Controlling Infection by Blocking Nodes and Links Simultaneously

Jing He[*] Hongyu Liang[†] Hao Yuan[‡]

Abstract

In this paper we study the problem of controlling the spread of undesirable things (viruses, epidemics, rumors, etc.) in a network. We present a model called the *mixed generalized network security model*, denoted by MGNS(d), which unifies and generalizes several well-studied infection control model in the literature. Intuitively speaking, our goal under this model is to secure a subset of nodes and links in a network so as to minimize the expected total loss caused by a possible infection (with a spreading limit of d-hops) plus the cost spent on the preventive actions. Our model has wide applications since it incorporates both node-deletion and edge-removal operations. Our main results are as follows:

1. For all $1 \leq d < \infty$, we present a polynomial time $(d+1)$-approximation algorithm for computing the optimal solution of MGNS(d). This improves the approximation factor of $2d$ obtained in [20] for a special case of our model. We derive an $O(\log n)$-approximation for the case $d = \infty$. Moreover, we give a polynomial time $\frac{3}{2}$-approximation for MGNS(1) on bipartite graphs.
2. We prove that for all $d \in \mathbb{N} \cup \{\infty\}$, it is $\mathcal{APX}$-hard to compute the optimum cost of MGNS(d) even on 3-regular graphs. We also show that, assuming the Unique Games Conjecture [14], we cannot obtain a $(\frac{3}{2} - \epsilon)$-approximation for MGNS(d) in polynomial time. Our hardness results hold for the special case GNS(d) in [20] as well.
3. We show that an optimal solution of MGNS(d) can be found in polynomial time for every fixed $d \in \mathbb{N} \cup \{\infty\}$ if the underlying graph is a tree, and the infection cost and attack probability are both uniform. Our algorithm also works for the case where there are budget constraints on the number of secured nodes and edges in a solution. This in particular settles an open question from [22] that asks whether there exists an efficient algorithm for the minimum average contamination problem on trees.

[*]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. Email: `lianghy08@mails.tsinghua.edu.cn`.

[†]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. Email: `he-j08@mails.tsinghua.edu.cn`.

[‡]Department of Computer Science, City University of Hong Kong, Hong Kong, China. Email: `haoyuan@cityu.edu.hk`.

1 Introduction

During the recent years, much effort has been devoted to the study on the structure of various types of networks such as social networks, wireless sensor networks, computer networks, transportation networks, and the World Wide Web. An important and active subject is to study the *information diffusion process* in the situations where we want some news, topics, thoughts or products to spread quickly in the network, such as viral marketing [9]. This idea is formalized by Kempe, Kleinberg and Tardos [13] as a combinatorial problem called the *influence maximization problem*, which has since then been extensively studied under various settings (see, e.g., [6, 11, 16, 21]).

In contrast, another important line of research is to study how to prevent or limit the spread of undesirable things through the network, such as the propagation of computer viruses and worms over computer networks, the fast spreading of malicious rumors through social networks, and the spread of infections or epidemics (such as Swine Flu and H1N1) among groups of people. In all these circumstances we need to eliminate or at least control the evolution of the bad things over the whole network, which is usually achieved by taking some preventive measures before the emergence of these undesirable things, and isolating or restricting the behaviors of some individuals if the infection has already been spread through the network. An important issue in real-world applications is the balance between the cost spent on prevention and the expected loss caused by infection. For example, installing anti-virus softwares on the computers is a natural response to the possible virus attack, but it may cost a lot of money and bring inefficiency to the protected computers due to high maintenance cost or memory requirement.

An elegant model that integrates both the security and infection costs has been formalized by Aspnes, Chang and Yampolskiy [3]. In their model, we seek for a subset of nodes on which we shall install the anti-virus softwares (call such nodes *secure*). A virus-attack is initiated by choosing one node from the network uniformly at random, and this node, if not secure, will infect all other nodes that are reachable from it in the network with all secure nodes removed. The goal is to minimize the cost for installing softwares (*security cost*) plus the expected total loss caused by the virus (*infection cost*). They consider both centralized (optimization) and game-theoretic settings. The model is substantially generalized by Kumar et al. [20] by allowing individual security and infection costs and arbitrary distribution of the virus-attack probability, and by introducing a parameter d into the model that represents the distance within the network that an infection can spread. This new model is called the *generalized network security model*, denoted GNS(d). Thus, GNS(d) is able to capture networks with less infection power or limited local information, such as ad hoc wireless networks. An issue with GNS(d) is that it lacks the power of modeling the action of restricting the interconnections between individuals in the network (instead of simply removing them from the network), which, in the graph language, corresponds to blocking edges in the graph instead of deleting nodes. In spirit of such consideration, the contamination minimization model where edges are supposed to be blocked is raised by [17] and has been further studied in, e.g., [18, 19, 22].

In this paper, we present a model for minimizing the spread of infection that unifies and further generalizes the two aforementioned approaches, which we call the *mixed generalized network security model*, denoted by MGNS(d). In our model, each node has its own *security cost* and *infection cost* as in GNS(d), and each edge has its own *link-blocking cost* that represents the lost caused by the removal of the edge. The attack probability distribution can be arbitrary as in GNS(d). The insecure node that is attacked initially will infect exactly those nodes that are within distance at most d from it in the *attack graph* obtained by removing all secure nodes and blocked edges from the original network. The cost of a solution is equal to the total expected infection cost of the nodes plus the cost for securing nodes and blocking edges in this solution. The goal is then to find a solution with minimum cost. Our main results in this paper, some of which improve on the previously best known results achieved for special cases of our model, are given in the following.

1. For all $1 \leq d < \infty$, we present a polynomial time $(d+1)$-approximation algorithm for computing the optimal solution of MGNS(d) based on the primal-dual method. This improves the approximation factor of $2d$ obtained in [20] for GNS(d), which is a special case of MGNS(d). (We note that it is possible to design a reduction from MGNS(d) to GNS($2d$), which will give us a $4d$-approximation for MGNS(d) using the algorithm in [20]. However, the reduction loses a lot of information about the topology of the underlying network.) For the case $d = \infty$, we derive an $O(\log n)$-approximation for MGNS(∞) that matches the result of [20] for GNS(∞). Moreover, we give a polynomial time $\frac{3}{2}$-approximation for MGNS(1) on bipartite graphs.

2. We prove that for all $d \in \mathbb{N} \cup \{\infty\}$, it is $\mathcal{APX}$-hard to compute the optimum cost of GNS(d) even if the graph is 3-regular and all costs and probability are uniform, thus ruling out the possibility of designing PTAS for the problem. We also show that, assuming the Unique Games Conjecture [14], we cannot obtain a $(\frac{3}{2} - \epsilon)$-approximation for GNS(d) in polynomial time. To our knowledge these are the first inapproximability results for GNS(d). Since GNS(d) is a special case of MGNS(d), all the hardness results trivially apply to MGNS(d).

