Network Decontamination with a Single Agent

Yassine Daadaa1 · Asif Jamshed1,2 · Mudassir Shabbir3

Received: 28 July 2013 / Revised: 16 September 2014 / Published online: 30 April 2015 © Springer Japan 2015

Abstract Faults and viruses often spread in networked environments by propagating from site to neighboring sites. We model this process of network contamination by graphs. Consider a graph $G = (V, E)$, whose vertex set is contaminated and our goal is to decontaminate the set $V$ using mobile decontamination agents that traverse along the edge set of $G$. Temporal immunity, $\tau(G) \geq 0$, is defined as the time that a decontaminated vertex of $G$ can remain continuously exposed to some contaminated neighbor without getting infected itself. The immunity number of $G$, $\iota_k(G)$, is the least $\tau(G)$ that is required to decontaminate $G$ using $k$ agents. We study immunity number for some classes of graphs corresponding to network topologies and present upper bounds on $\iota_1(G)$, in some cases, with matching lower bounds. Variations of this problem have been extensively studied in literature, but proposed algorithms have been restricted to monotone strategies, where a vertex, once decontaminated, may not be recontaminated. We exploit nonmonotonicity to give bounds which are strictly better than those derived using monotone strategies.

Keywords Network decontamination · Pursuit-evasion · Graph search · Decontamination with immunity

Yassine Daadaa
y.daadaa@ccis.imamu.edu.sa

Mudassir Shabbir
mudassir.shabbir@itu.edu.pk

1 College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, KSA
2 PredictifyMe Inc., Raleigh, NC 27601, USA
3 Department of Computer Science, Information Technology University, Arfa Technology Park, Lahore, Pakistan
1 Introduction

Faults and viruses often spread in networked environments by propagating from site to neighboring sites through links in the network. The process is called network contamination. Once contaminated, a network node might behave incorrectly, and it could cause its neighboring nodes to become contaminated as well, thus propagating faulty computations. The propagation patterns of faults can follow different dynamics, depending on the behavior of the affected node, and topology of the network. At one extreme we have a full spread behavior: when a site is affected by a virus or any other malfunction, the malfunction can propagate to all its neighbors. At other times, faults propagate only to sites that are susceptible to be affected. The definition of susceptibility depends on the application but often it is based on local conditions. For example, a node could be vulnerable to contamination if a majority of its neighbors are faulty, and immune otherwise (e.g., see [15, 16, 19]); or it could be immune to contamination for a certain amount of time after being repaired (e.g., see [8, 13]).

In fact we work with a variation of the notion of temporal immunity as defined in [8, 13]. We consider propagation of faults based on temporal immunity where a clean node can remain exposed to contaminated nodes for a predefined period of time after which it becomes contaminated. Actual decontamination is performed by mobile cleaning agents which move from host to host over network connections.

1.1 Previous Work

Graph Search The decontamination problem considered in this paper is a variation of a problem extensively studied in the literature known as graph search. The graph search problem was first introduced by Breish in [5], where an approach for the problem of finding an explorer that is lost in a complicated system of dark caves is given. Parsons ([21, 22]) proposed and studied the pursuit-evasion problem on graphs. Members of a team of searchers traverse the edges of a graph in pursuit of a fugitive, who moves along the edges of the graph with complete knowledge of the locations of the pursuers. The efficiency of a graph search solution is based on the size of the search team. The size of smallest search team that can clear a graph $G$ is called the search number, and is denoted in the literature by $s(G)$. In [20], Megiddo et al. approached the algorithmic question: Given an arbitrary $G$, how should one calculate $s(G)$? They proved that for arbitrary graphs, determining if the search number is less than or equal to an integer $k$ is NP-Hard. They also gave algorithms to compute $s(G)$ where $G$ is a special case of trees. For their results, they used the fact that recontamination of a cleared node does not help reduce $s(G)$, which was proved by LaPaugh in [17]. A search plan for $G$ that does not involve recontamination of cleared nodes is referred to as a monotone plan.

Decontamination The model for decontamination studied in literature is defined as follows. A team of agents is initially located at the same node, the homebase, and all the other nodes are contaminated. A decontamination strategy consists of a sequence of movements of the agents along the edges of the network. At any point in time each node of the network can be in one of three possible states: clean, contaminated,
or guarded. A node is guarded when it contains at least one agent. A node is clean when an agent passes through it and it has not yet been recontaminated by one of its neighbors, and contaminated otherwise. The solution to the problem is given by devising a strategy for the agents to move in the network in such a way that at the end all the nodes are clean.

The tree was the first topology to be investigated. In [2], Barrière et al. showed that for a given tree $T$, the minimum number of agents needed to decontaminate $T$ depends on the location of the homebase. They gave the first strategies to decontaminate trees. In [12], Flocchini et al. consider the problem of decontaminating a mesh graph. They present some lower bounds on the number of agents, number of moves, and time required to decontaminate a $p \times q$ mesh ($p \leq q$). They showed that at least $p$ agents, $pq$ moves, and $p+q-2$ time units are required to solve the decontamination problem. Decontamination in graphs with temporal immunity, which is similar to the model of decontamination used in this paper, was first introduced in [13] where the minimum team size necessary to disinfect a given tree with temporal immunity $\tau$ was derived. The main difference between the classical decontamination model, and the new model in [13] is that once an agent departs the decontaminated node is immune for a certain $\tau \geq 0$ (where $\tau = 0$ corresponds to the classical model studied in the previous work) time units to viral attacks from infected neighbors. After the temporal immunity time $\tau$ has elapsed, recontamination can occur.

Some further work in the same model was done in [9], where a two dimensional lattice is considered.

1.2 Definitions and Terminology

We will only deal with connected finite graphs without loops or multiple edges. For a graph $G = (V, E)$, and a vertex $v \in V$ let $N(v)$, the neighborhood of $v$, be the set of all vertices $w$ such that $v$ is connected to $w$ by an edge. Let $\text{deg}(v)$ denote the degree of a vertex $v$ which is defined to be the size of its neighborhood. The maximum and minimum degrees of any vertex in $G$ are denoted by $\Delta(G)$ and $\delta(G)$ respectively. The shortest distance between any two vertices $u, v \in V$ is denoted by $\text{dist}(u, v)$ and the eccentricity of $v \in V$ is the maximum $\text{dist}(u, v)$ for any other vertex $u$ in $G$. The radius of a graph, $\text{rad}(G)$, is the minimum eccentricity of any vertex of $G$ and the vertices whose eccentricity is equal to $\text{rad}(G)$ are called the center vertices. The diameter of a graph, $\text{diam}(G)$, is the maximum eccentricity over all the vertices in $G$. $K_n$ is the complete graph on $n$ vertices. $K_{m,n}$ denotes the complete bipartite graph where the size of two partitions is $m$ and $n$. An acyclic graph is known as a tree and a vertex of degree 1 in a tree is known as a leaf of the tree. The rest of the tree terminology used is standard [10].

A star graph, $S_n$, is a tree on $n+1$ vertices where one vertex has degree $n$ and the rest of the vertices are leaves. Sometimes a single vertex of a tree is labeled as the root of the tree. In this case the tree is known as a rooted tree. If we remove the root vertex from a rooted tree it decomposes into one or more subtrees; each such subtree along with the root is called a branch, denoted by $B_i$, of original tree. A $k$-ary tree is a rooted tree in which each vertex has no more than $k$ children. A 2-ary tree is
commonly referred to as a **binary tree**. Similarly, an **arm** is the set of vertices that lie on the path from root to a leaf, denoted by $A_i$.

Other classes of graphs will be defined as and when needed.

### 1.3 Decontamination Model Specification

Our decontamination model is a synchronous system. We assume that initially, at time $t = 0$, all vertices in the graph are contaminated. A **decontaminating agent** (henceforth referred to as an agent) is an entity, or a marker, that can be placed on any vertex. A concept similar to this is referred to in the literature as a **pebble** [6]. Assume that at some time step $k$, an agent is at $v \in V$. Then at the next time step, we may move the agent to any of the neighbors of $v$. Vertices visited in this process are marked **decontaminated**, or **disinfected**. Any vertex that the agent is currently placed on is considered to be decontaminated.

