 - (\Gamma_{1-\epsilon/2}(\mu) + \gamma)/\mu$, where the $\Gamma_{1-\epsilon/2}(\mu)$ is the Gaussian noise stability.*

We present two remarks about the Conjecture 13 from [2], which are relevant to our proof of Theorem 2.

▶ Remark 14. [2] The **Yes** instance of Conjecture 13 implies that the number of edges leaving each S_i is at most $4\epsilon|E|/q$, so the total number of edges not contained in one of the S_i is at most $2\epsilon|E|$.

▶ Remark 15. [2] The **No** instance of Conjecture 13 implies that for ϵ sufficiently small, there exists some constant c' such that $\Phi_G(S) \geq c'\sqrt{\epsilon}$, provided that $\mu \in [1/10, 1/2]$ and setting $\gamma \leq \sqrt{\epsilon}$. In particular, $|E(S, V \backslash S)| \geq \Omega(\sqrt{\epsilon}|E|)$, for any $|V|/10 \leq |S| \leq 9|V|/10$. [6]

---

[5] Note that the lower bound on μ can be taken arbitrarily close to 0. So the statement holds with $\mu \in [\epsilon', 1/2]$ for any constant $\epsilon' > 0$.

[6] Recall that Fact 12 is true for $\mu \in [\epsilon', 1/2]$ for any constant $\epsilon' > 0$. Therefore, Remark (15) can be strengthened and states, for any $\epsilon'|V| \leq |S| \leq (1 - \epsilon')|V|$, $|E(S, V \backslash S)| \geq \Omega(\sqrt{\epsilon}|E|)$. This will be a useful fact for proving hardness of approximation of $m(G, g)$ for finding many truthful nodes in Section 5 in full length paper [8].

Remark 14 follows from the definition of normalized edge expansion and the fact that sum of degree is two times number of edges. Remark 15 follows from Fact 12. The strong form of SSE Hypothesis 13, Remark 14 and Remark 15 will be particularly helpful for proving our SSE-hardness of approximation result (Theorem 2).

2.2 Preliminaries for Corruption Detection on Networks

We model networks as directed or undirected graphs, where each vertex in the network can be one of two types: truthful or corrupted. At times, we will informally call truthful vertices "good" and corrupt vertices "bad." We say that the corrupt party has *budget* b if it can afford to corrupt at most b nodes of the graph. Given a vertex set V, and a budget b, the corrupt entity will choose to control a subset of nodes $B \subseteq V$ under the constraint $|B| \leq b$. The rest of the graph remains as truthful vertices, i.e., $T = V \backslash B \subseteq V$. We assume that there are more truthful than corrupt nodes ($b < |V|/2$). It is easy to see that in the case where $|B| \geq |T|$, the corrupt nodes can prevent the identification of even one truthful node, by simulating truthful nodes (see e.g. [1]).

Each node audits and reports its (outgoing) neighbors' identities. That is, each vertex $u \in V$ will report the type of each $v \in \mathcal{N}(u)$, which is a vector in $\{0,1\}^{|\mathcal{N}(u)|}$. Truthful nodes always report the truth, i.e., it reports its neighbor $v \in T$ if v is truthful, $v \in B$ if v is corrupt. The corrupt nodes report their neighbors' identities adversarially. In summary, a strategy of the bad agents is composed of a strategy to take over at most b nodes on the graph, and reports on the nodes that neighbor them.

▶ **Definition 16** (Strategy for a corrupt party)**.** A strategy for the corrupt party is a function that maps a graph G and budget b to a subset of nodes B with size $|B| \leq b$, and a set of reports that each node $v \in B$ gives about its neighboring nodes, $\mathcal{N}(v)$.

▶ **Definition 17** (Computationally bounded corrupt party)**.** We say that the corrupt party is computationally bounded if its strategy can only be a polynomial-time computable function.

The task for the central agency is to find a good node on this corrupted network, based on the reports. It is clear that the more budget the corrupt party has, the harder the task of finding one truthful node becomes. It was observed in [1] that, for any graph, it is not possible to find one good node if $b \geq |V|/2$. If $b = 0$, it is clear that the entire set V is truthful. Therefore, given an arbitrary graph G, there exists a critical number $m(G)$, such that if the bad party has budget lower than $m(G)$, it is always possible to find a good node; if the bad party has budget greater than or equal to $m(G)$, it may not be possible to find a good node. In light of this, we define the critical number of bad nodes on a graph G. First, we formally define what we mean when we say it is impossible to find a truthful node on a graph G.