3. We show that an optimal solution of MGNS(d) can be found in polynomial time for every fixed $d \geq 1$ or $d = \infty$ if the underlying graph is a tree, and the infection cost and attack probability are both uniform. Our algorithm can handle all $d \leq O(\sqrt{\log n})$ in polynomial time on bounded-degree trees. Our algorithm also works for the case where budget constraints are put on the number of nodes and edges that can be secured and blocked respectively in a wanted solution. In particular, this settles an open question of [22] that asks whether there exists an efficient algorithm for the *minimum average contamination problem* on trees (which will be mentioned later in more detail). We remark that the tree structure, despite being special, has applications in hierarchically-organized networks such as company relationships.

Paper Organization. In the rest part of this section, we rigorously define our model and compare it with some previous work. In Section 2 we present approximation algorithms for MGNS(d). Hardness of approximation results for MGNS(d) are given in Section 3. Section 4 copes with tree instances of MGNS(d). Finally, in Section 5 we conclude the whole paper and propose some open problems and future research directions.

1.1 Our Model for Infection Control

In this subsection we explain the mixed generalized network security model MGNS(d) in more detail, where $d \in \mathbb{N}^+ \cup \{\infty\}$ is a parameter that, intuitively, reflects the "degree of infectivity" within the network. Although we will describe our model in terms of preventing virus-spreading in computer networks, one should keep in mind that the model is capable of many other situations where we wish to minimize the propagation of undesirable things. Specifically, our model MGNS(d) comprises the following ingredients:

Contact Graph, Costs and Strategy. The contact graph is an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ denotes the set of computers in a connected network, and $E \subseteq V^2$ specifies the underlying topology of the network. Thus, an edge $\{u, v\} \in E$ indicates that nodes (computers) u and v are directly connected, so that u can potentially affect v if it is infected by a computer virus or worm, and vice versa. For each $v \in V$, let C_v denote the *security cost* of v (for installing an anti-virus software on v), and L_v the *infection cost* of v (for recovering it from a virus attack). For each $e \in E$, let C'_e denote the *link-blocking cost* of e (for the lost caused by the removal of e). All the costs are non-negative. In a *strategy (solution)*, we need to decide on which nodes to install anti-virus softwares and which edges to block. A node with anti-virus software installed on it is called *secure*, and otherwise is called *insecure*. Similarly we have *blocked* and *unblocked* edges. A solution S is also identified with $V_S \cup E_S$, where $V_S \subseteq V$ is the set of secure nodes in S and $E_S \in E$ is the set of blocked edges in S. The *attack graph* of a solution is the graph obtained from G by removing all secure nodes and blocked edges.

Infection Model and Social Cost. We assume that the virus is initiated at *exactly one* node chosen from V according to the *attack probability distribution* $\{w_v \mid v \in V\}$, where $\sum_{v \in V} w_v = 1$. Write $w(S) := \sum_{v \in S} w_v$ for $S \subseteq V$. A secure node will neither suffer from the virus nor transmit the virus to other nodes (although it can be chosen as the attacked node), whereas an insecure node, if chosen as the attacked node, will infect exactly those nodes at distance at most d from it in the attack graph (including itself). For a strategy S, let $V_S^{\leq d}(v)$ denote the set of nodes at distance at most d from v in the attack graph of S. Then the *social cost* of S (denoted by $cost(S)$) is defined as:

$$cost(S) = \underbrace{\sum_{v \in V_S} C_v}_{\substack{\text{cost for} \\ \text{installing softwares}}} + \underbrace{\sum_{e \in E_S} C'_e}_{\substack{\text{cost for} \\ \text{blocking links}}} + \sum_{v \in V \setminus V_S} \underbrace{L_v \cdot w(V_S^{\leq d}(v))}_{\substack{\text{expected cost for} \\ \text{recovering } v \text{ from infection}}} .$$

Goal. In the *centralized* setting of MGNS(d), the goal is to find a strategy with minimum social cost, or *social optimum*. We can also define the *decentralized* (game-theoretic) model, in which the user needs to decide whether to install the anti-virus software on his/her computer and whether to disconnect some of the links with other users in the network. In this paper we concentrate on the centralized setting of MGNS(d), while leaving explorations of the decentralized model to future work.

1.2 Related Work

As stated before, our model MGNS(d) incorporates and generalizes several infection prevention models that have been studied recently. We list some problems considered in the literature that are either special cases of or related to the problem of computing the social optimum of MGNS(d).

- Consider the instances of MGNS(d) where $d = \infty$, $C'_e = \infty$ for all $e \in E$, all nodes have the same security cost C and infection cost L, and the attack probability distribution is uniform over nodes. When restricted on such instances, MGNS(d) coincides with the model proposed by Aspnes, Chang and Yampolskiy [3], who gave an $O(\log^{1.5} n)$-approximation for computing the social optimum, based on the sparsest cut algorithm of Arora, Rao and Vazirani [1]. The approximation ratio is subsequently improved to $O(\log n)$ independently by [5] and [20], which is also the currently best known result for this problem.

- Restricted on the instances where $C'_e = \infty$ for all $e \in E$ (i.e., all the edges should remain unblocked in any reasonable solution), our model is equivalent to the *generalized network security* model GNS(d) introduced by Kumar et al. [20]. They present a $2d$-approximation for computing the social optimum of GNS(d) for all $d < \infty$ by rounding a natural linear program for the problem. This result is subsumed by our $(d+1)$-approximation for MGNS(d). They also give an $O(\log n)$-approximation for GNS(∞) based on a reduction to the *minimum weighted vertex multicut problem* [10], improving the $O(\log^{1.5} n)$ factor of [3] and matching the result independently obtained in [5].

- Under the case where $d = \infty$, $C_v = \infty$ for all $v \in V$, $w_v = 1/n$ for all $v \in V$, and both the infection costs and link-blocking costs are uniform, the problem of computing the social optimum of MGNS(d) is similar to the *minimum average contamination problem* studied by Li and Tang [22], which originates from a (stochastic) link-blocking model initiated by Kimura, Saito and Motoda [17]. The difference between our setting and theirs is that they put a budget constraint K on the number of edges that can be removed from the network. In [22], a $(1+\epsilon, O(\frac{\log n}{\epsilon}))$-bicriteria approximation algorithm and a $(\frac{5}{3} - \epsilon)$-inapproximability result are given for the minimum average contamination problem. Note that their problem is harder than ours (with an additional budge constraint) and thus their hardness factor is stronger than ours. However, they only consider the case $d = \infty$, while our hardness result applies to all d. Also, our polynomial-time algorithm for tree instances of MGNS(d) holds for the budgeted case as well.

- Another related problem that has mainly been studied in the operations research forum is the *critical node problem* [2, 4, 8] defined as follows: given a node-weighted graph $G = (V, E)$, a connection cost $c(u, v)$ for each pair of nodes $\{u, v\} \in V^2$, and a parameter K, the goal is to find a subset of nodes whose total weight does not exceed K such that the total connection cost (counted for all connected pairs of nodes) is minimized. This problem is similar to MGNS(∞) with $C'_e = \infty$ for all $e \in E$ and $w_v = 1/n$ for all $v \in V$, but with additional budget constraints and more general cost functions. The problem is NP-complete on general graphs with unit costs and unit weights [2], and on trees with unit weights [8]. For the unit-cost case (which makes the problem fit in our model with $d = \infty$) in a tree of size n, Di Summa et al. [8] show that the problem is solvable in $O(n^7)$ time. Our polynomial-time algorithm for (budgeted) MGNS(d) on trees substantially generalizes their result to all fixed d.