A decontaminated vertex can get contaminated by uninterrupted exposure, for a certain amount of time, to a contaminated vertex in its neighborhood. For decontaminated $v$ if there is no agent placed on $v$ but some neighbor of $v$ is contaminated, we say that $v$ is **exposed**. For a decontaminated vertex $v$ we define the **exposure time** of $v$, $\Xi(v)$, as the duration of time $v$ has been exposed. Every time an agent visits $v$, or all vertices in $N(v)$ are decontaminated, we reset $\Xi(v) = 0$. We say that $G$ has temporal immunity $\tau(G)$ if a decontaminated vertex $v \in V$ can only be recontaminated if for uninterrupted $\tau(G)$ time units, there is a neighbor of $v$ (not necessarily unique) that is contaminated and an agent does not visit $v$ during that time period. Note that for any decontaminated vertex $v$ we have that $0 \leq \Xi(v) \leq \tau(G) - 1$.

Given a graph $G$, a temporal immunity $\tau$ and $k$ agents, our goal is to devise a decontamination strategy, which consists of choosing an initial placement for the agents and their movement pattern so that we can reach a state where all the vertices of $G$ are simultaneously decontaminated and we call the graph fully decontaminated. Figure 1 shows an example of a simple graph in the process of being decontaminated.

![Figure 1](image-url)

**Fig. 1** Figure illustrates variation in exposure times of vertices $a$ and $b$ at different time steps as the agent tries to decontaminate $G$, with $\tau(G) = 2$
A strategy is called \textit{monotone} if a decontaminated vertex is never recontaminated and is called \textit{nonmonotone} otherwise. The \textit{immunity number} of $G$ with $k$ agents, $\iota_k(G)$, is the least $\tau$ for which full decontamination of $G$ is possible. It is trivial to see that $\iota_k(G)$ is always finite for $k \geq 1$. In particular for a connected graph $G$ on $n$ vertices, $\iota_k(G) \leq 2(n-1)$ for $k \geq 1$ as the depth first traversal of the spanning tree of $G$ takes exactly $2(n-1)$ steps. However, in this paper we focus on the decontamination of graph by a single agent; this gives us the liberty to use shortened notation $\iota(G)$, and just $\iota$ when the graph is obvious from context, to mean $\iota_1(G)$, the immunity number of a graph using a single agent. In this section we prove bounds on $\iota$ for some simple graphs.

In Sect. 2 we consider spiders and $k$-ary trees in addition to giving asymptotically sharp upper and lower bounds on $\iota(G)$ where $G$ is a mesh graph. Section 3 studies the immunity number of general trees. We also give algorithms to decontaminate several graph topologies. Results are outlined in Table 1. To get a better idea of these concepts, let us begin with examples of some simple graphs.

**Proposition 1** Let $P_n$ be a path on $n$ vertices, then $\iota(P_n) = 0$, for all $n \geq 1$.

Consider the strategy of starting the agent at one leaf vertex and at each time step moving towards the other leaf vertex of the path. Clearly this process will decontaminate the path in $n-1$ time steps. Since none of the decontaminated vertices are exposed at any step, we do not need any immunity. Furthermore this strategy is monotone. A cycle can be decontaminated using a similar strategy.

**Proposition 2** If $C_n$ is a cycle on $n$ vertices, $\iota(C_n) = 2$, for all $n \geq 4$.

**Proof** To see that $\iota(C_n) \leq 2$ set the temporal immunity $\tau = 2$ and begin with the agent at any vertex of the cycle. At $t = 1$ arbitrarily choose one of its neighbors to move to. Henceforth, for $t = k \geq 2$, we always move our agent in a fixed, say counterclockwise, direction. It is straightforward to verify that we will end up with a fully decontaminated graph in at most $2n$ time steps. Note that this is a nonmonotone strategy.

If we set temporal immunity $\tau = 1$ then we will show that in a cycle with at least four vertices we can never decontaminate more than two (adjacent) vertices of the cycle. Suppose that four vertices $v_n, v_1, v_2, v_3$ appear in the cycle in that order. Assume that at time step $t = 0$ the agent is placed at $v_1$ and, without loss of generality, it moves to $v_2$ at the next time step. At $t = 2$ if the agent moves to $v_3$ then $v_1$ becomes contaminated due to its exposure to $v_n$ and we end up with only $v_2$ and $v_3$ decontaminated which is the same as not having made any progress. If, on the other hand, the agent had moved back to $v_1$ at $t = 2$ we would again have ended up with no progress since the agent would still have the same constraints on proceeding to its next vertex, therefore $\iota(C_n) > 1$. \hfill $\square$

The algorithm outlined above runs in $\Theta(n)$ time; in at most $2n$ time steps to be precise. We observe that this constant can be improved by compromising on the temporal immunity. Formally,

**Observation 1** A cycle on $n$ vertices can be decontaminated in $\frac{\tau}{\tau-1}n$ time steps where the temporal immunity $\tau \geq 2$. 

\copyright Springer
Table 1 A summary of our results. Here $i^*$ represents the immunity number of a graph when considering only monotone strategies.

| Graph topology                              | Upper bound on $i$ | Lower bound on $i$ | Lower bound on $i^*$ |
|---------------------------------------------|--------------------|--------------------|----------------------|
| Path $P_n$, with $n \geq 1$                 | 0 (Proposition 1)  | 0 (Proposition 1)  | 0 (Observation 7)    |
| Cycle $C_n$, with $n \geq 4$                | 2 (Proposition 2)  | 2 (Proposition 2)  | $n - 1$ (Observation 8) |
| Complete graph $K_n$, with $n \geq 3$       | $n - 1$ (Theorem 1)| $n - 1$ (Theorem 1)| $n - 1$ (Observation 9) |
| Complete bipartite graph $K_{m,n}$, with $3 \leq m \leq n$ | $2m - 1$ (Theorem 2)| $2m - 1$ (Theorem 2)| $2m - 1$ (Observation 10) |
| Spider graph $S_n$, with $n \geq 1$         | $4\sqrt{n}$ (Corollary 1) | –                 | $\Omega(n)$ (Observation 11) |
| Tree on $n$ vertices                        | $30\sqrt{n}$ (Theorem 6) | –                 | $\Omega(n)$ (Corollary 4) |
| Mesh $p \times q$, with $p \leq q$          | $p$ (Theorem 4)    | $\frac{p}{2}$ (Theorem 5) | $2p - 2$ (Observation 12) |
| General planar graph on $n$ vertices        | $n - 1$ (Theorem 7) | $\Omega(\sqrt{n})$ (Corollary 6) | $\Omega(n)$ (Corollary 5) |
| General graph on $n$ vertices               | $n - 1$ (Theorem 7) | $n - 1$ (Theorem 1) | $n - 1$ (Observation 13) |
Proof It takes \( n \) steps to complete the first cycle. By this time there are \( n/\tau \) contaminated vertices in the cycle. It takes \( n/\tau \) time steps to decontaminate them. Now there are \( n/\tau^2 \) contaminated vertices. If we continue in this fashion we get the following sum for the total time spent:

\[
\sum_{i=0}^{\infty} \frac{n}{\tau^i}.
\]

The conclusion follows. \( \Box \)

Path and cycle happen to be the simplest possible graphs that can be decontaminated easily with optimal constant immunity numbers as seen above. We now consider some dense graphs and show that they may require a much larger value of \( \tau \).

Theorem 1 Let \( K_n \) be a complete graph on \( n \) vertices, then \( \iota(K_n) = n - 1 \) for all \( n \geq 3 \).

Proof Let the vertex set \( V = \{v_1, v_2, \ldots, v_n\} \). Since the graph is fully connected, we can decontaminate \( K_n \) by making the agent visit all the vertices sequentially in any order giving us \( \iota(K_n) \leq n - 1 \).

