Multi-Robot Adversarial Patrolling: Facing Coordinated Attacks [*]

Efrat Sless
Department of Computer Science
Bar Ilan University, Israel
efrat.slas@live.biu.ac.il

Noa Agmon
Department of Computer Science
Bar Ilan University, Israel
agmon@cs.biu.ac.il

Sarit Kraus
Department of Computer Science
Bar Ilan University,Israel
sarit@cs.biu.ac.il

ABSTRACT

The use of robot teams is common for performing patrol tasks, in which the robots are required to repeatedly visit a target area (perimeter, in our case) controlled by an adversary, in order to detect penetrations. Previous work has focused on determining the optimal patrol algorithm when facing a general adversary that tries to penetrate once through the patrol path. There, the robots' goal is to detect penetrations, i.e., the robots do not change their behavior once a penetration is detected. Requiring the robots to physically inspect penetration attempts can have far reaching consequences on the performance of the patrol algorithm. Specifically, it creates vulnerability points along the patrol path that a knowledgeable adversary can take advantage of. In this work we investigate the problem of coordinated attacks, in which the adversary initiates two attacks in order to maximize its chances of successful penetration, assuming a robot from the team will be sent to examine a penetration attempt. We suggest an algorithm that computes the optimal robot strategy for handling such coordinated attacks, and show that despite its exponential time complexity, practical run time of the algorithm can be significantly reduced without harming the optimality of the strategy.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms

Keywords

Agent Cooperation::Multi-robot systems; Agent-based system development::Security aspects of agent systems;

1. INTRODUCTION

The problem of multi-robot patrol is well known in the robotic and agent community. Given an area and a team of robots, the robots should repeatedly visit the area in order to either maximize point-visit frequency criterion (e.g., [4, 5]) — frequency based patrol — or maximize the robots' chance of detecting penetrations through the patrol path (e.g., [1, 3]) — adversarial patrol. In this paper we concentrate on the latter, i.e., the robots' goal is to detect penetrations that are controlled by an adversary, in our case around a perimeter.

[*]This work was supported in part by ERC grant #267523.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

Previous work on multi-robot patrol has concentrated mainly on the detection problem [1, 3, 13]. There, the robots' task is to detect penetrations, i.e., once a penetration is detected, the robots report it and move on. The robots' world model does not change, thus their behaviors do not change. Therefore, the efficiency of the patrol strategy remains the same for any number of penetration attempts — coordinated or uncoordinated. However, in many realistic settings, the robot that detects the penetration has to handle it as well, for example by confining the penetrator or examining the penetration using additional sensors. Thus the penetration will necessarily influence The remaining team's ability to handle them, which makes the patrolling team more vulnerable to coordinated attacks.

When the adversary initiates a coordinated attack, it will try to penetrate through one location, and then (once a robot is removed from the team in order to handle that penetration attempt) it will try to penetrate again, this time through vulnerability points generated by the need to handle the first penetration. We distinguish between the behavior of the robots in three different phases: (Phase I) the steady-state prior to the extraction of the robot for handling the penetration, i.e., patrolling by k robots (Phase II) The reorganization phase: the team transfers from having k patrolling robots to having $k-1$, and (Phase III) The steady-state of patrolling by $k-1$ robots. Agmon *et al.* [1] established the optimal patrol algorithm for phases I and III. However they did not consider handling the penetration attempts and thus did not need to reorganize the robots. In this paper we examine the optimal behavior during the reorganization phase (Phase II), in which the robots are most vulnerable to additional attacks. One of the difficulties in finding the robots' strategies in the this phase is that the robots are not symmetric; one needs to compute specific strategies for each agent. This is in contrast to the steady phase where all agents follow the same strategy.

As the vulnerability points of the patrolling robots exist during the reorganization phase, a natural solution would be one that minimizes the duration of this phase. Surprisingly, we prove that this is not true. On the contrary, the shortest reorganization phase is inevitably deterministic and thus most vulnerable. Moreover, we show that as the reorganization time grows, the minimal probability of penetration detection along the perimeter grows until it levels up with probability of penetration detection of the steady state. Therefore, this allows us to focus our attention on computing an optimal strategy and limit the computation time.

In this paper we propose an algorithm that determines the optimal patrol strategy for the robots during the reorganization phase. The strategy is based on providing a probability distribution over all possible paths for each robot, leading it from its initial position to its final position (preparing for the next steady state). Unfortunately, the number of such paths is exponential in the size of the perimeter. However, we have shown that it is unnecessary to examine all possible paths, but only a small fraction of them, leading to another significant drop in the computation time of an optimal solution.

2. RELATED WORK

The problem of multi-robot patrolling has received considerable attention in the literature during the past decade. The problem has been investigated in different areas, for example perimeter patrol [1], patrol along an open fence [6], a $2D$ continuous environments [5] and graphs [4, 9, 10]. The problem has also been examined with respect to two different perspectives of the patrolling robots: *adversarial patrol* [1, 3, 11], where the robots' goal is to detect penetrations controlled by an adversary, and *frequency-based patrol* [4, 6, 9, 10], where the robots' goal is to optimize some point-visit frequency criterion, for example by maximizing the average frequency along the area. Another perspective is the game-theoretic domain [3, 13], which refers to a single robot in the patrol, but they also formulated their problems as non-linear optimization problems. Another approach is modeling the problem as a Stackelberg game, as in [14] which confronted a similar problem. They introduced a method of representing the exponential number of paths. However, in our work we assume that the penetration is not instantaneous, therefore we cannot use this representation. Moreover, they assume that the contribution of multiple patrol units covering the same edge is additive, thus the problem can be formulated as a linear programming problem, whereas in our settings multiple robots covering the same segment contribute the same as a single robot.

In this paper we concentrate on the problem of multi-robot adversarial perimeter patrol. In these types of problems, the adversarial model commonly depends on the adversarial knowledge. An adversary with full knowledge was considered in [1, 3], and an adversary with partial information was considered for example in [12].

In [2] it was shown that when the adversary has no knowledge about the patrolling robots and acts randomly, an algorithm maximizing the probability of penetration detection is the one that maximizes the point-visit frequency. Yet, to the best of our knowledge, the problem of coordinated attacks, i.e., handling more than one attack, has not been considered in previous.