2 Approximation Algorithm for MGNS(d)

In this section we concern with the computation of the social optimum of MGNS(d). As the problem is NP-hard, we focus on the perspective of approximation, and obtain the following results.

Theorem 1. *For any $d \geq 1$, there is a polynomial time $(d+1)$-approximation algorithm for computing the social optimum of MGNS(d). (Here d need not be a constant.)*

Theorem 2. *There is a polynomial time $O(\log n)$-approximation for the social optimum of MGNS(∞).*

Theorem 3. *There is a polynomial time $\frac{3}{2}$-approximation algorithm for computing the social optimum of MGNS(1) with bipartite contact graphs.*

2.1 The case $d < \infty$

We first consider the case $1 \le d < \infty$, and prove Theorem 1. Let $\mathcal{I}$ be an instance of MGNS(d) with contact graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$. If $C_i < w_i L_i$ for some $i \in V$, then clearly i should be secured in any optimum solution. Thus, we assume in what follows that $C_i \ge w_i L_i$ for all $i \in V$.

We write an integer program to formulate the social optimum of $\mathcal{I}$. For each $k \in V \cup E$, let x_k be a binary variable that is 1 if and only if k is secure (or blocked, depending on whether k is a node or an edge). For a path p, let V_p and E_p denote the sets of nodes and edges on p, respectively. For all $1 \le i < j \le n$, let $P^d_{i,j}$ denote the collection of all simple paths from i to j of length at most d (note that $P^d_{i,j}$ can be empty and can also be of exponential size), and $y_{i,j}$ be a binary variable that is 1 if and only if there exists at least one path $p \in P^d_{i,j}$ on which all nodes are insecure and all edges are unblocked. Thus, $y_{i,j} = 1$ iff i and j can infect each other in the attack graph. Then the following integer program IP1 characterizes precisely the social optimum of $\mathcal{I}$:

$$\begin{aligned}
\text{IP1: Min} \sum_{i \in V} C_i x_i + \sum_{\{i,j\} \in E} C'_{\{i,j\}} x_{\{i,j\}} \quad &+ \quad \sum_{i \in V} L_i \left(w_i(1 - x_i) + \sum_{j \in V \setminus \{i\}} w_j y_{i,j} \right) \\
\text{subject to: } y_{i,j} + \sum_{k \in V_p \cup E_p} x_k \quad &\ge \quad 1 \qquad \forall 1 \le i < j \le n \text{ and } p \in P^d_{i,j} \\
y_{i,j} \quad &= \quad y_{j,i} \qquad \forall 1 \le i < j \le n \\
x_k \quad &\in \quad \{0, 1\} \qquad \forall k \in V \cup E \\
y_{i,j} \quad &\in \quad \{0, 1\} \qquad \forall 1 \le i, j \le n, i \ne j \, .
\end{aligned}$$

We write $C'_i = C_i - w_i L_i$ for each $i \in V$ (with a little abuse of notation since C' is originally defined for edge costs), $L_{i,j} = w_i L_j + w_j L_i$ for all $1 \le i < j \le n$, and $C = \sum_{1 \le i \le n} w_i L_i$. Note that $C'_i \ge 0$ for all $i \in V$ by our assumption before. Rearranging terms, unifying the first two summations, and combining the occurrences of $y_{i,j}$ and $y_{j,i}$ in the objective function of IP1, we get a simpler yet equivalent formulation IP2 as follows:

$$\begin{aligned}
\text{IP2:} \quad \text{Min} \sum_{k \in V \cup E} C'_k x_k \quad + \quad \sum_{1 \le i < j \le n} L_{i,j} y_{i,j} + C \quad \text{subject to:} \\
y_{i,j} + \sum_{k \in V_p \cup E_p} x_k \quad &\ge \quad 1 \qquad \forall 1 \le i < j \le n \text{ and } p \in P^d_{i,j} \\
x_k \quad &\in \quad \{0, 1\} \qquad \forall k \in V \cup E \\
y_{i,j} \quad &\in \quad \{0, 1\} \qquad \forall 1 \le i < j \le n.
\end{aligned}$$

Observe that IP2, with the constant part C discarded, can be regarded as an instance of the weighted set cover problem when treating the length-at-most-d paths as the elements to be covered. When d is fixed, the instance of set cover is constructible in polynomial time. Also, in this set cover instance, every element appears in at most $2d + 2$ sets, because each constraint in IP2 involves at most $2d + 2$ variables (note that each $p \in P^d_{i,j}$ consists of at most $d + 1$ vertices and d edges). Therefore, a polynomial time $(2d+2)$-approximation exists for IP2 (see, e.g., [12]) and thus also for MGNS(d). Notice that, by reducing the problem to set cover, we can only handle constant d, and cannot hope for a poly-time $(2d + 2 - \epsilon)$-approximation due to the $(k - \epsilon)$-hardness of k-uniform hypergraph vertex cover [15], assuming the Unique Games Conjecture [14].

We next show that we can obtain an approximation factor of $d+1$ for all d (not necessarily fixed) by utilizing the special structure of IP2, thus saving a factor of 2 from the set cover approach. To achieve this, we relax the last two constraints of IP2 to $x_k \geq 0$ and $y_{i,j} \geq 0$ respectively, and ignore the constant part C in the objective function. This gives us a linear programming relaxation (which might still have super-polynomial size) of the original instance, which we call LP. (We do not state LP explicitly since it is very similar to IP2.) Obviously, $OPT(LP)+C \leq OPT(IP2) = OPT(IP1)$, where $OPT(P)$ is the optimum objective value of the mathematical program P.

We now write the dual formulation of LP. Let $P^d = \cup_{1 \leq i < j \leq n} P^d_{i,j}$. For each $p \in P^d$, introduce a dual variable z_p, which corresponds to the constraint $y_{i,j} + \sum_{k \in V_p \cup E_p} x_k \geq 1$ in LP (where i and j are the endpoints of p). The dual program DU can be written as follows:

$$\begin{array}{rcll}
\text{DU:}\quad \text{Max} \displaystyle\sum_{p \in P^d} z_p & & & \text{subject to:} \\
\displaystyle\sum_{p \in P^d_{i,j}} z_p & \leq & L_{i,j} & \forall 1 \leq i < j \leq n \\
\displaystyle\sum_{\substack{p \in P^d \\ k \in V_p \cup E_p}} z_p & \leq & C'_k & \forall k \in V \cup E \\
z_p & \geq & 0 & \forall p \in P^d.
\end{array}$$

By the strong duality theorem, $OPT(DU) = OPT(LP)$. We now find a solution to IP2 by Algorithm 1, which basically consists of a primal-dual procedure and a “pruning” phase. Since the number of variables in DU can be super-polynomial in n for non-constant d, the naïve implementation of Algorithm 1 may not run in polynomial time. Nevertheless, we will show later that the running time can be reduced to $n^{O(1)}$ regardless of d; stating the algorithm in its current form is just to simplify the analysis of its performance guarantee. Let S denote the solution to IP2 returned by Algorithm 1, and $Z = \{z_p \mid p \in P^d\}$ be the solution to DU obtained in Algorithm 1 (which is not explicitly returned). Let $value(S)$ denote the objective value of the solution S.