To see that this bound is actually tight we need to show that temporal immunity of \( n - 2 \) is not good enough for full decontamination. For this purpose set \( \tau = n - 2 \) and suppose that at time step \( t = k \) we have somehow managed to decontaminate all the vertices of \( K_n \) except one last vertex, say, \( v_n \). Assume without loss of generality that the agent is at \( v_{n-1} \). As long as the complete graph is not fully decontaminated, all the vertices which do not have the agent placed on them are exposed. This implies that the vertices \( v_1, \ldots, v_{n-2} \) have all been visited by the agent in the last \( n - 2 \) time steps, that is, \( \Xi(v_i) < n - 2 \) for \( 1 \leq i \leq n - 2 \). It also implies that since there is one agent, all these vertices have different exposure times, meaning that there is one vertex, say \( v_1 \), such that \( \Xi(v_1) = n - 3 \). At time step \( k + 1 \), if the agent moves to \( v_n \) and decontaminates it, then \( v_1 \) becomes contaminated hence we make no progress; there is still one contaminated vertex remaining in the graph. If on the other hand agent \( x \) is moved to \( v_1 \) to avoid its contamination, we will again have not made any progress. Moving the agent to any other vertex at \( t = k + 1 \) actually increases the number of contaminated vertices in the graph. \( \Box \)

The immunity number of complete bipartite graph depends upon the size of the smaller partition.

Theorem 2 Let \( G \) be a complete bipartite graph on the vertex sets \( A \) and \( B \) where \( |A| = m, |B| = n \) such that \( 3 \leq m \leq n \), then \( \iota(G) = 2m - 1 \).

Proof Let \( A = \{a_1, a_2, \ldots, a_m\} \) and \( B = \{b_1, b_2, \ldots, b_n\} \). Set the temporal immunity \( \tau = 2m - 1 \) and place an agent at \( a_1 \) at \( t = 0 \). Now we cycle through the vertices in \( A \) and \( B \) in an interleaved sequence as follows:

\[
a_1, b_1, a_2, b_2, a_3, b_2, \ldots, a_m, b_m, a_1, b_{m+1}, a_2, b_{m+2}, \ldots, b_n.
\]
When \( t < 2m \) none of the vertices are exposed long enough to be recontaminated. At \( t = 2m \) the agent returns to \( a_1 \), and thereafter none of the decontaminated vertices in \( B \) remain exposed while the vertices of \( A \) keep getting visited by the agent before their exposure time reaches \( \tau \). It follows that this monotone strategy fully decontaminates \( G \) in \( 2n - 1 \) time steps.

Our claim is that if \( \tau < 2m - 1 \) then it is not possible to fully decontaminate a partition during any stage of a given decontamination strategy. Consider a strategy that aims to fully decontaminate \( A \) at some point (and \( B \) is never fully decontaminated before that). Suppose that at time \( t = k \) there remains exactly one contaminated vertex in \( A \) (and that there were two contaminated vertices in \( A \) at \( t = k - 1 \)). Note that this implies that the agent is at some vertex in \( A \) at \( t = k \). Since \( B \) has never fully been decontaminated, it follows that there exists a vertex \( a_j \in A \) such that \( \Xi(a_j) = 2m - 3 \). Since it is a bipartite graph, it will take at least two additional time steps to reach the last contaminated vertex of \( A \), and if the temporal immunity is less that \( 2m - 1 \) the agent will fail to decontaminate \( A \) fully.

In the case where the decontamination strategy requires that \( B \) is fully decontaminated before \( A \), similar argument gives us a lower bound of \( \Omega(\gamma(G)) \) but we have already given a strategy that decontaminates \( A \) first which gives a better upper bound.

2 Spiders, \( k \)-ary Trees, and Mesh Graphs

Two important network topologies are star and mesh. They are extreme examples of centralization and decentralization respectively. In the following we study our problem on stars, spiders (the generalization of stars), \( k \)-ary trees, and mesh graphs. Some of the ideas and proof techniques developed in this section will feature again in the proof of the upper bound on immunity number for general trees that will be treated in the next section.

2.1 Spider and \( k \)-ary Trees

Let \( S \) be a star graph. The simple strategy of starting the agent at the center vertex and visiting each leaf in turn (via the center) gives us the bound on \( \gamma(S_n) \). This strategy takes \( 2n + \Delta \) time steps where \( \Delta \) is the maximum degree. Both the bounds thus achieved, on temporal immunity and the time complexity, are optimal.

**Lemma 1** Temporal immunity \( \tau = 1 \) is necessary and sufficient for any star graph.

**Proof** The strategy outlined above gives us the upper bound of \( \gamma(S_n) \leq 1 \). The matching lower bound argument is straightforward and we omit the details. \( \square \)

A spider is a graph that is structurally similar to a star graph. A spider is a tree in which one vertex, called the root, has degree at least 3, and the rest of the vertices have degree at most 2. Equivalently a spider consists of \( k \) vertex disjoint paths all of which have one endpoint that is connected to the root vertex. Such a spider is said to have \( k \) arms.
Let $S$ be a spider such that the degree of the root is $\Delta$. If $m$ is the length of the longest arm of $S$ then it is easy to verify that $\iota(S) \leq 2m$. However one can obtain a better estimate on $\iota(S)$.

**Theorem 3** Let $S$ be a spider on $n$ vertices such that the degree of the root is $\Delta$. If $m$ is the length of the longest arm of $S$ then $\iota(S) \leq \Delta + \sqrt{\Delta^2 + 4m}$.

**Proof** Arbitrarily order the arms of the spider $A_1, A_2, \ldots, A_\Delta$ and let the temporal immunity be $\tau^*$. Thus the agent, when starting from the root, can decontaminate $\tau^*$ vertices on an arm before the exposed root gets recontaminated. Our strategy is going to be an iterative one and in each iteration, we are going to let the root get contaminated just once in the beginning, and after we decontaminate it, we will make sure that it does not get recontaminated during the course of that iteration. At the end of iteration, $j$, we will have decontaminated all the arms of the spider from $A_1$ to $A_j$ along with the root. Since this is going to be a nonmonotone strategy, parts or whole of these arms may be recontaminated during the rest of the algorithm.

At the first iteration we start from the root, traverse $A_1$ to the end and return to the root. We proceed to decontaminate the rest of the spider using the following strategy. At the beginning of $j$th iteration, our agent is at the root of the spider and all the arms from $A_1, \ldots, A_{j-1}$ are fully decontaminated whereas $A_j, \ldots, A_\Delta$ are all fully contaminated (except for the root). The agent traverses each arm of the spider up to the farthest contaminated vertex and returns to the root in sequence starting from $A_j$ down to $A_1$. We will allow the root to get contaminated just once in this iteration, that is, when our agent is traversing $A_j$. We want to fine tune the temporal immunity such that once the agent has returned after visiting all the vertices in $A_j$, the root doesn’t get contaminated again in this iteration. Figure 2 illustrates a step in the general strategy to decontaminate a spider.

There are at most $m$ contaminated vertices in $A_j$. The number of vertices in $A_{j-1}$ contaminated in this process of decontaminating $A_j$ is thus $2m/\tau^*$. Similarly there are at most $2(m + 2m/\tau^*)/\tau^*$ vertices contaminated in $A_{j-2}$ during this iteration. The number of contaminated vertices in $A_1$ is given by $\sum_{i=1}^{j} \left( \frac{m}{(\tau^*/2)^i} \right)$. To make sure that the root is not recontaminated in this iteration we need to satisfy the following inequality

$$\sum_{i=1}^{j} \frac{m}{(\tau^*/2)^i} < \frac{\tau^*}{2}. \quad (1)$$

We claim that $\tau^* = \Delta + \sqrt{\Delta^2 + 4m}$ is good enough.

$$\sum_{i=1}^{j} \frac{m}{(\Delta + \sqrt{\Delta^2 + 4m/2})^i} < \frac{\Delta + \sqrt{\Delta^2 + 4m}}{2}. \quad (2)$$

The first term on the left amounts to at most $\sqrt{m}$ and every other term is less than one. Since the total number of terms is at most $\Delta$ the inequality holds and that completes the proof of the theorem. \hfill \Box

In particular, we have the following corollary:
Fig. 2 The general decontamination strategy for star graphs is presented here. The agent is at the root and the arms $A_1, \ldots, A_6$ are decontaminated, represented as white dots. Dotted line segments show the path followed by the agent in 7th iteration to decontaminate $A_7$.