The problem of handling events in frequency based patrol was considered, among others, by Elmaliach *et al.* [5] and by Fazli and Mackworth [7]. Jensen *et al.* [8] examined the problem of removing robots for re-charging. These events are predictable, and can be taken into consideration in advance as opposed to adversaries' attacks. Elmaliach *et al.* [5] considered the removal of robots due to failure, where they focused on minimizing the reorganization time. In all these cases, the robots' goal is to adjust to the new state with one less robot as quickly and efficiently as possible, by adopting a deterministic behavior. This behavior is necessarily vulnerable to additional penetration attempts by a knowledgeable adversary, thus cannot be used in our case.

3. PROBLEM SETTINGS

We are given k homogeneous robots $R = \{r_0, \dots, r_{k-1}\}$ that are required to patrol along a perimeter, i.e., a cyclic path around a closed polygon. The fence is partitioned into N segments $S = \{s_0, \dots, s_{N-1}\}$, with endpoints $ep_0, \dots, ep_{N-1}$, where each robot travels through one segment per time unit. s_i starts at ep_i and ends at $ep_i + 1$, The length of the segments and the robots' speed can differ from one segment to another, as long as the motion time required to travel through the segments is equal. Each robot can either move forward, backwards or stay put, at every step, where turning around has an associated cost, modeled in time. We denote the number of time units it takes the robots to turn around by τ. When an attempt to penetrate is detected, either by a robot or other sources, a robot must inspect it until the threat is removed, thus the robot is removed from the team. During that time it is still able to observe part of the perimeter (this value is quantified in Section 4).

3.1 Adversarial Model

The system consists of two coordinated adversaries A_0, A_1 (equivalent to one adversary initiating two attacks). Their joint goal is for at least one of them to successfully penetrate the perimeter, that is, without being detected by the robots. Both adversaries earn the same utility if they manage to achieve their joint goal. We assume the time it takes for each one of the adversaries to penetrate lasts t time units, during which it may be detected by a robot passing through the segment in which it resides in. In order to maximize their chances of successful penetration, the adversaries initiate a second penetration attempt if the first one is detected. We consider two cases for the second penetration with respect to the detection of the first attack:

1. **Known penetration period.** Second penetration initiated at some known time, or bounded time $[x_1, x_2]$ after the first penetration is detected.
2. **Unknown penetration time.** Second penetration is initiated at an unknown time.

We consider a strong adversarial model, in which the adversaries have full knowledge of the patrol strategy and the robots' positions and a-priory decide the segments to penetrate and the timing of both attacks.

3.2 Patrol Task

We define the Probability of Penetration Detection in a segment s_i, ppd_i, as the probability that an adversary trying to penetrate through s_i during t time units will be detected by some robot passing through s_i during that time period. We would like to find an optimal patrol algorithm for the robot team when confronting a strong adversary that will necessarily penetrate through the segment with minimal ppd, i.e., an algorithm that maximizes the minimal probability of penetration detection throughout the perimeter. Agmon *et al.* [1], proved that the optimal patrol algorithm that maximizes the minimal ppd along the perimeter for a team of k robots is one that requires the robots to be spread out uniformly (in time) along the perimeter , i.e., with a distance of d_k between every two consecutive robots. In their work, they define a nondeterministic patrol framework in which, at each time step, the robots either continue straight with a probability of p or turn around with a probability of $1-p$. Based on this framework, it is possible to find, in polynomial time, the optimal patrol strategy (characterized by the value p). We therefore distinguish between the behavior of the robots in three different phases: Phase I- the steady-state prior to the extraction of the robot for handling the penetration, i.e., patrolling by k robots; Phase II- the reorganization phase, where the team's composition changes from k patrolling robots to $k-1$ patrolling robots; and Phase III- the steady-state of patrolling by $k-1$ robots. Since the optimality of the patrol strategy is established in [1] for phases I and III,in this paper we focus our attention on the reorganization phase. We therefore handle the following problem:

Given k robots patrolling around the perimeter, where one robot is extracted from the team in order to handle a penetration at location s, determine the optimal behavior for the robots that will maximize the minimal probability of penetration detection along the perimeter.

4. THE REORGANIZATION PHASE

When a robot is extracted from the team to handle the first penetration, we wish to reorganize the remaining $k-1$ robots such that they will again be organized uniformly around the perimeter, to achieve optimal behavior [1], while the extracted robot remains close to the location of the initial penetration. The set of *final positions* of the robots is the set of positions in which all robots are placed uniformly along the perimeter. First, we are interested in determining the amount of time allocated for the reorganization phase, i.e., the time it will take the robots to reach their final positions. The duration of the reorganization phase is denoted by ρ. Note that the length of the path between a robot's current position and its final position varies from one robot to the other, and depends on its location relative to the penetration point (see Figure 1). A naive approach would be for each robot to go straight to its final position and wait until the reorganization phase ends, thus minimizing the time of the reorganization. However such an approach will create vulnerability points as there are segments that will not be visited during t time units, thus a knowledgeable adversary will manage to penetrate through those weak points. In the example in Figure 1, all points between the final positions of the robots and the current positions of the adjacent robots will not be visited during the reorganization phase using this naive algorithm. Thus these points can be chosen for penetration by the adversary. We are therefore interested in a non-deterministic algorithm that will guarantee that for each segment s_i around the perimeter, $\mathsf{ppd}_i > 0$ during the reorganization phase.

Determining final positions.

It is easy to see that as ρ grows, each robot has a greater range of segments it can visit during the reorganization phase, thus ppd_i potentially increases. Hence, theoretically, we would like ρ to have large values. However, this conflicts with the fact that we wish to minimize the reorganization phase length since an optimal patrol is achieved when all robots are organized uniformly. In order to find the ρ value that balances this trade-off, we start by finding the final positions for all the robots such that the reorganization time is minimized, i.e., the maximal distance traveled by some robot is minimized. We assume, without loss of generality, that r_0 is the extracted robot. The uniform distance between m robots is denoted by $d_m = \frac{N}{m}$. We also use D_i to denote the path length for robot i, $1 \le i \le k-1$ from its current position to its final position (which provides the minimal path length). The sign of D_i indicates the direction of the path: + for counter clockwise movement, - for clockwise. We begin by proving the following supporting lemma in order to find the final positions (by calculating the values of D_i 's). Let $D_{max} := \max_{1 \le i \le k-1} |D_i|$.