Lemma 1. *Z is a feasible solution to DU, and S is a feasible solution to IP2.*

Proof. Since all variables in a constraint of DU are frozen (and hence whose values cannot change thereafter) when the constraint goes tight, Z is clearly a feasible solution to DU. Now consider the solution S to IP2. Fix any $p \in P^d$. Since z_p is frozen when the algorithm terminates, at least one constraint in DU involving z_p is tight, and thus $y_{i,j}$ (where i and j are endpoints of p) or at least one x_k with $k \in V_p \cup E_p$ is set to 1 at Line 6 or 8 inside the WHILE loop. During the FOR loop, $y_{i,j}$ is changed back to 0 only if x_i or x_j is 1, and for each $\{i_1, i_2\} \in E_p$, $x_{\{i_1,i_2\}}$ is changed back to 0 only if x_{i_1} or x_{i_2} is 1. Thus, at least one of the variables in the constraint “$y_{i,j} + \sum_{k \in V_p \cup E_p} x_k \geq 1$” remains 1, which fulfills the constraint. By the arbitrariness of p, we know that S is a feasible solution to IP2. □

Lemma 2. *$value(S) \leq (d+1)OPT(IP2)$.*

Algorithm 1 Constructing a feasible solution for IP2

```
1: $x_k \leftarrow 0,\ \forall k \in V \cup E; y_{i,j} \leftarrow 0,\ \forall 1 \le i < j \le n.$
2: $z_p \leftarrow 0,\ \forall p \in P^d$; also, set all $z_p$ to be "unfrozen."
3: while there are still unfrozen variables do
4:   Choose any unfrozen variable, say $z_p$, that appears in some constraint of DU. Raise the value
     of $z_p$ until some constraint in DU, say $c$, becomes tight. (Pick an arbitrary one if there are
     more than one tight constraints.)
5:   if $c$ is "$\sum_{p \in P^d : k \in V_p \cup E_p} z_p \le C'_k$" for some $k \in V \cup E$ then
6:     $x_k \leftarrow 1$
7:   else if $c$ is "$\sum_{p \in P^d_{i,j}} z_p \le L_{i,j}$" for some $1 \le i < j \le n$ then
8:     $y_{i,j} \leftarrow 1$
9:   end if
10:  Freeze all variables that occur in some (newly appeared) tight constraint.
11: end while
12: for all $1 \le i < j \le n$ do
13:   if $x_i = 1$ or $x_j = 1$ then
14:     $y_{i,j} \leftarrow 0$; $x_{\{i,j\}} \leftarrow 0$ if $\{i,j\} \in E$.
15:   end if
16: end for
17: return $\{x_k \mid k \in V \cup E\} \cup \{y_{i,j} \mid 1 \le i < j \le n\}$.
```

Proof. For each variable v of IP2, let $c(v)$ denote the constraint in DU that corresponds to v. Call a constraint $c(v)$ *active* if $v = 1$ in the solution S. By Line 4 of Algorithm 1, every active constraint $c(v)$ (say) is tight, and hence the contribution of this v to $value(S)$ (which is the coefficient of v in the objective function of IP2) equals to the sum of z_p's contained in $c(v)$. Therefore, $value(S) = \sum_{p \in P^d} t_p z_p$, where t_p is the number of active constraints containing z_p.

Now fix an arbitrary $p = (i_0, i_1, \ldots, i_t) \in P^d, t \le d$. The set of constraints in which z_p appears is $\{c(y_{i_0,i_t})\} \cup \{c(x_{i_j}) \mid 0 \le j \le t\} \cup \{c(x_{\{i_j,i_{j+1}\}}) \mid 0 \le j \le t-1\}$, which can be partitioned into the following $t+1$ subsets:

$$\{c(x_{i_0}), c(x_{\{i_0,i_1\}})\}, \{c(x_{i_1}), c(x_{\{i_1,i_2\}})\}, \ldots, \{c(x_{i_{t-1}}), c(x_{\{i_{t-1},i_t\}})\}, \{c(x_{i_t}), c(y_{i_0,i_t})\}.$$

Due to the function of the FOR loop, at most one constraint from each subset is active. Thus z_p appears in at most $t+1 \le d+1$ active constraints. Recalling that the objective function of IP2 embraces an additional part C, we have

$$\begin{aligned} value(S) &\le C + (d+1)\sum_{p \in P^d} z_p \le C + (d+1)OPT(DU) \\ &= C + (d+1)OPT(LP) \le C + (d+1)(OPT(IP2) - C) \\ &\le (d+1)OPT(IP2), \end{aligned}$$

completing the proof of Lemma 2. □

Lemmas 1 and 2 ensure that S is a $(d+1)$-approximate solution to IP2. We next explain how to make Algorithm 1 run in poly-time for all d. Consider the following two operations:

(1) Find an unfrozen variable of DU if there exists at least one.

(2) Given a variable z_p, find all the constraints in DU that contain z_p.

Lemma 3. *If operations (1) and (2) can be done in polynomial time, then Algorithm 1 can be implemented to run in polynomial time.*

Proof. Suppose (1) and (2) can be done in polynomial time. Since DU has at most $\binom{n}{2} + n \leq n^2$ constraints and each time only one variable raises its value, we can keep the current LHS and RHS values of each constraint, and are thus able to know which constraints are tight. Hence Line 10 can be realized implicitly since a variable is frozen iff it appears in some tight constraint. To implement Line 4, we first apply (1) to find an unfrozen variable (say z_p) if there exists one, and then use (2) to find a constraint containing z_p that has the smallest difference between RHS and LHS values; this difference is exactly the amount that z_p can be raised. The other steps in Algorithm 1 can clearly be implemented to run in poly-time. The lemma is thus proved. $\square$

Lemma 4. *We can accomplish (1) and (2) in polynomial time.*

Proof. We use $c(v)$ to denote the constraint in DU that corresponds to the variable v of IP2. First note that (2) is easy to implement: For each variable z_p where p has endpoints i and j, z_p appears exactly in the constraints corresponding to $y_{i,j}$ or x_k for some $k \in V_p \cup E_p$. Thus we focus on (1). As shown in the proof of Lemma 3, we know the set of tight constraints in DU, and a variable is unfrozen if and only if it does not appear in any tight constraint. For $p \in P^d$, the variable z_p does not appear in $c(x_k)$ (where $k \in V \cup E$) iff $k \notin V_p \cup E_p$, and z_p does not appear in $c(y_{i,j})$ (where $1 \leq i < j \leq n$) iff p is not a path between i and j. We do the following: Construct a graph G' from G by deleting all $k \in V \cup E$ from G for which $c(x_k)$ is tight. Then, for every $1 \leq i < j \leq n$ such that $c(y_{i,j})$ is not tight, check whether there exists a path p from i to j in G' of length at most d; if so, then the corresponding variable z_p must be unfrozen due to our previous analysis. Also, by this procedure we will find an unfrozen variable if there exists at least one. Clearly this process can be finished in polynomial time. $\square$

Now Theorem 1 follows directly from Lemmas 1, 2, 3 and 4.

