Yale University
Department of Computer Science

Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Partition Problem

James Aspnes [1 2] Kevin Chang [1 3]
Aleksandr Yampolskiy [1 4]

YALEU/DCS/TR-1295
July 2004

[1]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA.
[2]Email: aspnes@cs.yale.edu. Supported in part by NSF grants CCR-0098078 and CNS-0305258.
[3]Email: kevin.chang@yale.edu. Supported by NSF grant CCR-0331548.
[4]Email: aleksandr.yampolskiy@yale.edu. Supported by NSF grants CCR-0098078 and ANI-0207399.

Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Partition Problem

James Aspnes *† Kevin Chang *‡ Aleksandr Yampolskiy *§

Abstract

We propose a simple game for modeling containment of the spread of viruses in a graph of n nodes. Each node must choose to either install anti-virus software at some known cost C, or risk infection and a loss L if a virus that starts at a random initial point in the graph can reach it without being stopped by some intermediate node. The goal of individual nodes is to minimize their individual expected cost. We prove many game theoretic properties of the model, including an easily applied characterization of Nash equilibria, culminating in our showing that allowing selfish users to choose Nash equilibrium strategies is highly undesirable, because the price of anarchy is an unacceptable $\Theta(n)$ in the worst case. This shows in particular that a centralized solution can give a much better total cost than an equilibrium solution. Though it is **NP**-hard to compute such a social optimum, we show that the problem can be reduced to a previously unconsidered combinatorial problem that we call the **sum-of-squares partition problem**. Using a greedy algorithm based on sparse cuts, we show that this problem can be approximated to within a factor of $O(\log^2 n)$, giving the same approximation ratio for the inoculation game.

1 Introduction

Consider a system in which n machines, each of which may or may not be protected against viruses, are connected by a network in the form of a graph, and any virus that infects some machine eventually infects all of its unprotected neighbors. If anti-virus software is available, a natural response would be to protect all the machines—but perhaps the anti-virus software itself creates costs, both in money and time to purchase and install the software and in reduced efficiency or usability of the protected machine. Suppose that protecting a machine by installing anti-virus software costs the owner of the machine C, but that having the machine be infected costs L, which may or may not be greater than C. If the virus spreads from some initial machine chosen uniformly at random, on which machines does it make sense to install the anti-virus software?

The answer will depend on whether we adopt the perspective of the owner of a single machine or of the society as a whole. When the anti-virus software costs more than the loss from infection, no economically rational machine owner will install the anti-virus software, every machine will be infected, and the system will incur a social cost of Ln. But for many graphs, selective inoculation of a few central machines can limit the spread of infection to a small subset of the graph, greatly reducing the total cost of infection in return for a small investment in anti-virus software. We can ask how much of an improvement a centralized solution can provide, and how easy it is to find a good centralized solution.

*Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA.
†Email: aspnes@cs.yale.edu. Supported in part by NSF grants CCR-0098078 and CNS-0305258.
‡Email: kevin.chang@yale.edu. Supported by NSF grant CCR-0331548.
§Email: aleksandr.yampolskiy@yale.edu. Supported by NSF grants CCR-0098078 and ANI-0207399.

After discussing some previous work on related problems (in Section 2), we give a complete characterization of the Nash equilibria for an anti-virus software installation game in which each machine's owner separately chooses whether or not to install the software, without regard to the effect on other machines. (This game is defined in Section 3.) We show (in Section 4) that finding either the most or least expensive equilibrium is **NP**-hard, but that some Nash equilibrium can be computed in $O(n^3)$ time and that any population of nodes will quickly converge to a Nash equilibrium by updating their strategies locally based on the other nodes' strategies. Unfortunately, the cost of any such Nash equilibrium may be badly suboptimal; the **price of anarchy** for this game is $\Theta(n)$ in the worst case. This shows that for many graphs and values of C and L, letting the users choose individually whether or not to inoculate their machines will give bad results.

We then consider (in Section 5) the possibility of a centralized solution in which a dictator computes and enforces an optimal inoculation plan. We show that essentially the same argument that shows that extreme Nash equilibria are hard to find applies to the optimal solution as well. However, we show that the problem of finding an optimal inoculation plan reduces to a graph partition problem in which we are asked to partition the graph by removing m nodes; the quality of the partition is measured by the sum of the squares of the sizes of its components. We give (in Section 6) a polynomial-time approximation algorithm that removes $O(\log^2 n)m$ nodes in order to achieve a partition with quality within $O(1)$ of the optimum.

Conclusions and open problems appear in Section 7.

2 Related work

In this section, we describe three classes of work related to this paper: virus propagation models, economic models of investment in security, and game-theoretic models of security. We then discuss some work on the graph partition problem that is related to the sum-of-squares partition problem we consider in Section 6.

2.1 Virus propagation models

Traditional epidemiological models characterize the viral infection in terms of birth rate and death rate of the virus [4, 9]. Usually, these models assume that an infected individual is equally likely to infect any other individual in the population; meanwhile, computer viruses usually spread via localized interactions. Kephart and White extended the traditional model by transferring it onto a directed random graph [16]. Later works (*e.g.*, [15, 17, 25]) studied virus propagation over other kinds of graphs, including Internet-like power-law graphs [22,23,26]. We do not restrict the network topology in any way and consider a general undirected graph. Our model is in some ways closer to models in percolation theory (see [18]): an infected node infects all of its unprotected neighbors, spreading infection throughout the graph until it is blocked by an anti-virus software.

2.2 Economic models of security

Our work is motivated in part by an observation that security technologies exhibit network externalities [1]. Specifically, the benefit obtained by using security technology (anti-virus software in our case) does not accrue only to the user of the security technology but rather to all users of the network. Aspnes *et al.* [3] also consider anti-virus immunization, and proposed studying how to encourage highly connected nodes to use anti-viral techniques.

We assume that costs of installation and infection are known. Alternatively, one could use risk analysis to estimate the costs and benefits from installing a security technology (see, for example, [13]), or estimate values based on empirical studies of the costs of security breaches [6, 10].

2.3 Game-theoretic models of security

Application of game theory to network security has yielded interesting results [11, 12, 24]. For example, Bell uses a simple game to study network reliability. In the game, the router tries to find a least cost path and a network tester tries to maximize this cost by failing links [5]. Kunreuther and Heal recently introduced the notion of **interdependent security** (IDS) games, in which decisions to adopt security technology by one agent affect other agents [20]. Kearns and Ortiz subsequently extended their paper and gave an algorithm for finding approximate Nash equilibria in this model [14].

Our work is similar to work on IDS games in certain respects: each agent in both our game and an IDS game makes a decision whether or not to invest money in a security technology, and this decision affects other agents. The main differences are that we assume that installing anti-virus software protects against all bad effects of viruses, while the IDS work concentrates on negative side-effects of security breaches even on protected parties; and we assume a restricted network topology that contains the spread of viruses, while the IDS work assumes a complete topology.