**Corollary 1** If $S$ is a spider on $n$ vertices then $\iota(S) = O(\sqrt{n})$.

*Proof* Let $S$ be rooted at a vertex $r$. If $\deg(r) = \Delta \leq \sqrt{n}$, if follows from Theorem 3 that $\iota(S) \leq \Delta + \sqrt{\Delta^2 + 4m} \leq 4\sqrt{n}$. Therefore, without loss of generality, assume that $\Delta > \sqrt{n}$.

Let $A_1, A_2, \ldots, A_\Delta$ denote the arms of $S$ with $|A_i| \leq |A_j|$, for all $i \leq j$. Again without loss of generality there are at most $\sqrt{n}$ vertices in each of the first $k$ arms and more than $\sqrt{n}$ vertices in the last $\Delta - k$ arms for some $0 < k < \Delta$. Consider a modified spider $S^* = S \setminus \bigcup_{1 \leq i \leq k} A_i$ with $r$ as root. By the pigeon hole principle, $\Delta(S^*) \leq \sqrt{n}$. So we can apply the technique used in the proof of Theorem 3 to decontaminate $S^*$ with $\tau \leq 4\sqrt{n}$. Once $S^*$ is decontaminated, we use the following lemma to decontaminate $(S \setminus S^*) \cup \{r\}$. The bound will follow because the height of this tree is less than $\sqrt{n}$ and an already decontaminated $S$ never gets recontaminated by the virtue of monotonicity in Lemma 2.

**Lemma 2** Any $k$-ary tree $T$ with height $h$ can be decontaminated with $\tau = 2h - 1$ using a monotone algorithm.

*Proof* First label the leaf vertices of $T$ so that $\ell_1, \ell_2, \ell_3, \ldots$ represents the order in which the leaves are visited if an in-order depth-first traversal is performed on $T$, starting from the root vertex. It is straightforward to verify that if we start with the agent at the root, and visit each leaf in order $\ell_1, \ell_2, \ell_3, \ldots$ returning to the root every time before visiting the next leaf, then $\tau = 2h - 1$ would be enough to decontaminate the entire $k$-ary tree. Note further that any leaf $\ell_i$ is never exposed after decontamination, and all non-leaf vertices, once decontaminated, are exposed for at most $2h - 1$ time units. Monotonicity follows.

The proof of Lemma 2 completes the proof of Corollary 1.

A perfect or full $k$-ary tree is a rooted tree where every vertex has $k$ children except the leaves. We observe the following corollary of Lemma 2.
Corollary 2 Let T be a perfect k-ary tree on n vertices, then \( \tau(T) = O(\log n) \). □

In case of a binary tree the bound on temporal immunity given by Lemma 2 can be slightly improved if we use the above strategy to first fully decontaminate the subtree rooted at the left child of the root, and then use the same method to decontaminate the subtree rooted at the right child of the root. Thus,

Observation 2 A binary tree with height h can be decontaminated with a temporal immunity of 2h − 3. □

2.2 Decontaminating a Mesh

A \( p \times q \) mesh is a graph with \( pq \) vertices arranged in a grid structure. It is convenient to work with planar drawing of the mesh graph where the vertices of \( G = (V, E) \) are embedded on the points with the integer coordinates of Cartesian plane. The vertices are named \( v(i,j) \) for \( 1 \leq i \leq q, 1 \leq j \leq p \) corresponding to their coordinates in the planar embedding. There is an edge between a pair of vertices if their euclidean distance is exactly 1. We can partition \( V \) into the column sets \( C_1, C_2, \ldots, C_q \) so that \( C_i = \{v(i,j) : 1 \leq j \leq p\} \) for all \( 1 \leq i \leq q \). Row sets \( R_1, R_2, \ldots, R_p \) are defined analogously, i.e., \( R_j = \{v(i,j) : 1 \leq i \leq q\} \).

We observe that the decontamination strategies proposed in [7] can be adapted to decontaminate a mesh graph with temporal immunity of \( 2p - 1 \). However, once again we can improve this bound by resorting to a nonmonotone strategy.

Theorem 4 Let \( G \) be a \( p \times q \) mesh where \( p \leq q \). Then \( \tau(G) \leq p \).

Proof We will describe a strategy (illustrated in Fig. 3) to decontaminate \( G \) non-monotonically. However, once we declare a column to be secure, we do not allow any of its vertices to be contaminated again.

Set the temporal immunity \( \tau = p \) and start with the agent at \( v(1,1) \). Proceed all the way up to \( v(1,p) \), move the agent to the next column onto \( v(2,p) \). Traverse down the column until we reach the vertex \( v(2,\lfloor \frac{p}{2} \rfloor + 1) \). Note that the vertices of \( C_1 \) had started getting recontaminated when the agent reached \( v(2,p - 1) \) because the exposure time of \( v(1,1) \) became equal to \( \tau \) at that point. Now move the agent back to \( C_1 \) onto \( v(1,\lfloor \frac{p}{2} \rfloor + 1) \) and proceed all the way down back to \( v(1,1) \). We declare that \( C_1 \) has been secured and none of its vertices will be recontaminated henceforth. It is pertinent to note that at this point, \( \Sigma(v(2,p)) = \tau - 1 = p - 1 \).

To decontaminate the rest of the columns we use the following scheme. Assume that we have declared all the columns \( C_1, C_2, \ldots, C_k \) to be secured and our agent is at \( v(k,1) \). We also know that \( \Sigma(v(k+1,p)) = \tau - 1 \). We move the agent to the next column onto \( v(k+1,1) \). At this point \( v(k+1,p) \) becomes contaminated leaving \( v(k,p) \) exposed. We follow the same strategy as the one that we followed when we were decontaminating \( C_1 \). We move the agent all the way up to \( v(k+1,p) \), move to \( C_{k+2} \), traverse all the way down to \( v(k+2,\lfloor \frac{p}{2} \rfloor + 1) \), revert back to \( C_{k+1} \) and move back down to \( v(k+1,1) \) declaring column \( C_{k+1} \) to be decontaminated. None of the vertices in \( C_k \) will be recontaminated since \( v(k,p) \) had the maximum exposure time due to \( v(k+1,p) \), and we were able to...
decontaminate $v_{(k+1, p)}$ before $v_{(k, p)}$ got contaminated. Similarly, it is not difficult for the reader to verify that none of the rest of the vertices of $C_k$ are exposed long enough to be recontaminated. 

**Corollary 3** Let $G$ be a mesh on $n$ vertices, then $\iota(G) \leq \sqrt{n}$. 

The algorithm in the proof of Theorem 4 takes $2n$ time units. But, 

**Claim 3** A $p \times q$ mesh on $n$ vertices can be decontaminated in $n$ steps with temporal immunity $2p$ where $p \leq q$. 

The claim 3 follows from the strategy outlined in [8].

**Remark 1** Strategy used in proof of Theorem 4 can also be used to decontaminate a cylinder graph (a mesh graph with an edge between the leftmost and the rightmost vertices on each row).

In the following we present an asymptotically sharp lower bound for mesh graphs, but first we would like to establish a graph isoperimetric result that we use in proof of lower bound.

**Lemma 3** Let $G = (V, E)$ be an $\sqrt{n} \times \sqrt{n}$ mesh graph, then for any $W \subset V$, $|W| = \frac{n}{2}$, size of a maximum matching between $W$ and its complement has size at least $\sqrt{n}$.