2.2 The case $d = \infty$

We next turn to the case $d = \infty$ and prove Theorem 2. We reduce MGNS(∞) to GNS(∞) as follows: Construct a graph G' by subdividing each edge $e \in E$ with a new vertex v_e. Let $w(v_e) = 0, C_{v_e} = C'_e$ and $L_{v_e} = 0$ for all $e \in E$. It is easy to argue that the problem of finding the social optimum of GNS(∞) on this new instance is equivalent to that of MGNS(∞) on the original one. Now, applying the poly-time approximation algorithm for GNS(∞) given in [20], we get a solution for MGNS(∞) with approximation ratio $O(\log |V(G')|) = O(\log n)$. This finishes the proof of Theorem 2.

We remark that a similar reduction can reduce an instance of MGNS(d) to that of GNS($2d$). Using the approximation algorithm in [20], we obtain a solution for MGNS(d) with approximation factor $4d$, which is nearly four times larger than the ratio guaranteed by Theorem 1. This is in part due to the fact that such a reduction loses some information of the graph topology, which is important to our algorithm.

2.3 Improved Approximation Ratio for MGNS(1)

We consider in this part the local infection model MGNS(1), i.e., each vertex, if infected, can only affect its neighbors. A 2-approximation follows directly from Theorem 1. We will prove Theorem 4, which shows that the approximation guarantee can be improved when considering a special (but natural) class of contact graphs, namely bipartite contact graphs.

Theorem 4. *There is a polynomial time $\frac{3}{2}$-approximation algorithm for computing the social optimum of MGNS(1) with bipartite contact graphs.*

Proof. Let $\mathcal{I}$ be an instance of MGNS(1) whose contact graph $G = (V, E)$ is bipartite. Consider the exact formulation IP2 of MGNS(d) given in Section 2.1. In our case, IP2 degenerates to the following program IP3:

$$\begin{array}{rrcll} \text{IP3:} \quad \text{Min} \displaystyle\sum_{i \in V} C'_i x_i + \sum_{\{i,j\} \in E} C'_{\{i,j\}} x_{\{i,j\}} & + & \displaystyle\sum_{1 \le i < j \le n} L_{i,j} y_{i,j} + C & \text{subject to:} \\ x_i + x_j + x_{\{i,j\}} + y_{i,j} & \ge & 1 & \forall \{i,j\} \in E \\ x_k & \in & \{0,1\} & \forall k \in V \cup E \\ y_{i,j} & \in & \{0,1\} & \forall 1 \le i < j \le n, \end{array}$$

where $\{C'_i\}, \{L_{i,j}\}$ and C are defined in the same way as before (in the paragraph between the descriptions of IP1 and IP2).

Notice that in IP3, for each $\{i,j\} \in E$, $x_{\{i,j\}}$ and $y_{i,j}$ both appear only once and in the same constraint. (Also, $y_{i,j}$ with $\{i,j\} \notin E$ does not appear in any constraint, so we can ignore them.) Thus, if $C'_{\{i,j\}} > L_{i,j}$ and $x_{\{i,j\}} = 1$, we are better off letting $y_{i,j} = 1$ and switching $x_{\{i,j\}}$ to 0 (which still gives a feasible solution). Similarly, if $C'_{\{i,j\}} \le L_{i,j}$ and $y_{i,j} = 1$, setting $x_{\{i,j\}} = 1$ and $y_{i,j} = 0$ does not increase the objective value. Thus, by comparing $C'_{\{i,j\}}$ and $L_{i,j}$, we can eliminate one variable from each constraint. Specifically, for each $\{i,j\} \in E$, the corresponding constraint can be simplified to $x_i + x_j + x_{\{i,j\}} \ge 1$ if $C'_{\{i,j\}} \le L_{i,j}$, and to $x_i + x_j + y_{i,j} \ge 1$ if $C'_{\{i,j\}} > L_{i,j}$.

We now construct from G a node-weighted hypergraph $H = (V(H), E(H))$ as follows: Let $V(H) = V \cup \{y_{i,j} \mid \{i,j\} \in E\} \cup \{x_{\{i,j\}} \mid \{i,j\} \in E\}$, and

$$\begin{aligned} E(H) \;=\; & \Big\{\{i, j, y_{i,j}\} \mid \{i,j\} \in E; C'_{\{i,j\}} > L_{i,j}\Big\} \\ & \cup \Big\{\{i, j, x_{\{i,j\}}\} \mid \{i,j\} \in E; C'_{\{i,j\}} \le L_{i,j}\Big\}. \end{aligned}$$

The node-weight function $w : V(H) \to \mathbb{R}^+ \cup \{0\}$ is defined as: $w(i) = C'_i$ for all $i \in V$; $w(x_{\{i,j\}}) = C'_{\{i,j\}}$ and $w(y_{i,j}) = L_{i,j}$ for all $\{i,j\} \in E$. Then it is easy to see that IP3 (with the constant part C discarded) is equivalent to the problem of finding a minimum-weight vertex cover of H. Note that H is a 3-uniform tripartite hypergraph, because G is bipartite. Thus, we can apply the $\frac{k}{2}$-approximation algorithm for vertex cover on k-uniform k-partite hypergraphs [23] to obtain a $\frac{3}{2}$-approximation for IP3 (note that adding C back cannot deteriorate the factor since $C \ge 0$), and hence also for MGNS(1) on bipartite contact graphs. This completes the proof of Theorem 4. □

3 Hardness of Approximation for GNS(d)

In this section we present inapproximability results for GNS(d), a special case of our model MGNS(d). Thus, all the hardness results trivially apply to MGNS(d).

Theorem 5. *For every $d \in \mathbb{N} \cup \{\infty\}$, computing the social optimum of GNS(d) is $\mathcal{APX}$-hard, even if the contact graph is 3-regular and all types of costs as well as the attack probability distribution are uniform.*

Proof. We will present an L-reduction (see e.g. [24]) from the vertex cover problem on 3-regular graphs, which is known to be $\mathcal{APX}$-hard [7]. Let $G = (V, E)$ be a 3-regular graph considered as an instance of the vertex cover problem. Assume without loss of generality that G is connected. We construct an instance $\mathcal{I}$ of GNS(d) as follows: Simply use G as the contact graph, and set $L_v = |V|, C_v = 3 - \epsilon$ (with $0 < \epsilon < 2$) and $w_v = 1/|V|$ for all $v \in V$.