2.4 Graph partition problems

In Section 6, we describe and provide an approximate solution for a new graph partitioning problem. Previous work on other forms of graph partitioning includes the approximation algorithm of Leighton and Rao [21] for **sparsest cut**, from which the same authors derive a pseudo-approximation algorithm for b**-balanced cuts**, where each side of the cut must have size $b|V|$ or greater. The case of $b = 1/2$ is **graph bisection**, for which Feige and Krauthgamer [8] give a good approximation algorithm. Even *et al.* [7] give $O(\log n)$-ratio pseudo-approximation algorithms for several balanced partitioning problems, including the ρ**-separator problem** and the k**-balanced partitioning problem**.

3 Our model

We represent network topology by an undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n-1\}$ is a set of network hosts and $E \subseteq V \times V$ is a set of (bidirectional) communication links. Our basic model for installing anti-virus software is a one-round game:

Players. Our game has n players corresponding to nodes of the graph. Initially, all nodes are insecure and vulnerable to infection.

Strategies. We denote the **strategy** of i by a_i. Each node i has two possible actions: **do nothing** and risk being infected or **inoculate itself** by installing anti-virus software. Node i's strategy a_i is the probability that it inoculates itself.

Nodes' choices can be summarized in a **strategy profile** $\vec{a} \in [0, 1]^n$. If a_i is 0 or 1, we say that node i adopts a **pure strategy**; otherwise, its strategy is **mixed**. We call nodes that install anti-virus software **secure** and denote the set of such nodes by $I_{\vec{a}}$. We associate an attack graph $G_{\vec{a}} = G - I_{\vec{a}}$ with $\vec{a}$. It is essentially the network graph with secure nodes and their edges removed (see also Figure 1). Note that both $I_{\vec{a}}$ and $G_{\vec{a}}$ are random variables unless all strategies are pure.

Attack model. After the nodes made their choices, the adversary picks some node uniformly at random as a starting point for infection. Infection then propagates through the network graph. Node i gets infected if it has no anti-virus software installed and if any of its neighbors become infected.

Individual costs. Suppose it costs C to install anti-virus software. If a node is infected, it suffers a loss equal to L. For simplicity, we assume that both C and L are known and are the same for all nodes; we discuss possible consequences of removing these assumptions in Section 7.

The cost of a mixed strategy $\vec{a} \in [0,1]^n$ to node i is

$$\mathrm{cost}_i(\vec{a}) = a_i C + (1 - a_i) L \cdot p_i(\vec{a}).$$

Here $p_i(\vec{a})$ is the probability of node i being infected given the strategy profile $\vec{a}$, *conditioned on the event that node i does not install the anti-virus software.* It is equal to the probability that some vulnerable node reachable from i without passing through a secure node is the initial point of infection. For pure strategies, this is just k_i/n, where k_i is the size of the connected component containing i in the attack graph $G_{\vec{a}}$.

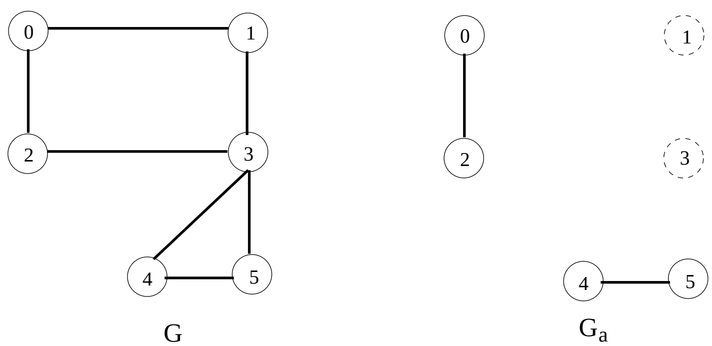

Figure 1: Sample graph G and its attack graph $G_{\vec{a}}$ for $\vec{a} = 010100$.

Social cost. The total social cost of a strategy profile is the sum of the individual costs. For pure strategies, there is a simple characterization of the total social cost in terms of the attack graph $G_{\vec{a}}$. Because each node in the same component of $G_{\vec{a}}$ has the same chance of infection, the k_i nodes in the i-th component between them face a loss of $k_i \cdot (Lk_i/n) = (L/n)k_i^2$. So the social cost is

$$\begin{aligned}
\mathrm{cost}(\vec{a}) &= \sum_{j=0}^{n-1} \mathrm{cost}_j(\vec{a}) \\
&= \sum_{j=0}^{n-1} a_j C + (1 - a_j) L \cdot p_j(\vec{a}) \\
&= C|I_{\vec{a}}| + \frac{L}{n} \sum_{i=1}^{l} k_i^2,
\end{aligned}$$

where $k_1, k_2, \ldots k_\ell$ are the sizes of the components in $G_{\vec{a}}$.

4 Nash equilibria

We consider first the choices that the nodes will make in the absence of coordination, by examining the Nash equilibria of the game defined in Section 3. The assumption that the nodes will reach a Nash equilibrium is a very strong one, as it requires assuming that the nodes are aware of each other's choices to install or not and that the nodes can evaluate C (printed on the box for the anti-virus software) and L (which is more problematic). It also assumes that the nodes can compute a Nash equilibrium in a reasonable amount of time, which is not always possible for some games. However, we can show that Nash equilibria for our game are easily characterized in terms of the sizes of the components of the attack graph (Section 4.1), and that a population will converge to some Nash equilibrium quickly even though finding the best or worst *pure* equilibrium as measured by total cost is **NP**-hard (Section 4.2).

We can further imagine that some of the difficulties of limited information could be overcome by considering an iterated game where nodes pay C to rent the anti-virus software in each round and update their strategies based on observations of losses to viruses and the strategies of other nodes in previous rounds; though we do not analyze this multi-round game formally, a simplified version is implicit in our convergence result. We also show that the hardness of finding the worst-case equilibrium does not prevent obtaining further information about its behavior; for example, its total cost is nondecreasing as a function of the inoculation cost C (Section 4.3).

Unfortunately, selfish behavior proves to be highly undesirable, because the cost of a Nash equilibrium solution may be very far from the social optimum. In Section 4.4, we prove that while the **price of anarchy**, defined as the ratio of total cost between the worst Nash equilibrium and the social optimum never exceeds n, this bound is tight up to constant factors for some graphs and choices of C and L.

4.1 Characterization of mixed and pure equilibria

A useful feature of the Nash equilibrium for our game is its simple characterization: there is always a component-size threshold $t = Cn/L$ such that each node will install the anti-virus software if it would otherwise end up in a component of vulnerable nodes with expected size greater than t, and will not install the software if it would otherwise end up in a component with expected size less than t. When the expected component size equals t, the node is indifferent between installing and not installing and may adopt a mixed strategy. The threshold arises in a natural way: it is the break-even point at which the cost C of installing the software equals the expected loss $L(t/n)$ of not installing.

We define $\vec{a}[i/x]$ to be the strategy vector that is identical to $\vec{a}$, except the i'th component a_i is replaced by x. Note that attack graph $G_{\vec{a}[i/0]}$ is the attack graph in which player i never installs the anti-virus software.

