Multi-Robot Adversarial Patrolling: Handling Sequential Attacks

Efrat Sless Lin

Department of Computer Science

Bar Ilan University,Israel

Noa Agmon

Department of Computer Science

Bar Ilan University,Israel

Sarit Kraus

Department of Computer Science

Bar Ilan University,Israel

Abstract

Robot teams are commonly used for security tasks, where they are required to repeatedly monitor an area in order to prevent penetrations, initiated by an adversary. Current research in this field focuses mainly on *detecting* penetration attempts, but not on *responding*. Requiring the robots to also inspect and handle the penetrations has a significant impact on the patrol, as each penetration attempt also influences the robots' behavior, making them vulnerable to multiple attacks. Moreover, a knowledgeable adversary can initiate two sequential attacks, where the second attempt exploits the vulnerable points caused by the requirement that a robot handle the first penetration attempt. In this work, we consider the problem of sequential attacks and examine different robot policies against such adversarial behavior. We provide an optimal patrol strategy for various penetration attempt patterns. Our novel approach considers a full history-length policy, while previous work only handled very limited lengths of history. The use of a longer history improves the results. Moreover, we show how to significantly reduce, in practice,

Email addresses: efrat.sless.lin@gmail.com (Efrat Sless Lin), agmon@cs.biu.ac.il (Noa Agmon), sarit@cs.biu.ac.il (Sarit Kraus)

the exponential space state of the problem, while maintaining the optimality of the solution.

Keywords: Multi-Robot Patrolling, Adversarial Modeling

1. Introduction

Robotic patrolling is becoming more and more prominent in the field of security [1, 2, 3, 4, 5]. In such problems, a team of robots is required to repeatedly monitor an area (a perimeter in our case) in the presence of an adversary who is trying to penetrate it. Such patrolling is applicable for guarding high security facilities, neighborhood watch, and more.

Previous work has focused on the robots' goal to detect penetration attempts [1, 6, 7]. That is, once a penetration is detected, the detecting robot reports it and leaves the penetration point to continue its patrol. Therefore, sequential penetration attempts, coordinated or not, do not influence the patrol strategy and its performance. However, in realistic scenarios, the robot might be required to handle the penetration as well, instead of leaving it. Possible ways to handle a penetration, for example, are to scan the penetration point with additional sensors or to confine the intruder.

In an optimal patrol strategy for patrolling a perimeter, as established by Agmon *et al.* [1], the robots are distributed uniformly along the perimeter and follow the same strategy, in a coordinated manner. However, they did not consider the need to handle a penetration attempt. Extracting a robot from the patrolling force to handle a penetration attempt means that an attack changes the structure of the team and causes them to be vulnerable to a consequent attack. Therefore, an adversary which knows the robots' patrol strategy and their locations can initiate two coordinated attacks in order to exploit the robots' responses.

After initiating the first attack at one location, the adversary will try to penetrate again through the vulnerable points that are caused by the need to handle the first penetration. Thus, the remaining robots should reorganize to an optimal patrol strategy. In order for the robots to start reorganizing, they are required to communicate reliably, either with each other, or to a center of command. Three different phases should be considered: Phase I, the steady state for k patrolling robots prior to the penetration attempt and extraction of the robot; Phase II, the reorganization phase (transitioning from having k to $k-1$ patrolling robots); and Phase III, the steady-state for $k-1$ patrolling robots.

An optimal behavior maximizing probability of penetration detection for each steady state (for k and $k-1$ robots) is well established (e.g., [1]), thus we focus on achieving optimal behavior during the reorganization phase (Phase II), in which the robots are most vulnerable to additional attacks. As an optimal patrol strategy is achieved for $k-1$ robots after the reorganization phase ends [1], an intuitive solution would be to minimize the duration of this phase. However, the shortest reorganization phase results in a deterministic solution, meaning it is the most vulnerable to a knowledgeable adversary. Moreover, we show that when the second penetration time is unbounded a-priori, any single reorganization time creates a vulnerable strategy and there is a need to randomize over different possible durations in order to achieve an optimal patrol.

There are two common approaches for the multi-robot patrolling problem. The first maximizes the point-visit frequency criterion (e.g., [8, 9]), assuming that the robots do not act against a knowledgeable adversary. The second maximizes the robots' probability of detecting penetrations through the patrol path, originated by an adversary (e.g., [1, 6]). Since we are dealing with knowledgeable adversaries we consider the latter approach for the patrol strategies. A patrol strategy defines the probabilities for the robots' next actions. The model for the patrol strategy is defined by the robots' patrol policy, for example a Markovian policy. The optimal patrol strategy is determined by the optimization criteria.

For the most part, only first-order Markovian models have been considered in the literature [1, 3, 10, 11], which means that the previous locations of the robots have no effect on their next action. It is clear that the longer the history length of the paths, the results achieved from the patrol can only improve, i.e. better penetration detection (or the same, at worst). However, as shown in previous work, the ability to increase the history length is very limited [2, 6, 12]. This is simply due to the fact that increasing the history length exponentially increases the problem size, making it unsolvable. Whereas, we manage to consider high-order Markovian policies for robots with full history lengths of their paths.

In this work we present a novel approach that is able to consider the full history length (the complete paths), even when the use of smaller history lengths fails to produce results. Our approach is based on performing preprocessing over all possible paths for each robot, which leads the robot from its current position to its final position where it finishes the reorganization phase, and then we find the optimal probability distribution over the paths for each robot. As the number of all possible paths is exponential in the amount of time allocated to the reorganization phase, we show that it is unnecessary to consider all of them, but only a small portion, without harming the optimality of the solution. This leads to a significant drop in the state space of the problem.

The main contribution of this work is that it addresses the problem of sequential attacks, in a full knowledge adversarial environment. Requiring robots to both detect and respond to events has not been considered in previous work with other adversarial models. An additional contribution is the technical ability to consider higher Markovian order policies for the robots, whereas in most previous works only the first order Markovian has been considered. The results are also supported by extensive simulations.

The rest of this work is organized as follows. In Section 2 we review previous works related to our research. Section 3 includes a description of the problem settings, the adversarial model and the patrol task. In Section 4 we define the reorganization phase, and we lay the foundations for this work in Section 5. In Section 6 we provide a polynomial optimal patrol algorithm for the case in which the time between the sequential attacks is bounded. The unbounded case is handled in Section 7, where we present our optimal algorithm and a Pareto optimization method for reducing the problem size. In Section 8 we show how we have implemented the optimization problem and conducted experiments with perimeters of different lengths (up to 84 segments) as well as experiments with more than 11,000 settings to show the efficiency of the Pareto optimization in practice. Finally, we conclude our work and suggest venues for future research in Section 9.

2. Related Work

The problem of multi-robot patrolling has received considerable attention in the literature over the past decade. The problem has been investigated in different areas, for example perimeter patrol [1], patrol along an open fence [13], $2D$ continuous environments [9] and graphs [8, 14, 15]. The problem has also been examined with respect to two different perspectives of the patrolling robots: *frequency-based patrol* [8, 13, 14, 15], where the robots' goal is to optimize some point-visit frequency criteria, (e.g., by maximizing the average frequency along the area), and *adversarial patrol* [1, 2, 6], where the robots' goal is to detect penetrations controlled by an adversary.

For optimizing frequency criteria, the *Idleness* notion was introduced in [8, 14, 15], which marks the time between two consecutive visits of some robot at a point. Marier *et al.* [15] considered online algorithms for solving the multi-agent patrol problem. They modeled the problem as a graph where the travel time of each edge is defined by a probability distribution which is a generalization of a certain travel duration. However, online algorithms optimize only for the current state at each time point (agents' current locations and *Idleness* of points) and, in order to determine the next state, consider only a subset of states reachable from said point. This, however, may lead to non-optimal results. Thus, in our approach we consider all possible states in order to ensure optimality.

Elmaliach *et al.* [9] considered patrolling a 2D area, determined the cyclic path, and placed the robots uniformly in order to achieve uniform frequency. In this type of problem, the resulting (optimal) patrol strategy would be deterministic. Such an approach is good for non-adversarial domains, or for random behavior of the adversary. It has been shown by Agmon *et al.* [16] 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. In more realistic scenarios, the adversary has some knowledge about the patrol. Previous work considered different adversarial models. An adversary with full knowledge was considered in [1, 6], and an adversary with partial information was considered, for example, in [5]. Villacorta *et al.* [5] followed an experimental approach to study deviations from the optimal behavior of the adversary, contrary to our assumption that a full knowledge adversary will be the best responder and attack the most vulnerable point.

A setting similar to our problem was investigated by Agmon *et al.* [1]. They considered the problem of perimeter multi-robot patrol in the presence of a full knowledge adversary. The authors modeled the problem as a Markov chain, and developed a polynomial-time algorithm for determining the optimal patrol strategy that maximizes the robots' chances of detecting the adversary. However, their solution deals only with the task of detecting one attacker at a time.

Another approach to adversarial patrol applies game-theory [2, 6, 7, 11, 17, 18]. In these works the problem is modeled as a Stackelberg game – a game consisting of a leader that sets its strategy first, and a follower that sequentially responds. Basilico *et al.* [6] considered a single robot patrolling along a graph which contains targets of varied interests of the adversary. They modeled the problem as an extensive form game with an infinite horizon, and adopted a leader-follower solution concept. The patrolling robot is considered the leader, and the adversary the follower that can observe the leader's actions and acts as the best responder. The adversary is considered to have full knowledge, and a penetration time greater than zero. The authors formulated a non-linear optimization problem to find the robot's optimal randomized strategy. As stated by the authors, non-linear problems are generally NP-hard. Though likely to be NP-hard, non-linear formulations are common and appear in addition in [3, 7, 11]. One of the models that we consider in our work is similar to the formulation devised by Basilico *et al.* [11], which extended the work of Basilico *et al.* [6] to consider multiple robots, which required adding more non-linear constraints to the previous formulation.

All of the aforementioned formulations considered a first order Markovian model, namely the next action depends only on the current state. Such a model was considered also in [1, 4]. In [6] the authors defined the strategy for a higher Markovian order, but performed experiments only for a first order Markovian policy, due to computational reasons. Basilico *et al.* [2] showed that a first order Markovian model performs well, even though not optimally, compared to a higher order. While we examined policies with varied levels of Markovian order. In our case they were also time-dependent strategies, based on the fact that reorganizing is time-dependent.

Kučera and Lamser [19] suggest a new approach for modeling and analyzing patrolling problems using regular strategies, that are not based on a Markovian model. Their approach is still nontrivial to compute optimally (constructable in exponential time), however they are able to adjust to different history lengths by using rounds of improvements to an original strategy. While their approach seems promising, they are still limited to a single patroller, and do not handle sequential (or dependent) attacks. Therefore handling such complicated scenarios by regular strategies is currently impractical.

Time-dependent patrolling strategies have also been considered in previous research. Yin *et al.* [17] considered multiple agents patrolling a transit system in order to detect illegal passengers. Agents move by trains which causes the patrol graph to change according to time, making the strategy time-dependent. They modeled the problem as a Stackleberg game, and followed an assumption that the patrol units can be represented as flow units. Their assumption results in a linear formulation. This does not hold in our case, as we assume that penetration is not instantaneous, thus multiple robots covering the same segment contribute the same as a single robot. Another time-dependent strategy was considered by Boansk *et al.* [3] who extended the work of Basilico *et al.* [11] to a time-dependent strategy in order to consider moving targets. They modeled the problem as a two player non-zero-sum game and formulated a non-linear optimization problem, with the same computation complexity issues of such formulations, as mentioned earlier.

To the best of our knowledge, the problem of sequential attacks, i.e., handling more than one attack that affects the patrol, has been considered previously only in the conference paper version of this work [20].

The problem of handling events was considered in a non-adversarial environment, or when the adversary has no knowledge. The latter was considered, among others, by Elmaliach *et al.* [9] and by Fazli and Mackworth [4]. Elmaliach *et al.* [9] considered the extraction of robots due to failure, where they focused on minimizing the reorganization time. In the work done by Fazli and Mackworth [4], the robots' goal was to detect events (equivalent to penetration attempts). The events occurred randomly, and the robots learned their distribution. Planning their paths was done in a decentralized manner. This is equivalent to an adversary with no knowledge, but with a pattern of its penetration attempts. Jensen *et al.* [21] extended the algorithm for the frequency-based patrol, and examined the problem of extracting robots for re-charging. These events are predictable, and can be taken into consideration in advance, as opposed to attacks by adversaries. In all of 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. Note that our suggested solution may be used in case a robot is extracted from the team due to a failure (yet still able to temporarily monitor parts of his surrounding area), if it is known that an adversary may take advantage of the vulnerability of the robots during their reorganization phase.

