Scheduling Intrusion Detection Systems in Resource-Bounded Cyber-Physical Systems

Waseem Abbas*
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37212
waseem.abbas@vanderbilt.edu

Aron Laszka†
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37212
aron.laszka@vanderbilt.edu

Yevgeniy Vorobeychik
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37212
yevgeniy.vorobeychik@vanderbilt.edu

Xenofon Koutsoukos
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37212
xenofon.koutsoukos@vanderbilt.edu

ABSTRACT

In order to be resilient to attacks, a cyber-physical system (CPS) must be able to detect attacks before they can cause significant damage. To achieve this, *intrusion detection systems* (IDS) may be deployed, which can detect attacks and alert human operators, who can then intervene. However, the resource-constrained nature of many CPS poses a challenge, since reliable IDS can be computationally expensive. Consequently, computational nodes may not be able to perform intrusion detection continuously, which means that we have to devise a schedule for performing intrusion detection. While a uniformly random schedule may be optimal in a purely cyber system, an optimal schedule for protecting CPS must also take into account the physical properties of the system, since the set of adversarial actions and their consequences depend on the physical systems. Here, in the context of water distribution networks, we study IDS scheduling problems in two settings and under the constraints on the available battery supplies. In the first problem, the objective is to design, for a given duration of time T, scheduling schemes for IDS so that the probability of detecting an attack is maximized within that duration. We propose efficient heuristic algorithms for this general problem and evaluate them on various networks. In the second problem, our objective is to design scheduling schemes for IDS so that the overall lifetime of the network is maximized while ensuring that an intruder attack is always detected. Various strategies to deal with this problem are presented and evaluated for various networks.

Categories and Subject Descriptors

K.6.5 [**Management Of Computing and Information Systems**]: Security and Protection

Keywords

Cyber-physical system, scheduling, intrusion detection systems, sensor networks, dominating sets, game theory

1. INTRODUCTION

Traditionally, cyber-security research has focused primarily on preventing attacks from successfully penetrating sensitive systems. However, as recent examples have shown, motivated and resourceful attackers may be able to compromise even highly secure and secluded systems. Consider, for example, the Stuxnet worm, which was able to penetrate nuclear facilities [15, 17], or the successful attack against RSA, a leading security company [23]. In light of these examples, we must focus not only on the "first lines" of defense but also on mitigating the effects of successful compromises, thereby increasing our system's resilience to attacks.

Mitigating the effects of successful compromises is possible only if attackers are not able to inflict substantial damage immediately after compromising the system, but only after some delay. This delay allows us to implement countermeasures and prevent the system from sustaining significant losses, which is the key to increasing resilience. Due to the unalterable physical attributes of CPS, many attacks against CPS are inherently limited in how quickly they can cause substantial damage. For example, in the Maroochy Shire water-services incident, the attack lasted multiple months [3]; as another example, the Stuxnet worm drastically reduced the lifetime of nuclear centrifuges, eventually destroying one-fifth of Iran's centrifuges [16]. Consequently, it is imperative that we are able to detect and react to attacks against our systems.

*The author contributed equally to this work.

†The author contributed equally to this work.

CPS-SPC'15, October 16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3827-1/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2808705.2808711 .

To this end, we can deploy *intrusion detection systems* (IDS), which may detect attacks and alarm human operators, who can then intervene. These systems try to detect attacks by looking for signatures of known attacks (e.g., known exploits) or for anomalies (i.e., suspicious activities). Since attackers may try to stay covert until they inflict damage, detection is a challenging task, and operating an IDS can require substantial resources. However, many cyber-physical systems consist of resource-constrained nodes, which limits the applicability of IDS, as the nodes may not be able to run an IDS continuously. For example, many cyber-physical systems build on wireless sensors, which use batteries as power sources. Since conserving battery power is crucial to prolonging the lifetime of a system, a computationally demanding IDS cannot be operated continuously. As another example, consider devices with limited computational power, which is not sufficient for running an IDS and serving the devices' primary function at the same time. Therefore, in CPS based on resource-constrained devices, we may be able to operate IDS only at a fraction of time.

Hence, we face the problem of determining the schedule of running IDS, that is, determining when each node should be running IDS. This scheduling problem has many unique challenging aspects. Firstly, if we used a deterministic schedule, the attacker might be able to predict which nodes are running IDS at the time of the attack. This information would allow the attacker to target nodes that are unprotected, thereby circumventing the IDS. Consequently, we must use a non-deterministic schedule if we want to secure our system. Secondly, the optimal schedule of running IDS must take into account characteristics of the physical part of the system, since these characteristics also determine actions of the attacker. For example, if a given subset of sensors measure some physical property of the system and the attacker's goal is to fake this property, then at least one of these sensors should be running IDS at any given time. Finally, following Kerckhoffs's principle, we should expect an attacker who knows everything about the algorithm that we use for scheduling (except for the randomization). Hence, we must optimize our schedule for a worst-case rational attacker, which can be modeled as a Stackelberg security game.

Since schedules must take into account physical aspects of the system, the optimal solutions have to be tailored for specific systems to some extent. In this paper, we consider the problem of scheduling host-based IDS on resource-constrained devices for *water-distribution networks*. In these networks, sensors such as pressure sensors are placed to observe events such as pipe leakages or bursts that result in pressure changes detected by the sensors. An attacker might compromise a subset of sensors altering their true observations. As a result, failure events might go undetected resulting in physical or monetary losses, or in the other case, false alarms might be generated causing the wastage of useful resources.

To this end, scheduling problems in IDS with limitations on the battery supplies, can be studied in two different settings. First, for a given duration of time, design a scheduling scheme for IDS with the objective of maximizing the probability of detecting an intruder attack during that time period. Second, design a scheduling scheme to maximize the lifetime of the network while imposing the condition that an intruder attack should always be detected. In other words, for how long can we operate the network with limited battery supplies and under the constraint that an attack is always detected? Both of these problems, though related, are of practical significance, and we study them in detail here.

Table 1: List of Symbols

Symbol	Description
V	junctions
E	links
S	sensors
D	detection distance
T	time horizon
B	battery power
$\mathcal{U}$	defender's utility

The remainder of this paper is organized as follows. In Section 2, we introduce our model of scheduling intrusion detection systems for water-network monitoring. Then, we formulate the first problem in which the objective is to design, for a given duration of time T, scheduling schemes for IDS so that the probability of detecting an attacker is maximized within that duration. In Section 3, we propose and compare various scheduling algorithms to achieve this objective. The numerical results for three different types of networks are presented in Section 4. In Sections 5 and 6, we study the second scheduling problem in which complete coverage of the network is required. Complete coverage means that any edge failure should be detected by some active sensor, and every active sensor should have an active IDS to ensure that any attack on an active sensor is detected. The objective there is to design a scheduling scheme to maximize the lifetime of the network while ensuring complete coverage. We formulate and present solutions to the problem in Section 5, and deal with the computational aspects of the solution in Section 6 along with various simulation results. Finally, we present our conclusions in Section 7.