Theorem 1 *(Characterization of mixed equilibria): Suppose $S(i)$ is the expected size of the insecure component that contains node i of the attack graph $G_{\vec{a}[i/0]}$, (i.e. $S(i) = np_i(\vec{a})$).*

Fix C, L. Let the threshold be $t = Cn/L$. A strategy profile $\vec{a}$ is a Nash equilibrium if and only if *(a) For all i such that $a_i = 1$, $S(i) \geq t$.*

(b) For all i such that $a_i = 0$, $S(i) \leq t$.

(c) For all i such that $0 < a_i < 1$, $S(i) = t$.

Proof:

Suppose $\vec{a}$ is a Nash equilibrium and consider node i. The expected cost to node i is $a_i C + (1 - a_i)(L/n)S(i)$.

1. Suppose $a_i = 0$. Then node i has expected cost $(L/n)S(i)$. If $(L/n)S(i) > C$, then i will want find the situation $a_i = 1$ with cost C preferable. Thus, we must have $S(i) \leq CL/n = t$.

2. Suppose $a_i = 1$. Then node i has expected cost C. If $(L/n)S(i) < C$, then i would find the situation $a_i = 0$ with expected cost $(L/n)S(i) < C$ preferable. Thus, we must have $S(i) \geq CL/n = t$.

3. Suppose $0 < a_i < 1$. If $(L/n)S(i) > C$, then i will find the situation $a_i = 1$ preferable. If $(L/n)S(i) < C$, then i will find the situation $a_i = 0$ preferable. Thus, we must have $S(i) = CL/n = t$.

Thus, $\vec{a}$ satisfies condition (a), (b), and (c) above.

Conversely, suppose $\vec{a}$ satisfies conditions (a), (b) and (c) of the theorem. Consider node i.

1. Suppose $a_i = 0$. Then node i will have expected cost $(L/n)S(i) < C$, and thus will not want to switch to a different a_i that puts any weight on installing at cost C.

2. Suppose $a_i = 1$. Then node i will have cost C, and thus will not want to switch to a different a_i that puts any weight on being insecure at expected cost $(L/n)S(i) \geq C$.

3. Suppose $0 < a_i < 1$. Then node i will have expected cost $a_i C + (1 - a_i)(L/n)S(i) = C$. Switching to any other strategy will have the same expected cost.

Thus, $\vec{a}$ is a Nash equilibrium. ∎

A special case of Theorem 1 is the following characterization for pure Nash equilibria. Because nodes in a pure Nash equilibrium do not make randomized choices, the attack graph is not a random object, but a determined graph. We have the same threshold conditions as before, but the removal of randomness simplifies the situation.

Corollary 2 *(Characterization of pure equilibria) Fix C, L. Let the threshold be $t = Cn/L$. A strategy profile $\vec{a}$ is a pure Nash equilibrium if and only if*

(a) Every component in attack graph $G_{\vec{a}}$ has size at most t.

(b) Inserting any secure node $j \in I_{\vec{a}}$ and its edges into $G_{\vec{a}}$ yields a component of size at least t.

For example, let $C = 0.5$ and $L = 1$, and consider the graph in Figure 1. The threshold for this graph is $t = Cn/L = 3$. Then Corollary 2 tells us that pure strategy $\vec{a} = 010100$ must be a Nash equilibrium for these C and L.

4.2 Computing pure Nash equilibria

Designing algorithms for finding mixed Nash equilibria or proving hardness results for finding optimized mixed equilibria would most likely involve estimating or otherwise manipulating the expected value of the sizes of components in the attack graph, which is at the very least a non-trivial problem. Furthermore, in the absence of central control, nodes attempting to calculate their best strategy based on a mixed strategy paradigm would possibly run into similar computational issues.

Thus, we turn our attention to the computation and hardness of pure Nash equilibria. Corollary 2 gives us a powerful tool with which to reason about pure Nash equilibria. We now show that computing the best or worst pure Nash equilibria is hard, but that finding some intermediate Nash equilibrium is easy. A consequence of this algorithm is that the existence of a pure Nash equilibrium is always guaranteed. (The existence of a mixed Nash equilibrium is a consequence of Nash's theorem.)

Theorem 3 *Both computing the pure Nash equilibrium with lowest cost and computing the pure Nash equilibrium with highest cost are **NP**-hard problems.*

Proof: We reduce VERTEX COVER to the decision problem "Does there exist a pure Nash equilibrium with cost less than c?" and we reduce INDEPENDENT DOMINATING SET to "Does there exist a pure Nash equilibrium with cost greater than c?"

Fix some graph $G = (V, E)$, and set $C/L = 1.5/n$ so that $t = Cn/L = 1.5$, where t is the component size threshold from Corollary 2. Then from Corollary 2, in any Nash equilibrium the components of the attack graph all have size at most 1, and any secure node is adjacent to some insecure node (as otherwise it could uninstall its software and be in a component of size at most 1). It follows that in a Nash equilibrium (a) every vulnerable node is either isolated or has all neighbors secure, and (b) every secure node has an insecure neighbor.

We now argue that G has a vertex cover of size k if and only if the inoculation game on G with the above settings of C and L has a Nash equilibrium with k or fewer secure nodes, or equivalently an equilibrium with social cost $Ck + (n-k)L/n$ or less, as each insecure node must be in a component of size 1 and contribute exactly L/n expected cost. Given a minimal vertex cover $V' \subseteq V$, observe that installing the software on all nodes in V' satisfies condition (a) because V' is a vertex cover, and (b) because V' is minimal. Conversely, if V' is the set of secure nodes in a Nash equilibrium, then V' is a vertex cover by condition (a). This concludes the proof that finding a minimum-cost Nash equilibrium is **NP**-hard.

For a maximum cost equilibrium, consider the set of *insecure* vertices. These consist of isolated nodes (which are already in components of size 1) and nodes that do not install the software because all their neighbors do. Given an independent dominating set $V' \subseteq V$, installing the software on all nodes *except* the nodes in V' satisfies condition (a) because V' is independent and (b) because V' is a dominating set. Conversely, the insecure nodes in any Nash equilibrium are independent by condition (a) and dominating by condition (b). This shows that G has an independent dominating set of size k if and only if it has a Nash equilibrium with no more than k insecure nodes, which occurs only if it has a Nash equilibrium with at least $n - k$ secure nodes or, equivalently, a cost of at least $C(n-k) + (L/n)(k)$. ∎

Theorem 3 says that finding extreme pure equilibria is hard. But what if we just want to converge to some equilibrium, but we don't care which one? Suppose we implement the process of convergence implied by the Nash equilibrium: at each step, some participant, whose current strategy is suboptimal given the others' strategies, switches. This is an easy process to implement because each participant can detect if its strategy is suboptimal using the $t = Cn/L$ component size threshold from Corollary 2.[1] But does this process converge to a Nash equilibrium? If it does, how long does it take?