▶ **Definition 18** (Impossibility of finding one truthful node)**.** Given a graph $G = (V, E)$, the bad party's budget b and reports, we say that it is *impossible to identify one truthful node* if for any $v \in V$, there exists a configuration of the identities of the nodes where v is bad, at most b nodes are bad, and the configuration is consistent with the given reports.

▶ **Definition 19** (Critical number of bad nodes on a graph G, $m(G)$)**.** Given an arbitrary graph $G = (V, E)$, we define $m(G)$ as the minimum number b such that there is a way to distribute b corrupt nodes and set their corresponding reports such that it is impossible to find *one* truthful node on the graph G, given G, the reports and that the bad party's budget is at most b.

For example, for a star graph G with $|V| \geq 5$, the critical number of bad nodes is $m(G) = 2$. If there is at most 1 corrupt node on G, the central agency can always find a good node, thus $m(G) > 1$. If there are at most 2 bad nodes on G, then the bad party can control the center node and one of the leaves. Then for any node v in the graph, there exists a configuration where v is bad, only two nodes are assigned bad, and the configuration is consistent with observed reports. Therefore, it is impossible for central agency to find one good node, $m(G) = 2$.

Given a graph G, by definition there exists a set of nodes of size $m(G)$ that can make it impossible to find a good node if they are corrupted. However, this does not mean that the corrupt party can necessarily find this set in polynomial time. Indeed, Theorem 2 establishes that they cannot always find this set in polynomial time if we assume the SSE Hypothesis (Conjecture 13) and that P $\neq$ NP.

3 Proofs and Main Lemmas

In the following section, we state our main results by first presenting the close relation of our problem to the k-vertex separator problem. Then we use this characterization to prove Theorem 1. This characterization will additionally be useful for the proofs of Theorems 2 and 3, which we will sketch in Section 3.2, and provide in Section 3.3 in our full length paper [8].

3.1 2-Approximation by Vertex Separation

▶ **Lemma 20** (2-Approximation by Vertex Separation)**.** *The critical number of corrupt nodes for graph G, $m(G)$, can be bounded by the minimal sum of k-vertex separator and k, $\min_k(S_G(k)+k)$, up to a factor of 2. i.e.,*

$$\frac{1}{2}\min_k\left(S_G(k)+k\right) \leq m(G) \leq \min_k\left(S_G(k)+k\right)$$

Proof of Lemma 20. The direction $m(G) \leq \min_k S_G(k) + k$ follows simply. Let $k^* = \arg\min_k(S_G(k)+k)$. If the corrupt party is given $S_G(k^*)+k^*$ nodes to corrupt on the graph, it can first assign $S_G(k^*)$ nodes to the separator, thus the remaining nodes are partitioned into components of size at most k^*. Then it arbitrarily assigns one of the components to be all bad nodes. The bad nodes in the connected components report the nodes in the same component as good, and report any node in the separator as bad. The nodes in the separator can effectively report however they want (e.g. report all neighboring nodes as bad). It is impossible to identify even one single good node, because all connected components of size k can potentially be bad, and all vertices in the separator are bad.

The direction $1/2\min_k(S_G(k)+k) \leq m(G)$ can be proved as follows. When there are $b = m(G)$ corrupt nodes distributed optimally in G, it is impossible to find a single good node by definition, and therefore, in particular, the following algorithm (Algorithm 1) cannot always find a good node:

Algorithm 1 Finding one truthful vertex on undirected graph G.

Input: Undirected graph G

- If the reports on edge (u, v) does not equal to $(u \in T, v \in T)$, remove both u, v and any incident edges. Remove a pair of nodes in each round, until there are no bad reports left.
- Call the remaining graph H. Declare the largest component of H as good.

Run Algorithm 1 on G, and suppose the first step terminates in i rounds, then:

- No remaining node reports neighbors as corrupt
- $|V| - 2i$ nodes remain in graph
- $\leq b - i$ bad nodes remain in the graph, because each time we remove an edge with bad report, and one of the end points must be a corrupt vertex.

Note that if two nodes report each other as good, they must be the same type (either both truthful, or both corrupt.) Since graph H only contains good reports, nodes within a connected component of H have the same types. If there exists a component of size larger than $b - i$, it exceeds bad party's budget, and must be all good. Therefore, Algorithm 1 would successfully find a good node.