2. MODEL

In this section, we introduce our game-theoretic model of IDS for water-distribution networks. For a list of symbols used in this paper, see Table 1. We will also use the following notation:

$$1_{\text{condition}} = \begin{cases} 1 \text{ if condition is holds,} \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

We model the water-distribution network as a graph $G(V, E)$, where V is the set of nodes corresponding to the junctions of pipes (i.e., links), and E is the set of links. Some of the nodes have pressure sensors for detecting leakages; hence, the set of sensors S is a subset of V. A leakage (i.e., failure) at pipe $\ell \in E$ is detected by a sensor whenever the distance between the sensor and the leakage is smaller than or equal to a certain threshold D. This distance threshold based model has been used in water networks in the context of sensor placement problems (e.g., [10]). We define the distance between a node and link in a natural way: if the link is connected to the node, their distance is 1; otherwise, their

distance is 1 plus the length of the shortest path to the node from the end of link which is closer to the node.

We assume that time is divided into T timeslots, which are denoted $1, \ldots, T$. Each sensor device is capable of running IDS; however, this consumes battery power, which is limited. Formally, we assume that each sensor can run IDS for at most B timeslots.

2.1 Strategies

We model the security problem as a two-player Stackelberg security game between a defender and an attacker. The defender's strategic choice is to select a schedule, which we represent as T subsets of S, denoted by S_1, S_2, and S_T. Since the schedule must be randomized, in practice the subsets are rearranged into a random order, and in each timeslot, those nodes will be running IDS that are members of the subset corresponding to the timeslot. Consequently, we can express the battery constraint as

$$\forall s \in S: \ \sum_{t=1}^{T} 1_{s \in S_t} \leq B. \tag{2}$$

The attacker selects a subset of sensors $A \subseteq S$, compromises them, and changes the leak report. We can assume that the attacker selects only subsets which lead to a fake leakage report. Due to the randomization, the attacker does not know which subset S_t is running IDS at the time of the attack. Consequently, the attacker does not have to choose the time of the attack, and the probability of the attack including a sensor that is running IDS is

$$\sum_{t=1}^{T} \frac{1}{T} 1_{A \cap S_t \neq \emptyset}. \tag{3}$$

Note that, following Kerckhoffs's principle, we assume that the attacker knows the schedule $(S_1, \ldots, S_T)$, only their order is unknown.

2.2 Payoff

We define the player's payoffs in a natural way: the defender's utility is the probability of detecting an attack, while the attacker's utility is the probability of not being detected. We assume that an attack is detected iff it includes a sensor that is running IDS. Thus, the the defender's utility is

$$\mathcal{U}(S_1, S_2, \ldots, S_T) = \min_{\substack{A \subseteq S: \\ A \text{ can fake leakage}}} \Pr[\text{attack } A \text{ is detected}] \tag{4}$$

$$= \min_{\substack{A \subseteq S: \\ A \text{ can fake leakage}}} \sum_{t=1}^{T} \frac{1}{T} 1_{A \cap S_t \neq \emptyset}. \tag{5}$$

Finally, we can formulate the problem of finding the optimal schedule as the following optimization problem:

$$\max_{\substack{(S_1,\ldots,S_T): \\ \forall s \in S: \ \sum_{t=1}^{T} 1_{s \in S_t} \leq B}} \left(\min_{\substack{A \subseteq S: \\ A \text{ can fake leakage}}} \sum_{t=1}^{T} \frac{1}{T} 1_{A \cap S_t \neq \emptyset} \right). \tag{6}$$

3. ANALYSIS

In this section, we study the problem of finding an optimal schedule in general.

3.1 Attacker's Best Response

First, observe that we can represent the attacker's strategy space simply by E: once the attacker has decided the link ℓ whose failure he is going to fake, the subset of sensors $A(\ell)$ to be compromised is given by the influence matrix M.

Consequently, the attacker's best response is simply to pick the link ℓ that leads to the lowest probability of getting caught. For a given schedule $(S_1, \ldots, S_T)$, we can compute this easily:

$$\forall \ell: \ \Pr[\ell \text{ not detected}] = 1 - \sum_{t=1}^{T} \frac{1}{T} 1_{A(\ell) \cap S_t \neq \emptyset}. \tag{7}$$

3.2 Defender's Optimal Schedule

First, we show that finding an optimal schedule for the defender is computationally hard. We formulate the defender's problem as a decision problem as follows.

Definition 1. **Secure Schedule Problem:** Given a graph $G = (V, E)$, a set of sensors $S \subseteq V$, a detection distance D, a time horizon T, a battery power B, and a threshold utility $\mathcal{U}^*$, determine if there exists schedule $(S_1, S_2, \ldots, S_T)$ such that the defender's utility $\mathcal{U}$ is at least $\mathcal{U}^*$.

THEOREM 1. *The Secure Schedule problem is NP-hard, even in the special case $D = 2$, $B = 1$, $T = 2$, and $\mathcal{U}^* = 1$.*

We show that the Secure Schedule problem is NP-hard by reducing a known NP-hard problem, the 2-Disjoint Set Covers problem [9] to the schedule problem. The 2-Disjoint Set Covers problem is defined as follows.

Definition 2. **2-Disjoint Set Covers Problem** Given a set U and a collection $\mathcal{C}$ of subsets of U, determine whether $\mathcal{C}$ can be partitioned into two disjoint set covers or not.

PROOF. Given an instance of the 2-Disjoint Set Covers problem (i.e., a base set U and a collection $\mathcal{C}$ of subsets), we construct an instance of the Secure Schedule problem as follows:

- for every $u \in U$, create two nodes, denoted node u_1 and node u_2, and connect them with a link;
- for every $C \in \mathcal{C}$, create a node, denoted node C, and connect it to every node u_1 such that $u \in C$;
- create two additional nodes, denoted node a_1 and node a_2, and connect both of them every node $C \in \mathcal{C}$;
- let the set of sensors S be the union of $\{a_1, a_2\}$ and the set of nodes corresponding to the elements of $\mathcal{C}$;
- let the detection distance be $D = 2$;
- let the time horizon be $T = 2$;
- let the battery power be $B = 1$;
- and let the threshold utility be $\mathcal{U}^* = 1$.

It is obvious that the above reduction can be carried out in time that is polynomial in the size of the input. Therefore, it remains to show that the Secure Schedule (SS) problem has a solution if and only if the 2-Disjoint Set Covers (2-DSC) problem does.

First, assume that 2-DSC has a solution, that is, there exists a partition $(\mathcal{C}_1, \mathcal{C}_2)$ such that both $\mathcal{C}_1$ and $\mathcal{C}_2$ cover the base set U. Then, let the defender's schedule be $(S_1 = \mathcal{C}_1 \cup \{a_1\}, S_2 = \mathcal{C}_2 \cup \{a_2\})$. Since $(\mathcal{C}_1, \mathcal{C}_2)$ is a partition, this schedule clearly abides the battery constraint, as each sensor is activated only in single timeslot. It remains to show that the defender's utility $\mathcal{U}$ is $\mathcal{U}^* = 1$, that is, we have to show that an attack against any link will be detected with certainty in both timeslots. Consider an arbitrary timeslot t and link $\ell \in E$. First, if ℓ is a link between some node $C \in \mathcal{C}$ and another node, then the distance between this link and either a_1 or a_2 is at most two, since every C is connected to both a_1 and a_2. Consequently, as node a_t is active in timeslot t, an attack targeting link ℓ is detected with certainty. Second, suppose that link ℓ is a link between some pair of nodes u_1 and u_2. Since $\mathcal{C}_t$ is a set cover, there exists a node $C \in \mathcal{C}_t \subset S_t$ that is connected to u_1. Consequently, the distance between link ℓ and a sensor (i.e., node C) is at most two. Therefore, the schedule (S_1, S_2) attains utility $\mathcal{U}^* = 1$, which means that SS has a solution.