Let OPT_{vc} and $OPT_{\mathcal{I}}$ denote the size of the minimum vertex cover of G and the social optimum of $\mathcal{I}$, respectively. Let $S \subseteq V$ be a vertex cover of G of size OPT_{vc}. In the instance $\mathcal{I}$ of GNS(d), if we make exactly the vertices in S secure and leave others insecure, the social cost will be $C_v|S| + L_v w_v(|V| - |S|) = (2 - \epsilon)|S| + |V|$, since every insecure node is isolated in the attack graph. As G is 3-regular, each vertex can cover at most 3 edges, and thus $|V| \leq |E| + 1 \leq 3|S| + 1 \leq 4|S|$. Therefore, $OPT_{\mathcal{I}} \leq (2 - \epsilon)|S| + |V| \leq (6 - \epsilon)|S| = (6 - \epsilon)OPT_{vc}$, finishing one direction of the L-reduction.

Now consider an arbitrary solution $S \subseteq V$ of $\mathcal{I}$. If there exist two insecure nodes (say u and v) that are adjacent in the attack graph, we change the solution to a new one (called S') by securing u, i.e., $S' = S' \cup \{u\}$. In this transformation, the additional security cost incurred is $C_u = 3 - \epsilon$, and the decreasing in the infection cost is at least $L_v w_u + L_u(w_u + w_v) = 3$. Hence, the new solution has a smaller total cost than S does. Whenever there exists an edge in the attack graph, we can repeat the above process to get a new solution with lower cost. Thus, we will finally obtain a solution S' (in at most $|E|$ steps) that is a vertex cover of G such that $cost(S') \leq cost(S)$. We also regard S' as a solution to the vertex cover problem. Note that the above argument also indicates that any optimum solution S^* to $\mathcal{I}$ must be a vertex cover of G. Similarly as before, we have $cost(S') = (2-\epsilon)|S'| + |V|$ and $OPT_{\mathcal{I}} = cost(S^*) = (2 - \epsilon)|S^*| + |V|$. As $2 - \epsilon > 0$, S^* must be a minimum vertex cover of G; i.e., $|S^*| = OPT_{vc}$. Now we have $||S'| - OPT_{vc}| = \frac{1}{2-\epsilon}|cost(S') - OPT_{\mathcal{I}}| \leq \frac{1}{2-\epsilon}|cost(S) - OPT_{\mathcal{I}}|$, which completes the other direction of the L-reduction. This concludes the proof of Theorem 5. □

Theorem 6. *Assuming Unique Games Conjecture, for any $d \in \mathbb{N} \cup \{\infty\}$ and any fixed $\epsilon > 0$, we cannot approximate the social optimum of GNS(d) to a factor of $\frac{3}{2} - \epsilon$ in polynomial time.*

Proof. Assume $\epsilon \leq \frac{1}{2}$, otherwise the statement is trivial. Let $G = (V, E)$ be a graph on which we wish to find a small vertex cover, and denote by OPT_{vc} the size of the minimum vertex cover of G. Construct an instance $\mathcal{I}$ of GNS(d) in the same way as that in the proof of Theorem 5, setting the security cost to be $3 - \frac{\epsilon}{3}$. Let $OPT_{\mathcal{I}}$ be the optimum social cost of $\mathcal{I}$. Similar to the proof of Theorem 5, we have $OPT_{\mathcal{I}} = (2 - \frac{\epsilon}{3})OPT_{vc} + |V|$. By [15], it is UGC-hard to distinguish whether $OPT_{vc} \geq (1 - \frac{\epsilon}{3})|V|$ or $OPT_{vc} \leq (\frac{1}{2} + \frac{\epsilon}{3})|V|$. Thus, it is UGC-hard to tell whether $OPT_{\mathcal{I}} \geq ((2 - \frac{\epsilon}{3})(1 - \epsilon) + 1)|V|$ or $OPT_{\mathcal{I}} \leq ((2 - \frac{\epsilon}{3})(\frac{1}{2} + \epsilon) + 1)|V|$, and therefore it is UGC-hard to approximate the social optimum of GNS(d) to a factor of

$$\frac{(2 - \frac{\epsilon}{3})(1 - \frac{\epsilon}{3}) + 1}{(2 - \frac{\epsilon}{3})(\frac{1}{2} + \frac{\epsilon}{3}) + 1} \geq \frac{3}{2} - \epsilon \, ,$$

which completes the proof of Theorem 6. □

4 Polynomial Algorithm for MGNS(d) on Trees

In this section we consider a special class of instances of MGNS(d); namely, the underlying contact graph of the instance is a tree, and the infection cost and attack probability are both uniform. We will present a polynomial time algorithm that computes the social optimum of such instances, for every fixed $d \geq 1$ or $d = \infty$. We define the minimum value of an empty set to be ∞. We have the following result:

Theorem 7. *For every fixed $d \geq 1$ or $d = \infty$, we can find in polynomial time an optimal solution of a tree-instance of MGNS(d) with uniform infection cost and attack probability.*

Proof. We first focus on the case where $d < \infty$, and then show how the algorithm can be adapted to handle the case $d = \infty$. Consider an instance $\mathcal{I}$ of MGNS(d) with contact graph $T = (V, E)$ being a tree, security cost $\{C_v \mid v \in V\}$, link-blocking cost $\{C'_e \mid e \in E\}$, uniform infection cost L, and uniform attack probability $1/n$. Root T at an arbitrary vertex. Assume without loss of generality that $V = \{1, 2, \ldots, n\}$ such that $i < j$ whenever i is a descendant of j (thus the root is n). For $1 \leq i \leq n$, let $i_1, i_2, \ldots, i_{c(i)}$ denote the set of children of i (where $c(i) = 0$ iff i is a leaf). For every $1 \leq i \leq n$ and $1 \leq s \leq c(i)$, let $T_{i,s}$ be the subtree of T induced by the vertex set $V_{i,s} := \{i, i_1, i_2, \ldots, i_s\} \cup \{j \mid j \text{ is a descendant of } i_t \text{ for some } 1 \leq t \leq s\}$. If i is a leaf, we let $T_{i,0}$ be the single-vertex tree containing only the vertex i. Clearly $T_{i,c(i)}$ is the full subtree of T rooted at i, and $T_{n,c(n)} = T$. We say (i, s) is an *admissible pair* if $T_{i,s}$ is defined. We set a total order on the set of admissible pairs, namely, define $(i, s) < (i', s')$ iff $i < i'$ or ($i = i'$ and $s < s'$).

We will use a dynamic programming approach in a bottom-up manner, during which we need to cope with the subproblems associated with the subtrees; that is, for each admissible pair (i, s), we have an instance $\mathcal{I}_{i,s}$ whose contact graph is $T_{i,s}$ and whose (all kinds of) costs and attack probabilities are the same as in $\mathcal{I}$. (Note that $\mathcal{I}_{n,c(n)}$ is exactly $\mathcal{I}$ itself.) For a solution $S_{i,s}$ of $\mathcal{I}_{i,s}$, we define the *signature* of $S_{i,s}$ to be a $(d+1)$-tuple $(n_0, n_1, \ldots, n_d)$, in which n_t $(0 \leq t \leq d)$ denotes the number of (insecure) nodes that are at distance exactly t from i in the attack graph of $S_{i,s}$. (Thus $n_0 = 0$ if i is secure and $n_0 = 1$ otherwise; we do not allow other values of n_0. It is also clear that $n_0 = 0$ implies $n_1 = \ldots = n_d = 0$.) For a signature sig and an integer $r \in \{0, 1, \ldots, d\}$, $sig[r]$ refers to the $(r+1)$-th element (from left) of sig; i.e., if $sig = (n_0, \ldots, n_d)$, then $sig[t] = n_t$ for $0 \leq t \leq d$. Let SIG denote the set of all possible signatures. It follows that $|SIG| \leq 2n^d$. Let $f_{i,s}(sig)$ denote the minimum cost of a solution to $\mathcal{I}_{i,s}$ with signature sig. Then the social optimum of $\mathcal{I}$ is exactly $\min_{sig \in SIG}\{f_{n,c(n)}(sig)\}$.