The final issue concerns the size of the problem, which is affected by the policy of the patrol. In [17] in order to confront the exponential number of paths, the authors proposed a compact representation, in which the agents' strategy considers a history of a length of two. As mentioned above, this representation cannot be applied when the penetration is not instantaneous. Basilico *et al.* [2] considered solving large instances of security games, and decreasing the problem size without utility loss by discarding dominated strategies. This was done by a computational exponential algorithm, but the computational time was shown to be negligible in practice. This method yields optimal results for medium sized problems. Another method presented involves utility loss, which handles large instances with sub-optimal results. These methods were applied to a first order Markovian model. The algorithm of Basilico *et al.* for removing dominated strategies works under their problem settings and model assumptions. Their settings are somewhat similar to ours, but differ in the fact that in our case every node is a target, and turning around is costly. Consequently their implementation cannot be applied in our settings to reduce the problem size.

Recent work by Basilico *et al.* [22] have considered the problem of detecting penetrations controlled by an adversary, where the environment has an alarm system, that informs the defender of a penetration. The alarm system is noisy, and may produce false positive alarms and missed detections, and determining an optimal strategy for the defender is $\mathcal{APX}$-hard. In our case, one can consider an initial penetration attempt as a false alarm, forcing the system to react to it. However, we consider a multi-robot setting, in which the robot inspecting the alarm (penetration) is extracted from the team, thus the focus is on the behavior in the transition phase (and not on the optimal way to react to the alarm).

This paper extends the work in [20], with an optimal patrol algorithm for the bounded case, and different policies considered for the robots for the unbounded case.

3. Problem Settings

The problem of handling sequential attacks consists of general settings regarding the patrol. In this section we formally describe the patrol settings, the adversarial model, the patrol task and provide a formal definition of the problem. Note that all notations are summarized in Table 1.

3.1. Patrol Settings

In our settings, there are k homogeneous robots $R = \{r_0, \ldots, r_{k-1}\}$ that are required to patrol along a perimeter, i.e., a cyclic path around a closed polygon. The perimeter is partitioned into n segments $S = \{s_0, \ldots, s_{n-1}\}$, with endpoints $ep_0, \ldots, ep_{n-1}$, such that s_i, $0 \leq i < n-1$ starts at ep_i and ends at ep_{i+1} and s_{n-1} starts at ep_{n-1} and ends at ep_0. Without loss of generality, we assume that the sections are numbered from s_0 to s_{n-1} in a clockwise order (see example in Figure 1). Each robot travels through one segment per time unit. 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. A segment is said to be visited at some time if at least one robot travels through that segment during that time. Each robot has a set of possible actions, denoted by A, and at every step can move forward, turn around or stay put, respectively $A = \{CONT, TURN, STAY\}$, where turning around has an associated cost, modeled in time. We use τ to denote the number of time units it takes the robots to turn around. When an attempt to penetrate is detected, a robot must investigate it until the threat is eliminated, necessitating the extraction of the robot from the team. During this inspection time the robot is still able to observe part of the perimeter (this value is quantified in Section 4.1).

3.2. Adversarial Model

The system consists of two coordinated adversaries (this is equivalent to one adversary initiating two attacks). We assume that the time it takes for each one of the adversaries to penetrate lasts t time units, during which it may be detected and captured by a robot traversing the segment in which it resides. Their joint goal is that at least one of them will successfully penetrate the perimeter (that is, without being detected by the robots). Therefore, there are three possible outcomes for the adversaries: $<$penetrated,outside$>$ (indicating that the first attempt succeeded, the second was not initiated), $<$captured,penetrated$>$ (first attempt detected, second succeeded), and $<$captured,captured$>$ (both attempts failed, adversary detected in both cases). Both adversaries earn the same utility if they manage to achieve their joint goal.

Their most preferred outcome is that one of them will penetrate and the other will stay outside the perimeter. It is not acceptable to the adversary that, in order to ensure that one of the attackers will penetrate for sure, the other attacker will always be captured, which could be achieved by a simultaneous attack. Basing this statement on the fact that patrolling algorithms require robots to maintain a constant distance between them [1], then for example, if the robots r_0 and r_1 are placed in segments s_0 and s_j s.t. $j > t$, then initiating penetration attempts in segments s_t and $s_{j-t+\tau+1}$ will result in at least one definite penetration. This is since during t time units r_0 cannot reach s_t and r_1 cannot reach $s_{j-t+\tau+1}$, meaning only r_0 can capture a penetration at $s_{j-t+\tau+1}$ and r_1 at s_t. However, since it is not acceptable, we assume sequential attacks, i.e., the second attack will be launched only if the first attacker is captured.

We consider three cases for the knowledge of the defender on the second penetration with respect to the detection of the first attempt (certain knowledge):

1. **Known penetration period.** The second penetration attempt is initiated at some known time.
2. **Bounded penetration time.** The second penetration is bounded in time $[t_1, t_2]$, after the first penetration attempt is detected.
3. **Unknown penetration time.** The second penetration attempt is initiated at an unknown time.

We consider a strong adversarial model, in which the adversary has full knowledge of the patrol strategy and the robots' positions at the time of the first attack. The adversary chooses the segments to penetrate and the timing of both attacks prior to the initiation of the first attempt. This assumption is applicable to cases where the adversary is required to organize itself in advance (for example due to terrain challenges, and/or the need to remain unobserved), and thus commit to certain penetration locations, without the ability to change those easily on the fly.

3.3. Patrol Task

We define the Probability of Penetration Detection in a segment s of a penetration initiated at time t_a, denoted by $\mathsf{ppd}(s, t_a)$, as the probability that an adversary trying to penetrate through s will be detected by some robot traversing s during t time units of the penetration attempt (the t time units necessary for the adversary to complete the penetration).[1] We would like to find a patrol algorithm that finds the optimal patrol strategies for the robot team when confronting a strong adversary that will necessarily penetrate through the segment with the minimal $\mathsf{ppd}(s, t_a)$ at time t_a, i.e., an algorithm that will maximize the minimal probability of penetration detection throughout the perimeter. More formally, the optimization criterion is $\mathit{maxmin}_{s,t_a}\mathsf{ppd}(s, t_a)$.

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 in which the robots are spread out uniformly (in time, thus in segments) along the perimeter. The distance between every two consecutive robots for k robots is denoted by d_k ($d_k = \frac{n}{k}$), i.e, the number of segments between the robots. They also showed that the minimal value of t such that the robots can have a probability greater than zero of detecting the adversary is $\frac{d_k+\tau}{2}$. This value is denoted by t^k_{min}. In their work, they defined 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 for k robots (characterized by the value p). Therefore, we choose to adapt a scheme where we distinguish between the behavior of the robots in three different phases and optimize each phase:

1. *Phase I*: the steady-state prior to the extraction of the robot handling the first penetration attempt, i.e., patrolling by k robots;
2. *Phase II*: the reorganization phase, where the team's composition changes from k patrolling robots to $k - 1$ patrolling robots;
3. *Phase III*: the steady-state of patrolling by $k - 1$ robots.

Since the optimality of the patrol strategy was established in [1] for phases I and III, in this research we focus on the reorganization phase. Thus, we are concerned with the following problem:

[1]We refer to the probability of penetration detection $\forall t_a$ along the perimeter as ppd.

Given k robots patrolling around the perimeter, where one robot is extracted from the team in order to handle a penetration at location $\mathbf{s_i}$, determine the optimal behavior for the robots that will maximize the minimal probability of penetration detection along the perimeter during the reorganization phase.

Table 1: Notations.

Notation	**Definition**
R	Set of robots
k	Number of robots
r_j	Robot j
S	Set of segments
n	Number of segments
s_i	Segment number i
ep_i	Endpoint i
A	Set of actions for robots
τ	Turning cost
t	Adversary's penetration cost
t_a	Adversary's penetration initiation time
t_0	Start time for the reorganization phase
t_1, t_2	bounds for the range of possible penetrations
d_m	Uniform distance between m robots
l_j	The path length of r_j
l_{max}	The maximal path length
$v_0(r)$	The initial position of robot r in the reorganization phase
$v_{end}(r)$	The final position of robot r in the reorganization phase
$\mathsf{ppd}(s, t_a)$	Probability of penetration detection of segment s initiated at time t_a

$G := (V, E)$	The graph that models the problem
u, v	A vertex consist of endpoint and direction
u', v'	A vertex with the same endpoint as v but counter direction
$W(u, v)$	The cost (time) to arrive from u to v
ρ	Reorganization time
π_r	A path of robot r
h	History of a robot actions
$I(s, \pi_r)$,$I(s, h)$	The contribution of π_r or history h to detecting a penetration attempt at segment s
$\mathsf{ppd}_r(s, t_a)$	Probability of penetration detection of segment s initiated at time t_a by robot r
$\mathsf{ppd}_{r_1 \dots r_j}(s, t_a)$	Probability of penetration detection of segment s initiated at time t_a by robots $r_1 \dots r_j$
$P_{r,\rho}(\pi)$	The probability of robot r to follow path π of length ρ
P_ρ	The set of $P_{r,\rho}$
$\mathsf{ppd}(P, s, t_a)$	The ppd when the robots use probability distribution P
t^m_{min}	The lower bound of t for m robots
$\rho_{ub}[t_1, t_2]$	The upper bound for ρ to optimize $\mathsf{ppd}(t_1, t_2)$
$\pi[t_1 : t_2]$	the sub-path of π from time t_1 to time t_2
$\|\pi\|$	The length of π
$\|h\|$	The length of the history h
H	The set of histories
$P_{r,h,a}$	The probability of robot r to perform action a based on history h
Γ	set of ρs

4. The Reorganization Phase

The reorganization phase starts when a penetration is detected by one robot (that is assigned with handling the penetration), and its teammates reorganize into their new positions. We denote the first penetration detection time by t_0, and consider all times with respect to it (i.e. set $t_0 = 0$). When a robot is extracted from the team to handle the first penetration, the remaining $k - 1$ robots need to be organized such that they will again be spread uniformly around the perimeter, to achieve optimal behavior [1]. The extracted robot remains close to the location of the initial penetration to handle the penetration, and monitor the close environment for the detection of additional penetrations (refer to Section 4.1 for the exact size of the monitored area). This robot's involvement in the general patrolling task diminishes, and by the end of phase II its sole responsibility is handling the penetration it detected. In order for the robots to start reorganizing they are required to communicate reliably, either directly with each other, or through a central command. 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 set of *final positions*, and then we will determine the amount of time to allocate 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 shortest 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. We use l_j to denote this path length for robot $j,\ 1 \leq j \leq k - 1$. The sign of l_j indicates the direction of the path: + for clockwise movement and - for counterclockwise movement. See illustration in Figure 1.

Determining final positions. It is apparent that as ρ grows, each robot has a greater range of segments it can visit during the reorganization phase, thus $\mathsf{ppd}(s, t_a)$ potentially increases. Hence, theoretically, it is preferable that ρ will have large values. However, this contradicts the fact that the reorganization phase length should be minimized, since an optimal patrol is achieved when all robots are organized uniformly. In order to find the ρ value that will balance this trade-off, we start by finding the final positions for all the robots such that the reorganization time is minimized, i.e., minimize the maximal distance that needs to be traveled by some robot from the team.

We assume, without loss of generality, that r_0 is the extracted robot. We begin by proving the following supporting lemma in order to find the final positions (by calculating the values of l_j 's). Let $l_{max} := \max_{1 \leq j \leq k-1} |l_j|$.

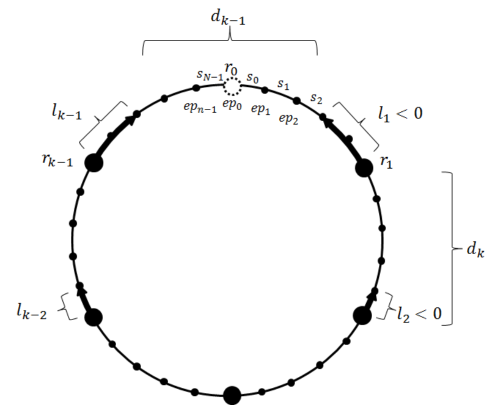


Figure 1: *An illustration of the deterministic reorganization phase. The current positions of the robots are represented by large points. Each robot goes straight to its final position. The robots to the right of r_0 travel in counterclockwise direction (thus their path length is negative), and the robots to the left of r_0 travel in clockwise direction (thus their path length is positive). d_k denotes the distance between two consecutive robots along the perimeter in phase I, and d_{k-1} denotes this distance in phase III (by $k-1$ robots). The unvisited segments are vulnerable to attacks.*

Lemma 1. *For k robots that are spread uniformly around the perimeter with a distance of d_k segments between every two consecutive robots, once robot r_0 is extracted from the system, $l_j = l_1 + \frac{j-1}{k-1} d_k$ for $2 \leq j \leq k-1$, and $l_{max} = |l_1| = |l_{k-1}| = \frac{1}{2}\frac{k-2}{k-1} d_k$.*

Proof. After reorganizing, the distance between robots r_j and r_{j-1}, $j = 2 \ldots k-1$, is $d_k + l_j - l_{j-1}$. 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 $d_k + l_j - l_{j-1} = \frac{k}{k-1} d_k$, and we find that $l_j = l_{j-1} + \frac{1}{k-1} d_k$. In addition, it is easy to verify that $l_j = l_1 + \frac{j-1}{k-1} d_k$.