**Proof** For ease of understanding let us say that a vertex is colored white if it is in set $W$, and black otherwise. An edge is monochromatic if both its endpoints have the same color, and bichromatic otherwise. Let $R_1, R_2, \ldots, R_{\sqrt{n}}$, and $C_1, C_2, \ldots, C_{\sqrt{n}}$ be the row and column sets respectively. We observe following four possible cases:
Case 1 For each row $R_i$, $0 < |R_i \cap W| < \sqrt{n}$:
Since $R_i$ contains vertices of both colors, it is clear that there will be at least one bichromatic edge. We pick one such edge from each $R_i$. As these edges are disjoint, we have a matching of size at least $\sqrt{n}$.

Case 2 There exist two rows $R_i, R_j$, such that $|R_i \cap W| = 0$, and $|R_j \cap W| = \sqrt{n}$:
We interchange the roles of rows and columns. The claim then follows from Case 1.

Case 3 There exists a row $R_i$, such that $|R_i \cap W| = 0$, and for every row $R_j$, $|R_j \cap W| \neq \sqrt{n}$:
We present below a scheme to match vertices in this case.

We will use two markers $b$ (for bottom row), and $c$ (for current row). In the beginning, both point to the first row of the mesh, i.e., $b := c := 1$.

1. Locate the minimum $x \geq c \geq b$, such that $R_x \cap W = \emptyset$. If we can not find such an $x$, go to Step 3.
   (a) Now locate the maximum $y < x \leq b$, such that $|R_y \cap W| \geq x - y + 1$. For all the white vertices in $R_y$ we have black vertices in corresponding columns of $R_x$. So for each such column there exists a pair of rows $R_i, R_{i+1}$ with a bichromatic edge in that column where $y \leq i < x$. We set $c := x + 1$, and call rows $R_j$ matched if $y \leq j \leq x$.
   (b) If we cannot find such a $y$, then we look for a minimum $z > x$, such that the row corresponding to $z$ contains at least $z - x + 1$ points from $W$, i.e., $|R_z \cap W| \geq z - x + 1$. For all the white vertices in $R_z$, we can find bichromatic edges as above. We set $c := z + 1$ and $b := x + 1$ and we call rows $R_j$ matched if $x \leq j \leq z$.

2. Repeat Step 1. Failure to find both $y$ and $z$ at any step would imply a contradiction because there are not enough black vertices as assumed. In the worst case

\[
\left| \bigcup_{i=1}^{x-1} R_i \right| \leq (x - 1) \frac{\sqrt{n}}{2}
\]

(alternating complete black and white rows), and

\[
\left| \bigcup_{j=x+1}^{\sqrt{n}} R_j \right| < (\sqrt{n} - (x + 1)) \frac{\sqrt{n}}{2}.
\]

3. Match all unmatched rows as in Case 1.

Case 4 There exists a row $R_i$, such that $|R_i \cap W| = \sqrt{n}$ and for every row $R_j$, $|R_j \cap W| \neq 0$:
The claim in this case follows directly from the proof of Case 3 by reversing the roles of $W$ and $\overline{W}$.

This concludes the proof of Lemma.

Note that bound in Lemma 3 is tight when $W$ is a rectangular subgrid. We do not know of a tight example which is not rectangular in shape. We observe that since
\[\Delta = 4\] for mesh, Lemma 3 also follows from vertex and edge isoperimetric inequalities proved in [3, 4] up to a constant factor.

**Theorem 5** If \(G\) is a \(p \times q\) mesh with \(p \leq q\), then \(\iota(G) > \frac{p^2}{2}\).

**Proof** Let us assume the opposite, i.e., a decontaminating algorithm does exist with \(\tau = \frac{p^2}{2}\). For simplicity assume that \(G\) is a \(p \times p\) mesh and ignore the agent’s moves for the remaining \(p \times (q - p)\) vertices if any. Let \(n = p^2\), then at some time step during this algorithm we will have exactly \(\frac{n^2}{2}\) decontaminated vertices. Lemma 3 implies that at this stage at least \(p\) vertices of \(G\) are exposed through at least \(p\) disjoint edges to contaminated vertices. At least half of these vertices will get contaminated in the next \(p/2\) time units regardless of the agent’s moves. It follows that no decontaminating algorithm exists with assumed temporal immunity. \(\square\)

### 3 General Trees

To upper bound \(\iota\) for general trees, we will try to adapt the strategy used to decontaminate \(k\)-ary trees.

Assuming \(T\) to be rooted at a center vertex (choose one arbitrarily if there are two center vertices) visit each of the leaves of \(T\) in the depth-first search order, every time returning to the center vertex, as in the case of spider. In the worst case this approach may require a temporal immunity \(\tau = 2 \cdot rad(T) = diam(T) + 1\) to fully decontaminate \(T\) but the diameter of a tree on \(n\) vertices can easily be \(O(n)\). We will use nonmonotonicity to our advantage by letting a controlled number of vertices get recontaminated so that we get a much stronger bound even for trees with large diameters. But first we present a key lemma.

**Lemma 4** Any rooted tree \(T\) with height \(h\) can be decontaminated, monotonically, with temporal immunity \(\tau \geq \alpha h\), in \(cn\) time step, where \(n\) is the number of vertices in \(T\), and \(c \leq 4 \frac{\alpha^2/2}{\alpha^2/2 - 1}\) for any positive number \(\alpha > 2\).

**Proof** Assuming an arbitrary tree with height and immunity as above, we present an algorithm with the claimed time complexity. Informally the plan is to group vertices into clusters of “appropriate” sizes and decontaminate each cluster by performing a depth-first search traversal while keeping the path from each cluster to the root decontaminated. The details of this scheme follow.

Define level of a vertex as the distance between the root and the vertex. Let \(l\) be the maximum integer such that there exists a subtree rooted at level \(l\) with more than \(z\) (to be fixed later) vertices. Let \(P_1\) be one such subtree rooted at a vertex \(p_1\) at level \(l\). By maximality of \(l\) the subtrees \(S_i\) for \(1 \leq i \leq k\) rooted at the children \(s_1, s_2, \ldots, s_k\) of \(p_1\) have less than \(z\) vertices each. Now let \(j\) be the largest integer such that \(|S_1 \cup \cdots \cup S_j \cup \{p_1\}| < z\), we define \(X_1 := S_1 \cup \cdots \cup S_j \cup \{p_1\}\). We similarly define \(X_2, \ldots, X_u\), as maximal subtrees all rooted at \(p_1\) making sure that \(z/2 < |X_i| < z\), with possible exception of \(X_u\) which might be of smaller cardinality. Figure 4 shows an example of such a grouping.

As the first step of the decontamination process, the agent starts at the root of \(T\), walks its way to \(p_1\) and performs a depth-first search traversal on each \(X_i\) one by one.
The next step is to walk up a step to the parent of \( p_1 \), say \( p_2 \) and recurse. Let \( P_2 \) be the subtree rooted at \( p_2 \). Arbitrarily choose any subtree \( R \subseteq P_2 \setminus P_1 \), at minimum possible distance from \( P_2 \) (e.g., potentially \( R = P_2 \)), with the property that for all subtrees \( R_i \) of \( R \), \( |R_i| < \tau \) as before. We will group \( R_i \)'s into \( X_j \)'s as before but this time after performing a depth-first search traversal on each \( X_j \), we will pay a visit to \( p_2 \), making sure it remains decontaminated. Once \( P_2 \) is decontaminated, we proceed to \( p_3 \) the parent of \( p_2 \) and repeat the process until \( p_j \) becomes the root of \( T \), and we are done with the decontamination process. If we fix \( \tau := (h \frac{\alpha}{2} - h) \), it is easy to see that \( \tau = ah \) is enough for the decontamination process. We can always group any tree into at most \( 2^n \) such subtrees each of the required size. Note that this clustering can be performed in linear time as follows. For each vertex \( v \) define a variable \( size(v) \) that counts the number of vertices in the subtree rooted at \( v \). The value of \( size(v) \) for all the vertices in the tree can be computed using depth first search algorithm. Now we run the following recursive algorithm to cluster the vertices. Initial input to this routine is the root of the tree.

\[
\text{CLUSTER}(\text{head})
\]

- Define two lists \textit{small\_trees} and \textit{clusters}.
  - For every child of \textit{head}.
    - If \( size(\text{child}) \) is more than \( \tau \) then recursively call CLUSTER(\text{child}). Add the clusters returned by the recursive call to the \textit{clusters} list.
    - Otherwise put the child in the \textit{small\_trees} list.
  - Define a new list \textit{cluster}. The \( size(\text{cluster}) \) is defined as the number of vertices in the cluster.
  - For all vertices \textit{vertex} in the \textit{small\_trees} list.
    - If \( size(\text{cluster}) + size(\text{vertex}) \) is less than \( \tau \) then add all the vertices in the subtree rooted at \textit{vertex} to \textit{cluster}.
    - Otherwise add \textit{cluster} to the \textit{clusters} list and define new empty \textit{cluster}.
  - Add the last \textit{cluster} to the \textit{clusters} list and return \textit{clusters}.