Second, assume that the SS has a solution, that is, there exists a schedule (S_1, S_2) such that any attack will be detected. Then, we show that $(\mathcal{C}_1 = S_1 \cap \mathcal{C}, \mathcal{C}_2 = S_2 \cap \mathcal{C})$ is a solution to 2-DSC. First, it is obvious that $(\mathcal{C}_1, \mathcal{C}_2)$ is a partition, since every sensor is a member of at most one S_t due to the battery constraint. It remains to show that every $\mathcal{C}_t$ is a set cover. For the sake of contradiction, suppose that this is not true, i.e., there exist a $\mathcal{C}_t$ and an element u such that u is not an element of any $C \in \mathcal{C}_t$. This implies that S_t does not contain any node that is connected to u_1; hence, the distance between the link (u_1, u_2) and any sensor is at least three. However, this contradicts our initial assumption that the schedule (S_1, S_2) attains utility $\mathcal{U}^* = 1$, since an attack against link ℓ is detected with probability 0.5 (or less). Therefore, the original claim must hold, that is, partition $(\mathcal{C}_1, \mathcal{C}_2)$ is a solution to 2-DSC. □

3.3 Algorithms for Finding a Schedule

Since the problem of finding an optimal schedule is computationally hard, in practice, we have to use either heuristic algorithms that are not guaranteed to find an optimal solutions, or specific algorithms for special cases. Here, we introduce and discuss three heuristic algorithms, which we will evaluate numerically in the following section.

3.3.1 Simple Greedy

The scheduling problem resembles the set covering problem closely, since we have to "cover" the set of links using a set of sensors, each of which can cover a given subset of the links. Since the straightforward greedy algorithm is known to be an approximation algorithm for the set covering problem, we can expect it to perform well for the scheduling problem as well. Hence, we formulate a greedy algorithm for the scheduling problem as follows (see Algorithm 1): start with an empty schedule $(S_1 = \emptyset, \ldots, S_T = \emptyset)$, and iteratively add assign the sensors to the sets S_t, always picking a feasible combination (s, t) of a sensor and a timeslot that increases the defender's utility the most.

3.3.2 Overlap Minimization

The greedy algorithm works well for the set covering problem because the set-covering objective function is submodular. Unfortunately, the defender's utility in the scheduling problem is not submodular. Moreover, in most steps of the algorithm, there is no combination (s, t) that could increase the defender's utility. For example, if we start with an empty schedule, the utility remains zero until we reach complete coverage, that is, until every single link is covered by some node in at least one timeslot. This poses a problem for the greedy algorithm, since it makes an arbitrary choice if no combination can increase the objective function.

Algorithm 1 Simple Greedy

```
for all t = 1, ..., T do
    S_t ← ∅
end for
S' ← S
while S' ≠ ∅ do
    (s, t) ← argmax_{(σ∈S', τ∈{1,...,T})} U(S_1, ..., S_τ ∪ {σ}, ..., S_T)
    A_t ← A_t ∪ s
    S' ← {s ∈ S | s has been assigned to less than B time slots}
end while
```

To evade the problems caused by the behavior of the original objective function, we introduce an alternative objective function. First, recall that the original objective function is based on the minimum "covered" link, that is, the link that is detected in the least number of timeslots. After all the sensors have been assigned to B timeslots, the number of timeslots in which a link ℓ is detected can be computed as the total number of sensors that can detect ℓ minus the total number of overlaps, where an overlap is a pair of sensors that can both detect ℓ and run IDS in the same timeslot. Formally, given that every sensor has been assigned to B timeslots, we have

$$\sum_{t=1}^{T} \frac{1}{T} 1_{A(\ell) \cap S_t \neq \emptyset} = \frac{\#\text{ of sensors covering } \ell - \#\text{ of overlaps on } \ell}{T} . \quad (8)$$

Now, observe that the first term is constant, since it is determined by the structure of the network. Hence, in order to maximize the minimal coverage, we have to minize the number of overlaps.

Based on the above observation, we introduce a greedy *overlap minimization* algorithm, which is defined as follows (see Algorithm 2, where $C(s)$ denotes the set of links whose failures node s can detect): start with an empty schedule $(S_1 = \emptyset, \ldots, S_T = \emptyset)$, and proceed in B iterations; in each iteration, assign each sensor to a timeslot in which it overlaps with the minimum number of other sensors. As we will see in the numerical results, this approach for minimizing "wasted" sensing leads to very good results in practice.

3.3.3 Repeated Set Cover

An alternative approach for finding a good schedule is to break down the problem into T subproblems: instead of trying to find a good schedule for T timeslots, we try to find a covering set for each individual timeslot. More formally, given an algorithm for set covering, we can find a good schedule as follows (see Algorithm 3): iterate over the timeslots, and find a covering set for each timeslot using the sensors that have not been assigned to B timeslots; if no complete

Algorithm 2 Overlap Minimization

for all $t = 1, \ldots, T$ **do**
  $S_t \leftarrow \emptyset$
end for
for all $b = 1, \ldots, B$ **do**
  for all $s \in S$ **do**
    $t \leftarrow \text{argmin}_\tau \left| C(s) \cap \bigcup_{\sigma \in S_\tau} C(\sigma) \right|$
    $A_t \leftarrow A_t \cup s$
  end for
end for

cover exists, use all available to achieve maximal coverage. The rationale behind this algorithm is that – with a good set cover algorithm – we will use a small number of sensor in each timeslot, and we can achieve complete coverage for a large fraction of the timeslots. Note that we can plug any set cover algorithm into the above heuristic, e.g., a greedy approximation algorithm.

Algorithm 3 Repeated Set Cover

for all $t = 1, \ldots, T$ **do**
  $S' \leftarrow \{s \in S \mid s$ has been assigned to less than B time slots$\}$
  if $\forall \ell : A(\ell) \cap S' \neq \emptyset$ **then**
    $S_t \leftarrow$ greedy minimum cover of E using S'
  else
    $S_t \leftarrow S'$
  end if
end for

4. NUMERICAL RESULTS

In this section, we present numerical results on the simple greedy (Algorithm 1), the overlap minimization (Algorithm 2), and the repeated set cover (Algorithm 3) algorithms. Note that we evaluated the latter using a greedy algorithm for the set covering step.

We have evaluated our algorithms on three types of networks:

- *Random geometric graphs*: In these graphs, the nodes are drawn from the area of a unit square uniformly at random, and two nodes are connected if their distance is less than a given threshold, which we chose to be 0.15 for the experiments. We used this random-graph model in the numerical evaluation because it can capture the geographic nature of water-distribution networks.
- *Barabási-Albert (BA) random graphs* [6]: We generated random networks starting with cliques of 2 nodes and connecting every additional node to 2 existing ones. B-A graphs are widely used to construct synthetic graphs as their heavy-tailed degree distribution resembles real-world technological networks.
- *Water-distribution network*: We also compared our algorithms using a water-distribution network from [21], which has 126 nodes, 168 pipes, one reservoir, one pump, and two storage tanks. The layout of the network is illustrated in Figure 9. This benchmark water-distribution network has been extensively studied in the context of sensor placement problems for water quality.

For each of the two random network types, we generated 1000 graphs, each having 100 nodes. Then, for each value of the battery power B, we plotted the average utility over these graphs.