In order to show the maximal value of l_j (l_{max}), notice that there are two robots r_1 and r_{k-1}, within a distance of $2d_k$ from each other after the extraction of r_0, and the rest of the robots remain within a distance of d_k from each other. Since $2d_k \geq \frac{k}{k-1} d_k$ for all $k \geq 2$, the minimal duration of reorganizing is achieved when these two robots head towards each other in order to reduce the distance between them.

Thus $l_{k-1} - l_1 = 2d_k - \frac{k}{k-1} d_k = \frac{k-2}{k-1} d_k$ holds. The minimum path length is achieved when $l_{k-1} = -l_1 = \frac{1}{2}\frac{k-2}{k-1} d_k$. Hence the minimum of the longest path length for all robots, denoted by l_{minmax}, satisfies that $l_{minmax} \geq \frac{1}{2}\frac{k-2}{k-1} d_k$. The maximum of $|l_j|$ is achieved at boundary points $j = 1, k-1$ and $|l_1| = |l_{k-1}|$

holds, therefore $l_{max} = max|l_j| = |l_1| = l_{k-1} = \frac{1}{2}\frac{k-2}{k-1}d_k$. Since l_{max} is the longest path length, $l_{minmax} \leq l_{max}$ holds, thus $l_{minmax} = l_{max}$. □

Following Lemma 1, we can calculate the value of l_j as follows:

$$l_j = l_1 + \frac{j-1}{k-1}d_k = -\frac{1}{2}\frac{k-2}{k-1}d_k + \frac{j-1}{k-1}d_k = \frac{j-\frac{1}{2}k}{k-1}d_k \tag{1}$$

The initial position of $r_j, j = 1 \ldots k-1$, is denoted by $v_0(r)$ and equals $ep_{j \cdot d_k}$, thus we can calculate the final position of r_j (which is at a distance of l_j from its current position) as follows:

$$v_{end}(r_j) = ep_{j \cdot d_k + l_j} = ep_{j \cdot d_k + \frac{j-\frac{1}{2}k}{k-1}d_k} = ep_{\frac{kd_k \cdot (j-\frac{1}{2})}{k-1}} \tag{2}$$

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

After establishing the final positions by means of Equation (2), we will determine the reorganization time. 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 needed for reorganization. However, such an approach would create vulnerability points as there are segments that will not be visited during t time units. This is defined more formally by the following definition.

Definition 4.1. *Segment $s \in S$ is said to be vulnerable if there exists t_a such that* $\textit{ppd}(s, t_a) = 0$.

Consequently, a knowledgeable adversary would manage to penetrate successfully through these vulnerable points. In the example in Figure 1, none of the points between the final positions of the robots and the current positions of the adjacent robots will be visited during the reorganization phase using this naive algorithm. Thus, these vulnerable 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 around the perimeter, $\text{ppd}(s, t_a) > 0$ during the reorganization phase.

Note that since each segment corresponds to one time unit of travel time, we refer to time and distance interchangeably. The minimal ρ is equivalent to the maximal distance a robot will travel (straight) to its final final position, after turning to its direction (if needed). Therefore the following Corollary is derived directly from Lemma 1.

Corollary 1. *The minimal ρ for reorganizing equals* $\frac{1}{2}\frac{k-2}{k-1}d_k + \tau$

4.1. Monitoring Requirement From the Extracted Robot

The transition from k robots to $k-1$ cannot be instantaneous, as the extraction of a robot leaves segments that cannot be covered by the remaining robots during t time units. Thus, extraction creates segments that a knowledgable adversary will penetrate sucessfully through them. This means that the extracted robot needs to monitor several segments before its extraction is completed. We are interested in guaranteeing that $\mathsf{ppd}(s, t_a) \geq 0$ for every $s \in S$ and $t_a \geq t_0$. For that, every segment must be covered between t_a to $t_a + t$ for every $t_a \geq t_0$. Therefore, it is necessary and reasonable to require that a range of the segments close to the penetration location, that cannot be visited during that time by some robot $r_i \neq r_0$, will be observed by the extracted robot r_0. We turn to detrmine the number of segments that r_0 needs to monitor for every t' time units after t_0. The number of segments that cannot be visited during a period of t time units after t' time units equals $2(d_k - (t' + t))$. This is due to the fact that the number of segments between r_1 and r_{k-1} is $2d_k$. During t time units, r_1 can visit at most t of the segments between r_1 and r_{k-1}. The same holds for r_{k-1}. Therefore, the number of remaining unvisited segments is $2(d_k-(t'+t))$. These are the segments that must be monitored by the extracted robot r_0. Note that as t' increases (more time has passed since the initial penetration occurred), this value decreases, i.e., r_0 needs to monitor less segments (and is not required to monitor any segment at all after $d_k - t$ time units).Now, that we have finished defining the requirements of the reorganization phase, we turn to model the problem.

5. Problem Modeling

Even though the final positions are determined in advance, there are still many strategies that can lead each robot from its current position to its final position during ρ time units. As stated before, because a deterministic approach does not perform well against a full-knowledge adversary, we need a randomized strategy. Unlike the steady state where the world is symmetric, thus enabling each robot to execute the same patrol strategy, in our problem there is no symmetry, making it necessary to calculate a different strategy for each robot. Randomizing at each endpoint does not guarantee that the robot will arrive at its final position at the end of the reorganization phase. Therefore we randomize over the possible paths that start at the current locations of the robots and end at their final positions after ρ time units.

In this section we describe the graphical modeling of the problem, and lay the foundations and algorithms for finding paths for reorganization, and calculating the resulting ppd values determined by the possible paths, along with some given probability distribution over the possible paths. This will be used for determining the optimal strategy for the patrolling robots, based on the adversarial model.

5.1. Modeling the Problem as a Graph

Similar to [1], we model the problem as a directed graph in order to capture the directionality of the robots' movements (facing forward is different than facing backward, considering the time τ it takes the robots to turn around). The graph $G = (V, E)$ is constructed as follows (see the illustration in Figure 2):
For each pair of an endpoint and a direction in the original problem we construct a vertex $v = (ep_i, CW)$, and $v' = (ep_i, CCW)$, where CW stands for clockwise, and CCW for counter-clockwise. Edges are directed according to the direction of the vertices, with a uniform weight of 1. If τ equals 1 (turning cost, equivalent to the time it takes to turn), then we connect (v, v') to (v', v). If $\tau > 1$, we add $\tau - 1$ vertices between v and v', such that the cost of arriving at v' from v and vice versa is τ.

Note that G is actually a Markov chain representing the possible states of the system, which corresponds to the location and direction of the robot.

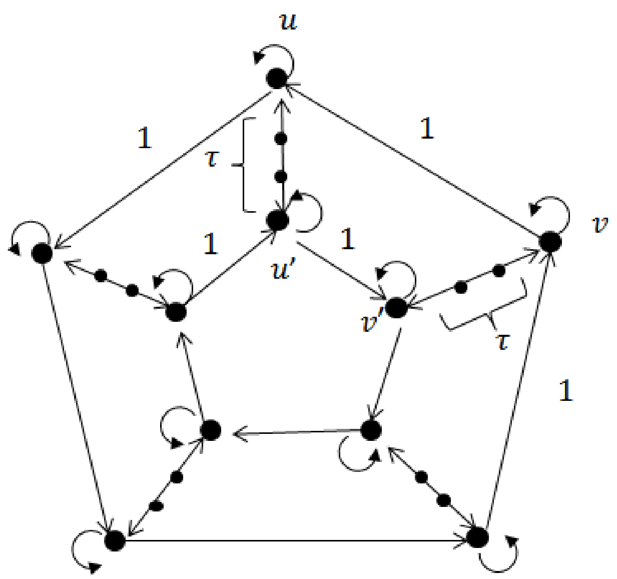


Figure 2: *Example of a weighted graph that takes into account the cost of turning around, τ.*

5.2. Finding All Possible 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. This number is calculated by the function $\mathsf{NumberOfPaths}(u, v, \rho)$, which, given the endpoints of segments in a given direction, uses the entry (u, v) of the adjacency matrix M of the graph G (the Markov chain, as constructed above), in the power of ρ (denoted by $M^\rho(u, v)$):

$$\mathsf{NumberOfPaths}(u, v, \rho) = M^\rho(u, v)$$

The time complexity of calculating $\mathsf{NumberOfPaths}$ is defined by the time complexity of computing M^ρ, which is $O(n^c \cdot \log(\rho))$ for some fixed c, $2 < c < 3$ [23] (Chapter 11.2.5).

Algorithm $\mathsf{FindPaths}(G, v_0(r), v_{end}(r), \rho)$ finds all paths of length ρ. The algorithm considers the modeled graph G, the initial position and direction, encompassed by $v_0(r)$, and the final position and direction encompassed by $v_{end}(r)$. We denote by $W(u, v)$ the time it takes to arrive from u to v, for example, for some u', $W(u, u') = \tau$.

This algorithm uses a Depth-First-Search (DFS) traversal on the graph G, which recursively generates the paths, and each step continues only to branches that can yield a valid path of length up to ρ (until terminating in depth ρ). The algorithm uses the function $\mathsf{NumberOfPaths}$ defined above to verify that the current examined set of paths in nonempty. Clearly, as it uses a rigorous traversal of paths, it returns a complete set of paths between u and $goal$ of length ρ. The time complexity of $\mathsf{FindPaths}$ is $O(\#\Pi \cdot n^c \log(\rho))$ and at most $O(n^{\rho+c} \log(\rho))$ for some fixed c, $2 < c < 3$.

5.3. ppd Computation

The values of the ppd throughout the perimeter depend on the probability of each path being chosen. Suppose that for each robot r we are given a set of paths, denoted by Π_r. We can compute in advance the contribution of each path to every segment at every time of penetration initiation, denoted by t_a. Let π_r be some path of robot r, $\pi_r \in \Pi_r$, and define $\pi_r[t_a : t_a + t]$ to be a sub-path of π_r between times $t_a, t_a + t$. The contribution of this path to the probability of detection, denoted by $I(s, \pi_r)[t_a]$, indicates whether or not a robot following π_r visits segment s during

t time units, starting at t_a. More formally,

$$I(s,\pi_r)[t_a] \leftarrow \begin{cases} 1 & \pi_r[t_a : t_a + t] \text{ contains } s \\ 0 & \text{otherwise} \end{cases}$$

For every distribution of probabilities of selecting a path over Π_r, denoted by P_r, we can calculate the probability for each robot to visit a segment s, denoted by $\mathsf{ppd}_r(s,t_a)$, $\forall s$, using the following Equation (3).

$$\mathsf{ppd}_r(s,t_a) \;=\; \sum_{\pi_r} P_r(\pi_r) \cdot I(s,\pi_r)[t_a] \tag{3}$$

Finally, ppd values are calculated by the complement of the probability that none of the robots will visit segment s.

$$\mathsf{ppd}(s,t_a) = 1 - \prod_{l=1}^{k-1}(1 - \mathsf{ppd}_{r_l}(s,t_a)) \tag{4}$$

This equation is non-linear, and it is computationally harder to solve non-linear constraints than linear or quadratic constraints. However, Equation (4) is equivalent to a set of quadratic equations. In order to show this, we define $\mathsf{ppd}_{r_1\ldots r_j}(s,t_a)$ as the probability of robots $r_1\ldots r_j$ to detect a penetration attempt to segment s initiated at time t_a. By definition:

$$\mathsf{ppd}(s,t_a) = \mathsf{ppd}_{r_1\ldots r_{k-1}}(s,t_a) \tag{5}$$

Similarly, $\mathsf{ppd}_{r_1\ldots r_j}(s,t_a)$ is computed by the following Equation (6).

$$\mathsf{ppd}_{r_1\ldots r_j}(s,t_a) = 1 - \prod_{l=1}^{j}(1 - \mathsf{ppd}_{r_l}(s,t_a)) \tag{6}$$

It is easy to see that the following holds for $j > 1$:

$$\mathsf{ppd}_{r_1\ldots r_j}(s,t_a) = 1 - (1 - \mathsf{ppd}_{r_1\ldots r_{j-1}}(s,t_a))(1 - \mathsf{ppd}_{r_j}(s,t_a)) \tag{7}$$

After simplification we get the following recursive calculation:

$$\mathsf{ppd}_{r_1\ldots r_j}(s,t_a) = \mathsf{ppd}_{r_1\ldots r_{j-1}}(s,t_a)(1 - \mathsf{ppd}_{r_j}(s,t_a)) + \mathsf{ppd}_{r_j}(s,t_a) \tag{8}$$

The above equations assist us in iteratively calculating the ppd for each segment, by integrating the contribution of each robot (by its paths).