LEMMA 1. *For k robots that are spread uniformly around the perimeter with a distance of $d = N/k$ segments between every two consecutive robots, once robot r_0 is removed from the system, $D_j = D_1 - \frac{j-1}{k-1} d_k$ for $2 \le j \le k-1$, and $D_{max} = D_1 = \frac{1}{2}\frac{k-2}{k-1} d_k$.*

PROOF. After reorganizing the distance between robot r_j ,r_{j-1}, j=2..$k-1$ is $d_k + D_{j-1} - D_j$. The robots are distributed uniformly along their final positions, hence they are within a distance of $\frac{k}{k-1} d_k$ from each other, therefore $d_k + D_{j-1} - D_j = \frac{k}{k-1} d_k$, and we find that $D_j = D_{j-1} - \frac{1}{k-1} d_k$. It is easy to verify that $D_j = D_1 - \frac{j-1}{k-1} d_k$.

In order to show that the maximal value of D_j (D_{max}), we notice that there are two robots r_1, r_{k-1}, within a distance of $2d_k$, and the rest of the robots remain within a distance of d_k. Since $2d \ge \frac{k}{k-1} d_k$ for all $k \ge 2$, the minimal duration of reorganizing is achieved when those two robots will head in opposite directions in order to reduce the distance between them.

Thus $D_1 + D_{k-1} = 2d_k - \frac{k}{k-1} d_k = \frac{k-2}{k-1} d_k$ holds. The minimum path length is achieved when $D_1 = D_{k-1} = \frac{1}{2}\frac{k-2}{k-1} d_k$. Hence the minimum of the longest path length for all robots, denoted by min, satisfies that $min \ge \frac{1}{2}\frac{k-2}{k-1} d_k$. The maximum of $|D_j|$, is achieved at the boundary points $j = 2, k-1$ and $|D_2| < |D_1| = |D_{k-1}|$ holds, therefore $D_{max} = max|D_j| = D_1 = D_{k-1} = \frac{1}{2}\frac{k-2}{k-1} d_k$. Since D_{max} is the longest path length it holds that $min \le D_{max}$, hence $min = D_{max}$. □

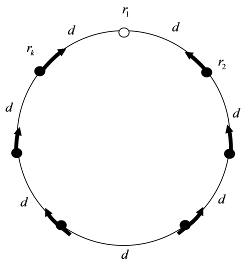


Figure 1: ***Illustrating of the deterministic reorganization phase. The current positions of the robots are represented by the black points. Each robot goes straight to its final position. The unvisited segments are vulnerable to attacks.***

Following Lemma 1, we can calculate the values of D_j as follows:

$$D_j = D_1 - \frac{j-1}{k-1} d_k = \frac{1}{2}\frac{k-2}{k-1} d_k - \frac{j-1}{k-1} d_k = \frac{\frac{1}{2}k - j}{k-1} d_k \quad (1)$$

The current position of r_i, $i = 1 \dots k-1$, is $i \cdot d_k$, thus we can calculate the final position of r_i (which is at a distance of D_i from its current position) as follows:

$$id_k - D_j = id_k - \frac{\frac{1}{2}k - i}{k-1} d_k = \frac{kd_k \cdot (i - \frac{1}{2})}{k-1} \quad (2)$$

Similarly we can compute the final position of each robot given its initial position, and the penetration segment.

Monitoring requirement from the extracted robot.

We are interested in visiting all segments during all periods of t time units. We wish to examine the range of segments close to the penetration location that cannot be visited during that time by the remaining robots, but can be observed by the extracted robot, or any other sensing element. The number of segments that cannot be visited during a period of t time units starting from t_1 equals $2(d_k - (t_1 + t))$. This is due to the fact that the number of segments between r_1, r_{k-1} is $2d_k$. During t time units, r_1 can visit at most t of the segments between r_1, r_{k-1}. The same holds for r_{k-1}. Therefore the number of remaining unvisited segments is $2(d_k - (t_1 + t))$. These are the segments that need to be monitored, either by the extracted robot, or other sensing elements (security cameras for instance).

5. PRELIMINARIES

Even after determining the final positions, there are still many ways each robot can travel from its current position to its final position during ρ time units. A deterministic approach is no good against a full-knowledge adversary. Unlike the steady state, where the world is symmetric, enabling each robot to execute the same patrol strategy, in our problem there is no symmetry thus it is necessary to calculate a strategy for each of the robots. Randomizing at each endpoint does not guarantee the robot will arrive to its final position at the end of the reorganization phase. Therefore we randomize over the possible paths that start at the current location of the robots, and end at the final position after ρ time units.

Modeling the problem by a graph.

Similar to [1], we model the problem as a directed graph in order to capture the directionality of the robots' movement (facing forward is different than facing backward, considering the time τ it takes the robots to turn around). The graph G is constructed as follows (see the illustration in Figure 2):
For each pair in the original problem of end point and direction we construct a vertex. To take into account the cost of turning, we construct chains of length τ between vertices with different direction.
$V = \{v|v = (ep_i, direction)\}$
$v = (ep_i, forward)$ $v' = (ep_i, backward)$
C =$\{$ chains with length of τ between u, u' and $u', u|$ u is an endpoint$\}$
E =$\{(v, u), (u', v')|u, v$ are endpoints of the same segment$\}\cup C$
$W : VxV \to R$
$W(u, v) = 1$ $(u, v) \in E$
$W(u, u') = W(u', u) = \tau$ the cost of weights between u, u'

Algorithm 1 calculates the number of paths of length $\mathcal{L}$ between two given endpoints in a given direction, using the adjacency matrix of G, as defined in [11]. u,v are the corresponding vertices to the directed endpoints.