![Figure 4](image_url)  
**Fig. 4** Example illustrating grouping of \( S_j \) into \( X_j \)
Each vertex is visited at most once in the above routine, therefore the procedure can be completed in $O(n)$ time steps. Now $S$ can be found and labeled in linear time. The agent spends at most $2z$ time units on depth-first traversal and $2h$ time units on visiting some $p_j$ potentially at distance $h$ for each such subtree. The total amount of time spent in the process is

$$
\leq 2n \times 2(z + h)
\leq 2n \times 2 \left( q + \frac{q}{\alpha - 2} \right)
= 2n \times 2 \left( 1 + \frac{1}{\alpha - 2} \right)
= 4n \times \frac{\alpha - 1}{\alpha - 2}
$$

\[\square\]

**Theorem 6** Let $T$ be a tree on $n$ vertices, then $\iota(T) = O(\sqrt{n})$.

**Proof** Now let $r$ be a center vertex of $T$ and let $k$ be the number of leaves in $T$. Recall that an arm $A_i$ is a set of vertices that lie on the path from $r$ to a leaf $\ell_i$ for all $1 \leq i \leq k$. Given a tree $T$, a vertex $v$, and an integer we denote by $T_x(v)$ a subtree of $T$ that is obtained by removing all vertices from $T$ that are at distance more than $x$ from $v$, e.g., $T_x(r)$ is $T$ truncated at depth $x$. Assume without loss of generality that leaves $\ell_i$ are sorted in the depth-first order. This implies an ordering on arms $A_i$. Note that $A_i \setminus \{r\}$ are not disjoint in general. We proceed as below.

- For $i = 1$ to $k$
  - Perform an auxiliary step and apply Lemma 4 on $T_{2\sqrt{n}}(r)$ with $\alpha = 3$.
  - Move the agent from $r$ towards leaf $\ell_i$ performing auxiliary decontamination steps Lemma 4 on $T_{10}\sqrt{n}(v_j)$ rooted at $v(j)$ if and only if
    - $\deg(v_j) > 2$, and
    - $v_j$ is at a distance more than $5\sqrt{n}$ from the last vertex we did auxiliary decontamination on.

To analyze this algorithm, we find the following definition useful.

**Definition 1** A vertex $v$ in tree $T$ is called secured at some time step $i$, if it never gets contaminated again.

The agent decontaminates a new arm $A_i$ in the $i$th iteration of the algorithm. We observe that

**Lemma 5** The following invariants hold for every step of the algorithm:

1. Root $r$ is secured at iteration 1.
2. For any secured vertex $v$, and a contaminated vertex $w$, which is in the same branch as $A_i$, $\text{dist}(v, w) > \sqrt{n}$ at the start of iteration $i + 1$.
3. All vertices $v_j \in A_i$ are secured at the start of iteration $i + 1$. 
Proof We fix $\tau = 30\sqrt{n}$. Let $\Gamma(i)$ be the time spent by the algorithm at iteration $i$. Then $\Gamma(i)$ can be broken down into three parts: (1) the time spent performing auxiliary decontamination at $r$, (2) the time spent visiting $\ell_i$, and (3) time spent at each auxiliary step on the way to $\ell_i$, which is less than $12 \cdot |T_{10\sqrt{n}}(v_j)|$ where $v_j$ for each vertex $v_j$ we used in the auxiliary step. We have

$$\Gamma(i) \leq 12n + n + 12\sum_{v_j} |T_{10\sqrt{n}}(v_j)|$$

$$\leq 12n + n + 24n$$

$$= 37n,$$

where we use the fact that $\sum_{v_j} |T_{10\sqrt{n}}(v_j)|$ cannot be more than twice the number of total vertices, since each vertex is in at most two such truncated trees. Since $\tau = 30\sqrt{n}$, after performing an auxiliary decontamination step on tree with $2\sqrt{n}$ height, it takes $\geq 60n$ for the contamination to reach the root which is less than the time spent in one iteration.

This gives the first invariant (i).

Now a vertex $v$ is secured only if $v \in A_j$ for some $j \leq i$. If $v$ lies in a different branch than the one $A_i$ lies in then invariant (ii), and (iii) follow from (i) i.e. if $r$ is secured then contamination has no way to spread from one branch to another, and if a branch has been decontaminated, it will not get recontaminated. For any contaminated vertex $w$, $\text{dist}(r, w) > \sqrt{n}$ implies that $\text{dist}(v, w) > \sqrt{n}$, for any $v$ in a fully decontaminated branch. So we assume without loss of generality that $v$ lies in the same branch as $A_i$. By the order in which we decontaminate leaves, it is clear that after iteration $i$, the closest secured vertex to any contaminated vertex, lies in arm $A_i$. A direct consequence of performing auxiliary decontamination during iteration $i$ is that any contaminated vertex is at distance more than $5\sqrt{n}$ from closest $v \in A_i$. When we have completed iteration $i$, it is still more than $4\sqrt{n}$ distance away, and this implies invariant (ii).

For any contaminated vertex $w$, any $u \in A_i$, and any $v \in A_j$ with $j < i$, all contained in the same branch it holds that $\text{dist}(u, w) < \text{dist}(v, w)$. It follows that $v \in A_j$ for $j < i$ never get contaminated during decontamination process of this branch. This along with (i) implies (iii).

The claim completes the proof of Theorem with $\iota = 30\sqrt{n}$. 

It is easy to see that

**Claim 4** The algorithm in the proof of Theorem 6 to decontaminate general trees is bounded by $\min(O(\Delta \cdot n), O(n^{1.5}))$.

However, trivially

**Claim 5** A tree can be decontaminated in $2n$ time steps with $\tau = n - 1$.

So general trees are the only case where we gain in the asymptotic time complexity of the strategy if we allow more temporal immunity than is actually required for the strategy with the best known bound on the immunity.
4 Monotone Strategy Bounds

For comparative purposes we present some observations and results on monotone decontamination strategies on different classes of graphs. We will use the notation $\iota^*(G)$ to represent the immunity number of a graph $G$ when considering only monotone strategies.

**Observation 6** At any time step during a monotone decontamination strategy, each exposed vertex, $v$, of a graph $G$ has a distinct exposure time such that

$$\text{dist}(v, v_a) - 1 \leq \Xi(v) \leq \tau(G) - 1$$

where $v_a$ is the vertex where the agent is currently positioned.

**Proof** Since it is a monotone strategy once the agent leaves a vertex, it is either exposed immediately or is never exposed to start with. If the vertex is exposed, one should note that the exposure time increases by one unit at every time step until the vertex eventually becomes unexposed due to decontamination of all of its neighboring vertices or the exposure time is reset to zero due to the agent visiting the vertex again. So in effect the exposure time of a vertex $v$ is actually measuring the period of time the agent last visited the exposed vertex. Finally note that an agent can be at only one vertex at a single time step. The conclusion follows. \(\square\)

It is easy to verify the following observation on paths.

**Observation 7** Let $P_n$ be a path on $n$ vertices, then $\iota^*(P_n) = 0$, for all $n \geq 1$. \(\square\)

Requiring monotonicity pushes the immunity number of a cycle, $C_n$, from 2 to $n - 1$.