6. Bounded Penetration Period

In some cases, once a penetration attempt is detected, we can assume that a sequential attack will occur within a given time frame, for example, between now and sunrise, or knowing that the adversary must spend some time to regroup, but will still initiate an additional attack before the moon rises. In all those cases, the time frame for the next attack is known to us (specifically, it is bounded). Therefore, in this section we consider the case where the time of the second penetration, denoted by t_a, is bounded and known to the defender. Formally, $t_1 \leq t_a \leq t_2$ for some known t_1, t_2.

In this case it is sufficient to find an algorithm that maximizes the ppd between $[t_1, t_2]$ regardless of the ppd values in the times outside that range. Therefore we can choose the reorganization time ρ such that $\rho > t_2$, and only need to consider the sub-paths between $[t_1, t_2 + t]$ for every robot. Moreover, between $[t_2, \rho]$ a deterministic path towards the final position of each robot can be used. The number of possible sub-paths between $[t_1, t_2 + t]$ is affected by the reorganization time, ρ. This is because the final positions after ρ time units might be reachable from every position at time t_2. The greater ρ is, the more possible sub-paths exist between $[t_1, t_2 + t]$ and thus the greater ppd values. In order to achieve the *optimal* results we consider the state space of sub-paths to be all possible sub-paths, and compute ρ accordingly.

We denote by $\pi[t_1 : t_2]$ the sub-path of π from time t_1 to time t_2, and let $|\pi|$ denote the length of a path π. If $\pi[t_1 : t_2]$ is a sub-path of π of length ρ, then clearly for every $\rho' > \rho$ there exists π' of length ρ', such that $pi'[t_1 : t_2] = pi[t_1 : t_2]$. This means that for $\pi[t_1 : t_2]$ there exists a minimal ρ such that $\pi[t_1 : t_2]$ is contained in the set of sub-paths for paths of length ρ', $\rho' > \rho$. We wish to extend this so that we are interested in finding such a ρ that its set of sub-paths between $[t_1, t_2]$ contains all possible sub-paths between $[t_1, t_2]$. We denote this bound by $\rho_{ub}[t_1, t_2]$. That is, using $\rho = \rho_{ub}[t_1, t_2]$ yields the *maxmin*$_{s,t_a}\mathsf{ppd}(s, t_a)$ during $[t_1, t_2]$, and any greater ρ yields the same value.

Lemma 2. $\rho_{ub}[t_1, t_2] := 2t_2 + 2t + l_{max} + \tau$ *is an upper bound for the minimal* ρ *such that maxmin*$_{s,t_a}\mathsf{ppd}(s, t_a)$ *remains the same for every* $\tilde{\rho} > \rho$ *during* $[t_1, t_2]$.

Proof. In order to prove that $\rho_{ub}[t_1, t_2]$ is an upper bound, it is sufficient to prove that:
$\{\pi[t_1 : t_2] \text{ such that } |\pi| = \rho\} = \{\pi[t_1 : t_2] \text{ such that } |\pi| = \rho_{ub}[t_1, t_2]\}$ for $\rho \geq \rho_{ub}[t_1, t_2]$, which means that sub-paths of length $t_2 - t_1 + 1$ from all paths of length ρ are the same as sub-paths of paths of length $\rho_{ub}[t_1, t_2]$, hence they have

the same ppd. A path of length $t_2 + t$ can be at most $t_2 + t + l_{max}$ from its final position. In order for a path of length ρ to be of that distance, ρ needs to be greater than $2t_2 + 2t + l_{max} + \tau$. $\square$

We turn to present the main theorem of this section regarding the computational complexity of the algorithm when the time frame of the second penetration is known.

Theorem 1. *If the time range of the second penetration is known, then a polynomial time algorithm exists that guarantees optimal* ppd *values.*

In order to prove this theorem, we consider several cases in the following Lemmas, 3-7. We distinguish between the case where the penetration time is known exactly ($t_1 = t_2$), and the more general case of a range of values for the penetration time. We partition the problem into four different cases of ranges of $[t_1, t_2]$. For each case we find optimal patrolling algorithm and the optimal resulting *maxmin*ppd. A summary of the cases and our results is presented in Table 2.

Table 2: Optimal values of ppd with respect to the second penetration time range, $[t_1, t_2]$.

Case	Range	*maxmin*ppd	Section
A	$t_1 = t_2$	$\frac{1}{2}$	6.1
B	$t_1 \geq d_k - \frac{t}{2}$ or $t_2 \leq \frac{1}{2}(t_1 + t - t^k_{min})$,$t_1 \leq t - t^k_{min}$	$\frac{1}{2}$	6.2
C	otherwise if $t_1 \geq d_k - t$ or $t_2 \leq t + t_1 - t^k_{min}$	$\frac{1}{3}$	6.2
D	otherwise	$\frac{t(k-1)}{2kd_k}$	6.2

6.1. Known Exact Penetration Time

We examine the first case, in which the exact time that the adversary is going to initiate a second penetration is known (Case A of Table 2). This assumption is not necessarily realistic, but its results provide an upper bound to the guaranteed probability of detection of all other cases, thus important to analyze. Since the penetration time is known, only the period of $[t_a, t_a + t]$ is of interest to us. During a single period of t time units, it is sufficient for the robots to use only two paths that together visit all points in the area, as $t > \frac{d_k}{2}$. This yields ppd values of $\frac{1}{2}$. We prove this in the following lemma.

Lemma 3. *In the case where t_a is known a-priori, an optimal patrol algorithm with polynomial time complexity exists which guarantees that maxmin$_s$ppd$(s, t_a) = \frac{1}{2}$.*

Proof. When the distance between each pair of consecutive robots is at most $2t$ during the reorganization phase, then all segments can be visited by one of two paths for each robot: one forward and one backward, yielding ppd value of at least $\frac{1}{2}$. Therefore, in order to prove the lemma, it is sufficient to show that at each time step between $[t_a, t_a + t]$, the distance between every two consecutive robots (except for r_0) is less than $2t$.

In order for $k - 1$ robots to be able to handle a second penetration in the steady state, t must be greater than $\frac{d_{k-1}+\tau}{2}$, therefore $2t \geq d_{k-1}$. In the steady state the distance between each pair of consecutive robots is less than $2t$, which means that also during the reorganization the distance between them is less than $2t$. The greatest distance between two consecutive robots is between r_1 and r_{k-1} and equals $2d_k$. Reducing this distance to $2t$ requires $d_k - t$ time steps. Since the initial distance between the rest of the pairs of robots equals d_k, which is already less than $2t$, the remaining robots are required to move less than $d_k - t$ segments to ensure a distance of at most $2t$ between them. Thus, for $t_a \geq d_k - t$, the robots can assure that they are distant at most $2t$ at time t_a. Then, if each robot moves forward or backward with a probability of $\frac{1}{2}$, all segments will be covered for t time units with ppd$(s, t_a) = \frac{1}{2}$. Hence the lemma holds for $t_a \geq d_k - t$ as *maxmin$_s$*ppd $= \frac{1}{2}$.

If $t_a < d_k - t$, then the segments that are required to be visited between $[t_a, t_a + t]$ are segments $s_{d_k-(t+t_a)} \dots s_{n-(d_k-(t+t_a))}$ (all segments besides $2(d_k - (t + t_a))$ segments around ep_0). Let r_1, r_{k-1} head towards ep_0 for t_a time units. They are within a distance of t from the segments that need to be visited. The distance between every remaining robot and their consecutive one can be less than $2t$ in less than t_a time units, because robots r_1, r_{k-1} move only t_a segments away. The number of segments where the middle robots move in opposite directions is less than the number required for full reorganization. Therefore, $\forall s \in S$ $\{s_{n-(d_k-(t+t_a))} \dots, s_0, \dots, s_{d_k-(t+t_a)}\}$, the probability of a robot visiting it is at least $\frac{1}{2}$, completing the proof. $\square$

This result is the best that can be achieved, because the assumption for the second penetration time is the strongest. Thus we achieve an immediate Corollary for an upper bound of *maxmin*$_{s,t_a}$ppd(s, t_a) for all settings.

Corollary 2. *maxmin*$_{s,t_a}$*ppd*$(s, t_a) \leq \frac{1}{2}$ *for any setting.*

6.2. Known Penetration Time Range

We now assume that the exact timing of the second attack is unknown, however it is bounded between a given range. That is, t_a is a-priori known to be within $[t_1, t_2]$ for some known t_1, t_2. In the following lemmas we analyze the effect of the different values of t_1, t_2 on the *maxmin*$_{s,t_a}$ppd(s, t_a) that can be achieved for $t_a \in [t_1, t_2]$.

In the following lemma we consider Case B (Table 2), where $t_2 \leq \frac{1}{2}(t_1 + t - t^k_{min})$ and $t_1 \leq t - t^k_{min}$, or $t_1 \geq d_k - \frac{t}{2}$.

Lemma 4. *A patrol algorithm exists such that* $min_{s,t}$*ppd* $= \frac{1}{2}$ *for* $t \in [t_1, t_2]$ *if and only if* $t_2 \leq \frac{1}{2}(t_1 + t - t^k_{min})$ *when* $t_1 \leq t - t^k_{min}$*, or* $t_1 \geq d_k - \frac{t}{2}$.

Proof. For each robot we determine two paths, as illustrated in Figure 3 (a). For r_1,r_{k-1}, the first path is towards ep_0, and on the second path the robot proceeds x segments towards ep_0 and then turns around. The remaining robots have two paths in opposite directions, where they proceed x segments in each direction and then turn around. The segments between r_1 and r_2 must be visited during $t_1 + t$ time units. Due to the fact that r_{k-1} heads in the opposite direction, the paths for $r_2 \ldots r_{k-2}$ must be symmetric, hence they will visit at most $\frac{d_k}{2}$ segments, therefore $x - t_1 + x + \tau + \frac{d_k}{2} \leq t$ (to guarantee a nonzero probability of reaching each segment) which yields $x \leq \frac{1}{2}(t_1 + t - \frac{d_k}{2} - \tau) = \frac{1}{2}(t_1 + t - t^k_{min})$. The segments with a distance of $x+1$ from r_1 and r_{k-1}'s initial positions will not be visited after the first visit, which means that the patrol algorithm is appropriate for values of t_a such that $t_a \leq x$, resulting in $t_2 \leq \frac{1}{2}(t_1 + t - t^k_{min})$. If $t_1 > t - t^k_{min}$ then there cannot be a path that visits both the segment within the distance of $\frac{1}{2}(t_1+t-t^k_{min})$ and the segment within the distance of $\frac{d_k}{2}$ in the opposite direction.

If $t_1 \geq d_k$, then during $d_k - t$ time units paths exist such that the robots can be within a distance of at most $2t$ from each other. From that state we determine for each robot both a path going forward and a path that turns around for $\frac{t}{2} - \tau$ time units and then turns around again and continues forward, as illustrated in Figure 3 (b). All segments will be visited after $d_k - t + \frac{3t}{2}$, which means that from $t_a = d_k - t + \frac{t}{2}$ all segments will be visited during the following period of t time units. Hence if $t_1 \geq d_k - \frac{t}{2}$ then $\min_{s,t}$ ppd $= \frac{1}{2}$ for $t \in [t_1, t_2]$. The amount of time between a visit in a segment to the next visit will not be greater than t time