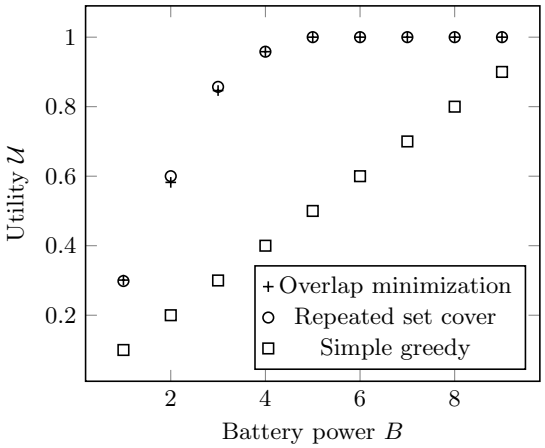


Figure 1: Comparison of various algorithms for scheduling on *geometric graphs* with $|V| = 100$, $S = V$, $D = 2$, $T = 10$.

Figure 1 compares our scheduling algorithms on random geometric graphs. We see that both the overlap minimization and the repeated set cover algorithms perform well, the latter being slightly better. On the other hand, the simple greedy algorithm performs much worse than the other two. In fact, the output of the simple greedy algorithm is actually equal to a naïve solution that assigns each sensor to the first B sets $S_1, \ldots, S_B$, achieving B/T utility.

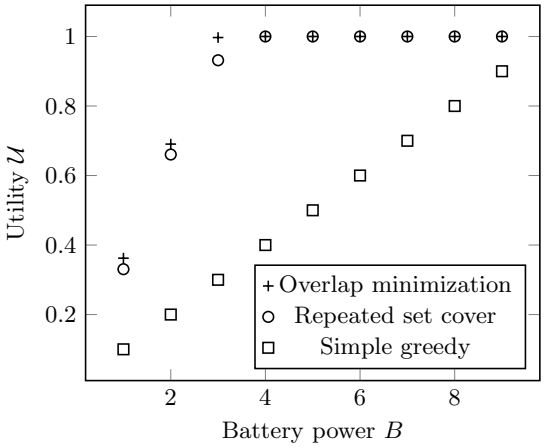


Figure 2: Comparison of various algorithms for scheduling on *BA graphs* with $|V| = 100$, $S = V$, $D = 2$, $T = 10$.

Figure 2 compares our scheduling algorithms on random BA graphs. Similarly to the case of geometric graphs, we see that both the overlap minimization and the repeated set cover algorithms perform very well. However, in this

case, the former performs slightly better. Finally, the performance of the simple greedy algorithm is again abysmal.

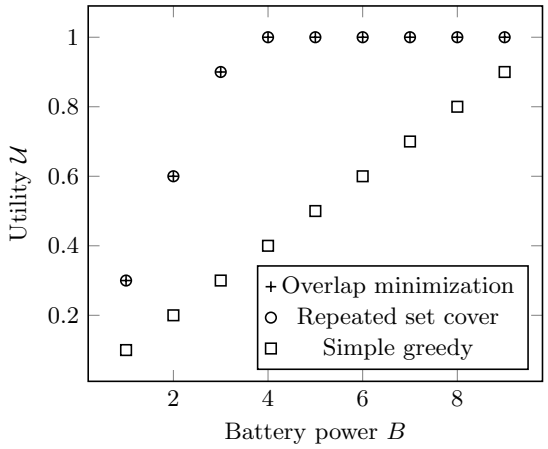


Figure 3: Comparison of various algorithms for scheduling on the *water-distribution network* with $S = V$, $D = 2$, $T = 10$.

Figure 3 compares our scheduling algorithms on the water-distribution network. Again, we see that the both the overlap minimization and the repeated set cover algorithms perform well, while the simple greedy algorithm performs much worse.

5. MAXIMIZING THE NETWORK LIFETIME WHILE ENSURING COMPLETE COVERAGE

So far, we have focused on a setup where a scheduling scheme is designed for a fixed duration of time with the objective of maximizing the probability of detecting an attack. In this section, we study a scheduling problem with the objective of maximizing the overall lifetime of the network while insisting on the *complete coverage* of the whole network. Complete coverage here means the following conditions are satisfied:

(a) Every edge failure $\ell \in E$ (leakage in a pipe) can be detected by some sensor under the model presented in Section 2.

(b) An IDS is active on a node whenever a sensor at the node is turned on to detect an edge failure, i.e., an attack on an active node will always be detected.

We assume that every node in the network is equipped with a sensor and an IDS, and the objective is to schedule their turning on/off to maximize the the overall lifetime of the network while ensuring complete coverage. Before proceeding further, we define the *closed neighborhood* of a node as following:

Definition 3. The closed neighborhood of a node i, denoted by N_i, is the union of i and the set of vertices that are directly connected to the node i through an edge.

Now, if $D = 2$ is assumed in the sensing model (i.e., an edge failure is detected by the nodes in the closed neighborhoods of the end nodes of the edge), then to achieve (a) above, IDS and sensors need to be placed at the nodes that form a so-called *vertex 2-cover* of the underlying graph of the network. The vertex 2-cover is defined as

Definition 4. (Vertex 2-Cover) A vertex 2-cover $\mathcal{C} \subset V$ is a subset of vertex set such that if $\ell(u, v)$ is an edge of the graph, then

$$\exists x \in \mathcal{C} : \{x\} \cap (N_u \cup N_v) \neq \emptyset$$

For the sake of simplicity, we say that a node v is *active* if the corresponding sensor and IDS on the node are turned on. Thus, *to have a complete coverage with $D = 2$, the set of active nodes should form a vertex 2-cover.* Now, owing to the limited battery power available at each node, there is a constraint on the duration for which a node can be active. Thus, the problem of maximizing the overall lifetime of the network while maintaining complete coverage is related to finding distinct vertex 2-covers under the constraints on the number of times a node can be included in such vertex 2-covers.

5.1 Dominating Set Based Problem Formulation

Now, we see that instead of working with vertex 2-covers of a graph for the complete coverage problem, we can utilize the notion of more widely studied *dominating sets*.

Definition 5. A dominating set is a set of vertices, denoted by $\mathcal{D}$, such that for every $i \in V$, there exists some $j \in \mathcal{D}$ such that $i \in N_j$.

We observe that if $\mathcal{D}$ is a dominating set, then by the definition, for every edge $\ell(u, v) \in E$, there are some $x, y \in \mathcal{D}$ such that $\{x\} \cap N_u \neq \emptyset$ and $\{y\} \cap N_v \neq \emptyset$. This leads to the following observation:

Observation 1. *A dominating set of a graph is also its vertex 2-cover.*

In other words, *the network is guaranteed to be completely covered whenever the set of active nodes form a dominating set.* Thus, in a way, the objective is to find distinct dominating sets in a graph for the active nodes so that the overall lifetime of the network is maximized. The problem of finding distinct dominating sets under certain constraints has been of great interests owing to its wide variety of applications (e.g., [4, 13, 14, 22]). Thus, we use a dominating set based formulation of the complete coverage problem to maximize the network lifetime.