Algorithm 1 NumberOfPaths($u, v, \mathcal{L}$)

```
1: Construct $G$
2: $A \leftarrow$ Adjacency matrix of $G$
3: Return $A^{\mathcal{L}}(u, v)$
```

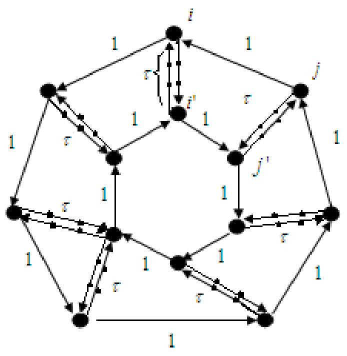


Figure 2: ***Example of a weighted graph that takes into account the cost of turning, τ.***

5.1 Algorithm For Finding All Paths

The behavior of the robots during the reorganization phase has to be random in order to handle a knowledgeable adversary. In this case, we randomize over the possible choices of paths leading the robots from their current position to their final position during ρ time units. First, we need to find the possible paths for each robot before further evaluation. Algorithm $FindPaths$ receives the current position v and the final position $goal$ of a given robot, and finds all paths with length of ρ. The algorithm considers the modeled graph G, the current position and direction, encompassed by v, and the final position and direction encompassed by $goal$. $path$ and $cost$ are the current path and its cost (initially empty and 0). The results are stored in $paths$ (initially empty).

Algorithm 2 FindPaths($G, v, goal, \rho, path, cost, paths$)

```
1: add $v$ to $path$
2: if $cost = \rho$ then
3:     add $path$ to $paths$
4:     Return
5: if $cost > \rho$ then
6:     Return
7: for each $v'$ in $neighbors(v)$ do
8:     if $numberOfPaths(v', goal, \rho - (cost + W(v, v'))) > 0$ then
9:         $FindPaths(G, v', goal, \rho, path, cost + W(v, v', paths))$
10: return $paths$
```

The algorithm recursively (DFS[1] based) generates the paths, each step continues only to branches that can yield a valid path.

The complexity of $NumberOfPaths$ is polynomial in N, i,e, $O(N^c)$ for some fixed c. The time complexity of $FindPaths$ is linear in output size, and is $O(N^{\rho+c})$, since the number of paths as calculated by $numberOfPaths$ is $A^\rho(v, goal)$, where A is the adjacency matrix of graph G. We can bound this value by defining matrix B_{NxN} such that $B_{ij} = 1$, $i = 1..N, j = 1\dots N$. Since $0 \leq A_{ij} \leq 1, i = 1\dots N, j = 1\dots N$. It holds that $0 \leq A_{ij} \leq B_{ij}, i = 1\dots N, j = 1\dots N$ and thus $A^\rho_{ij} \leq B^\rho_{ij}$, and $B^\rho_{ij} = N^\rho$.

5.2 ppd Computation

The values of the ppd depend on the probability of each path to be chosen. Suppose that for each robot we are given a set of paths and a distribution over this set, Algorithm $calcPPD$ computes the values of ppd's, at a given time and illustrates the influence of the distribution on the ppd values.
Input for $calcPPD$:
problem settings: S := segments set,R := robots set
$paths$:= set of all paths for all robots: $\{path_r\}$, where $path_r$ is the set of possible paths for robot r
$path[t_1 : t_2]$:= sub-path of $path$ between t_1, t_2
t_a := inspected time of penetration initiation
P := The sets of distribution over paths for all robots, $\{P_r\}$ where P_r is the distribution of robot r over $path_r$.
$prob_r(s)$:= the probability for robot r to visit segment s.
Result: $ppd[t_a]$

Algorithm 3 CalcPPD($paths, t, S, R, t_a, P$)

```
1: for each $s$ in $S$ do
2:     $ppd(s) \leftarrow 0$
3:     for each $r$ in $R$ do
4:         $prob_r(s) \leftarrow 0$
5:         for each $path$ in $paths_r$ do
6:             $I(s, r, path)[t_a] \leftarrow \begin{cases} 1 & path[t_a : t_a + t] \text{ visits } s \\ 0 & \text{otherwise} \end{cases}$
7:             $prob_r(s) \leftarrow prob_r(s) + P_r(path) \cdot I(s, r, path)[t_a]$
8:         $\mathsf{ppd}(s) \leftarrow \mathsf{ppd}(s) \cdot (1 - prob_r(s)) + prob_r(s)$
9: return ppd
```

[1]Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking.

$prob_r(s) = \sum_{path} P_r(path) \cdot I(s, r, path)[t_a]$, i.e, the probability that r will visit segment s between $[t_a, t_a + t]$. Next we prove a lemma that will assist us in iteratively calculating the ppd for each segment, iterating the contribution of each robot.

LEMMA 2. *Assume that $F_1(s) = prob_{r_1}(s)$, and that $F_{j+1}(s) = F_j(s) \cdot (1 - prob_{r_{j+1}}(s)) + prob_{r_{j+1}}(s)$, $1 \le j \le k-2$ then $\mathsf{ppd}(s) = F_{k-1}(s)$.*

PROOF. We prove by induction that $F_j(s) = 1 - \prod_{l=1}^{j}(1 - prob_{r_l}(s))$ This statement holds for $F_1(s)$; We will now prove that if it holds for $F_j(s)$ it holds for $F_{j+1}(s)$ $F_{j+1}(s) = F_j(s) \cdot (1 - prob_{r_{j+1}}(s)) + prob_{r_{j+1}}(s) = (1 - \prod_{l=1}^{j}(1 - prob_{r_l}(s)))(1 - prob_{r_{j+1}}) + prob_{r_{j+1}}(s) = 1 - prob_{r_{j+1}}(s) - \prod_{l=1}^{j+1}(1 - prob_{r_l}(s)) + prob_{r_{j+1}}(s) = 1 - \prod_{l=1}^{j+1}(1 - prob_{r_l}(s))$, as required.

It holds that $\mathsf{ppd}(s) = 1 - \prod_{l=1}^{k-1}(1 - prob_{r_l}(s))$, which is the complement of the probability that none of the robots will visit segment s. Therefore $\mathsf{ppd}(s) = F_{k-1}(s)$. □

6. BOUNDED PENETRATION PERIOD

In this section we consider the case where the time of the second penetration, denoted by t_a, is bounded, and formally that $x_1 \le t_a \le x_2$ for some known x_1, x_2.