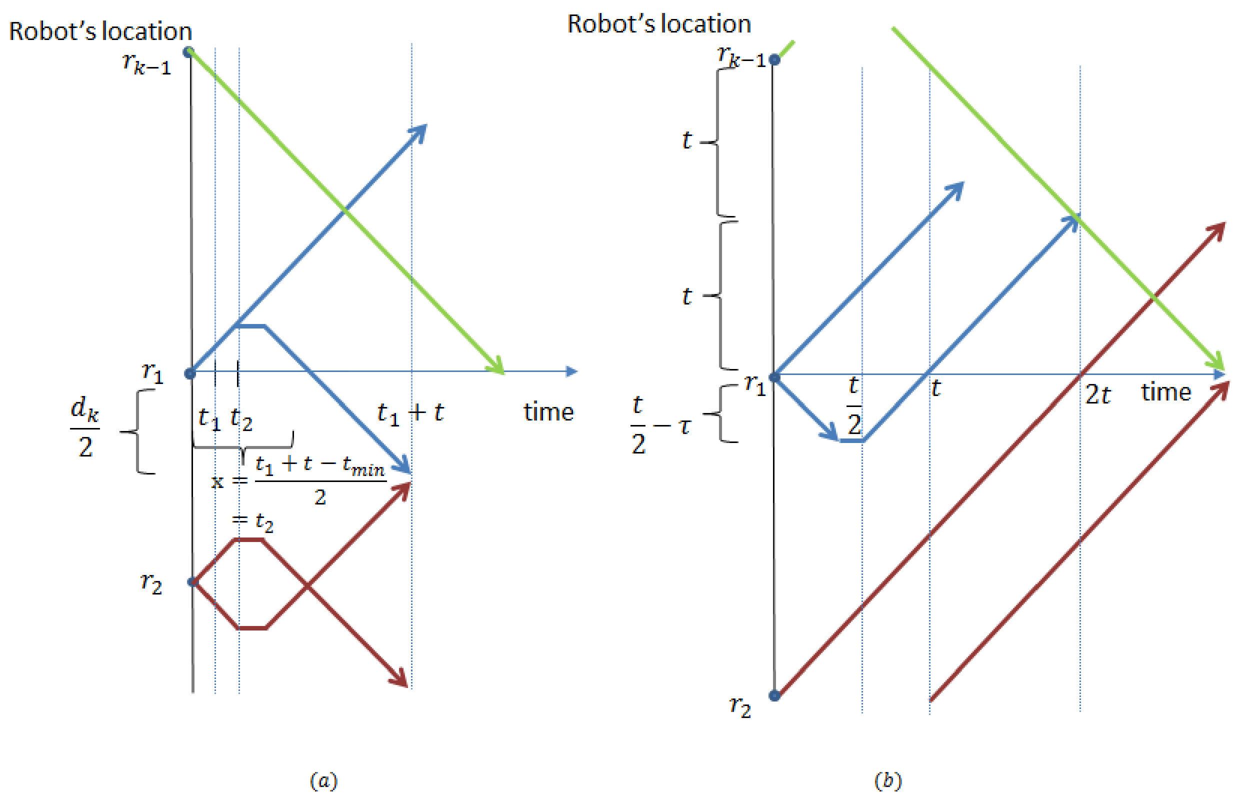


Figure 3: *The paths for the robots to achieve* ppd $= \frac{1}{2}$ *(a) when* $t_2 < \frac{1}{2}(t_1 + t - t^k_{min})$*. (b) when* $t \geq d_k - \frac{t}{2}$*. The* y*-axis represents the robots' location, while the* x*-axis shows the progress along time.*

units. An overlap by r_j, r_{j+1} for some $1 \leq j \leq k-2$ in the segments not between their location can occur after d_k time steps. Thus the second visit is carried out by the same robot. This results in paths of either the first or second case. They are sufficient to visit the segments if t_1, t_2 satisfies the inequalities. By Corollary 2 we find that $\frac{1}{2}$ is the greatest result possible, hence the result is optimal for this case. □

In the following lemma we consider Case C (Table 2), where $\frac{1}{2}(t_1+t-t^k_{min}) < t_2 \leq t_1 + t - t^k_{min}$ when $t - t^k_{min} < t_1 < d_k - \frac{t}{2}$, or $t_1 \geq d_k - t$.

Lemma 5. *A patrol algorithm exists such that* $maxmin_{s,t_a}$ ppd$(s, t_a) = \frac{1}{3}$ *for* $t_a \in [t_1, t_2]$ *if and only if* $\frac{1}{2}(t_1+t-t^k_{min}) < t_2 \leq t_1+t-t^k_{min}$ *when* $t-t^k_{min} < t_1 < d_k-\frac{t}{2}$*, or* $t_1 \geq d_k - t$.

Proof. We first examine the case where $t_1 < d_k - t$. For each robot we determine three possible paths, as illustrated in Figure 4. For r_1, r_{k-1} the paths are: (1) A path heading toward ep_0; (2) A path waiting t time units and then heading toward ep_0; (3) A path waiting x time units and then turning away from ep_0. The first two paths visit the segments toward ep_0 for $2t$ time units. The third path must visit $\frac{d_k}{2}$ segments within t time units starting at t_1. Hence $x = t_1 + t - \frac{d_k}{2}$, and the greatest value t_2 can have is x, since s_1 would not be visited during t time units after $x + 1$. Therefore $t_2 \leq x = t_1 + t - \frac{d_k}{2}$. If $t_a \leq d_k$ then there are no overlaps between the segments outside r_j, r_{j+1}. This means that the lowest $\mathsf{ppd}(s, t_a)$ for $t_a \in [t_1, t_2]$ equals the lowest probability given to one of these paths.

Let us observe three segments with different possible penetration times: segment $s^{(1)} = s_{d_k-(t+t_1)}$ when $t_a^{(1)} = t_1$, segment $s^{(2)} = s_{d-(\frac{1}{2}(t_1+t-t^k_{min})+\tau)}$ when $t_a^{(2)} = \frac{1}{2}(t_1 + t - t^k_{min}) + 1$, and segment $s^{(3)} = s_{d+\frac{d}{2}}$ when $t_a^{(3)} = t_1$.

Only a direct path for robot r_1 toward ep_0 can visit segment $s_{d_k-(t+t_1)}$ during the first t time units after t_1. This path cannot visit segment $s^{(2)}$ or segment $s^{(3)}$, for $t_a^{(2)}, t_a^{(3)}$, respectively.

The distance between $s^{(2)}$ and $s^{(3)}$ equals $\frac{1}{2}(t_1+t-t^k_{min})+\tau+\frac{d_k}{2} = \frac{1}{2}(t_1+t+t^k_{min})$. When $t_1 > t - t^k_{min}$ then this distance is greater than t. Therefore, no single path exists that visits both $s^{(1)}$ and $s^{(2)}$ during the same t time units. If r_2 does not visit $s^{(3)}$ when $t_a^{(3)} = t_1$, then it must be visited by r_1 and we hence no path of r_1 can visit more than one of these segments in the examined penetration time. If r_2 does have a path that can visit $s^{(3)}$, then it does not visit the segment $s_{d+\frac{t}{2}}$ when $t_a = t_1$. The distance between this segment and s_{2d_k} is $d_k - \frac{t}{2}$, which is greater than t. In other words, if r_3 does not visit the latter segment then r_2 must visit it, and hence r_2 has no path that can visit more than one of these three segments. Similarly, it is proven that no robot exists with a path that allows it to visit more than one of the three segments it must visit. Therefore, any additional path would influence the ppd of only one of the segments, and the optimal solution is to have a uniform distribution over three paths.

If $t_1 \geq d_k - t$, then at t_1 the robots can be within a distance of at most $2t$ from each other. We determine a path forward for each robot, a path that turns around for $\frac{t}{2} - \tau$ time units and then turns around again and continues forward, as well as a path backward. After $\frac{t}{2}$ all segments are visited by all pairs of the first two paths of each robot. The third path is required to be visited between $d_k - t$ to t. No additional path can visit all of the segments that the two paths have already visited. Hence, $\mathsf{ppd}(s, t_a) = \frac{1}{3}$ is optimal for $t_a \in [t_1, t_2]$. $\square$

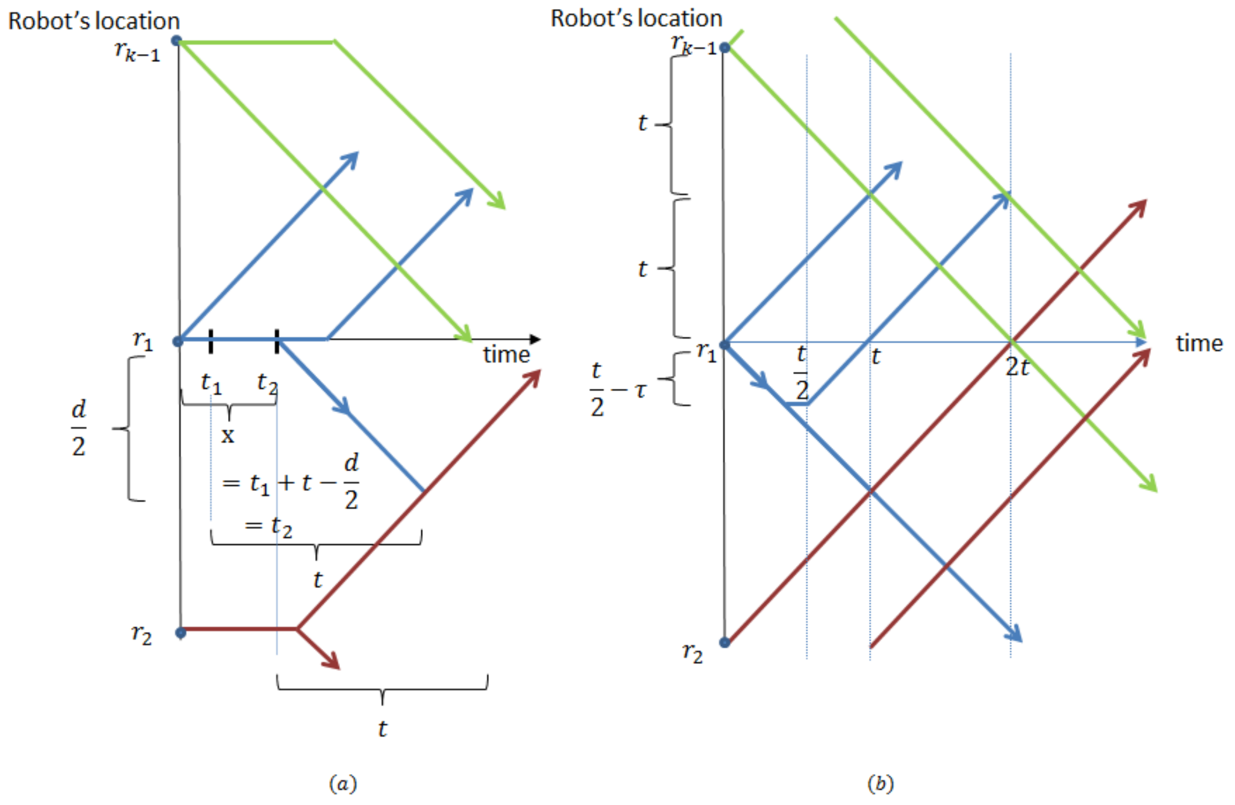


Figure 4: *The paths for the robots to achieve* $\mathsf{ppd} = \frac{1}{3}$ *(a) when* $t_2 < t + t_1 - t^k_{min}$*; (b) when* $t_1 \geq d_k - t$*. The* y*-axis represents the robots' location, while the* x*-axis shows the progress along time.*

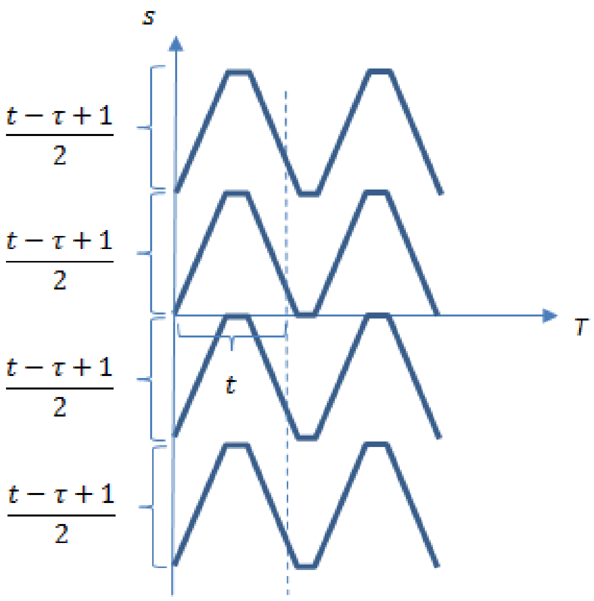


Figure 5: *Distinct cyclic paths that visit their segments (y-axis) for any range of t time units (x-axis).*

We now prove the last case, Case `D` (Table 2). In order to prove the guaranteed probability of detection in this case, we must first prove a supporting lemma regarding the partitioning of the segments into distinct cyclic paths. Afterwards we will allocate each robot such paths, which will yield the desired lower bound. The paths are illustrated in Figure 5.

Lemma 6. *Let π be a cyclic path that consists only of segments within a distance of at most x from the starting position. Then π visits each of its segments during any range of t time units if $x \leq \frac{t-\tau+1}{2}$.*

Proof. The time to return to the segment of the starting position is $x-1+x+\tau = 2x+\tau-1$. In order to visit every segment within any range of t it is then required that $2x+\tau-1 \leq t$. Thus the distance is $\frac{t-\tau+1}{2}$. $\square$

The final case, Case `D`, applies when the conditions required for cases `A, B, C` do not hold. In the following lemma we provide a lower bound for the optimal ppd values for all of the possible settings in this case. The paths that we use are of the form presented in the previous lemma with a uniform probability distribution.

Lemma 7. *A polynomial patrol algorithm exists that guarantees that in the worst case maxmin*$_{s,t_a}$ppd$(s,t_a) \geq \frac{t(k-1)}{2kd_k}$*, which is at least* $\frac{1}{4}$.