By choosing an appropriate potential function, we can show that this process does indeed converge to a Nash equilibrium quickly:

Theorem 4 *Starting from any pure strategy profile $\vec{a}$, if at each step some participant with a suboptimal strategy switches its strategy, the system converges to a pure Nash equilibrium in no more than $2n$ steps.*

Proof: Let $t = Cn/L$. For any strategy profile $\vec{a}$, consider the set $S_{\text{big}}(\vec{a})$ of "big" components of $G_{\vec{a}}$ of size greater than t and the set $S_{\text{small}}(\vec{a})$ of "small" components of $G_{\vec{a}}$ of size less than or equal to t. Define a potential function Φ by

$$\Phi(\vec{a}) = \sum_{A \in S_{\text{big}}(\vec{a})} |A| - \sum_{A \in S_{\text{small}}(\vec{a})} |A|.$$

It is easy to see that $-n \leq \Phi(\vec{a}) \leq n$ for any $\vec{a}$. We will now show that each step of the process reduces Φ by at least one. There are two main cases:

1. Some node i switches from insecure to secure. In this case i was previously an element of a component in S_{big} of size $m > t$. This former component becomes one or more new components with total size $m - 1$; if all of the resulting components are big, Φ is reduced by exactly one; otherwise, Φ is reduced by more than one as some components move from the positive to the negative side of the ledger.

2. Some node i switches from secure to insecure. In this case the resulting component containing i has $m \leq t$ elements, and it replaces one or more old components with total size $m - 1$. As both the new component and the old components are small, the net effect on Φ is to decrease it by one.

If each step reduces Φ by one, the number of steps must be less than the difference between the initial and final value of Φ, which is at most $n - (-n) = 2n$. ∎

As a special case, we can start with $\vec{a} = 1^n$ and converge to an equilibrium from above by checking each node once. Each such test requires computing the size of the component in the attack graph, which takes time $O(|V| + |E|) = O(n^2)$ using depth-first search; this gives:

Corollary 5 *A Nash equilibrium can be computed in time $O(n^3)$.*

It is not hard to see that the $2n$ in Theorem 4 is close to the best possible bound, although a more careful analysis might reduce it slightly. A lower bound of n steps is trivial: in a system with $C < L/n$ and no players secure in the initial strategy profile, it takes n steps for all players to

[1]We must assume in this implementation either that the choice to install software or not is reversible, or that each player can observe the other players' intended actions and respond accordingly.

install the anti-virus software. To get closer to $2n$, consider a line with $t = \sqrt{n} - \frac{1}{2}$. Now consider an execution of the process where initially players 1 through $n - \sqrt{n}$, in increasing order, install to escape the single overlarge component; but then all players not at positions $k\sqrt{n}$ for some k uninstall; this takes $2n - 2\sqrt{n}$ steps.

We also have:

Corollary 6 *A pure Nash equilibrium always exists.*

4.3 Consequences of changes in the inoculation cost

Though Theorem 3 suggests that we cannot hope to characterize the worst pure Nash equilibrium exactly, we can give a description of how it reacts to changes in the inoculation cost C.

Theorem 7 *The cost of the worst pure Nash equilibrium is a non-decreasing function of C when C ranges over $[2L/n, L)$.*

Proof: Fix some price of anti-virus software, $C \geq 2L/n$ so that $\lfloor Cn/L \rfloor \geq 2$.

Suppose we increase the price from C to $C' = C + \epsilon$ $(\epsilon > 0)$. We denote the worst-cost Nash equilibrium when the price is C by $\vec{a}$ and the worst-cost equilibrium when the price is C' by $\vec{b}$.

If price increment is $\epsilon \leq L/n$, then the threshold (in Theorem 1) increases by at most one; that is, $\lfloor C'n/L \rfloor \leq \lfloor Cn/L \rfloor + 1$. We consider two cases:

Case 1: $\vec{a}$ is a Nash equilibrium for C'. This case is easy. Because $\vec{b}$ is a worst-cost Nash equilibrium for C', we have:

$$\text{cost}_C(\vec{a}) < \text{cost}_{C'}(\vec{a}) \leq \text{cost}_{C'}(\vec{b}).$$

Case 2: $\vec{a}$ is *not* a Nash equilibrium for C'. This can happen only if $\lfloor C'n/L \rfloor = \lfloor Cn/L \rfloor + 1$. Specifically, there must exist a node $w \in I_{\vec{a}}$ such that adding it into attack graph $G_{\vec{a}}$ yields a component of size $\lfloor Cn/L \rfloor$ but not $\lfloor C'n/L \rfloor$. Let us denote the sizes of components adjacent to w in G by $k_1, \ldots, k_s$.[2] We then have: $\sum_{i=1}^{s} k_i = \lfloor Cn/L \rfloor - 1$.

We define a new strategy $\vec{a'} = \vec{a}[w/0]$, which is the same as $\vec{a}$ except we no longer install anti-virus software on node w. Moreover,

$$\begin{aligned}
\text{cost}_{C'}(\vec{a'}) - \text{cost}_C(\vec{a}) &= \frac{L}{n} \lfloor Cn/L \rfloor^2 - \left(C + \frac{L}{n} \sum_{i=1}^{s} k_i^2 \right) \\
&\geq \frac{L}{n} \left(\lfloor Cn/L \rfloor^2 - \left(\sum_{i=1}^{s} k_i \right)^2 \right) - C \\
&= \frac{L}{n} \left(\lfloor Cn/L \rfloor^2 - (\lfloor Cn/L \rfloor - 1)^2 \right) - C. \qquad (1)
\end{aligned}$$

Equation (1) is non-negative whenever

$$2 \lfloor Cn/L \rfloor - 1 \geq Cn/L,$$

[2] We say that a component $K \subseteq V$ is adjacent to node w if $\exists v \in K$ s.t. $(v, w) \in E$.

which always holds by assumption for all $C \geq 2L/n$.

We repeat this process until there do not exist any nodes violating Nash equilibrium condition. At each step, the cost of our new strategy does not decrease. Therefore, if at the end we get a Nash equilibrium $\vec{d}$, then

$$\text{cost}_C(\vec{a}) \leq cost_{C'}(\vec{a}) \leq \text{cost}_{C'}(\vec{d}) \leq cost_{C'}(\vec{b}).$$

Hence,

$$\begin{aligned} cost_{C'}(\vec{b}) - cost_C(\vec{a}) &\geq cost_{C'}(\vec{d}) - cost_C(\vec{a}) \\ &\geq cost_{C'}(\vec{a'}) - cost_C(\vec{a}) \\ &\geq 0. \end{aligned}$$

Because we chose C arbitrarily, our argument holds for all values of ϵ.

∎

4.4 Price of anarchy

The notion of the **price of anarchy** was introduced by Koutsoupias and Papadimitriou in [19]. It is defined as the worst-case ratio between the cost of a Nash equilibrium and the cost of the optimal solution, and is thus a measure of how far away a Nash equilibrium can be from the social optimum.[3] In our game, we show that the price of anarchy is quite high, $\Theta(n)$.

This is a consequence of two simple lemmas:

Lemma 8 *(Lower bound). Let G be the star graph $K_{1,n}$. Let the price of the anti-virus software be $C = L(n-1)/n$. Then*

$$\rho(G, C, L) = n/2.$$

Proof: The given C and L satisfy $t = Cn/L = n-1$. From Corollary 2, it follows that installing anti-virus software on exactly one node is a Nash equilibrium. If pure Nash strategy $\vec{a}$ installs anti-virus software on some node that is not the center node, the cost will be $C + L(n-1)^2/n = L(n-1)$.