As stated in the preceding sections, our goal is to find the minimal ρ that satisfies the trade-off between increasing the reorganization time that will increase the ppd, and minimization of the reorganization time (since the optimal patrol algorithm for $k-1$ robots is known to have the robots placed uniformly along the perimeter). We therefore want to find the minimal ρ value that provides the *maxmin*ppd between $[x_1, x_2]$.

We provide a series of constraints regarding the value of ρ. First, it is necessary that $\rho > t$, otherwise there are segments that will not be visited during a period of t time units from the beginning of the reorganization phase.

The existence of a path from a current position to the final position that visits a segment guarantees that if there is a probability greater than zero of choosing it, then the probability of penetration detection at that segment is greater than zero.

As stated in previous sections, using a greater ρ improves the results. Since at some point increasing ρ yields the same sub-paths of a fixed length, the results will remain the same. We proceed to bound this point in time, and use $b[x_2]$ to denote the bound. That means, using $\rho = b[x_2]$ yields the *maxmin*ppd, and any greater ρ yields the same. For that we denote by $|path|$ the length of $path$.

LEMMA 3. *$b[x_2] := 2x_2 + 2t + D_{max} + \tau$ is a bound for the minimal ρ such that maxmin*ppd *remains the same for every $\tilde{\rho} > \rho$.*

PROOF. In order to prove that $b[x2]$ is a bound, it is sufficient to prove that: $\{path[x_1 : x_2] : |path| = \rho\} = \{path[x_1 : x_2] : |path| = b[x_2]\}$ for $\rho \ge b[x_2]$ which means that sub-paths of length x_2 from all paths of length ρ are the same as sub-paths of paths of length $b[x_2]$, hence they have the same ppd. A path of length $x_2 + t$ can be distant at most $x_2 + t + D_{max}$ from its final position. In order for a path of length ρ to be of that distant, ρ needs to be greater than $2x_2 + 2t + D_{max} + \tau$. □

Optimization problem.

After setting all preliminaries, we can formulate the problem as a QCP (quadratic constraint problem). The formulation considers the contribution of each $path_i$ of robot j to segment s and maximizes the minimal value of ppd's of all segments. The decision variables are the probabilities of choosing each path $P_r(path)$, and extra variables to formulate the problem as a QCP. The paths for each robot are computed by: $path_r = FindPaths(G, currentPos, finalPosition(currentPos), \rho, [], 0)$ The constraint of $maxmin_i\ \mathsf{ppd}_i$ is equivalent to stating that $\{\max m | \mathsf{ppd}_i \ge m\}$.

The ppd values are computed iteratively using Lemma 2, to generate quadratic constraints, instead of higher order.

$$\begin{array}{ll}
max\ m & \\
\text{S.T} & \\
s \in S, t_a \in [x_1, x_2] & \mathsf{ppd}_s[t_a] = calcPPD(paths, t, S, R, t_a, \\
. & \qquad\qquad \{P_r\}) \\
r \in R, & \sum_{path} P_r(path) = 1 \\
\forall path, r \in R & P_r(path) \ge 0 \\
s \in S, t_a \in [x_1, x_2] & \mathsf{ppd}_s[t_a] \ge m
\end{array}$$

Patrol algorithm.

The final algorithm is PatrolRearrange which computes the optimal distribution and randomizes the path for each robot.

Algorithm 4 PatrolRearrange$(R, S, \{init_r\}, t, \tau, x_1, x_2)$

```
1: construct $G$
2: for each $r$ in $R$ do
3:     $goal \leftarrow CalcFinalPosition(init_r)$
4:     $\rho \leftarrow max\{2t + 2x_2 + D_{max} + \tau, \frac{k}{k-1}d + \tau\}$
5:     $paths_r \leftarrow FindPaths(G, init_r, goal, \rho, [], 0)$
6: $P_r \leftarrow solve(paths, k, n, x_1, x_2)$
7: return random $path$ from $paths_r$ with distribution $P_r$
```

The time complexity of the problem is the complexity required to solve a non-linear optimization problem with $O(k \cdot N^{2t} \cdot (x_2 - x_1))$ constraints and variables. This differs between solvers. The call to $solve$ uses a solver to solve the non-linear optimization problem, which takes into consideration the paths of all robots, the number of robots, the number of segments, and the inspected area.

6.1 Lower bound for ppd values

In the following lemma we prove a lower bound on the minimal probability of penetration detection guaranteed by Algorithm PatrolRearrange during the reorganization phase. This lower bound is relatively high, and the implied "cost" to the system is discussed in the next section.

LEMMA 4. *If $t_a = x$, during the reorganization phase for some known x then the* ppd *values during the first t time units in the reorganization phase guaranteed by performing* PatrolRearrange *satisfies that $\mathsf{ppd}_l = 0.5$ for $l = d_u(k) - t \ldots n - (d_u(k) - t)$.*

PROOF. Each robot has a path where during t time units, it visits segments that are at a distance of t forward and a path that visits backwards. Since $2t \ge d + \tau$, a pair of adjacent robots has paths that cover all segments between them during t time units. Assume we determine two possible paths for each robot: one in which the robot visits the segments forward to its position, and one that it visits segments backward from its position. There exist such paths for each robot such that any two robots ensures that all segments at a distance of d between them are covered. If each of these two paths is chosen with a probability of 0.5 for each robot, then the values of ppd's in this example are at least 0.5, since all segments that are at a distance of t are covered. This implies that the resulting ppd's from solving the optimization problem necessarily satisfy that $\mathsf{ppd}_l \ge 0.5$ for $l = (d_u(k) - t) \ldots n - (d_u(k) - t)$. □

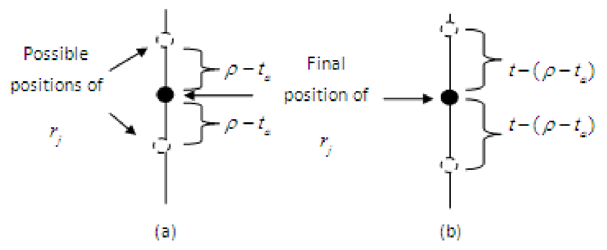


Figure 3: ***(a) represents the most distant positions of*** r_j ***after*** $t_a \leq \rho$ ***time units (during reorganization). (b) represents the most distant positions of*** r_j ***during the remaining period of*** t ***(after the reorganization phase).***

6.2 Unknown Penetration Time

In this section we consider the time of the second penetration attempt to be unbounded. In the previous case we maximized the minimal ppd during a bounded period. Which enabled the selection of a ρ that results the optimal solution, even higher than the steady state. This is due to the fact that we overlooked the segments afterwards.