Since Algorithm 1 cannot find a good node, the bad party must have the budget to corrupt the largest component of H, which means it has size at most $b - i$. Hence, $S_G(b - i) \leq 2i$. Plugging in $b = m(G)$, we get that

$$m(G) = \frac{2i}{2} + b - i \geq \min_k (S_G(k)/2 + k) \geq \frac{1}{2} \min_k (S_G(k) + k),$$

where the first inequality comes from $2i \geq S_G(b - i)$. ◀

Furthermore, the upper bound in Lemma 20 additionally tells us that if corrupt party's budget $b \leq m(G)/2$, the set output by Algorithm 1 is guaranteed to be good.

▶ Theorem 1 (restated). Fix a graph G and suppose that the corrupt party has a budget $b \leq m(G)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without knowledge of either $m(G)$ or b. Furthermore, the central agency's algorithm runs in linear time (in the number of edges in the graph G).

Proof of Theorem 1. Suppose the corrupt party has budget $b \leq m(G)/2$. Run Algorithm 1. We remove $2i$ nodes in the first step, and separate the remaining graph H into connected components. Notice each time we remove an edge with bad report, at least one of the end point is a corrupt vertex. So we have removed at most $2b \leq m(G) \leq \lceil |V|/2 \rceil$ nodes. Therefore, the graph H is nonempty, and the nodes in any connected component of H have the same identity. Let $k^* \geq 1$ be the size of the maximum connected component of H. We can conclude that $S_G(k^*) \leq 2i$, since $2i$ is a possible size of k^*-vertex separator of G.

Notice there are at most $b - i \leq m(G)/2 - i$ bad nodes in H by the same fact that at least one bad node is removed each round. By the upper bound in Lemma 20,

$$b - i \leq m(G)/2 - i \leq \min_k (S_G(k) + k)/2 - i \leq (2i + k^*)/2 - i \leq \frac{k^*}{2}.$$

Since $k^* \geq 1$, the connected component of size k^* exceeds the bad party's remaining budget $k^*/2$, and must be all good.

Algorithm 1 is linear time because it loops over all edges and removes any "bad" edge that does not have reports (T, T) (takes $\leq |E|$ time when we use a list with "bad" edges at the front), and counts the size of the remaining components ($\leq |V|$ time), and thus is linear in $|E|$. ◀

▶ Remark 21. Both bounds in Lemma 20 are tight. For the lower bound, consider a complete graph with an even number of nodes. For the upper bound, consider a complete bipartite graph with one side smaller than the other.

A tight lower bound and a tight upper bound example can be found in full-length version of our paper [8].

We end by discussing that the efficient algorithm given in this section does not address the regime when the budget of the bad party, b, falls in $m(G)/2 < b \leq m(G)$. Though by definition of $m(G)$, the central agency can find at least one truthful node as long as $b \leq m(G)$, by, for example, enumerating all possible assignments of good/bad nodes consistent with the report, and check the intersection of the assignment of good nodes. However, it is not clear that the central agency has a polynomial time algorithm for doing this. Of course, one can always run Algorithm 1, check whether the output set exceeds $b - i/2$, and concludes that the output set is truthful if that is the case. However, there is no guarantee that the output set will be larger than $b - i/2$ if $m(G)/2 < b \leq m(G)$. We propose the following conjecture:

▶ **Conjecture 22.** *Fix a graph G and suppose that the corrupt party has a budget b such that $m(G)/2 < b \leq m(G)$. The problem of finding one truthful node given the graph G, bad party's budget b and the reports is NP-hard.*

3.2 SSE-Hardness of Approximation for $m(G)$

In this section, we present the hardness of approximation result for $m(G)$ within any constant factor under the Small Set Expansion (SSE) Hypothesis [19]. Specifically, we give essential steps for proving Theorem 2.

▶ Theorem 2 (restated). For every $\beta > 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V, E)$, it is SSE-hard to distinguish between the case where $m(G) \leq \epsilon \cdot |V|$ and $m(G) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes for a graph to within any constant factor is SSE-hard.

In order to prove Theorem 2, we construct a reduction similar to [2], and show that the bad party can control auxiliary graph of the **Yes** case of SSE with $b = O(\epsilon|V'|)$ and cannot control the auxiliary graph of the **No** case of SSE with $b = \Omega(\epsilon^{0.51}|V'|)$. In the following, we present the arguments for the **No** case.