An optimal strategy for the star with the given C and L is $\vec{a}^* = (1, 0, \ldots, 0)$ (*i.e.*, only the center node installs anti-virus software.) Its cost is $C + L(n-1)/n = 2L(n-1)/n$.

The price of anarchy is therefore

$$\frac{L(n-1)}{2L(n-1)/n} = \frac{n}{2}.$$

∎

Lemma 9 *(Upper bound). Fix any graph G and costs C, L. Then*

$$\rho(G, C, L) \leq n.$$

[3] Because our game has a random component, the cost is an expected cost.

Proof: Let $\vec{a}^*$ denote the optimum solution.

If $C > L$, no node in a Nash equilibrium will install anti-virus software. Hence, there is only one Nash equilibrium $\vec{a} = 0^n$, whose cost is Ln. If the optimum solution contains at least one secure node, then then $\text{cost}(\vec{a}^*) \geq C > L$. (Otherwise, $\vec{a}^* = 0^n$ and $\rho(G, C, L) = 1$.) We thus have:

$$\rho(G, C, L) \leq \frac{Ln}{L} = n.$$

If $C \leq L$, then the expected cost of the worst Nash equilibrium $\vec{a}$ is at most Cn, because the expected cost to each node is at most C (if the expected cost to a node is greater than C, then it will want to switch to installing the software with probability 1.) If the optimum solution contains at least one secure node, then $\text{cost}(\vec{a}^*) \geq C$. Otherwise, the optimum solution contains no secure nodes and hence $\text{cost}(\vec{a}^*) \geq L \geq C$.

$$\rho(G, C, L) \leq \frac{Cn}{C} = n.$$

∎

5 Optimization

Allowing users to selfishly choose whether or not to install anti-virus software may be grossly inefficient, relative to the social optimum. An alternative approach to this problem is for a benevolent dictator to attempt to maximize social welfare by centrally computing a solution and imposing it on all nodes.

Difficulties with this approach arise from the hardness of computing the optimum solution to the inoculation problem. In the first two sections, we give a characterization of the optimum solution and use it to show that the inoculation problem is **NP**-hard.

This suggests computing an approximate solution. We can find in polynomial time a solution with approximation ratio at most $O(\log^2 n)$; such a solution is substantially better than the $\Theta(n)$ ratio derived from the worst Nash equilibrium.

5.1 Characterization

We have a graph-theoretic characterization of optimum strategies, similar to our characterization of Nash equilibria in Theorem 1:

Theorem 10 *Fix C, L and let $t = Cn/L$. If $\vec{a}$ is an optimum strategy, then every component in attack graph $G_{\vec{a}}$ has size at most* $\max(1, (t + 1)/2)$.

Proof: Strategy $\vec{a}$ partitions G into disjoint components. Pick some component from attack graph with at least two nodes; we call it $K \subseteq V$ ($k = |K|$). (If we can't find a component with at least two nodes, then all components in attack graph must have size one, and the theorem follows.)

If we install an anti-virus on some node of K, we may get m new components in $G_{\vec{a}}$, where $0 \leq m \leq k - 1$. We denote component sizes by $k_1, \ldots, k_m$, where $\sum_{i=1}^{m} k_i = k - 1$. Because $\vec{a}$ is an optimal strategy, installing anti-virus on an extra node cannot improve the total cost. Therefore, we have:

$$\begin{aligned} & C + \frac{L}{n}\left(\sum_{i=1}^{m} k_i^2\right) \geq \frac{Lk^2}{n} \\ \Leftrightarrow \quad & \\ & k^2 - \left(\sum_{i=1}^{m} k_i^2\right) \leq t. \end{aligned} \tag{2}$$

If $m = 0$, then Equation (2) becomes:

$$k \leq \sqrt{t} \leq (t+1)/2.$$

Meanwhile, for $m > 0$,

$$\begin{aligned} k^2 - \left(\sum_{i=1}^{m} k_i^2\right) &\geq k^2 - \left(\sum_{i=1}^{m} k_i\right)^2 \\ &= k^2 - (k-1)^2 \\ &= 2k - 1. \end{aligned} \tag{3}$$

Combining Equations (2) and (3), we get:

$$k \leq (t+1)/2.$$

■

5.2 Hardness of optimal solution

Unfortunately, the optimal solution is hard to compute.

Theorem 11 *It is **NP**-hard to compute an optimal strategy.*

Proof: The proof is by reduction from VERTEX COVER and is similar to the proof of Theorem 3. ■

5.3 Reduction to sum-of-squares partition

Because it is unlikely that we can find an optimal solution, we naturally consider approximation algorithms.

The optimization problem that defines the inoculation problem can be posed as follows: choose the set of secure nodes $I_{\vec{a}}$ that minimizes the following objective function:

$$C|I_{\vec{a}}| + \frac{L}{n} \sum_{V \in \phi(I_{\vec{a}})} |V|^2,$$

where $\phi(I_{\vec{a}})$ is the set of connected components created by the removal of nodes in $I_{\vec{a}}$.

For the purposes of our approximation algorithm for the inoculation problem, we assume that we can "guess" $m = |I_{\vec{a}}|$, the number of secure nodes in an optimum configuration. This assumption is without loss of generality, because we can run our algorithm on all possible choices of $m = 1, \ldots n$ and take the best solution.

Thus, a solution to the inoculation problem is reduced to finding a solution to the problem of removing m nodes from a given graph to minimize the sum of the squares of the sizes of the surviving components. We discuss this problem in Section 6.

6 Sum-of-squares partitions

In Section 5.3, we came across the following problem, which we now analyze in more detail.

Problem 12 *(Sum-of-Squares Partition Problem) By removing a set F of at most m nodes, partition the graph into disconnected components $H_1, \ldots, H_k$, such that $\sum_i |H_i|^2$ is minimum.*

Though we have arrived at this combinatorial optimization problem via our study of containing computer viruses, it may be of independent interest. Note that it inherits **NP**-hardness from the inoculation problem. The *edge version* of the sum-of-squares-partition problem is similar, but asks for the removal of m edges, rather than nodes, to disconnect the graph.