As an example, consider the case in which $N = 84$, $k = 7$ and $t = 8, t_a = 0$. In this case, $\rho_0 = 15$, which means that although we handle the first t time units, the remaining $\rho_0 - t = 7$ time units are overlooked. This allows the robots to be very well prepared against a coordinated attack, but not against general attacks that may occur at any time. Following this example, the minimal ppd in the steady state for $k = 7$ robots is 0.15, the value for $k - 1 = 6$ robots is 0.05, and during the (first t time units of the) reorganization phase is 0.5. However, if a penetration is initiated after 12 time units (and not at time 0), the minimal ppd drops to 0.

We start by proving that there are points of vulnerability during the reorganization phase, in every possible choice of a single ρ for reorganization. In the bounded case we could increase ρ in order to avoid them, because the penetration time was bounded. The problem arises when the penetration can occur at any time during the reorganization phase, even during those vulnerability points. Therefore, increasing ρ in this case would not solve the problem.

LEMMA 5. *For* $\rho - t \leq t_a \leq \rho$*, there exists a segment* i *such that* $\mathsf{ppd}_i[t_a] = 0$ *for* $\rho - \frac{1}{2}d_{k-1} \leq t_a \leq \rho - t + \frac{1}{2}d_{k-1}$.

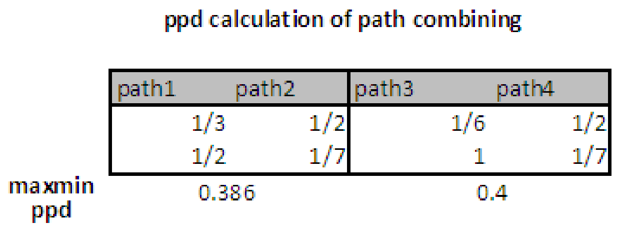
PROOF. At time t_a each of r_j is at a distance of $\rho - t_a$ from its final position. During the remaining $t - (\rho - t_a)$ time units, the robots are in the steady phase, and each r_j can be at a distance of at most $t - (\rho - t_a)$ from its final position. Thus during t time units each robot is at distance of at most $max(\rho - t_a, t - (\rho - t_a))$. This is illustrated in Fig. 3 If none of the segments is overlooked then each pair of robots cover all of the segments between their final positions, at a distance of d_{k-1}. Thus it is essential that $2max(\rho - t_a, t - (\rho - t_a)) \geq d_{k-1}$, which implies either $t_a \leq \rho - \frac{1}{2}d_{k-1}$ or $t_a \geq \rho - t + \frac{1}{2}d_{k-1}$, meaning that when $\rho - \frac{1}{2}d_{k-1} \leq t_a \leq \rho - t + \frac{1}{2}d_{k-1}$ segments with $\mathsf{ppd}_i[t_a] = 0$ exist. □

The above lemma implies that if the opponent has full knowledge of the patrol scheme, and of ρ in particular, it can choose to penetrate during that time and it is guaranteed to succeed. As a corollary ρ must be randomized in order to ensure $\mathsf{ppd}_i[t_a] \neq 0$ for all t_a.

Since all robots must end the reorganization phase at the same time, all robots must draw paths for the same ρ. This means that each robot can draw a path only after a ρ is drawn.

Optimization Problem.

Similar to the first case of bounded time, we wish to formulate the optimization problem in this case. As illustrated by the following example in Fig. 4 calculating the optimal paths probabilities for each ρ and combining them together might yield a solution that is worse than computing all probabilities together. Therefore all paths of different ρ's must be considered together. The set of multiple ρ's is denoted by Γ.


ppd calculation of path combining

	path1	path2	path3	path4
	1/3	1/2	1/6	1/2
	1/2	1/7	1	1/7
maxmin ppd	0.386		0.4	

Figure 4: ***Assuming there are four paths with ppd's as described above, for some two segments. Each of*** $path_3, path_4$ ***has a smaller*** $maxmin_i \mathsf{ppd}_i$ ***than*** $path_1, path_2$***. Yet, when maximizing the combination of each pair*** $maxmin_i \mathsf{ppd}_i$ ***of*** $path_3$ ***and*** $path_4$ ***is greater.***

$$
\begin{array}{ll}
max\ m & \\
\text{S.T} & \\
s \in S, t_a \in [x_1, x_2], & \mathsf{ppd}_{s,\rho}[t_a] = calcPPD(paths_\rho, t, S, R, t_a, \{P_r\}) \\
r \in R, & \sum_{path} P_{r,\rho}(path) = 1 \\
\forall path, r \in R & P_{r,\rho}(path) \geq 0 \\
\sum_\rho q_\rho = 1 & \\
\forall path, \rho \in \Gamma & q_\rho(path) \geq 0 \\
s \in S, t_a \in [x_1, x_2] & total_p pd_s[t_a] = \sum_\rho ppd_{s,\rho}[t_a] \\
s \in S, t_a \in [x_1, x_2] & total_p pd_s[t_a] \geq m
\end{array}
$$

The optimal patrol algorithm is an extension of PatrolRearrange. After solving the optimization problem, first a ρ is drawn with probability of q_ρ, and then a path with length ρ is drawn for each robot.

6.3 Bounding the reorganization time

In the previous section we proved a bound for the value of ρ that yields the maximum minimal ppd. In this section we prove that there exists a ρ that bounds the improvement towards the *maxmin*ppd.