**Observation 8** Let $C_n$ be a cycle on $n$ vertices, then $\iota^*(C_n) = n - 1$, for all $n \geq 3$.

**Proof** Let $C_n$ be a cycle on $n$ vertices labeled consecutively as $v_1, v_2, \ldots, v_n$. It is straightforward to see that $\tau = n - 1$ is sufficient to decontaminate a cycle on $n$ vertices monotonically by just moving the agent, say counterclockwise, at each time step until all the vertices have been visited. To show that $\tau = n - 2$ is not enough for monotone decontamination assume that at time step $k$ the agent has just managed to decontaminated all but one vertex, $v_n$, of the cycle and the agent is currently at $v_{n-1}$. Note that by virtue of the monotonicity constraint decontaminated vertices always form a connected path in $C_n$. This fact, together with Observation 6, implies that $\Xi(v_n) = n - 3$. At the next time step $k + 1$ the vertex $v_1$ will get recontaminated regardless of the move that the agent makes. \(\square\)

The following two observations follow directly from the decontamination strategies illustrated in the proofs of their nonmonotone counterparts.

**Observation 9** Let $K_n$ be a complete graph on $n$ vertices, then $\iota^*(K_n) = n - 1$, for all $n \geq 3$. \(\square\)

**Observation 10** Let $K_{m,n}$ be a complete bipartite graph on $m + n$ vertices where $3 \leq m \leq n$, then $\iota^*(K_{m,n}) = 2m - 1$. \(\square\)
The spider graph shows a quadratic increase in the required temporal immunity if the agent is restricted to using a monotone decontamination strategy.

Observation 11 Let $S$ be a general spider graph on $3n + 1$ vertices, then $\iota^*(S) = \Theta(n)$.

Proof To see that $\iota^*(S) = O(n)$ assume that the degree of the root is $\Delta$. If $m$ is the length of the longest arm of $S$ then using a naive strategy of visiting each arm sequentially, starting at the root and traversing each arm to the end and returning to the center shows that temporal immunity $\tau = 2m (= O(n))$ is enough to fully decontaminate $S$.

To lower bound the required temporal immunity using monotone strategy, let the root vertex be $r$, with degree 3, connected to arms $A_1$, $A_2$ and $A_3$ each consisting of $n$ vertices. Let $a_1$, $a_2$ and $a_3$ be the end vertices (farthest from $r$) of the arms $A_1$, $A_2$ and $A_3$ respectively. Any monotone strategy will decontaminate the end vertices in some order and without loss of generality let that order be $a_1$, $a_2$, $a_3$. When the agent decontaminates $a_2$ we note that by virtue of the monotonicity constraint the only vertices in $S$ that are still contaminated are $a_3$ and possibly a connected subpath of $A_3$ extending from the vertex $a_3$ towards the root. Let $v$ be the vertex exposed by some contaminated vertex of $A_3$. This $v$ may be the root itself or a vertex of $A_3$ only. It is clear that the immunity needs to be at least $n$ for the agent to reach $v$ from $a_2$ before $v$ is recontaminated. In fact, if we consider the time it took the agent to reach $a_2$ after passing through $r$ we can conclude that we have a matching lower bound of $2n (= 2m)$ on $\iota^*(S)$. $\square$

Corollary 4 Let $T$ be a general tree on $n$ vertices, then $\iota^*(T) = \Omega(n)$. $\square$

Corollary 5 There exist planar graphs on $n$ vertices such that $\iota^* = \Omega(n)$. $\square$

Using notation from Sect. 2.2 for vertex labeling we present a simple approach to monotonically decontaminate a $p \times q$ mesh, where $p \leq q$. We start with our agent at $v(1,1)$ at $t = 0$, proceed to visit all vertices in the column till we reach $v(1,p)$, move right one step to $v(2,p)$ and proceed all the way down to $v(2,1)$. This process may now be continued by moving the agent to $v(3,1)$ and going on to decontaminate the entire graph column by column until we reach the last vertex. Clearly a temporal immunity of $2p - 1$ is enough for this strategy to decontaminate the entire graph. In [7] the same strategy was used, albeit under a slightly different notion of immunity, to get a similar upper bound. The following lower bound for monotone decontamination on a mesh is almost optimal.

Observation 12 Let $G$ be a $\sqrt{n} \times \sqrt{n}$ mesh, then $\iota^*(G) \geq 2\sqrt{n} - 2$.

Proof Let us consider the corner vertices of the mesh. During any monotone strategy the agent will decontaminate these four vertices in some order. Without loss of generality assume that this order is $v_1$, $v_2$, $v_3$, $v_4$. Let $C_1$ be the set of vertices in the same column as $v_1$. Similarly $R_1$ is defined to be the set of vertices in the same row as $v_1$. Suppose that the agent has decontaminated $v_2$ at time step $k$. This second corner vertex, $v_2$, will be an element of at most one of the sets $R_1$ and $C_1$. Once again, without
loss of generality assume that $v_2 \notin R_1$. Then there will be a vertex, $v_x$ in $R_1$ that is exposed and $\mathcal{E}(v_x) \geq \sqrt{n} - 2$ by Observation 6. To avoid recontamination the agent needs to visit $v_x$ or its contaminated neighbors which it cannot do so before at least another $\sqrt{n} - 1$ (the shortest possible distance between $v_2$ and $v_1$ or a neighbor of $v_1$) time steps. Hence the temporal immunity needs to be at least $2\sqrt{n} - 2$.  

Lastly, by Observation 9:

**Observation 13** Let $G$ be a general tree on $n \geq 3$ vertices, then $\iota^*(G) \geq n - 1$.  

**5 Discussion**

In this paper we refined the decontamination with immunity model proposed in [13] and presented new or improved bounds for immunity number of common network topologies along with the decontamination algorithms. The time efficiency of these algorithms was not a focus of this work but we do note that most of the schemes presented here can be easily implemented in linear time. The relation between the temporal immunity and the time complexity requirement is worth studying further. Another pertinent computational question might be to check whether a given graph can be decontaminated with immunity $k$; we do not expect a polynomial time algorithm for this problem.

This paper leaves open the challenge to close the gap between upper and lower bounds on the immunity number of general trees. We showed that for any tree $T$, $\iota(T) = O(\sqrt{n})$. Using a somewhat involved argument, it can be shown that there exist trees $T$ on $n$ vertices for which $\iota(T) = \Omega(n^{\frac{1}{3} + \epsilon})$ for some constant $\epsilon > 0$.

Another interesting topology to consider for this problem is that of planar graphs. It follows directly from Theorem 5 that

**Corollary 6** There exist planar graphs on $n$ vertices with immunity number $\iota > \frac{\sqrt{n}}{2}$.

We believe that this is actually the true bound up to a constant factor.

**Conjecture 1** Any planar graph $G$ on $n$ vertices can be decontaminated with $\tau(G) = O(\sqrt{n})$.

A similar bound for a slightly different problem of bounding the search number of a graph, lends some credence to the above conjecture. Search number, $s(G)$, is the minimum number of agents needed to decontaminate a graph with $\tau = 0$. Following was proved in [1], but we note that proof we present here is simpler, shorter, and more intuitive.

**Theorem 14** Any planar graph $G = (V, E)$ on $n$ vertices can be decontaminated with $s(G) = O(\sqrt{n})$ agents where vertices of $G$ don’t have any immunity.