Our goal in this section is to prove the following theorem:

Theorem 13 *Let* OPT *be the optimum objective function value for the Sum-of-Squares Partition Problem on G with the removal of at most m nodes. We can find a set F of $O(\log^2 n)m$ nodes, such that their removal creates disconnected components $H_1, \ldots, H_l$ such that $\sum_i |H_i|^2 \leq O(1) \cdot \mathrm{OPT}$.*

An immediate consequence of this theorem is the existence of an approximation algorithm for the inoculation problem:

Corollary 14 *If* OPT *is the cost of the optimum solution for the inoculation problem, there exists a polynomial-time approximation algorithm that finds a solution with cost at most $O(\log^2 n) \cdot \mathrm{OPT}$.*

Proof: Suppose an optimum solution contains m secure nodes, and the sizes of the insecure node components are $k_1, \ldots, k_p$, so that $\mathrm{OPT} = Cm + L/n \sum_i k_i^2$. Using our approximation algorithm for the sum-of-squares partition problem, we can find a set of $O(\log^2 n)m$ secure nodes such that the sum of the squares of the corresponding insecure components is at most $O(1) \sum_i k_i^2$. Thus, the cost of the approximate solution is:

$$\begin{aligned} O(\log^2 n) \cdot Cm + O(1) \cdot \frac{L}{n} \sum_i k_i^2 &\leq O(\log^2 n) \cdot Cm + O(\log^2 n) \cdot \frac{L}{n} \sum_i k_i^2 \\ &= O(\log^2 n) \cdot \mathrm{OPT}. \end{aligned}$$

∎

6.1 Proof of Theorem 13

Our proof of Theorem 13 is based on the algorithm **PartitionGraph** given in Figure 2. It uses known algorithms for sparse cuts. The sparse cut literature has focused on edge cuts. However, in the case of our problem, cuts that involve removing nodes in order to disconnect the graph are more applicable. Fortunately, the Leighton-Rao approximation algorithm for finding sparse cuts extends to node cuts: there is a well known process for converting a node cut algorithm in an undirected graph to an edge cut algorithm in a directed graph. Since the Leighton-Rao sparse cut algorithm extends to edge cuts for directed graphs, it can be extended to node cuts. In particular, we will use the following fact, which is implicit in the discussion of balanced node cuts in [21].

Theorem 15 *There exists an $O(\log n)$-approximation algorithm for the following problem: Given graph G, find a node cut that partitions the nodes G into three sets: two sets defining disconnected subgraphs with node sets, V_1 and V_2, and a set of removed nodes R, such that the quantity*

$$\frac{(|V_1| + \frac{|R|}{2})(|V_2| + \frac{|R|}{2})}{|R|} \tag{4}$$

is maximized.

We refer to the quantity in expression (4) as the **sparsity** of the cut. In the literature, sparsity is usually given as the the inverse of expression (4), and finding the sparsest cut is cast as a minimization problem. We have presented it as a maximization problem, since this is more natural for our application.

Improvements to the approximation ratio for sparsest (edge) cut have recently been discovered [2]. However, the authors do not mention whether or not their results extend to sparse cuts for directed graphs.

Our algorithm for solving the sum-of-squares partition problem, **PartitionGraph** (see Figure 2), achieves the approximation results claimed in Theorem 13. The general approach of the algorithm is similar to the greedy $\log n$-approximation algorithm for set cover. A high-level description is that we repeatedly remove the node cut that gives us the best per-node-benefit, quantified as its **cost-effectiveness**.

Suppose we have a connected subgraph H with k nodes. If node cut R creates connected components with node sets V_1 and V_2, this cut has decreased the objective function value ($\sum$ size of connected component ^{2}) by $k^2 - |V_1|^2 - |V_2|^2$. We thus define the **cost-effectiveness** of node cut R by $(k^2 - |V_1|^2 - |V_2|^2)/|R|$. The cost-effectiveness of R is equal to

$$\begin{aligned}
\frac{k^2 - |V_1|^2 - |V_2|^2}{|R|} &= \frac{(|V_1| + |V_2| + |R|)^2 - |V_1|^2 - |V_2|^2}{|R|} \\
&= \frac{|R|^2 + 2|V_1||V_2| + 2|R|(|V_1| + |V_2|)}{|R|} \\
&= \frac{2|V_1||V_2|}{|R|} + |R| + 2(k - |R|) \\
&= \frac{2|V_1||V_2|}{|R|} + 2k - |R|.
\end{aligned}$$

We then have the following relation between finding sparse cuts and cost-effectiveness.

Lemma 16 *Let H be a graph with k nodes. If α^* is the maximum cost-effectiveness of all node cuts of H, the Leighton-Rao sparsest node cut algorithm will find a cut with cost-effectiveness at least $\alpha^*/(c \log k)$, for some constant c.*

Proof: The sparsity of a node cut that removes node set R and partitions the remaining nodes of H into connected components with node sets V_1 and V_2 is given by:

$$\begin{aligned}
\frac{(|V_1| + \frac{|R|}{2})(|V_2| + \frac{|R|}{2})}{|R|} &= \frac{|V_1||V_2| + \frac{|R|^2}{4} + \frac{|R|}{2}(|V_1| + |V_2|)}{|R|} \\
&= \frac{|V_1||V_2|}{|R|} + \frac{|R|}{4} + \frac{1}{2}(k - |R|) \\
&= \frac{|V_1||V_2|}{|R|} + \frac{k}{2} - \frac{|R|}{4}.
\end{aligned}$$

We then have the following relationships between the cost-effectiveness of a cut, α, and its sparsity, β.

$$\beta = \frac{|V_1||V_2|}{|R|} + \frac{k}{2} - \frac{|R|}{4} = \alpha/2 - k/2 + |R|/4 \geq \alpha/4.$$

and

$$\alpha > 2\beta.$$

Thus, we know there exists a node cut with sparsity at least $\alpha^*/4$ (i.e. the cut with the highest cost-effectiveness). The sparsest cut algorithm on H will find a node cut with sparsity at least $\alpha^*/(c \log k)$, for some constant c. This node cut will have cost-effectiveness at least $2\alpha^*/(c \log k)$. ∎

Input: A Graph G. A set of marked nodes F.

1. Use the Leighton-Rao algorithm to find an approximate most cost-effective cut in each connected component of G.
2. Let $H_1, \ldots, H_k$ be the components of G in which the Leighton-Rao algorithm found a cut that removes at most $(20c \log n)m$ nodes, where c is the constant from Lemma 16. If no such component exists, then halt and output F as the final set of marked nodes.
3. Otherwise, choose the component H_j from among those considered in Step 2, for which the cost-effectiveness is highest. Let the cut be $H_j = V_1 \cup V_2 \cup R$, where R is the final set of marked nodes.
4. Set $F = F \cup R$. Replace H_j by V_1 and V_2 in G. If $|F| > (36c \log^2 n)m$, then halt and output F as the set of marked nodes.
5. Otherwise, repeat.

Figure 2: Algorithm **PartitionGraph**

We now give some lemmas that characterize the behavior of the **PartitionGraph** algorithm. Fix an optimum solution for the sum-of-squares partition problem; let F^* be an optimum set of m removed nodes.

Lemma 17 ***PartitionGraph*** *outputs at most $O(\log^2 n)m$ marked nodes.*

Proof: Since the algorithm halts as soon as we augment the set of marked nodes such that $|F| > (36c\log^2 n)m$, we know that at the beginning of each iteration, F contains at most $(36c\log^2 n)m$ marked nodes. The lemma follows, because we add at most $(20c\log n)m$ marked nodes per iteration. ∎

In the next few proofs, we will abuse notation slightly and denote the order of graph G (i.e. the number of nodes) by $|G| = |V(G)|$. We will also denote an "intersection" of a graph G and a node set V as $G \cap V$, which is the set of nodes that G and V share.