LEMMA 6. *When* $max\Gamma \rightarrow \inf$*, the solution of the optimization problem converges to the maxmin*ppd *of the reorganization phase.*

PROOF SKETCH. The solution is monotonic with respect to ρ, since if a larger ρ does not improve the results it will have a probability of 0. In addition the solution is bounded (it represents probability). Therefore the process of adding more ρ's converges. Due to the fact that every possible path is contained in $\{paths\}_\rho \rightarrow \inf$, the optimal paths for the reorganization are also contained in this set. Consequently a solution to an optimization problem containing these paths will yield the *maxmin*ppd of the reorganization phase. □

The following corollary states the existence of a bound for adding larger ρ's.

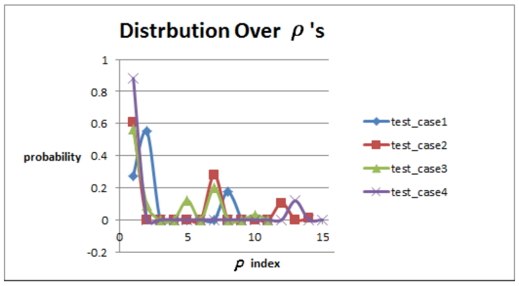


Figure 5: *Probability distribution over ρ's in optimal solutions with $\epsilon = 0.01$, for the following settings: (1) $k = 8$, $d = 7$, $t = 5$ (2) $k = 4$, $d = 6$, $t = 5$ (3) $k = 6$, $d = 5$, $t = 4$ (4) $k = 3$, $d = 8$, $t = 7$*

COROLLARY 1. *For every $\epsilon > 0$ there exists a ρ_0 such that adding any $\rho > \rho_0$ would not change more than ϵ of the solution.*

PROOF. This is proven by using Cauchy's criterion on previous lemma. □

This is demonstrated in Fig. 5, which shows the finite bound for achieving the optimal ppd for a certain ϵ, and that lower ρ values have a higher probability.

6.4 Reducing the Problem Size

So far, our suggested algorithm considers every possible path for the optimization problem. Not only does this mean a very large input is considered, it is also unnecessary. Some of these paths clearly have less contribution than others, and there is no need to consider them in the optimization problem.

Definition 1. A path $path_1$ dominates $path_2$ if $I(s, r, path_1)[t_a] \geq I(s, r, path_2)[t_a]$, for every $s \in S$, $0 \leq t_a \leq \rho$.

Let P_r be a distribution function over the paths for robot r. We wish to examine the effect of transferring the probability from $path_2$ to $path_1$ that dominates it. We denote the resulting distribution function by $\tilde{P}_r$, and the probability of robot r to visit segment s according $\tilde{P}_r$ by $\widetilde{prob}_r(s)$.

LEMMA 7. *If $path_1$ dominates $path_2$ then $\widetilde{prob}_r(s)[t_a] \geq prob_r(s)[t_a]$, for every $s \in S$, $0 \leq t_a \leq \rho$.*

PROOF. $\widetilde{prob}_r(s)[t_a] - prob_r(s)[t_a] = \sum_{path} \tilde{P}_r(path) \cdot I(s, r, path)[t_a]$- $\sum_{path} P_r(path) \cdot I(s, r, path)[t_a]$ = $(P_r(path_1) + P_r(path_2) \cdot I(s, r, path_1)[t_a])$ - $P_r(path_2) \cdot I(s, r, path_2)[t_a]$ $= (I(s, r, path_1)[t_a] - I(s, r, path_2)[t_a]) \cdot P_r(path_2) + P_r(path_1) \cdot I(s, r, path_1)[t_a]$. Since $path_1$ dominates $path_2$ the above expression is greater than 0. □

COROLLARY 2. *If $path_1$ dominates $path_2$, then $calcPPD(paths, t, S, R, t_a, \tilde{P}_r) \geq calcPPD(paths, t, S, R, t_a, P_r)$.*

PROOF. The difference between the ppd's equals $(1 - \prod_l (1 - \widetilde{prob}_l(s))) - (1 - \prod_l (1 - prob_l(s))) = \prod_l (1 - prob_l(s)) - \prod_l (1 - \widetilde{prob}_l(s)) = \prod_{l \neq r} (1 - prob_l(s))(1 - prob_r(s) - (1 - \widetilde{prob}_r(s))) = \prod_{l \neq r} (1 - prob_l(s))(\widetilde{prob}_r(s) - prob_r(s))$. By Lemma 7 the above expression is greater than 0. □

By Corollary 2 we can perform a Pareto optimization over the paths for each robot and examine only these paths in the optimization problem.

Number of Pareto-paths.

In this section we discuss the complexity of finding all Pareto-paths. As the following lemma states, evidently the number of Pareto-paths can be exponential with respect to ρ. We consider the case where $t = 1$, $N \geq 3$.

LEMMA 8. *The number of Pareto-paths in the above case is $\Omega(4^{\rho})$.*

PROOF. Let us consider 4 sub-paths with a length of 12:

1. $s_2, 0, s_2, s_1, 0, s_1, s_2, 0, s_2, s_1, 0, s_1$
2. $s_2, 0, s_2, s_1, 0, s_1, s_2, s_3, 0, s_3, s_2, 0$
3. $s_2, s_3, 0, s_3, s_2, s_1, 0, s_1, s_2, 0, s_2, 0$
4. $s_2, s_3, 0, s_3, s_2, 0, s_2, s_3, 0, s_3, s_2, 0$

In this case the contribution to segment s_i at time t_1 is 1 if $path[t_1] = s_i$ Therefore, these paths are not dominated by each other. Moreover, there cannot be a valid path that has the same contribution except for at least one 0 that is replaced.

The starting and ending position of all paths is ep_1 facing forward. Therefore we can construct paths by concatenating these sub-paths. Every combination of these sub-paths cannot dominate another combination since their sub-paths are non-dominating. Thus, we receive a sub-set of the Pareto-paths. The number of such combinations for paths of length ρ is $4^{(\rho/12)}$. □

7. EXPERIMENTAL RESULTS

In this section we analyze the results of the experiments we conducted with more than 11,000 settings for the problem of reducing input size.

In Section 6.4 we have shown that, theoretically, the number of pareto paths can be exponential in the input size. However, in this section we show that *practically*, in most cases the number of pareto paths is significantly lower than this theoretical upper bound.