Given an undirected d-regular graph $G = (V, E)$, construct an auxiliary undirected graph $G' = (V', E')$ in the following way [2]. Let $r = d/2$. For each vertex $v^i \in V$, make r copies of v^i and add to the vertex set of G', denoted $v^i_1, \cdots, v^i_r$. Denote the resulting set of vertices as $\tilde{V} = V \times \{1, \cdots, r\}$. Each edge $e^k \in E$ of G becomes a vertex in G', denoted e^k. Denote this set of vertices as $\tilde{E}$. In other words, $V' = \tilde{V} \cup \tilde{E} = V \times \{1, \cdots, r\} \cup E$. There exists an edge between a vertex v^i_j and a vertex e^k of G' if v^i and e^k were adjacent edge and vertex pair in G. Note that G' is a bipartite d-regular graph with $d/2|V| + |E| = 2|E|$ vertices.

▶ **Lemma 23.** *Suppose $q = 1/\epsilon$, and G can be partitioned into q equi-sized sets $S_1, \cdots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$. Then the bad party can control the auxiliary graph G' with at most $3\epsilon|E| = 1.5\epsilon|V'|$ nodes.*

Proof of Lemma 23. Notice by Remark (2.1.1), the total number of edges in G not contained in one of the S_i is at most $2\epsilon|E|$.

This implies that a strategy for the bad party to control graph G' is as follows. Control vertex $e^k \in \tilde{E}$ if $e^k \in E$ is not contained in any of the S_is in G. Call the set of such vertices $E^* \subseteq \tilde{E}$. Let $S^*_i \subseteq V'$ be the set that contains all r copies of nodes in $S_i \subseteq V$. Control one of the $S^*_i s$, say S^*_1. Corrupt nodes in S^*_1 report their neighbors in S^*_1 as good, and report E^* as bad. Nodes in E^* can effectively report however they want; suppose they report every neighboring node as bad. Then, it is impossible to identify even one truthful node, since assigning any S^*_i as corrupt is consistent with the report and within bad party's budget.

If $q = 1/\epsilon$, this strategy amounts to controlling $2\epsilon|E| + d/2 \cdot |V|/q = 3\epsilon|E|$ nodes on G'. Notice, this number is guaranteed to be smaller than $1/2|V'|$, as long as $q > 4$, because the bad party controls less than $2/q$ of all the "edge vertices" $\tilde{E}$, and controls less than $1/q$ of all the "vertex vertices" $\tilde{V}$. ◀

Note that, different from the argument in [2], we cannot take r to be arbitrarily large (e.g. $> O(|V||E|)$). This is because when r is large, $2\epsilon|E| + r \cdot |V|/q = O(\epsilon(|E| + |V'|)) = O(\epsilon|V'|)$, and will not be comparable with the $O(\sqrt{\epsilon}|E|)$ in Lemma 24.

▶ **Lemma 24.** *Let $G = (V, E)$ be an undirected d-regular graph with the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$. If bad party controls $O(\epsilon^{0.51}|E|) = O(\epsilon^{0.51}|V'|) < 1/2|V'|$ nodes on the auxiliary graph G' constructed from G, we can always find a truthful node on G'.*

Combining Lemma 23 and Lemma 24, Theorem 2 follows in standard fashion. The proofs for Lemma 24 and Theorem 2 are presented in our full-length paper [8].

We also obtain the following Corollary 25 from Theorem 2.

▶ **Corollary 25.** *Assume the SSE Hypothesis and that $P \neq NP$. Fix any $\beta > 1$. There does not exist a polynomial-time algorithm that takes as input an arbitrary graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\beta \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

In summary, the analysis in this section tells us that given an arbitrary graph, it is hard for bad party to corrupt the graph with minimal resources. Moreover, it is still hard for the bad party to corrupt the graph even if they are given a budget of $\beta m(G)$, for any $\beta \geq 1$. On the other hand, if the budget of the bad party is a factor of two less than $m(G)$, a good node can always be detected with an efficient algorithm, e.g. using Algorithm 1. This contrast highlights that the corruption detection problem of finding one good node (and, as later proven, finding any arbitrary fraction of good nodes) is easier for the good party and harder for the bad party.

Finally, the existence of efficient algorithms for good party to detect one good node on directed graphs (i.e. Theorem 4), and detect any fraction of the good nodes (i.e. Theorem 5) follows from analogous notions of vertex-separators to Definition 7. The hardness of approximation results for finding any arbitrary fraction of good nodes, i.e. 6 can be proven following similar constructions and arguments as in the proof sketch of Theorem 2. Then, we give a simple a gadget reduction that extends the result to $\delta \in [1/2, 1)$. The full details of this proof can be found in the full version of our paper, together with the proofs of Theorem 3, 4, 5, 6.