Proof. The number of paths with a distance of $\frac{t-\tau+1}{2}$ that visit n segments is $\frac{2n}{t-\tau+1}$. We allocate the paths uniformly between the robots, and for each robot define a uniform distribution over its paths. Then for each path of $k-1$ robots the probability is $\frac{(t-\tau+1)(k-1)}{2n} = \frac{(t-\tau+1)(k-1)}{2kd_k}$. Since $t \geq \frac{1}{2}\frac{k}{k-1}d_k + \tau$, it holds that the probability is greater than or equal to $\frac{1}{4}$. □

Thus, by Lemmas 3-7 we have proven Theorem 1, which states that there exists a polynomial time algorithm yielding the optimal ppd values. Moreover, we have shown that when the penetration period is known to us the optimal *maxmin*ppds are $\frac{1}{2}$ at best, and $\frac{1}{4}$ at worst.

7. Unbounded Penetration Period

In the most general (and probable) case, when handling one penetration attempt, it is unclear when the second penetration will occur. That is, the time of the second penetration attempt is unbounded. We formulate a general optimization problem and consider three specific models for computing the ppd. But first we discuss the differences between this case and the bounded case, which will lead us to conclude that the reorganziation time must be randomized.

In the previous case we maximized the minimal ppd during a bounded period, which enabled the selection of a ρ (reorganization time) that may result in a ppd value that could even be higher than the steady state. This is due to the fact that we were able to neglect the time after the second penetration occurred (unlike the steady state, that must guarantee a uniform behavior over time, since a penetration may occur at any given time).

As an example, consider the case in which $n = 84$, $k = 7$, $t = 8$, $t_a = 0$. Let us choose $\rho = 15$, which means that although the first $t = 8$ time units are handled, the remaining $\rho - t = 7$ time units are ignored during the optimization. This allows the robots to be very well prepared against a sequential attack during that time, 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, for $k - 1 = 6$ robots it is 0.05, and during the reorganization phase (the first t time units) it is 0.5. However, if a penetration is initiated after 12 time units (and not at time 0) while the original patrol scheme in use, then the minimal ppd drops to 0. The reason is that the probabilities were chosen to handle a penetration attempt only at time 0. Segments with $\mathsf{ppd} = 0$ are points of vulnerability for a definite successful attack by the adversary. This means that a patrol scheme designed against a bounded penetration period cannot be used in the unbounded case.

We start by proving that there are points of inevitable 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 is bounded. The problem arises when the penetration can occur at any time during the reorganization phase, even during the vulnerability points. Therefore, increasing ρ in this case would not solve the problem, thus we must randomize the ρ value, to avoid vulnerability.

Lemma 8. *For $\rho - \frac{1}{2}d_{k-1} \leq t_a \leq \rho - t + \frac{1}{2}d_{k-1}$, a segment s exists such that $\mathsf{ppd}(s, t_a) = 0$.*

Proof. After ρ time units all robots are located at their final positions of the reorganization phase (established by Equation (2)). For $\rho - t \leq t_a \leq \rho$, at time t_a each r_j is at a distance of at most $\rho - t_a$ 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 of the reorganization phase. Thus, during t time units each robot is at a distance of at most $\max(\rho - t_a, t - (\rho - t_a))$ from its final position. This is illustrated in Figure 6. If none of the segments is overlooked then each pair of robots visit all of the segments between their final positions, at a distance of d_{k-1}. Thus, it is necessary that $2\max(\rho - t_a), t - (\rho - t_a)) \geq d_{k-1}$, 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 $\mathsf{ppd}(s, t_a) = 0$ exist. □

The above lemma implies that if the opponent has full knowledge of the patrol scheme, namely, algorithm and robots positions, and in particular, ρ and the final positions of the robots after reorganizing, it can choose to penetrate during that time and it will be guaranteed to succeed.

Corollary 3. *$maxmin_{s,t_a} ppd(P, s, t_a) = 0$ for a single reorganization time*

As a corollary, ρ must be randomized in order to ensure $\mathsf{ppd}(s, t_a) \neq 0$ for all t_a. This is only true in the unbounded penetration period case, as the adversary's initiation time is not limited. In the bounded penetration period case, it is sufficient to choose a greater enough ρ. In that case, the robots' arrival to their final positions does not need to be optimized and the robots only need to maximize the number of segments visited during the period and choose their paths accordingly.

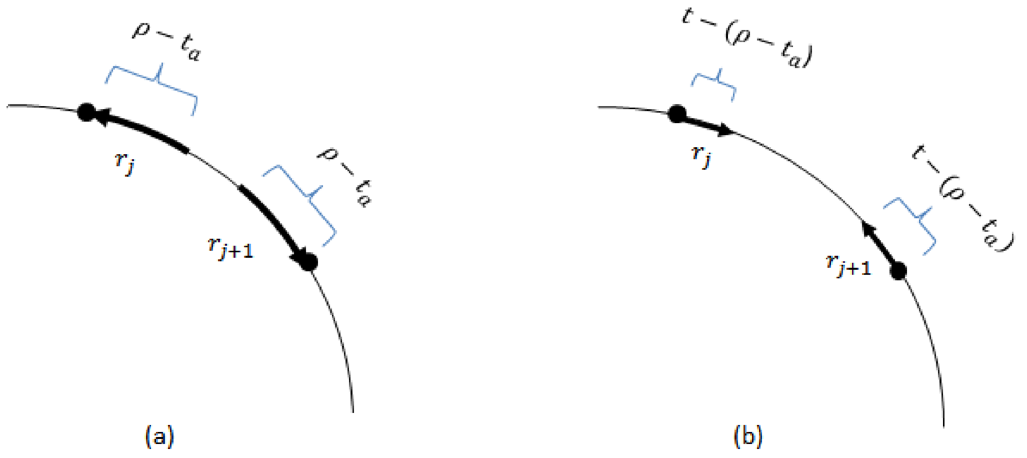


Figure 6: *The limitations on the distance of the robots from the final positions. (a) represents the most distant positions of the robots after $t_a \leq \rho$ time units (during reorganization). (b) represents the most distant positions of the robots during the remaining period of t (after the reorganization phase).*

Optimizing the entire reorganization phase requires considering the reorganization constraints and possible vulnerability points when choosing the ρ values. If we were to guarantee only that $\min \mathsf{ppd} > 0$, then using two different values of ρ would be sufficient, with some probability greater than zero for each one of them. Since all robots must end the reorganization phase at the same time, all robots must draw paths for the same ρ. Consequently each robot can draw a path only after a ρ is drawn. Hence, in order to *optimize* the coverage of the vulnerability points, there is a need to consider the different reorganization times along with the different paths for each case. We discuss the robots' behavioral model in Section 7.1, the solution for the optimization problem in Section 7.2, an approximation to the optimal solution in Section 7.3, and report experimental evaluation of the approximation vs optimal solution in Section 7.4.

7.1. Robots' Strategy Model

In most of the previous work, the patrol strategy of the robot was considered to be a first order Markovian, where the next position is only affected by the current position. We consider the robots' strategy model to be Markovian, with a history length $|h|$, of $1 \leq |h| \leq \rho$. The parameter $|h|$ affects the state space of strategies which the patrol algorithm considers. Obviously, optimal results are guaranteed only within each state space of strategies with the defined Markovian order. It is clear that larger history lengths can only improve results, because they contain all of the strategies with shorter history lengths. This introduces a trade-off between optimality and the size of the state space. The size of the state space is exponential in $|h|$: at each time step, a robot may decide to perform one of three actions: $\{$CONT,TURN,STAY$\}$. Thus for a given initial position, there are $3^{|h|}$ different possible histories with length $|h|$ starting at that position. Since there are n possible initial positions, there are $n \cdot 3^{|h|}$ different histories. A patrol strategy consists of ρ decision points, therefore the size of the state space is $O(\rho \cdot n \cdot 3^{|h|})$

7.2. Optimization Problem

In order to solve the problem when the penetration period is unbounded, we formulate it as an optimization problem that finds probabilities for robots' actions that will maximize the minimal ppd. In this subsection we discuss the different parts of the general reorganization algorithm for the robots in three steps:

I Given a set of possible ρ values, formulate the general optimization problem (Section 7.2.1)

II Compute the ppd constraints, given a certain reorganization time ρ (Section 7.2.2)

III Combining I+II into a general optimal algorithm, maximizing the minimal ppd value during the reorganization phase (Section 7.2.3)

7.2.1. General Formulation

The patrol scheme first randomizes its reorganization phase length and then randomizes over the paths of each robot. The set of multiple ρs is denoted by Γ, and the probability for each ρ by q_ρ. Nevertheless, when considering various values of ρ, all ppds of different ρs must be computed at the same time using the same optimization problem.

Separating the solving process into two phases, where the first finds the optimal ppds for each ρ separately and the second finds the optimal probability distribution over ρ, is useless. This is a result of Corollary 3 which states that $\mathit{maxmin}_{s,t_a}\mathsf{ppd}(P_\rho, s, t_a)) = 0$, thus $\underset{q_\rho}{\mathit{maxmin}} \sum_\rho (q_\rho \cdot \underset{P_\rho}{\mathit{maxmin}_{s,t_a}}\mathsf{ppd}(P_\rho, s, t_a))$ is 0 for every settings. This motivates the use of a combined solution, that randomizes over the combination of both, together.

Therefore, all paths of different ρs must be considered together. This leads to the general formulation of the optimization problem, where each specific model

defines the constraints of the computation of $\mathsf{ppd}_\rho(s, t_a)$.
$optimiaztionProblem(R, S, \Gamma, T, \Pi)$:

$$\begin{aligned}
\max \quad & m && (9)\\
\mathsf{ppd}_\rho(s, t_a) \text{ constraints} \quad & s \in S, t_a \leq \rho && (10)\\
\mathsf{ppd}_\rho(s, t_a) = steady(s, t_a) \quad & s \in S, t_a > \rho && (11)\\
\sum_{\pi \in \Pi_{r,\rho}} P_{r,\rho}(\pi) = 1 \quad & r \in R, \rho \in \Gamma && (12)\\
P_{r,\rho}(\pi) \geq 0 \quad & r \in R, \rho \in \Gamma, \pi \in \Pi_{r,\rho} && (13)\\
\sum_{\rho \in \Gamma} q_\rho = 1 \quad & && (14)\\
q_\rho \geq 0 \quad & \rho \in \Gamma && (15)\\
\mathsf{ppd}(s, t_a) = \sum_\rho q_\rho \cdot \mathsf{ppd}_\rho(s, t_a) \quad & s \in S, t_a \in T && (16)\\
\mathsf{ppd}(s, t_a) \geq m \quad & s \in S, t_a \in T && (17)
\end{aligned}$$

Constraint (10) calculates the ppds contributed by each ρ with respect to the distribution $\{P_{r,\rho}\}$. In the following sub-section we will present different models for representing the ppd constraints. Constraint (11) considers the values of the ppds after the reorganization phase, where the ppd values are calculated using the patrol scheme introduced by Agmon *et al.*[1]. Constraints (12)-(15) are probability properties for $P_{r,\rho}$ and q_ρ. The combined ppd equals the expected value of the ppd_ρs, as shown by constraint (16). Finally, constraint (17) guarantees that m is a minimum of the ppd values. Combining this while maximizing m by the objective function in (9) results in m equaling the $maxmin_{s,t_a}\mathsf{ppd}(s, t_a)$.

7.2.2. ppd Constraints

First we begin by determining the constraints for the ppd calculation based on a single reorganization duration ρ. We will later use this to randomize over possible ρ values.

We examine three different models of formulations to calculate the ppd based on the policy of the robots. All three models optimize the ppd for its policy, that is, the models of formulations are used to solve the same optimization problem: maximizing the minimal probability of detection (given a ρ value), and provide the calculation of ppd values for this purpose. The difference between these models is the amount of pre-process required for the formulation.

The first model—**BGA extension**—considers any Markovian order $|h|$ and adds helper variables to compute the ppd. This is an extension of previous work (e.g., [6]). This model does not require any pre-processing for computing the ppd.

The second model—**HISTORY-T**—enables any $|h|$, but with an exponential of t instead of $|h|$ to be considered. The model requires pre-processing of computing the contribution of paths of length t. This model is a novel formulation.

The third model—**FULL model**—considers histories of ρ length and thus includes all of the solutions of histories of different lengths. The model requires pre-process of computing contribution of paths of length ρ.

Surprisingly, the first two models failed to produce results in experiments, while the third model, which considers all of the histories, succeeded. This is further discussed in Section 8.3.