The number of Pareto-paths depends on t, ρ, $dist$, N, where $dist$ is the actual distance between a robot's current and final position.

We have examined over 11,000 different settings, where the examined parameters were: $k = 2..10$,$d = 3 \ldots 6$, $t = 1 \ldots d - 1$, $\rho = 2 \ldots 12$, $dist = 0 \ldots \rho$.

Fig. 6 illustrates that as t grows the number of Pareto-paths drops exponentially. Fig. 7 and Fig. 8 show that even when the number of Pareto-paths grows exponentially as ρ grows, the number of Pareto-paths is much smaller than the number of paths.

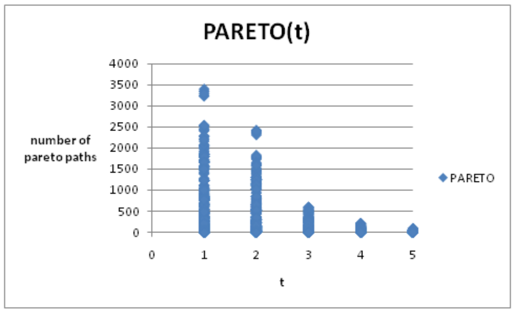


Figure 6: *Exponential drop in the number of Pareto-paths with respect to t.*

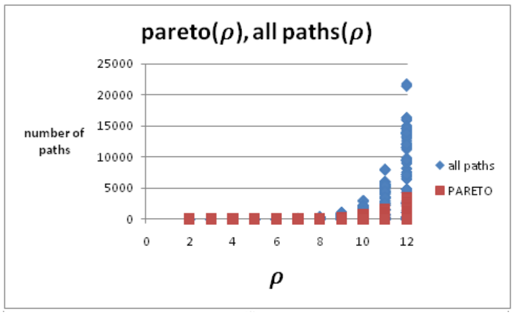


Figure 7: ***Number of paths and Pareto-paths as a function of*** ρ.

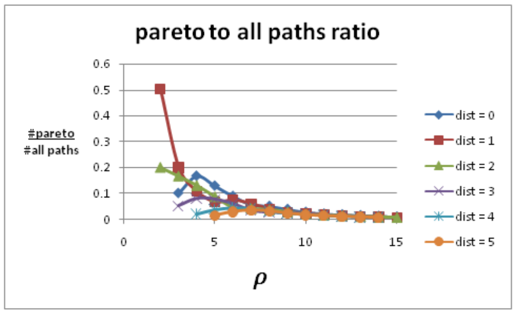


Figure 8: ***Exponential drop in the ratio between the number of Pareto-paths and all paths, with respect to*** ρ***, smaller*** $dist$ ***yields higher ratio. In this case*** $k = 3, d = 3, t = 2$.

8. CONCLUSIONS

We examined the problem of preparing against a coordinated attack. First we considered a bounded period of penetration attempts, which enables the allocation of a single reorganization time, as large as needed. We presented its optimization problem, and provided a proven bound for the reorganization time. For a penetration attempt where the time is unknown, we showed that it is not suffice to use a single reorganization time, and it must be randomized in order to avoid vulnerability points. The resulting optimization is an extension to the first case, and not an optimization over the results of the first case. The ppd during the reorganization phase wont be greater than the ppd of the steady state afterwards. Any results that are greater than that would have no effect, as the adversary would then choose to penetrate at the steady state. Another interesting result is that when randomizing over multiple ρ's using the deterministic approach with a high probability yields higher values of ppd. This is due to the fact that the ppds of the steady state are proven to be optimal, and the deterministic approach minimizes the reorganization time. When using multiple ρ's we are able to avoid the vulnerability points that are created by the deterministic approach. Then we introduced a method of reducing the number of paths, by using Pareto-paths. The number of these paths is exponential in the worst case, but considerably small in practice. Moreover, a bound for ρ exist.

We limited this work to randomization of the paths towards the steady state which minimizes the distance traveled by the robots in the reorganization phase. Future work warrants examination of randomization over the possible final positions. Moreover, it would be interesting to examine randomization during the reorganization phase (without having the robots commit to a path once the first attack is detected, as assumed in this work). Though we concentrated on two adversaries, the extension of multiple adversaries is straight forward. Another venue would be to examine the influence of the time of inspection of a robot, which might lead to a more robust patrol against multiple attacks.

9. REFERENCES

[1] N. Agmon, G. Kaminka, and S. Kraus. Multi-robot adversarial patrolling: facing a full-knowledge opponent. *Journal of Artificial Intelligence Research*, 42(1):887–916, 2011.

[2] N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceeding of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 55–62, 2008.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceeding of the Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 57–64, 2009.

[4] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, pages 302–308, 2004.

[5] Y. Elmaliach, N. Agmon, and G. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3):293–320, 2009.

[6] Y. Elmaliach, A. Shiloni, and G. Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *Proceeding of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 63–70, 2008.

[7] P. Fazli and A. Mackworth. Multi-robot repeated boundary coverage under uncertainty. In *IEEE International Conference on Robotics and Biometrics (ROBIO)*, pages 2167–2174, 2012.

[8] E. Jensen, M. Franklin, S. Lahr, and M. Gini. Sustainable multi-robot patrol of an open polyline. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4792–4797, 2011.

[9] A. Machado, G. Ramalho, J. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Multi-agent Based Simulations (MABS)*, pages 155–170, 2003.

[10] J. Marier, C. Besse, and B. Chaib-draa. Solving the continuous time multiagent patrol problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 941–946, 2010.

[11] J. Rodrìgueza. On the laplacian spectrum and walk-regular hypergraphs. *Linear and Multilinear Algebra*, pages 51:285–297, 2003.

[12] P. Villacorta and D. Pelta. Exploiting adversarial uncertainty in robotic patrolling: A simulation-based analysis. *Advances in Computational Intelligence*, pages 529–538, 2012.

[13] Y. Vorobeychik, B. An, and M. Tambe. Adversarial patrolling games. In *Proceeding of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1307–1308, 2012.

[14] Z. Yin, A. X. Jiang, M. P. Johnson, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, M. Tambe, and J. P. Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. *AI Magazine*, 33(4):59–72, 2012.