References

1. Noga Alon, Elchanan Mossel, and Robin Pemantle. Corruption Detection on Networks. *CoRR*, abs/1505.05637, 2015. `arXiv:1505.05637`.
2. Per Austrin, Toniann Pitassi, and Yu Wu. Inapproximability of Treewidth, One-Shot Pebbling, and Related Layout Problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 13–24, 2012. `doi:10.1007/978-3-642-32512-0_2`.
3. Walid Ben-Ameur, Mohamed-Ahmed Mohamed-Sidi, and José Neto. The k -separator problem: polyhedra, complexity and approximation results. *J. Comb. Optim.*, 29(1):276–307, 2015. `doi:10.1007/s10878-014-9753-x`.

4 Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree. *J. Algorithms*, 18(2):238–255, 1995.

5 Anton T. Dahbura and Gerald M. Masson. An $O(n^{2.5})$ Fault Identification Algorithm for Diagnosable Systems. *IEEE Trans. Computers*, 33(6):486–492, 1984. doi:10.1109/TC.1984.1676472.

6 Odd-Helge Fjeldstad. Fighting fiscal corruption: lessons from the Tanzania Revenue Authority. *Public Administration and Development: The International Journal of Management Research and Practice*, 23(2):165–175, 2003.

7 S. Louis Hakimi and A. T. Amin. Characterization of Connection Assignment of Diagnosable Systems. *IEEE Trans. Computers*, 23(1):86–88, 1974. doi:10.1109/T-C.1974.223782.

8 Yan Jin, Elchanan Mossel, and Govind Ramnarayan. Being Corrupt Requires Being Clever, But Detecting Corruption Doesn't. *arXiv*, 2018. arXiv:1809.10325.

9 Tiko Kameda, S Toida, and FJ Allan. A diagnosing algorithm for networks. *Information and Control*, 29(2):141–148, 1975.

10 David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the Spread of Influence through a Social Network. *Theory of Computing*, 11:105–147, 2015. doi:10.4086/toc.2015.v011a004.

11 Subhash Khot. On the Power of Unique 2-Prover 1-Round Games. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, page 25, 2002. doi:10.1109/CCC.2002.1004334.

12 Jon G Kuhl and Sudhakar M Reddy. Distributed fault-tolerance for large multiprocessor systems. In *Proceedings of the 7th annual symposium on Computer Architecture*, pages 23–30. ACM, 1980.

13 Euiwoong Lee. Partitioning a Graph into Small Pieces with Applications to Path Transversal. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1546–1558, 2017. doi:10.1137/1.9781611974782.101.

14 Shachindra N. Maheshwari and S. Louis Hakimi. On Models for Diagnosable Systems and Probabilistic Fault Diagnosis. *IEEE Trans. Computers*, 25(3):228–236, 1976.

15 Thebeth Rufaro Mukwembi and Simon Mukwembi. Corruption and its detection: a graph-theoretic approach. *Computational and Mathematical Organization Theory*, 23(2):293–300, June 2017. doi:10.1007/s10588-016-9227-z.

16 Richard P Nielsen. Corruption networks and implications for ethical corruption reform. *Journal of Business ethics*, 42(2):125–149, 2003.

17 Maarten Oosten, Jeroen HGC Rutten, and Frits CR Spieksma. Disconnecting graphs by removing vertices: a polyhedral approach. *Statistica Neerlandica*, 61(1):35–60, 2007.

18 Franco P. Preparata, Gernot Metze, and Robert T. Chien. On the Connection Assignment Problem of Diagnosable Systems. *IEEE Trans. Electronic Computers*, 16(6):848–854, 1967. doi:10.1109/PGEC.1967.264748.

19 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 755–764, 2010. doi:10.1145/1806689.1806788.

20 Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between Expansion Problems. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 64–73, 2012. doi:10.1109/CCC.2012.43.

21 Gregory F. Sullivan. A Polynomial Time Algorithm for Fault Diagnosability. In *FOCS*, pages 148–156. IEEE Computer Society, 1984.

22 Jie Xu and Shi-ze Huang. Sequentially t-Diagnosable Systems: A Characterization and Its Applications. *IEEE Trans. Computers*, 44(2):340–345, 1995. doi:10.1109/12.364544.