We define the history as a sub-path of length $|h|$ segments in a specific time. This definition is affected by the reorganizing requirement, which requires the robots' actions be taken into account with respect to the robots' locations, rather than only the actions. This is due to the fact that the positions of the robots with respect to their final positions should affect the probabilities of the next action. In addition, the history model contains time specification, as the current time of the patrol with respect to the time of reorganization should also affect the probabilities. The history, H, is modeled by a sequence of vertices, $\{v_i\}_{i=1}^{|h|}$. The decision variables are of the form $P_{r,h,a}$ where $r \in R$, $a \in A$ (the actions) and $h \in H$, meaning the probability that robot r will perform an action a given the last $|h|$ vertices that occurred are encompassed by h.

The first model, the BGA extension, is an extension of similar formulations from previous work [2, 3, 6]. The original formulation considered a set C of n cells represented by a graph G, to be patrolled by a single patroller, with Markovian strategy of order 1. The intruder has penetration cost of d_i (t in our notations). To compute the probability of capturing the intruder they used helper variables $\gamma_{i,j}^{h,w}$ to denote the probability that the patroller reaches cell j from cell i in h steps without passing through cell w. Capturing an intruder at cell q given that the patroller is in cell i is with probability of $1 - \sum_{i \in C \setminus q} \gamma_{s,i}^{d,q}$.

The new formulation (BGA extension) extend this model to support reorganization constraints, greater history length and multiple robots[2]. This model requires an exponential number of paths in $|h|$, but for small values of $|h|$, it is considerably small. The main issue is the helper variables γ which consider the probability of visits between any pair of vertices at any time. Most of these cases

[2]We omit the mathematical formulation in this paper

are impossible, and therefore it is unnecessary to include them in the optimization problem. In the original formulation of the BGA model the size of the problem was $O(n^3 \cdot t)$. Previous work discusses ways to reduce this problem size. In the original BGA model they improved the problem size by assuming that the number of targets that the adversary wishes to penetrate through is much smaller than the number of nodes. This assumption does not hold in our formulation. Moreover, in our problem we also consider multiple robots, multiple possible penetration times, different re-organization times and re-organization constraints which results in $O(\rho \cdot n^3 T^2 k)$ size of the helper variables γ, which is significantly greater than the original formulation's size. Though the size of the helper variables is polynomial in the parameters of the problem, they can still be very large[3]. Therefore, from our broad experience and experiments, solvers fail to solve this problem due to the high number of variables.

These variables are required for computing the contribution of each robot to each segment. To simplify this, in our next model, HISTORY-T, we extract this computation to a preprocessing stage, and thus avoid the need to evaluate the contribution when solving the optimization problem. In order to compute the contribution to a specific segment at a certain point in time, we need to consider only sub-paths of length t. By computing all of these sub-paths in advance, we can calculate their absolute contributions (visits/not visits), and use it along all ρ points in time. The size of this model is exponential with an exponent that is at least t, regardless of the history length, as all histories of length t are computed. But, compared to the previous model, even though it was polynomial, the size of this model is considerably small, and the formulation is simpler. Also, it enables considering greater history values without the need to compute the actual histories of that length. However, because it still contains many non-linear constraints, when combined with the reorganization requirements, the solvers (MINOS[4],filter[5], SNOPT[6], Ipopt[7]) fail to solve instances of the problem. It is possible that this model will perform well on settings of other patrolling problems, without reorganizing constraints. If so, it will enable the consideration of large history

[3]For example, even in a small setting with only five robots, thirty segments and a maximal reorganization time of 10, we obtain more than fifty million variables, without even considering different history lengths.

[4]http://www.sbsi-sol-optimize.com/asp/sol_product_minos.htm

[5]http://www-unix.mcs.anl.gov/ leyffer/papers/SQP_manual.pdf

[6]http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm

[7]https://projects.coin-or.org/Ipopt

lengths in problems in which only small lengths of history have been addressed to date.

In order to simplify the optimization problem even further, in the third model: the FULL model we extract both the contribution computation and the reorganization constraints to a preprocessing stage. This is done by focusing on $h = \rho$ and only consider paths, π_r that are valid for reorganizing, that is start with $v_0(r)$ and end with $v_{end}(r)$. This means that we pre-compute all possible π_r and their contribution $I(s, \pi_r)[t_a] \leftarrow \begin{cases} 1 & \pi_r[t_a : t_a + t] \text{ contains } s \\ 0 & \text{otherwise} \end{cases}$.
It is worth noting that this is the optimal Markovian patrolling policy, for each ρ. This is because every model of history length h' can be described by any higher $h \geq h'$. The paths are of size ρ, hence there is no benefit from any policy with $h > \rho$. Considering $h = \rho$ simplifies the model, as our decision variables are of the form $P_{r,\rho}(\pi)$, where π is of length ρ, instead of a conditional probability.

The FULL model constraints for the ppd values are derived from Equations (5)-(8):

$$\mathsf{ppd}_r(s, t_a) = \sum_{\pi_r} P_r(\pi_r) \cdot I(s, \pi_r)[t_a] \qquad (18)$$
$$r \in R, s \in S, t_a \in T$$
$$\mathsf{ppd}_{r_1 \ldots r_j}(s, t_a) = \mathsf{ppd}_{r_1 \ldots r_{j-1}}(s, t_a)(1 - \mathsf{ppd}_{r_j}(s, t_a)) + \mathsf{ppd}_{r_j}(s, t_a) \qquad (19)$$
$$j > 1, r_j \in R, s \in S, t_a \in T$$
$$\mathsf{ppd}(s, t_a) = \mathsf{ppd}_{r_1 \ldots r_{k-1}}(s, t_a) \qquad (20)$$
$$s \in S, t_a \in T$$

Constraint (20) is quadratic and results from multiple robots that visit the same segment during t time units. That is possible because the robots are not binded to move in the same direction. Without this constraint the optimization problem would be a simple linear programming problem, solved in polynomial time.

Hereinafter we will consider the formulation of the FULL model.

7.2.3. The Patrol Algorithm

The final algorithm for determining the paths for the robots is presented in Algorithm 1, PatrolRearrange which computes the optimal distribution and randomizes the path for each robot.

First, we calculate the final position for each robot and then find all paths

Algorithm 1 PatrolRearrange$(R, S, \{v_0(r)\}, t, \tau, T, \Gamma)$

Input:

R: the group of robots
S: the set of segments
$\{v_0(r)\}$: the set of initial positions of the robots
t: the time it takes the adversary to penetrate
τ: the time it takes a robot to turn around
T: the range of considered penetration times
Γ: set of ρs

Output: the patrol algorithm

```
1: construct $G$
2: for each $r$ in $R$ do
3:     $v_{end}(r) \leftarrow CalcFinalPosition(v_0(r))$
4:     for each $\rho$ in $\Gamma$ do
5:         $\Pi_{r,\rho} \leftarrow$ FindPaths$(G, v_0(r), v_{end}(r), \rho)$
6: $problem \leftarrow optimizationProblem(R, S, \Gamma, T, \Pi)$
7: $P, Q \leftarrow solve(problem)$
8: randomize $\rho$ from $\Gamma$ with distribution $Q$
9: return randomize $\pi$ from $\Pi_{r,\rho}$ with distribution $P_{r,\rho}$
```

from its initial position to its final position by means of algorithm FindPaths. We construct the optimization problem as descibed in Section 7.2.1 with FULL model ppd constraints and by calling $solve$ we use 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. After solving the optimization problem, first a ρ is drawn with a probability of q_ρ, and then a path with a length of ρ is drawn for each robot.

7.3. 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 for any $\epsilon > 0$ there exists an upper bound for ρ that yields a solution within a distance of ϵ from the optimal solution. We define the sequence $\{maxmin\mathsf{ppd}_\rho\}$ such that each element is the *maxmin*ppd achieved by solving the optimization problem when the length of the reorganization phase varies between $1 \dots \rho$.

Lemma 9. *When* $max\Gamma \to \infty$, $\{maxmin\mathsf{ppd}_\rho\}$ *converges to the maxmin*ppd *of the reorganization phase.*

Proof. The solution is monotonic with respect to ρ, because if a greater ρ does not improve the results it will have a probability of 0. In addition the solution is bounded (it represents a probability). Therefore, the process of adding more ρs converges. Due to the fact that every possible path is contained in $\{\Pi\}_{\rho \to \infty}$, the optimal paths for 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 greater ρs.

Corollary 4. *For every* $\epsilon > 0$ *a* ρ_0 *exists such that adding any* $\rho > \rho_0$ *would not change more than* ϵ *of* $maxmin\mathsf{ppd}_{\rho_0}$.

Proof. Cauchy's criterion [24] states that for a sequence a_i to converge it is sufficient and necessary that for every $\epsilon > 0$ a fixed number n_0 exists such that $|a_i - a_j| < \epsilon$ for all $i, j > n_0$. By Lemma 9 we find that $\{maxmin\mathsf{ppd}_\rho\}$ converges. By applying Cauchy's criterion to this sequence we find that for every ϵ there exists n_0 such that $|maxmin\mathsf{ppd}_i - maxmin\mathsf{ppd}_j| < \epsilon$ for all $i, j > n_0$. We set $\rho_0 = n_0 + 1$ and obtain $\rho \geq \rho_0 > n_0$ $|maxmin\mathsf{ppd}_\rho - maxmin\mathsf{ppd}_{\rho_0}| < \epsilon$. □

7.4. Experimental Results

Since the solution to the optimization problem is an approximation of the optimal solution, we were interested in examining the results in practice.

We conducted experiments based on 10 different setting combinations where the number of robots varied between 3 and 8, and the distance between them varied between 5 and 12 (resulting in graphs with up to 84 segments and 168 vertices). These parameters correspond to similar problem sizes in related work. We used the MINOS solver [8] to solve the optimization problem. The results are presented in Figure 7 and Figure 8.

Figure 7 demonstrates the convergence of the ppd. The theoretical analysis shows that *maxmin*ppd converges to the optimal solution. Since the true optimal solution is unknown, we compared the ppd results in the experiments to the results of the steady state. This is sufficient as the optimization problem consists of the steady state phases as well and thus the optimal ppd values are at most the values of the steady state. Figure 7 also demonstrates Lemma 8, which states the

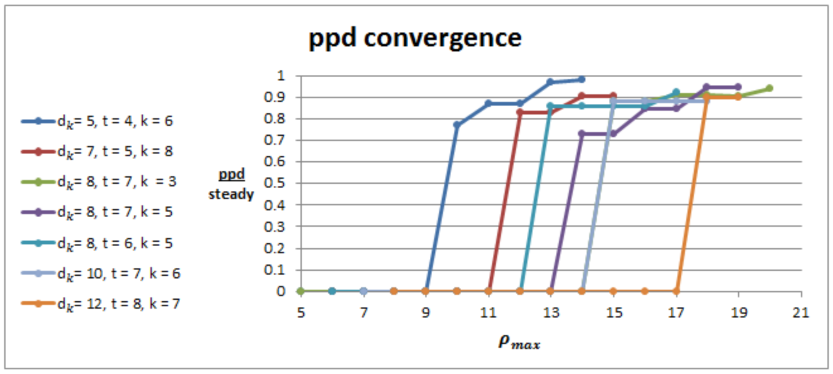


Figure 7: *The ratio between the maxmin*ppd *to the steady state, with respect to the maximal* ρ *used, when all of the* ρ*s in the range* $[\rho_{min}, \rho_{max}]$ *are used.*

vulnerability times for each ρ, and therefore ρs that cover each other's vulnerabilities must be used in order to achieve $\mathsf{ppd} > 0$. Respectively, in the graph the ppd values start from 0 and increase only when a ρ large enough to eliminate the vulnerabilities is used. When vulnerabilities remain, adding the constraint that *maxmin*$\mathsf{ppd} > \epsilon$ for some small ϵ enables solvers to pre-solve the constraints and determine that there are vulnerabilities that are not covered. In these cases the

[8]http://sbsi-sol-optimize.com/asp/sol_product_minos.htm

results are conclusive, in contrast to general solutions of the solver, which are approximations of the optimal solution and better results are possible. After reaching a ρ such that all vulnerabilities are covered, the ppd values increase to more than 70% of the ppd of the steady state. When considering greater ρ's, the values of the ppd attained reached $88 - 98\%$ of the *maxmin*ppd of the steady state. As it is a non-linear optimization problem, the solutions are approximations and it is possible that higher values can be achieved.

The graph in Figure 8 shows the probability assigned to each value of ρ in each of the approximated solutions, along with the ppd achieved. It is interesting to see that even though higher ρs are required, the minimal ρ (the deterministic case) is assigned the highest probability. This is due to the fact that the ppd of the steady state is optimal, and the minimal ρ minimizes the time in which it is achieved. Even though the results of the probability distributions might only yield an approximated solution, we know the value that is guaranteed by following these distributions and its comparison to the steady state.