**Proof** We partition $V$ into three sets $V_1$, $V_2$, and $S$ using Planar Separator Theorem [18], where $|V_1| \leq \frac{2n}{3}$, $|S| \leq 3\sqrt{n}$, owing to improvements in [11], and for any $v \in V_1$, and any $w \in V_2$, edge $vw \notin E$. We place $3\sqrt{n}$ agents on $S$ to make sure that contamination cannot spread from $V_1$ to $V_2$, or vice versa. Let $G_1 = (V_1, E_1)$ be the
subgraph of $G$ where an edge of $E$ is in $E_1$ if both its endpoints are in $V_1$. Similarly, define $G_2 = (V_2, E_2)$. Now let’s say it takes $s(G_1)$ agents to decontaminate $G_1$, once $G_1$ is fully decontaminated, we can reuse all those agents to decontaminate $G_2$. Since both $G_1$ and $G_2$ are also planar graphs, this gives us an obvious recurrence for $s(G)$:

$$s(G) \leq \max(s(G_1), s(G_2)) + 3\sqrt{n}$$

So the total number of agents required is at most $3\sqrt{n} + 3\sqrt{\frac{2n}{3}} + \ldots = O(\sqrt{n})$ which completes the proof. \hfill $\square$

Technique used in proof of Theorem 14 may help devise a similar proof for the conjectured bound on immunity number of planar graphs. In any case, we do believe that the Planar Separator Theorem may be beneficial in that case as well.

The current upper bound on $\tau(G)$ of a planar graph $G$ is implied from the following theorem.

**Theorem 7** Any connected graph $G = (V, E)$ on $n$ vertices can be decontaminated with $\tau = n - 1$.

**Proof** For an arbitrary graph $G$ we present a strategy to decontaminate $G$ with the claimed immunity. Our strategy is a modified depth first search traversal of the graph.

Start with an agent on an arbitrary vertex $v_1$, and at each time step keep walking the agent to successive *unvisited* neighbors in the depth first fashion. If we exhaust all $V$ then we are done since we visited all vertices before first vertex got recontaminated. Otherwise the agent follows some path $v_1, \ldots, v_{k-1}, v_k$ such that all neighbors of $v_k$ have already been visited. We label a vertex as a *terminal* vertex and plan never to visit it again. So for the rest of the decontamination process, we will assume that $v_k$ does not exist. We traverse the agent back along $v_k, v_{k-1}, \ldots, v_1$ to reach $v_1$, and then come back along same path to reach $v_{k-1}$. This time the agent moves to some other neighbor of $v_{k-1}$ if any, and continue as before either finding another another terminal vertex and deleting it too or finding a cycle on rest of the vertices. In either case, process completes in finite time. Since the agent decontaminated the terminal vertices, they cannot contaminate any other vertex after they have been visited. And since, every time the agent encounters a terminal vertex it goes back to $v_1$, and visits all its neighbors (all of which lie on agent’s path back to $v_1$) in the next less than $n - 1$ steps, the terminal vertices cannot get contaminated again. Vertices that are not terminal are decontaminated at the end of the process because they are visited in the traversal on cycle which takes at most $n - 1$ steps after we leave $v_1$. The claim follows. \hfill $\square$

On the other hand this might tempt one to conjecture that $\iota(G)$ is an increasing graph property, i.e., if we add new edges to $G$ then the immunity number can only go up. But this is not the case as illustrated in Fig. 5.

**Observation 15** Immunity number is not an increasing graph property. [14]

**Proof** Consider the following counter-example: let $G$ be a spider with $2\sqrt{n}$ arms labeled $A_1, A_2, \ldots, A_{2\sqrt{n}}$, where $|A_i| := \sqrt{n} - 1$ when $i$ is even; otherwise $A_i$ has just one vertex.
Now construct $G^*$ by adding edges $vw$ where $v \in A_i$, $w \in A_{i+1}$, for all $i \equiv 1 \pmod{2}$ then we can decontaminate $G^*$ with $\tau(G^*) = 2$.

Now we show that $\iota(G) > 2$. Let us assume otherwise, i.e. there is a strategy to decontaminate $G$ with $\tau = 2$. We will only consider agent’s moves in the longer arms with $\sqrt{n} - 1$ vertices each. At some time step $t$ the agent must have decontaminated exactly half of the vertices in these arms. There are two cases:

(i) At least a quarter of the arms are completely decontaminated and at least a quarter of the arms are completely contaminated. In this case the agent cannot move further away than the vertices that are at distance 3 from the root before decontaminating all the vertices that are adjacent to the root. If the agent goes up to a distance $k$ then by the time the agent returns to the root, the root would have contaminated at least $\sqrt{n} \cdot \frac{k}{4}$ vertices. Furthermore, the agent cannot decontaminate more than 3 vertices adjacent to the root that lie in the completely contaminated arms. In any case no more than $\frac{n^2}{4} + 3$ vertices can be decontaminated in this case.

(ii) At least a quarter of the arms have both contaminated and decontaminated vertices. In the next 3 time steps the agent can decontaminate at most 3 vertices but at least $\sqrt{n} - 3$ decontaminated vertices, exposed in the arms mentioned above, will be contaminated. Hence the agent makes no progress in this case as well.

The conclusion follows. $\Box$

Other interesting problems related to the topics covered in this paper include natural generalizations of the problem to directed graphs and weighted graphs. One may also investigate the behavior of immunity number of random graphs.

Acknowledgments We are grateful to Devendra Desai, Jeff Kahn, and William Steiger for their insightful comments and discussions. The work of the third author was partially supported by NSF Grant 0944081.

References

1. Alon, N., Mehrabian, A.: Chasing a fast robber on planar graphs and random graphs. J. Graph Theory (2014)
2. Barrière, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Capture of an intruder by mobile agents. In Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 200–209. ACM (2002)
3. Bollobás, B., Leader, I.: Compressions and isoperimetric inequalities. J Comb. Theory Ser. A 56(1), 47–62 (1991)
4. Bollobás, B., Leader, I.: Edge-isoperimetric inequalities in the grid. Combinatorica 11(4), 299–314 (1991)
5. Breisch, R.: An intuitive approach to speleotopology. Southwest Cavers 6(5), 72–78 (1967)
6. Chung, F.: Pebbling in hypercubes. SIAM J. Discrete Math. 2(4), 467–472 (1989)
7. Daadaa, Y.: Network Decontamination with temporal immunity. Ph.D. thesis, University of Ottawa (2012)
8. Daadaa, Y., Flocchini, P., Zaguia, N.: Network decontamination with temporal immunity by cellular automata. In Cellular Automata, pp. 287–299. Springer (2010)
9. Daadaa, Y., Flocchini, P., Zaguia, N.: Decontamination with temporal immunity by mobile cellular automata. In International Conference on Scientific Computing (CSC), pp. 172–178 (2011)
10. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2010)
11. Djidjev, H.N.: On the problem of partitioning planar graphs. SIAM J. Algebr. Discrete Methods 3(2), 229–240 (1982)
12. Flocchini, Paola, F.L., Song, L.X.: Size optimal strategies for capturing an intruder in mesh networks. In Proceedings of the International Conference on Communications in Computing (CIC), pp. 200–206, Las Vegas, USA, (2005)
13. Flocchini, P., Mans, B., Santoro, N.: Tree decontamination with temporary immunity. In Algorithms and Computation, pp. 330–341. Springer (2008)
14. Kahn, J.: Personal communication (2013)
15. Kutten, S., Peleg, D.: Fault-local distributed mending. J Algorithms 30(1), 144–165 (1999)
16. Kutten, S., Peleg, D.: Tight fault locality. SIAM J. Comput. 30(1), 247–268 (2000)
17. La Paugh, A.S.: Recontamination does not help to search a graph. J. ACM 40(2), 224–245 (1993)
18. Richard, J.: Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. SIAM J. Appl. Math. 36(2), 177–189 (1979)
19. Luccio, F., Pagli, L., Santoro, N.: Network decontamination with local immunization. In: Proceedings of the 20th International Conference on Parallel and Distributed Processing, pp. 264–264. IEEE Computer Society (2006)
20. Megiddo, N., Hakimi, S.L., Garey, M.R., Johnson, D.S., Papadimitriou, C.H.: The complexity of searching a graph. J ACM 35(1), 18–44 (1988)
21. Parsons, T.D.: Pursuit-evasion in a graph. theory and applications of graphs. Lecture Notes in Mathematics, pp. 426–441. Springer-Verlag (1976)
22. Parsons, T.D.: The search number of a connected graph. In: Proceedings of the 9th South-Eastern Conference on Combinatorics, Graph Theory, and Computing, pp. 549–554 (1978)