Lemma 18 *Suppose after a number of iterations, our graph G consists of k connected components $H_1, \ldots, H_k$, and let $S = \sum |H_i|^2$.*

Either there exists a component H_i such that the Leighton-Rao algorithm will find a node cut in H_i with at most $20cm\log n$ removed nodes and cost-effectiveness at least $S/(18cm\log n)$,

or $S \leq 72\mathrm{OPT}$ (or possibly both).

Proof: We assume that $S > 2\mathrm{OPT}$. Note that the node cut defined by the set $F^* \cap G$ divides G into a graph with objective function value at most OPT. The node cut thus induces a cost decrease of at least $S/2$.

Define $F_i^* = F^* \cap H_i$ and $m_i = |F_i^*|$. Also, let the subgraph induced by removing vertices in $F_i^* \cap H_i$ from H_i be composed of connected components H_i^j for $j = 1, \ldots, r_i$. (i.e, the optimum set of marked nodes partitions H_i into these components.) Note that $\sum_i \sum_j |H_i^j|^2 \leq \mathrm{OPT}$.

Since the total reduction in our objective function value from removing $\cup_i F_i^*$ from G is at least $S/2$ due to our assumption that $S > 2\mathrm{OPT}$, we have:

$$\sum_i \left(|H_i|^2 - \sum_j |H_i^j|^2 \right) \geq S/2, \tag{5}$$

because the outer summand on the left hand side of the inequality is the amount the objective function is reduced in each component.

Let I be the set of indices i for which $\left(|H_i|^2 - \sum_{j=1}^{r_i} |H_i^j|^2\right)/m_i \geq S/(4m)$ (i.e. the per node benefit is at least $S/(4m)$.)

We have two cases.

1. There exists an $i \in I$ such that for all $j = 1, \ldots, r_i$, $|H_i^j| \leq 1/3|H_i|$. We assume that $m_i < 1/50|H_i|$, because otherwise removing all nodes in H_i will give us a trivial node cut with cost-effectiveness at least $|H_i|^2/(50m_i) > S/(18cm\log n)$ for sufficiently large n. With this assumption, we know that there exists a set $R \subseteq F_i^*$ that defines a node cut of H_i^j that creates two connected components, V_1 and V_2 such that $1/3|H_i| \leq |V_1|$ and $1/3|H_i| \leq |V_2|$. The cost-effectiveness of this cut will be

$$2\frac{|V_1||V_2|}{|R|} + 2|H_i| - |R| \geq \frac{2|H_i|^2}{9m_i} \geq \frac{2\left(|H_i|^2 - \sum_{j=1}^{r_i} |H_i^j|^2\right)}{9m_i} \geq S/(18m).$$

   Lemma 16 guarantees that the Leighton-Rao algorithm will find a cut in H_i with cost-effectiveness at least $S/(18cm\log n)$. The node cut output by the algorithm cannot contain more than $20cm\log n$ nodes. Such a node cut would have cost-effectiveness at most $S/(20cm\log n)$, since any cut in G can decrease the objective function value by at most S, which is less than the guaranteed cost-effectiveness of $S/(18cm\log n)$.

2. For all $i \in I$, there exists a j^* such that $|H_i^{j^*}| > 1/3|H_i|$. Also, note that $\mathrm{OPT} > \sum_{i\in I} |H_i^{j^*}|^2$.

   Claim: $\sum_{i\in I}\left(|H_i|^2 - \sum_j |H_i^j|^2\right) \geq S/8$.

   Proof of claim: Let $\overline{I}$ be the set of intervals such that $\left(|H_i|^2 - \sum_{j=1}^{r_i} |H_i^j|^2\right)/m_i \leq S/(4m)$. Recalling equation (5), we get

$$S/2 \leq \sum_{i\in I}\left(|H_i|^2 - \sum_j |H_i^j|^2\right) + \sum_{i\in \overline{I}}\left(|H_i|^2 - \sum_j |H_i^j|^2\right).$$

   Also, we have

$$\begin{aligned}
\sum_{i\in\overline{I}}\left(|H_i|^2 - \sum_j |H_i^j|^2\right) &= \sum_{i\in\overline{I}} m_i \frac{\left(|H_i|^2 - \sum_j |H_i^j|^2\right)}{m_i} \\
&\leq \sum_{i\in\overline{I}} m_i S/(4m) \\
&\leq S/(4m)\sum_{i\in\overline{I}} m_i \leq S/4.
\end{aligned}$$

   Combining these two inequalities proves the claim. We have the inequalities:

$$\mathrm{OPT} > \sum_{i\in I} |H_i^{j^*}|^2 \geq \sum_{i\in I}\frac{1}{9}|H_i|^2 \geq \frac{1}{9}\frac{S}{8},$$

   where we used our claim for the last inequality. Thus, $\mathrm{OPT} \geq 72S$.

∎

We now give the proof of Theorem 13.

Let a_j be the number of connected components that comprise the graph at the beginning of the jth iteration, and let those connected components be $H_1^j, \ldots, H_{a_j}^j$. Let $S_j = \sum_{i=1}^{a_j} |H_i^j|^2$ be the value of the objective function at the beginning of the j'th iteration; thus $S_0 \leq n^2$ is its initial value. Let l be the number of iterations the algorithm needs to terminate, and S_{l+1} be the objective function's final value.

We wish to show that after the algorithm terminates, we have reduced the objective function value to $S_{l+1} = O(1)\cdot\mathrm{OPT}$. Let F be the final set of marked nodes removed from G. If the algorithm terminates at Step 2 of the l'th iteration because the Leighton-Rao algorithm only found node cuts with more than $(20c\log n)m$ removed nodes, then from Lemma 18 we know that $S_{l+1} \leq 72\mathrm{OPT}$. Thus, we assume this does not occur. Furthermore, we assume that $S_{l+1} \geq 72\mathrm{OPT}$ (in order to apply the "either" part of Lemma 18 to all iterations.)

In order to reason about the decrease in the objective function value after each iteration, we impute to each node in F a per-node-decrease in the objective function value, given by the cost-effectiveness of its node cut. We then show that the total imputed decrease in the objective function is at least $n^2 - O(1)\cdot\mathrm{OPT}$, from which the theorem will follow.

More formally, suppose the set of marked nodes is given by the sequence $F = \{f_1, \ldots, f_k\}$, where the removed nodes are ordered in the order in which they were removed from the graph: nodes removed at an earlier iteration occur earlier in the sequence. We have $k = |F| = \Theta(\log^2 n)m$.

Let b_j be the iteration in which f_j was removed. We impute to f_j the value $\delta_j =$ cost-effectiveness of cut removed in iteration b_j. From Lemma 18, we know that $\delta_j \geq S_{b_j}/(18cm\log n)$.

Set $T_0 = S_0$ and $T_i = T_{i-1} - \delta_i$ to be the value of the objective function after node f_i's per-node-decrease contribution has been made. Note $T_k = S_{l+1}$.

Claim: For all i, $T_i \leq T_{i-1} - T_{i-1}/(18cm\log n)$

Proof of claim: Proving the claim reduces to proving that $\delta_i = T_{i-1} - Ti \geq T_{i-1}/(18cm\log n)$. Fix an i. We have two cases.