Let B be the time for which a node can be an active node, and is a measure of the battery power of the node. Let $\mathcal{S}$ be the set of all dominating sets in a graph. If t_i be the time for which the nodes in $S_i \in \mathcal{S}$ are active, then the problem is to find a subset $S = \{S_1, \cdots, S_r\} \subseteq \mathcal{S}$ so that

1. the network lifetime, $T = \sum_{i=1}^{r} t_i$ is maximized.
2. no node is active for more than B duration, i.e.,

$$\sum_{i=1}^{r} t_i \mathbf{1}_{\{v\} \cap S_i \neq \emptyset} \leq B, \ \ \forall v$$

For the sake of simplicity, we assume $t_i = 1$ from here onwards. Then the objective is to obtain the maximum number of distinct dominating sets in a graph such that a node belongs to at most B distinct dominating sets.

5.2 Approaches to Lifetime Maximization

In this subsection, we will consider two different approaches to maximize the network lifetime while ensuring complete coverage.

5.2.1 Disjoint Dominating Set Based Approach

One way to approach this problem is to partition the vertex set such that each set in the partition is a dominating set, and all dominating sets are pair-wise disjoint. Such a partition is known as the *domatic partition*, and the maximum number of (disjoint dominating) sets that can be obtained is known as the *domatic number*, denoted by γ. Since dominating sets are pairwise disjoint in such a partition, each vertex belongs to only one of the dominating sets. Moreover, since each node can be active for B time slots, each dominating set can remain active for B time slots. If only one dominating set is active at any time instant, which is sufficient for the complete coverage, then the lifetime of the network achievable through this approach is

$$T_{dom} = \gamma B \tag{9}$$

The domatic partition problem has been extensively studied in the literature, and is known to be NP-hard [11]. Various sensor scheduling schemes that utilize domatic partitions have been proposed to maximize the network lifetime while ensuring complete coverage (e.g., [22, 20, 25]).

5.2.2 Non-Disjoint Dominating Set Based Approach

Another way to approach the network lifetime maximization while maintaining complete coverage is by using the *non-disjoint dominating sets* of active nodes. Using this approach, it is possible to obtain $T \geq T_{dom}$. As an illustration, consider the network shown in Figure 4. The graph has a domatic number of 2 [12] and therefore, $T_{dom} = 2B$. However, it is possible to obtain $T = \frac{5}{2}B = \frac{5}{4}T_{dom}$ if non-disjoint dominating sets are used as follows: Obtain five distinct (non-disjoint) dominating sets such that each node is included in at most two of the dominating sets as shown in Figure 4. Activate each dominating set for the $B/2$ duration. Since there are five dominating sets and each node has a battery power for B duration, the network remains completely covered for $5\left(\frac{B}{2}\right)$ time.

It can be noted that the overall network lifetime using the non-disjoint dominating set based approach directly depends, and is proportional to the number of distinct dominating sets that can be obtained under the constraint on the number of times a node can appear in a dominating set. Moreover, the network lifetime obtained this way is always going to be better or equal to the one obtained through disjoint dominating set based strategy. The problem of finding the maximum number of dominating sets under the constraints on the number of times a vertex can be included in a dominating set is related to the notion of (r, s)-configurations [1, 12] as defined below.

Definition 6. Let s, r be two positive integers, and $L = \{1, \cdots, r\}$ be the set of labels, then (r, s)-configuration of a graph is the assignment of s distinct labels from the set L to each vertex in the graph such that for every $i \in L$ and every vertex v, the label i is assigned to v or one of its neighbors.

An example of $(5, 2)$-configuration is shown in Figure 4. Note that the set of vertices corresponding to a particular label in L constitute a dominating set. So, if a graph has an (r, s)-configuration, it is possible to have r distinct dominating sets in which a vertex can be included in at most s such dominating sets. Thus, for a given s, finding the maximum r, denoted by r^*, for which (r^*, s)-configuration of a graph exists is of particular interest. In fact, for a given s, the maximum lifetime of the network that can be achieved using non-disjoint dominating set based strategy is $T = (r^*/s)\, B$. Hence, *the optimal network lifetime achieved using the non-disjoint dominating set based strategy under the battery power constraints (B) and complete coverage condition is*

$$T = \left\lfloor \max\left(\frac{r^*}{s}\right) \right\rfloor B \;\geq\; T_{dom} = \gamma B \tag{10}$$

where r^* and s are such that an (r^*, s)-configuration of the graph exists.

A non-disjoint dominating set based strategy can now be outlined as follows:

1 For a given graph, compute an (r, s) configuration such that (r^*/s) is maximum.

2 Let each time slot span B/s time unit.

3 Activate the nodes in a dominating set for the duration of one time slot.

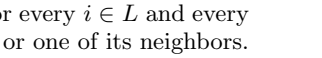
4 Since there are r^* distinct dominating sets, repeat step 3 for all such dominating steps.

The overall lifetime of the network obtained as a result of the above strategy is $\left(\frac{r^*}{s}\right) B$. It can be noted that the non-disjoint dominating set based strategy is strictly better than the disjoint dominating set based approach whenever $(r^*/s) > \gamma$. An important question is therefore, for a given s, which graphs have $(r^*/s) > \gamma$? In this regard, first we note that every connected graph has $\gamma \geq 2$, and therefore, for a given s, r^* is always at least $2s$. However, there exists many graphs for which $\gamma = 2$, but $r^* > 2s$. For instance, many *cubic graphs*[1] have a domatic number of 2, e.g., the one shown in Figure 4. However, the following theorem asserts that all cubic graph have $r^* \geq \frac{5}{2}s$ for a given s.

THEOREM 2. *[12] Any cubic graph has an (r, s)-configuration with $r = \lfloor 5s/2 \rfloor$, and such a configuration can be found in polynomial time.*

Recently, it has been shown in [2] that the above result is true even for a bigger class of graphs as stated in Theorem 3. Here, $K_{1,6}$ is a star graph with one central node of degree six, and six end nodes each with a degree one.

THEOREM 3. *[2] Let G be a graph such that*

- *G has a minimum degree at least two,*
- *no subgraph of G is isomorphic to $K_{1,6}$, and*
- *$G \neq \{$*  *$\}$;*

then G has an (r, s)-configuration with $r = \lfloor \frac{5s}{2} \rfloor$.

[1] graphs in which each vertex has a degree three.

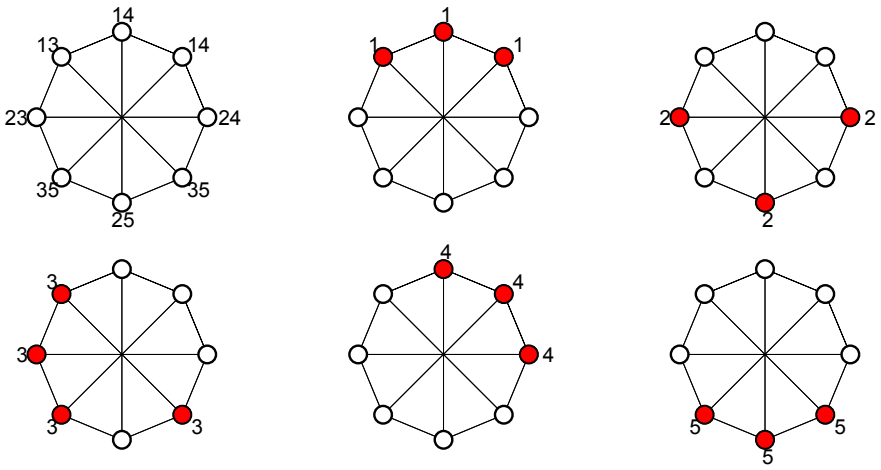


Figure 4: A graph has five distinct dominating sets indicated by the nodes with the same labels. Each node belongs to two distinct dominating sets.