We illustrate in detail how to compute $f_{i,s}(sig)$ for all possible admissible pairs (i, s) and $sig \in SIG$. We compute them in the increasing order of (i, s). If $(i, s) = (1, 0)$ (in which case i is a leaf), the computation is trivial: for every $sig = (n_0, \ldots, n_d) \in SIG$, set $f_{1,0}(s) = C_i(1-n_0)+Ln_0/n$ if $n_1 = \ldots = n_d = 0$, and set $f_{1,0}(sig) = \infty$ otherwise.

Next suppose we want to compute $f_{i,s}(sig)$, given that the values of $f_{i',s'}(sig')$ are known for all $(i', s') < (i, s)$ and $sig' \in SIG$. Assume $sig = (n_0, \ldots, n_d)$. If i is a leaf, the computation is similar to the base case, so we assume that i is not a leaf and thus $c(i) \geq 1$.

Case 1: $s = 1$. In this case $T_{i,s} = T_{i,1}$ just consists of $T_{i_1,c(i_1)}$, the vertex i, and the edge $\{i, i_1\}$. We consider two subcases.

- **Case 1.1:** $n_0 = 0$; i.e., i is secure. Then the edge $\{i, i_1\}$ need not be blocked. Set $f_{i,1}(sig) = \infty$ if $n_t > 0$ for some $1 \leq t \leq d$. When $n_1 = \ldots = n_d = 0$, it is easy to see that

$$f_{i,1}(sig) = C_i + \min_{sig_1 \in SIG}\{f_{i_1,c(i_1)}(sig_1)\}.$$

- **Case 1.2:** $n_0 = 1$; i.e., i is insecure. We examine two more subcases guided by whether the edge $\{i, i_1\}$ is blocked. Let f^{ub} (resp. f^b) be the minimum cost of a solution to $\mathcal{I}_{i,1}$ with signature sig, in which $\{i, i_1\}$ is unblocked (resp. blocked). If $\{i, i_1\}$ is not blocked, then any vertex at distance t from i_1 in the attack graph will be at distance $t+1$ from i. Also, we need to calculate the cost incurred from the mutual influence of i and nodes that are reachable from i within distance d in the attack graph. Hence we get:

$$f^{ub} = \frac{L}{n}\left(1 + 2\frac{\sum_{i=1}^{d} n_i}{n}\right) + \min_{\substack{sig_1 \in SIG \\ (\forall 0 \le t \le d-1) sig_1[t] = sig[t+1]}} \{f_{i_1,c(i_1)}(sig_1)\}.$$

  If $\{i, i_1\}$ is blocked, then i itself forms a connected component in the attack graph. Thus, $f^b = \infty$ if $n_t > 0$ for some $1 \le t \le d$. When $n_1 = \ldots = n_d = 0$, we have:

$$f^b = C'_{\{i,i_1\}} + \frac{L}{n} + \min_{sig_1 \in SIG} \{f_{i_1,c(i_1)}(sig_1)\}.$$

  Finally, we have $f_{i,1}(sig) = \min\{f^{ub}, f^b\}$, and thus Case 1 is finished.

Case 2: $2 \le s \le c(i)$. In this case, $T_{i,s}$ consists of $T_{i,s-1}$, $T_{i_s,c(i_s)}$, and the edge $\{i, i_s\}$. Again we divide it into two subcases.

- **Case 2.1** $n_0 = 0$; i.e., i is secure. Then all nodes reachable from i in the attack graph are from $T_{i,s-1}$, and thus the signature of the solution restricted on $T_{i,s-1}$ must also be sig. Therefore, we have:

$$f_{i,s}(sig) = \min_{sig_1 \in SIG} \{f_{i,s-1}(sig) + f_{i_s,c(i_s)}(sig_1)\}.$$

  (Note that we do not add the security cost C_i since it is already counted in $T_{i,s-1}$.)

- **Case 2.2:** $n_0 = 1$; i.e. i is insecure. Let g^{ub} (resp. g^b) be the minimum cost of a solution to $\mathcal{I}_{i,s}$ with signature sig in which the edge $\{i, i_s\}$ is unblocked (resp. blocked). Similarly as Case 2.1, we have:

$$g^b = C'_{\{i,i_s\}} + \min_{sig_1 \in SIG} \{f_{i,s-1}(sig) + f_{i_s,c(i_s)}(sig_1)\}.$$

  When computing g^{ub}, we should take into account the cost caused by the influence between nodes in $T_{i,s-1}$ and those in $T_{i_s,c(i_s)}$. (Note that i itself is contained in $T_{i,s-1}$.) Let v_1 be a node at distance t_1 from i in $T_{i,s-1}$, and v_2 be a node at distance t_2 from i_s in $T_{i_s,c(i_s)}$. Then clearly v_1 and v_2 can affect each other if and only if $t_1 + t_2 + 1 \le d$. Thus we have:

$$g^{ub} = \min_{\substack{sig_1, sig_2 \in SIG \\ sig_1[0]=1 \\ (\forall 1 \le t \le d) sig_1[t] + sig_2[t-1] = sig[t]}} \{\frac{2L}{n} \cdot \sum_{\substack{0 \le t_1, t_2 \le d-1 \\ t_1 + t_2 \le d-1}} sig_1[t_1] \cdot sig_2[t_2] + f_{i,s-1}(sig_1) + f_{i_s,c(i_s)}(sig_2)\}$$

  Finally we have $f_{i,s}(sig) = \min\{g^{ub}, g^b\}$, and Case 2 is completed.

After finishing the computation of all sub-instances, we can find the social optimum of $\mathcal{I}$ since it equals to $\min_{sig \in SIG}\{f_{n,c(n)}(sig)\}$.

Now we analyze the running time of the above algorithm. The time spent on computing each term $f_{i,s}(sig)$ is at most $O(d \cdot |SIG|^2)$. To see this, first observe that it is obviously true for all computations except g^{ub}. When computing g^{ub}, an upper bound of $O(d^2 \cdot |SIG|^2)$ is simple by enumerating sig_1, sig_2, t_1 and t_2. To reduce it by a factor of d, note that the summation term can be rewritten as

$$\sum_{\substack{0 \le t_1, t_2 \le d-1 \\ t_1+t_2 \le d-1}} sig_1[t_1] \cdot sig_2[t_2] = \sum_{0 \le t_1 \le d-1} sig_1[t_1] \cdot \sum_{0 \le t_2 \le d-1-t_1} sig_2[t_2],$$

and thus we only need the d cumulative sums

$$\{S_i \mid 0 \le i \le d-1; S_i = \sum_{t=0}^{i} sig_2[t]\}$$

to compute this summation. Clearly these sums can be obtained in $O(d)$ time, resulting in a total computation time of $O(d \cdot |SIG|^2)$ for g^{ub}.