1. $b_i = b_{i-1}$ (i.e. f_i and f_{i-1} were removed in the same iteration.) Then $\delta_i \geq S_{b_i}/(18cm\log n)$, but $S_{b_i} > T_i$, since S_{b_i} is the objective function value at the beginning of iteration b_i, whereas T_i is the objective function value "during" iteration b_i.

2. $b_i = b_{i-1} + 1$ (i.e. f_i was removed in the iteration after f_i was removed.) Then $\delta_i \geq S_{b_i}/(18cm\log n) = T_i/(18cm\log n)$, since in this case T_i is the objective function value at the start of iteration b_i.

This proves the claim.

We therefore have $T_k \leq T_0(1-1/(18cm\log n))^k \leq n^2(1-1/(18cm\log n))^k$. Since $k > 36cm\log^2 n$, it follows that $S_{l+1} = T_k = O(1) \leq O(1) \cdot \text{OPT}$. ∎

The algorithm given above can be adapted in a straightforward way to yield an algorithm for the edge cut version of the sum-of-squares partition problem (instead of taking sparse node cuts, take sparse edge cuts), from which an analog to Theorem 13 may be derived. The analysis of an edge cut algorithm would perhaps be slightly easier than our analysis for node cuts, since node cuts modify the node set, causing minor complications. Furthermore, such an algorithm can use sparse cut algorithms with better approximation ratios, and thus remove only $O(\log^{3/2} n)m$ nodes to achieve the same constant approximation ratio for the sum of squares.

7 Conclusions and future research

We have described a simple economic game that represents the difficult problem of choosing on which nodes to install anti-virus software to contain the spread of computer viruses in a network. The Nash equilibria of this game have a simple characterization, and we can show that in the worst case, the ratio between the social cost of a Nash equilibrium and a social optimum can be linear in the number of nodes.

Our model makes some very strong simplifying assumptions: every infected node eventually infects all unprotected neighbors; the costs of installing the anti-virus software and becoming infected are known and equal for all nodes; the virus imposes no costs on protected nodes; and nodes can observe which of the other nodes intend to install the anti-virus software and adjust their own strategies in response. None of these assumptions correspond completely to reality, but we believe that as a first step the resulting model is a reasonable compromise between accuracy and analyzability, and that the results obtained with the model (especially the characterization of Nash equilibria) are similar to what one might expect with a more complex model that took into account limited information and learning by individual nodes. The natural next step is to incorporate more details in the model and see if such changes affect the results; this might involve both theoretical work to predict the effect of changes and experimental or observational work to study how real-world decision-makers choose whether or not to deploy specific security mechanisms.

We have also shown how a near-optimal deployment of anti-virus software can be computed by reduction to the **sum-of-squares partition problem**, a new variant of classical graph partitioning problems where the goal is to remove m vertices so as to minimize the sum of the squares of the sizes of surviving components. Though it is **NP**-hard to solve this problem exactly, we give a polynomial-time $O(\log^2 n)$-approximation algorithm for sum-of-squares partition, which yields a corresponding approximation algorithm for anti-virus software deployment. This algorithm may be of use as a network administration tool for choosing how to deploy anti-virus software to minimize the combined costs of deployment and infection and as a public-health tool for designing inoculation strategies for containing outbreaks of highly-infectious diseases when a good approximation to the interaction graph can be computed but the initial source of contagion is unknown. Whether or not a polynomial-time algorithm with a better approximation ratio exists remains open.

8 Acknowledgments

The authors would like to thank Joan Feigenbaum, Hong Jiang, and Yang Richard Yang for many useful discussions.

References

[1] R. Anderson. Why information security is hard - an economic perspective, 2001. Available at `http://www.cl.cam.ac.uk/~rja14/econsec.html`.

[2] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 222–231, 2004.

[3] J. Aspnes, J. Feigenbaum, M. Mitzenmacher, and D. Parkes. Towards better definitions and measures of Internet security. In *Workshop on Large-Scale Network Security and Deployment Obstacles*, 2003.

[4] N. T. Bailey. *The Mathematical theory of infectious diseases and its applications.* Hafner Press, 1975.

[5] M. G. H. Bell. The measurement of reliability in stochastic transport networks. In *Proceedings of 2001 Intelligent Transportation Systems*, pages 1183–1188, 2001.

[6] K. Campbell, L. A. Gordon, M. P. Loeb, and L. Zhou. The economic cost of publicly announced information security breaches: Empirical evidence from the stock market. *Journal of Computer Security*, 2003.

[7] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28:2187–2214, 1999.

[8] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31:1090–1118, 2002.

[9] J. C. Frauenthal. *Mathematical modeling in epidemiology.* Springer-Verlag, New York, 1980.

[10] L. A. Gordon and M. P. Loeb. The economics of information security investment. *ACM transactions on information and system security*, pages 438–457, 2002.

[11] S. N. Hamilton, W. L. Miller, A. Ott, and O. S. Saydjari. Challenges in applying game theory to the domain of information warfare. In *4th Information survivability workshop (ISW-2001/2002)*, Vancouver, Canada, 2002.

[12] S. N. Hamilton, W. L. Miller, A. Ott, and O. S. Saydjari. The role of game theory in information warfare. In *4th Information survivability workshop (ISW-2001/2002)*, Vancouver, Canada, 2002.

[13] K. Hoo. How much is enough? A risk-management approach to computer security. Consortium for Research on Information Security Policy (CRISP) Working Paper., 2000.

[14] M. Kearns and L. Ortiz. Algorithms for interdependent security games. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[15] J. O. Kephart, D. M. Chess, and S. R. White. Computers and epidemiology. In *IEEE Spectrum*, pages 20–26, 1993.

[16] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Symposium on Security and Privacy*, pages 343–361, 1991.

[17] J. O. Kephart and S. R. White. Measuring and modeling computer virus prevalence. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1993.

[18] H. Kesten. *Percolation Theory for Mathematicians*, volume 2. Birkhäuser, Boston, 1982.

[19] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th annual symposium on theoretical aspects of computer science*, pages 403–413, 1999.

[20] H. Kunreuther and G. Heal. Interdependent security. *Journal of Risk and Uncertainty (Special Issue on Terrorist Risks)*, 2003.

[21] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the Association for Computing Machinery*, 46(6):787–832, 1999.

[22] R. Pastor-Satorras and A. Vespignani. Epidemics and immunization scale-free networks. In S. Bornholdt and H. Schuster, editors, *Handbook of graphs and networks: from the genome to the Internet*, pages 113–132, Berlin, 2002. Wiley-VCH.

[23] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. In *Physical Review Letters*, volume 65, 2002.

[24] P. F. Syverson. A different look at secure distributed computation. In *IEEE Computer Security Foundations Workshop (CSFW 10)*, pages 109–115. IEEE Computer Society Press, June 1997.

[25] C. Wang, J. C. Knight, and M. C. Elder. On computer viral infection and the effect of immunization. In *ACSAC*, pages 246–256, 2000.

[26] C. Zou, D. Towsley, and W. Gong. Email virus propagation modeling and analysis. Technical Report CSE-03-04, University of Massachusetts, Amherst, 2002.