The above result is particularly useful as the $R - disk$ proximity graph model[2], which is often used to model the limited range communication in networks such as wireless sensor networks, is always $K_{1,6}$-free. As pointed out in [2], a large number of graphs in this family have a domatic number of 2, thus, non-disjoint dominating set based strategy is strictly better than the disjoint dominating set based strategy in those cases.

6. COMPUTING (R, S)–CONFIGURATIONS

In this section, using a game theoretic setting, we present an algorithm to assign s labels from a set of r labels to each node such that the number of distinct labels in the closed neighborhoods of all nodes is maximized. In the case (r, s)-configuration of a graph exists, each node should have all r labels in its closed neighborhood.

Let $L = \{1, 2, \cdots, r\}$ be the set of r labels, and Q be the set of all s-element subsets of Q. Note that $|Q| = \binom{r}{s}$. Moreover,

$$f : V \longrightarrow Q \tag{11}$$

i.e., f is a set function that assigns $q \in Q$ to each vertex in a graph. Similarly, if N_i represents the closed neighborhood of a vertex i, then we define $F(i)$ as follows:

$$F(i) = \bigcup_{j \in N_i} f(j) \tag{12}$$

If the assignment of labels to nodes, as in (11), is a valid (r, s)-configuration, then $F(i) = L$, $\forall i$. Thus, the objective is to obtain an assignment of labels that maximizes the following:

$$\text{Objective:} \quad \max_{f} \sum_{i \in V} |F(i)| \tag{13}$$

[2]An edge exists between two vertices whenever the Euclidean distance between them is at most R.

Note that the maximum value in (13) is $r|V|$, which is achieved whenever the assignment of labels results in an (r, s)-configuration.

6.1 Game-Theoretic Formulation

Game theory concepts have been extensively employed to solve the locational optimization problems such as maximizing coverage on graphs (e.g., [24, 26]) and distributed control of multiagent systems (e.g., [5, 19]). In one of the approaches, the idea is to determine a *potential* function that captures the overall global objective. The players' individual utility functions are then appropriately aligned with the global objective such that the change in the utility of the player as a result of unilateral change in strategy equals the change in the global utility represented by the potential function. The players' strategies are then designed to ensure that local actions lead to the global objective. It turns out that this sort of problem formulation and design can be realized using a class of non-cooperative games known as *potential games*, which are now extensively used for various distributed control optimization problems.

A finite strategic game $\mathcal{G}(P, A, U)$ consists of a set of players $P = \{1, 2, \cdots, n\}$, action space $A = A_1 \times A_2 \times \cdots \times A_n$ where A_i is a finite action set of the player $i \in P$, and a set of utility functions $U = \{U_1, U_2, \cdots, U_n\}$ where $U_i : A \to \mathbb{R}$ is a utility function of the i^{th} player.

If $a = (a_1, \cdots, a_i, \cdots, a_n) \in A$ denotes the joint action profile, we let a_{-i} denote the action of players other than the player i. Using this notation, we can also represent a as (a_i, a_{-i}).

A game is a *potential game* if there exists a potential function, $\phi : A \to \mathbb{R}$ such that the change in the utility of the player i as a result of a unilateral deviation from an action profile (a_i, a_{-i}) to (a'_i, a_{-i}) is equal to the corresponding change in the potential function. More precisely, for every player i, $a_i, a'_i \in A_i$, and $a_{-i} \in A_{-i}$,

$$U_i(a_i, a_{-i}) - U_i(a'_i, a_{-i}) = \phi(a_i, a_{-i}) - \phi(a'_i, a_{-i}) \tag{14}$$

In the case of potential games, there exists algorithms such as log-linear learning (LLL) [7, 8] and binary log-linear learning (BLLL) [18] guaranteeing that only the joint action profiles that maximize the potential function are stocahstically stable. The basic idea behind these algorithms is to have a *noisy* best response dynamics, in which the noise parameter allows the selection of suboptimal action occasionally by the players. The probability of selecting a suboptimal action is dependent on the pay-off difference between the optimal and suboptimal cases. Thus, our objective is to design a potential game corresponding to the labeling problem on graphs and use the learning algorithms for the potential games to achieve the desired labeling.

6.1.1 A potential game for the graph labeling

We design a potential game $\mathcal{G}(P, A, U)$ to obtain a labeling of a graph that achieves the objective in (13), which results in an (r, s)-configuration of a graph if it exists. In our game, vertex set is the set of players, i.e., $P = V$, and for each player $i \in V$, the action set A_i is the set of all s elements subsets of the labeling set $L = \{1, \cdots, r\}$, i.e., $A_i = Q$, $\forall i \in V$. We also need to have a potential function that captures the global objective. For this, we define I_x as the set of vertices with the label x, i.e.,

$$I_x \ = \ \{v \in V : x \in f(v)\} \tag{15}$$

A potential function is then defined as

$$\phi(a) \ = \ \sum_{x=1}^{r} \left| \bigcup_{j \in I_x} N_j \right| \tag{16}$$

Note that $\phi(a)$ is simply the number of vertices having a label $x \in L$ in their closed neighborhoods, summed over all the labels, which is equivalent to the $\sum_{i \in V} |F(i)|$ in (13). Thus, $\phi(a)$ indeed captures the global objective.

Moreover, we define utility function of the player i as the total number of labels made available by a_i to the vertices in N_i that would not have been available to them otherwise. An example is illustrated in Fig. 5. More precisely, we define $U_i(a_i, a_{-i})$ as

$$U_i(a_i, a_{-i}) \ = \ \sum_{x=1}^{r} a_{ix} \left| N_i \setminus \bigcup_{k \in I_x \setminus \{i\}} N_k \right| \tag{17}$$

where,

$$a_{ix} = \begin{cases} 1 & \text{if } x \in a_i (= f(i)) \\ 0 & \text{otherwise.} \end{cases}$$

With utility function as defined in (17) and potential function as in (16), it turns out that the game designed above is indeed a potential game. Since our graph labeling problem can be formulated as a potential game, using the results in [18] we deduce that *if players adhere to the binary log linear algorithm (stated below), then the objective in (13) is achieved*. In other words, if s unique labels from a total of r labels are assigned to nodes as per below algorithm, then the number of distinct labels in the closed neighborhood of every node is maximized, which simply means that an (r, s)-configuration of a graph (if exists for a given r and s) is obtained.

Note that initially the nodes are assigned an s-element subset of labels randomly. Afterwards, in each iteration, a random node selects with a certain probability a subset of s-labels that improve the overall labeling to attain the objective in (13), which is eventually achieved.

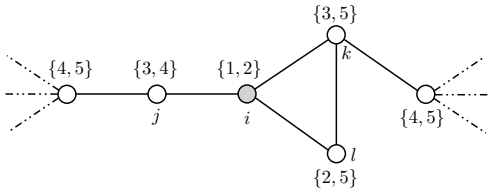


Figure 5: Here. $L = \{1, \cdots, 5\}$ and each node has a 2-element subset of L. Moreover, $N_i = \{i, j, k, l\}$. For all the four nodes in N_i, node i is the only one with the label 1; and for the node j, node i is the only one with the label 2. Thus, $U_i(a_i, a_{-i}) = 4 + 1 = 5$.