As there are at most $n^2|SIG|$ such terms, it costs $O(dn^2|SIG|^3)$ time to compute all of them. In the last step it takes $O(|SIG|)$ time to find the minimum cost among $|SIG|$ terms. Thus, the total running time of our algorithm is $O(dn^2|SIG|^3) \le n^{O(d)}$, which is polynomial in n for every fixed d. Notice also that it is very easy to modify the algorithm so that it can output an optimal solution; one just need to store the best solutions to all the sub-instances, which incurs an additional $n^{O(d)}$ time.

We now consider the case $d = \infty$, which is in fact simpler than the previous case. We still use the dynamic programming approach on the subproblems $\{\mathcal{I}_{i,s}\}$ in a bottom-up matter. The difference from the previous case is that we change the definition of the signature. For a solution $S_{i,s}$ of the instance $\mathcal{I}_{i,s}$, we define the signature of $S_{i,s}$ to be a pair $(n_0, n_{\ge 1})$, where $n_0 \in \{0,1\}$ denotes whether i is secure or not, and $n_{\ge 1}$ represents the the number of nodes other than i that are in the same component with i in the attack graph induced by $S_{i,s}$. All the computations will go through analogously (with considerable simplifications especially when dealing with g^{ub}). This completes the proof of Theorem 7. □

By taking the size of $|SIG|$ into account, we can handle a larger range of d.

Corollary 1. *For all $d \le O(\sqrt{\log n})$, we can find in polynomial time an optimal solution to MGNS(d) if the instance has uniform infection cost and attack probability, and its contact graph is a tree of bounded degree.*

Proof. Note that the algorithm presented in Theorem 7 actually runs in $O(dn^2|SIG|^3)$ time for the case $d < \infty$. Thus, it can also handle super-constant d when $|SIG|$ is small. Suppose the underlying tree of the instance has maximum degree Δ. Then, we have

$$|SIG| \le \prod_{t=0}^{d} (\Delta^t + 1) = \Delta^{O(d^2)},$$

since at most Δ^t nodes are at distance t from i in the subtree $T_{i,c(i)}$. Hence, the running time becomes $O(\Delta^{O(d^2)} dn^2)$, which leads to Corollary 1. □

Finally, we note that the algorithm can be easily modified to also handle the following generalization of our problem with additional *budget constraints* on the number of secured nodes and links.

Budgeted MGNS(d). Given an instance of MGNS(d) with two additional integers K and K', we want to find a minimum-cost solution in which at most K nodes are secured and at most K' edges are blocked.

We sketch the idea of how to apply the algorithm of Theorem 7 on Budgeted MGNS(d) as well. We add two dimensions to the signature of a solution S that represent the number of secured nodes and blocked edges in S, respectively. We then rewrite the state-transition functions to take these two dimensions into account (in an obvious way). All the computations go through similarly. Hence, we know that Theorem 7 and Corollary 1 both apply to the budgeted case as well. This in particular settles an open problem from [22] that asks if there is a polynomial time algorithm for the minimum average contamination problem, which corresponds to the special case of MGNS(d) on trees where every node has security cost ∞ and all other costs as well as the attack probability distribution are uniform.

5 Conclusions and Future Research

We propose in this paper the mixed generalized network security model MGNS(d), which generalizes several other models for infection control. We present approximation and inapproximability results for the problem of computing the optimum solution of MGNS(d), and exact polynomial-time algorithms for tree instances with uniform infection cost and attack probability distribution. Some of our results lead immediately to improvements upon the previously best known results achieved for some special cases of our model.

There are many interesting questions left that deserve further explorations. Regarding the optimization of social cost, a big open question is whether we can break the $O(\log n)$ factor for MGNS(∞) or GNS(∞), or there is a matching hardness of approximation result. Also for MGNS(d) where $d < \infty$, there remains a large gap between the upper bound of $d + 1$ and the lower bound of $\frac{3}{2} - \epsilon$ on the approximation ratio. Another research issue is the formulation and investigation of the decentralized or game-theoretic counterpart of our model, where a user can decide whether to install an anti-virus software, and might also be able to block some of the links to other users. Finally, incorporating other propagation models (e.g., the independent cascade model, or the linear threshold model) into MGNS(d) may lead to more accurate modeling of some applications.

References

[1] S. Arora, S. Rao, and U. Vazirani. Expander flows, geomeric embeddings and graph partitioning. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, 2004.

[2] A. Arulselvan, C.W. Commander, L. Elefteriadou, and P.M. Pardalos. Detecting critical nodes in sparse graphs. *Comput. Oper. Res.*, 36(7):2193–2200, 2009.

[3] J. Aspnes, K.L. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.*, 72(6):1077–1093, 2006. Preliminary version in SODA 2005.

[4] S. Borgatti. Identifying sets of key players in a social network. *Comput. Math. Org. Theory*, 12:21–34, 2006.

[5] P.-A. Chen, M. David, and D. Kempe. Better vaccination strategies for better people. In *Proceedings of the 11th ACM Conference on Electronic Commerce (ACM-EC)*, 2010.

[6] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[7] M. Chlebík and J. Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354:320–338, 2006.

[8] M. Di Summa, A. Grosso, and M. Locatelli. Complexity of the critical node problem over trees. *Comput. Oper. Res.*, 38(12):1766–1774, 2011.

[9] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[10] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996. Preliminary version in STOC 1993.

[11] A. Goyal, F. Bonchi, and L.V. S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM)*, 2010.

[12] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraph. *SIAM J. Comput.*, 31(5):1608–1623, 2002.

[13] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[14] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, 2002.

[15] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. Preliminary version in CCC 2003.

[16] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2006.

[17] M. Kimura, K. Saito, and H. Motoda. Minimizing the spread of contamination by blocking links in a network. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

[18] M. Kimura, K. Saito, and H. Motoda. Solving the contamination minimization problem on networks for the linear threshold model. In *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence (PRICAI)*, 2008.

[19] M. Kimura, K. Saito, and H. Motoda. Blocking links to minimize contamination spread in a social network. *ACM Trans. Knowl. Discov. Data.*, 3(2), 2009.

[20] V.S. Anil Kumar, R. Rajaraman, Z. Sun, and R. Sundaram. Existence theorems and approximation algorithms for generalized network security games. In *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS)*, 2010.

[21] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2007.

[22] A. Li and L. Tang. The complexity and approximability of minimum contamination problems. In *Proceedings of the 8th International Conference on Theory and Applications of Models of Computation (TAMC)*, 2011.

[23] L. Lovász. On minimax theorems of combinatorics. Doctoral Thesis, Mathematiki Lapok, 26:209–264, 1975.

[24] V.V. Vazirani. *Approximation Algorithms.* Springer, 2004.