Algorithm 4 Binary Log-Linear Learning [18]

1: **Initialization:** Pick a small $\epsilon \in \mathbb{R}_+$, and an $a \in A$.
2: **Repeat**
3: Pick a random node $i \in V$, and a random $a_i' \in A_i$.
4: Compute $P_\epsilon \ = \ \frac{\epsilon^{U_i(a_i', a_{-i}(t))}}{\epsilon^{U_i(a_i', a_{-i}(t))} + \epsilon^{U_i(a_i, a_{-i}(t))}}$.
5: Set $a_i \leftarrow a_i'$ with probability P_ϵ.
6: **End Repeat**

6.2 Simulations

As in Section 4, we simulate the algorithm for three different networks including,

- *Random geometric graph* with $n = 100$ nodes distributed uniformly at random over a unit square area. We assume that an edge exists between two nodes whenever the (euclidean) distance between them is at most 0.15.
- *Barabási-Albert (BA) graph* with 100 nodes in which each new node is connected to two already existing nodes as per preferential attachment strategy.
- *Water distribution network* with 126 nodes and 168 pipes as described in Section 4 [21].

For all the networks, we assume r and s to be 5 and 2 respectively, i.e., each element is assigned two labels from the set $\{1, \cdots, 5\}$ with the objective that all five labels are available in the closed neighborhood of every node in the network. Since the random geometric graph and the BA graph each have 100 nodes, the maximum value in (13) is $|V|r = 100 \times 5$, and a labeling that achieves this value is indeed a $(5, 2)$-configuration. To keep a track of how far a given labeling is from becoming an (r, s)-configuration, we define *deficiency* of the labeling as

$$\text{Deficiency} = |V|r - \sum_{i \in V} \left| \bigcup_{j \in N_i} f(j) \right| \tag{18}$$

where $f(i)$, as previously, is the set of labels assigned to node i. If a labeling is an (r, s)-configuration, then its deficiency is zero. The output of Algorithm 4 for the random geometric and BA graphs are shown in Figures 6 and 7 respectively.

For the water network, we note that nine nodes have a degree of 1, i.e., have only one neighbor. Since every node has

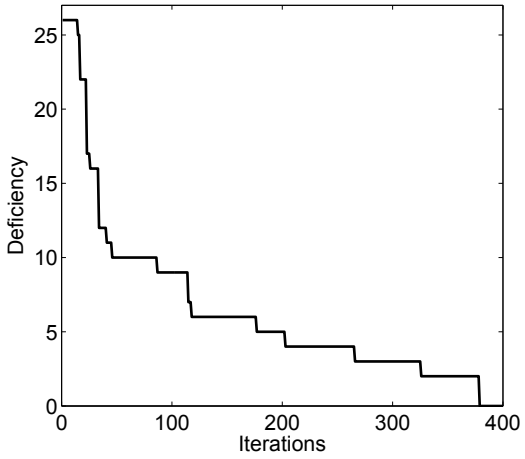


Figure 6: Deficiency of labeling as a function of (BLLL) iterations in a random geometric graph with 100 nodes.

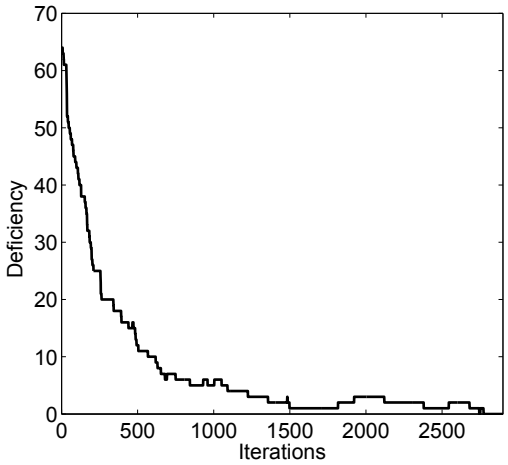


Figure 7: Deficiency of labeling as a function of (BLLL) iterations in a BA graph with $n = 100$.

two distinct labels, each of these nine nodes will be missing at least one of the five total labels in their closed neighborhoods. In other words, every labeling with $s = 2$ will have a deficiency of at least nine. Figure 8 illustrates the best possible labeling (having a deficiency of nine) obtained as a result of Algorithm 4. The nodes containing each of the five labels are also shown separately in Figure 9.

7. CONCLUSIONS

The deployment of intrusion detection systems can add significantly to the system's resilience against attacks by detecting them early enough, thus minimizing the losses incurred. However, in resource-constrained systems, such as sensor networks with limited battery supplies, a continuous operation of IDS for extended period of times becomes a major issue. Efficient scheduling schemes for running IDS at various nodes are therefore required. Here, we studied

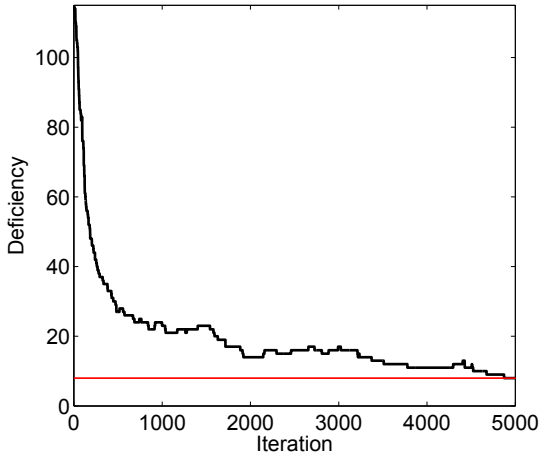


Figure 8: Deficiency of labeling as a function of (BLLL) iterations in the water network. The horizontal line indicates the best possible value (nine) of deficiency here.

scheduling schemes for IDS in the context of water distribution networks. In one of the problems, we fixed the time duration and designed schedules to maximize the detection probability during that time, while posing constraints on the available battery supplies to the individual IDS. We proposed and compared three algorithms, including simple greedy, overlap minimization, and repeated set cover. The latter two algorithms performed significantly better than the first one when evaluated for random geometric, Barabási-Albert, and water distribution networks. In another setting, we insisted that IDS needs to be scheduled in such a manner that every attack is detected, and the objective was to maximize the overall lifetime of the network while respecting the constraints on the available battery supplies to the individual IDS. We proposed a non-disjoint dominating set based strategy to obtain such a schedule by using the notion of (r, s)-configurations in graphs, and showed that this approach is better than the widely used disjoint-dominating set based strategy for the network lifetime maximization. We seek to extend this work towards more general scenarios and physical models of other infrastructure networks.

Acknowledgments

This work was supported in part by the National Science Foundation (CNS-1238959), Air Force Research Laboratory (FA8750-14-2-0180), and National Institute of Standards and Technology (70NANB13H169).

8. REFERENCES

[1] W. Abbas and M. Egerstedt. Characterizing heterogeneity in cooperative networks from a resource distribution view-point. *Communications in Information and Systems*, 14:1–22, 2014.

[2] W. Abbas, M. Egerstedt, C.-H. Liu, R. Thomas, and P. Whalen. Deploying robots with two sensors in $k\_\{1, 6\}$-free graphs. *arXiv preprint arXiv:1308.5450*, 2014.

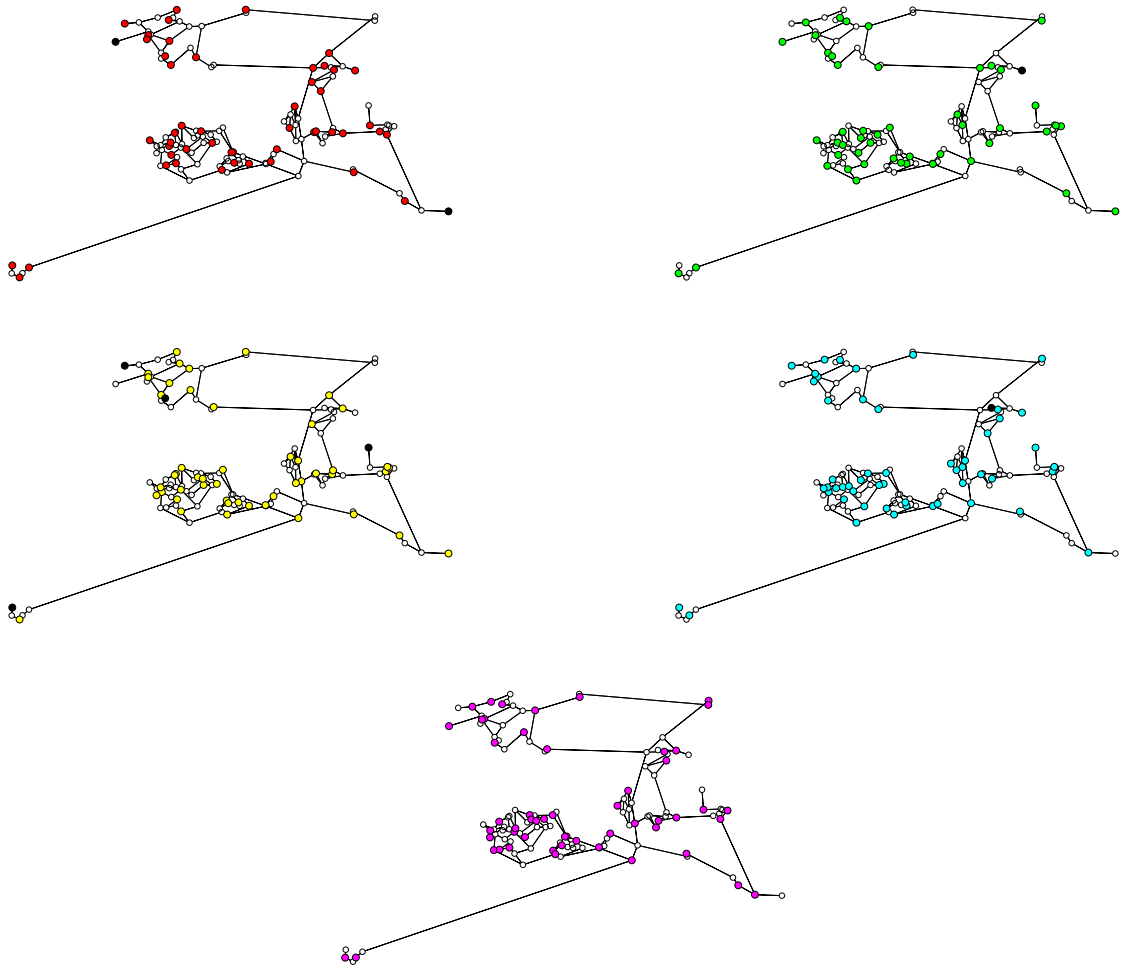


Figure 9: For $r = 5$ and $s = 2$, labeling of the water network with the deficiency of nine (best possible) is shown. The set of nodes containing label i are also shown for each $i \in \{1, 2, \cdots, 5\}$. Moreover, nodes (with degree 1) missing a specific label are colored black.

[3] M. Abrams and J. Weiss. Malicious control system cyber security attack case study – Maroochy Water Services, Australia. `http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study_report.pdf`, Jul 2008.

[4] N. Ahn and S. Park. A new mathematical formulation and a heuristic for the maximum disjoint set covers problem to improve the lifetime of the wireless sensor network. *Ad Hoc & Sensor Wireless Networks*, 13(3-4):209–225, 2011.

[5] G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[6] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[7] L. E. Blume. The statistical mechanics of strategic interaction. *Games and economic behavior*, 5(3):387–424, 1993.

[8] W. A. Brock and S. N. Durlauf. Discrete choice with social interactions. *The Review of Economic Studies*, 68(2):235–260, 2001.

[9] M. Cardei and D.-Z. Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005.

[10] A. Deshpande, S. E. Sarma, K. Youcef-Toumi, and S. Mekid. Optimal coverage of an infrastructure network using sensors with distance-decaying sensing quality. *Automatica*, 49(11):3351–3358, 2013.

[11] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating thedomatic number. *SIAM Journal on computing*, 32(1):172–195, 2002.

[12] S. Fujita, M. Yamashita, and T. Kameda. A study on r-configurations—a resource assignment problem on graphs. *SIAM Journal on Discrete Mathematics*, 13(2):227–254, 2000.

[13] S. Henna and T. Erlebach. Approximating maximum disjoint coverage in wireless sensor networks. In *Ad-hoc, Mobile, and Wireless Network*, pages 148–159. Springer, 2013.

[14] K. Islam, S. G. Akl, and H. Meijer. Maximizing the lifetime of wireless sensor networks through domatic partition. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pages 436–442. IEEE, 2009.

[15] Kaspersky Lab. Kaspersky Lab provides its insights on Stuxnet worm. `http://www.kaspersky.com/about/news/virus/2010/Kaspersky_Lab_provides_its_insights_on_Stuxnet_worm`, Sep 2010. Accessed: June 21st, 2015.

[16] M. B. Kelley. The Stuxnet attack on Iran's nuclear plant was 'far more dangerous' than previously thought. *Business Insider*, `http://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11`, Nov 2013. Accessed: June 21st, 2015.

[17] D. Kushner. The real story of Stuxnet. *IEEE Spectrum*, 50(3):48–53, 2013.

[18] J. Marden and J. Shamma. Revisiting log-linear learning: Asynchrony, completeness, and pay-off based implementation. *Games and Economic Behavior*, 75(2):788–808, 2012.

[19] I. Menache and A. Ozdaglar. Network games: Theory, models, and dynamics. *Synthesis Lectures on Communication Networks*, 4(1):1–159, 2011.

[20] T. Moscibroda and R. Wattenhofer. Maximizing the lifetime of dominating sets. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8–pp. IEEE, 2005.

[21] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, et al. The battle of the water sensor networks: A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[22] S. V. Pemmaraju and I. A. Pirwani. Energy conservation via domatic partitions. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 143–154. ACM, 2006.

[23] RSA FraudAction Research Labs. Anatomy of an attack. `https://blogs.rsa.com/anatomy-of-an-attack/`, Apr 2011. Accessed: June 21st, 2015.

[24] A. Y. Yazicioglu, M. Egerstedt, and J. S. Shamma. A game theoretic approach to distributed coverage of graphs by heterogeneous mobile agents. In *Estimation and Control of Networked Systems*, volume 4, pages 309–315, 2013.

[25] J. Yu, Q. Zhang, D. Yu, C. Chen, and G. Wang. Domatic partition in homogeneous wireless sensor networks. *Journal of Network and Computer Applications*, 37:186–193, 2014.

[26] M. Zhu and S. Martínez. Distributed coverage games for energy-aware mobile sensor networks. *SIAM Journal on Control and Optimization*, 51(1):1–27, 2013.