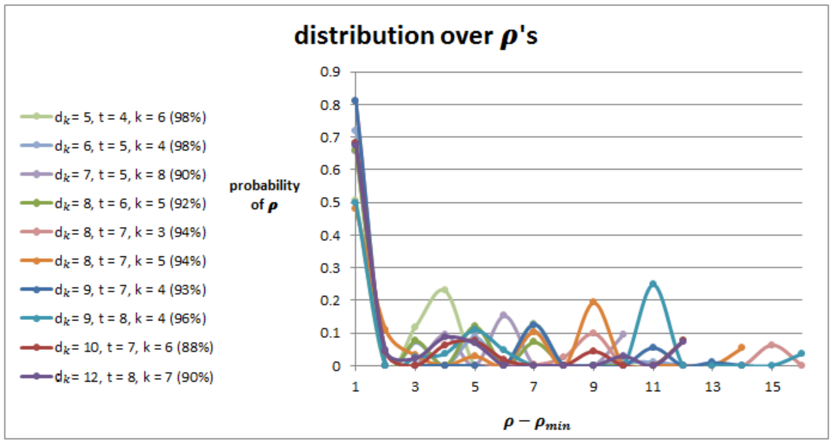


Figure 8: *Probability distribution over ρs for each case, with the percentage of the* ppd *achieved with respect to the* ppd *of the steady state. The values of ρs are scaled with respect to the minimal ρ of each case.*

8. Reducing the Problem Size

As the problem size is exponential in $|h|$, there is a need to reduce its size. Hitherto, our suggested algorithm (PatrolRearrange) considers every possible history path for the optimization problem. Not only does this entails consideration of a very large input, but it is also unnecessary, as some of the history paths clearly make less contributions than others, and there is no need to consider them in the optimization problem. To this end, we define and examine the dominance of paths (rather than sub-paths).

Definition 8.1. *A path π_1 dominates π_2 if $I(s, \pi_1)[t_a] \geq I(s, \pi_2)[t_a]$, 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 would like to examine the effect of transferring the probability from π_2 to π_1, which dominates it. We use $P_r{}'$ to denote the resulting distribution function, and $\mathsf{ppd}_r(P_r, s, t_a)$ to denote the value of ppd by robot r according to P_r.

Lemma 10. *If π_1 dominates π_2 then $\mathsf{ppd}_r(P_r{}', s, t_a) \geq \mathsf{ppd}_r(P_r, s, t_a)$, for every $s \in S$, $0 \leq t_a \leq \rho$.*

Proof. First we calculate the difference between the probabilities $\mathsf{ppd}_r(P_r{}', s, t_a)$ and $\mathsf{ppd}_r(P_r, s, t_a)$ and then we prove that it is greater than 0. By using Equation (3) we find that the difference $\mathsf{ppd}_r(P_r{}', s, t_a) - \mathsf{ppd}_r(P_r, s, t_a)$ equals
$\sum_\pi P_r{}'(\pi) \cdot I(s, \pi)[t_a] - \sum_\pi P_r(\pi) \cdot I(s, \pi)[t_a]$. Since $P_r{}'$ differs from P_r only by the probability for π_1, π_2, then the only remaining terms from the subtraction of $\mathsf{ppd}_r(P_r{}', s, t_a) - \mathsf{ppd}_r(P_r, s, t_a)$ are those regarding π_1 and π_2. The probability for π_1 by P_r' is $(P_r(\pi_1) + P_r(\pi_2))$, and 0 for π_2. The probability for π_1 by P_r is $P_r(\pi_1)$, and $P_r(\pi_2)$ for π_2. Thus, the difference equals $(P_r(\pi_1) + P_r(\pi_2)) \cdot I(s, \pi_1)[t_a] - P_r(\pi_1) \cdot I(s, \pi_1)[t_a] - P_r(\pi_2) \cdot I(s, \pi_2)[t_a]$. This results in $(I(s, \pi_1)[t_a] - I(s, \pi_2)[t_a]) \cdot P_r(\pi_2)$. Thus, since π_1 dominates π_2, the above expression is greater than 0. □

We have proven that the probability for a robot to visit any segment is greater when using a path that dominates another path, than when using the dominated path. As a corollary we found that this also holds for ppd values.

Corollary 5. *If π_1 of robot r dominates π_2 of robot r, then $\mathsf{ppd}(P', s, t_a) \geq \mathsf{ppd}(P, s, t_a)$ for every $s \in S$, $0 \leq t_a \leq \rho$.*

Proof. By Equation (4) $\mathsf{ppd}(s, t_a) = 1 - \prod_{l=1}^{k-1}(1 - \mathsf{ppd}_{r_l}(s, t_a))$. The subtraction between the ppds equals $(1 - \prod_l(1 - \mathsf{ppd}_{r_l}(P_{r_l}{}', s, t_a))) - (1 - \prod_l(1 - \mathsf{ppd}_{r_l}(P_{r_l}, s, t_a)))$. The simplified expression is $\prod_l(1 - \mathsf{ppd}_{r_l}(P_{r_l}, s, t_a)) - \prod_l(1 - \mathsf{ppd}_{r_l}(P_{r_l}{}', s, t_a))$. As $\mathsf{ppd}_{r_l}(P_{r_l}, s, t_a) = \mathsf{ppd}_{r_l}(P_{r_l}{}', s, t_a)$ for all $r_l \neq r$, the product $\prod_{r_l \neq r}(1 - \mathsf{ppd}_l(P_r, s, t_a))$ is mutual for both terms and thus the subtraction equals $\prod_{r_l \neq r}(1 - \mathsf{ppd}_l(P_{r_l}, s, t_a))(1 - \mathsf{ppd}_r(P_r, s, t_a) - (1 - \mathsf{ppd}_r(P_r{}', s, t_a)))$. After simplifying the expression the result is $\prod_{r_l \neq r}(1 - \mathsf{ppd}_{r_l}(P_r, s, t_a)) \cdot (\mathsf{ppd}_r(P_r{}', s, t_a) - \mathsf{ppd}_r(P_r, s, t_a))$. By Lemma 10 $\mathsf{ppd}_r(P_r{}', s, t_a) - \mathsf{ppd}_r(P_r, s, t_a) > 0$ and thus the above expression is greater than 0. $\square$

By Corollary 5 we conclude that the optimal patrol algorithm does not use dominated paths, since other paths that would yield better results exist. Thus we should examine only paths that are not dominated by any other path.

Definition 8.2. *A path π_1 is said to be a Pareto path if there is no $\pi_2 \in \Pi$ such that $\mathsf{ppd}_1(s, t_a) < \mathsf{ppd}_2(s, t_a)$ for all $s \in S, 0 \leq t_a \leq \rho$.*

Pareto paths can be computed by a brute-force method. First, paths with equal contribution to a previous path are eliminated. Then, the remaining paths are compared with each other to eliminate dominated paths. In the following subsections we perform a theoretical analysis of the number of Pareto-paths and present experimental results showing that although their number is theoretically exponential, *in practice* this number is significantly and sufficiently lower than the total number of paths.

8.1. Size of Pareto-paths Space

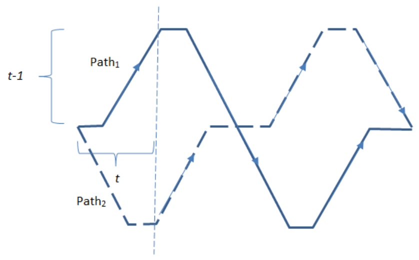
The purpose of using Pareto-paths is to reduce the exponential number of paths. In this section we discuss the complexity of the space of all Pareto-paths. As the following lemma states, evidently the number of Pareto-paths is still exponential with respect to their length, ρ (the same holds for h). To prove that we provide a very loose lower bound that is exponential in ρ.

Lemma 11. *The number of Pareto-paths is $\Omega(2^{\rho/4t})$.*

Proof. We prove this lemma by constructing an example, such that for every ρ, t has at least $2^{(\rho/4t)}$ paths. Let us consider two sub-paths with a length of $4t$.

1. TURN, CONT $t-1$ steps, TURN, CONT $2(t-1)$ steps, TURN, CONT $t-1$, TURN
2. CONT $t-1$ steps, TURN, CONT $t-1$ steps, TURN, TURN, CONT $t-1$ steps, TURN, CONT $t-1$ steps

The sub-paths and their contributions are illustrated in Figure 9. These paths are not dominated by each other. Moreover, there cannot be two valid paths that have the same contribution. This is because a contributing to $t-1$ segments



Path$_1$

	0	*t*	2*t*	3*t*
S1	1	1	0	0
S2	1	1	0	0
S3	1	1	0	0
S4	1	1	0	0
S5	0	0	1	1
S6	0	0	1	1
S7	0	0	1	1
S8	0	0	1	1

Path$_2$

	0	*t*	2*t*	3*t*
S1	0	0	1	1
S2	0	0	1	1
S3	0	0	1	1
S4	0	0	1	1
S5	1	1	0	0
S6	1	1	0	0
S7	1	1	0	0
S8	1	1	0	0

Figure 9: Two non-dominated sub-paths and their contributions at t_a: $0, t, 2t, 3t$.

requires continuing in the same direction (either forwards or backwards). The initial position and direction determines the direction of the traversal over the segments.

These two sub-paths start and end in the same position and in the same direction. Therefore we can construct paths by concatenating these sub-paths. No combination of these sub-paths can 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 $2^{(\rho/4t)}$. □

8.2. Experimental Results

As we showed in Section 5.2, the number of total paths is exponential in the input size. In sub-section 8.1 we have proven theoretically that the number of Pareto-paths is also exponential. However, in this section we show that in practice, in most cases the number of Pareto-paths is significantly lower than the exponential number of paths. We analyzed the results of the experiments we conducted with more than 11,000 settings for the problem of reducing the input size. This result is significant as it entails that we can apply this method successfully in order to reduce the number of paths for unbounded penetration time problems. We examined the following settings, and also examined the parameters that affect the number of paths and Pareto-paths. The number of paths depends on ρ, the length of the path over time, $dist$, the actual distance between a robot's current and final position (l_j for some robot j), and n since the fence is cyclic and paths from u to v can originate from different directions. The number of Pareto-paths depends on these parameters, but it also depends on t, as it affects the contribution of each path. Consequently, we examined k, d_k ($n = kd_k$), t, ρ and $dist$. We generated settings with these parameters varying between different ranges that correspond to similar problem sizes in related work: $k = 2 \ldots 10$, $d_k = 3 \ldots 6$, $t = 1 \ldots d - 1$,

$\rho = 2 \ldots 12$, $dist = 0 \ldots \rho$. We examined the influence of each parameter. Parameter n has a very minor influence on the number of paths, as almost no paths visit all n segments. The most influential parameters were t and ρ. As t grows, the contribution of each path is greater, and thus there are fewer Pareto-paths. Figure 10 shows that not only does the number of Pareto-paths decreases as t grows, but it drops exponentially.

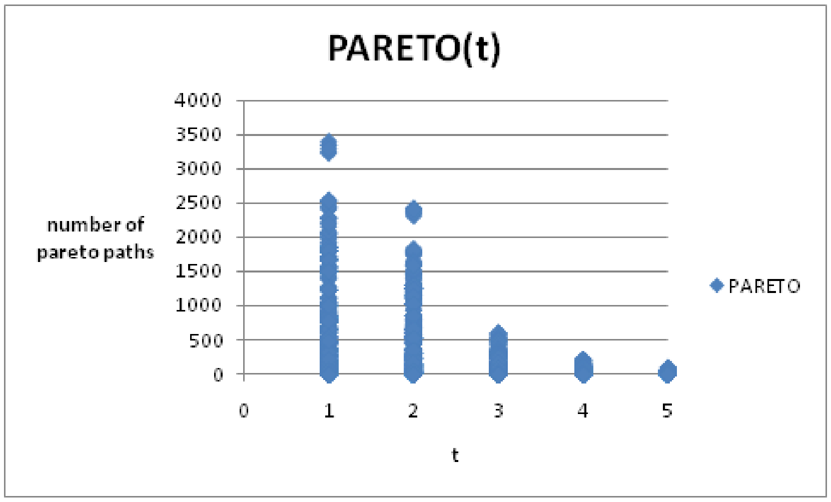


Figure 10: *Exponential drop in the number of Pareto-paths with respect to t.*

The ρ parameter has a reverse affect. As ρ grows, i.e. greater reorganization time, both the number of paths and Pareto-paths increases exponentially. Figure 11 shows 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. For example, when ρ equals 12 the number of paths is 21745, as opposed to only 3355 Pareto-paths.

The actual ratio between the number of Pareto-paths and the number of paths is shown in Figure 12. As ρ grows the ratio decreases, even though there are more Pareto-paths in general. This supports the hypothesis that the number of Pareto-paths grows much slower than the growth of the number of paths, and demonstrates the effectiveness of this method in practice.

In Figure 12 we can also see that as ρ grows $dist$ has no effect over the ratio between the number of Pareto-paths and the number of paths. The main influence of the parameter $dist$ is realized in the possible sizes of ρ. Since the minimal ρ is at least the size of $dist$, choosing a greater size of $dist$ requires that a greater size

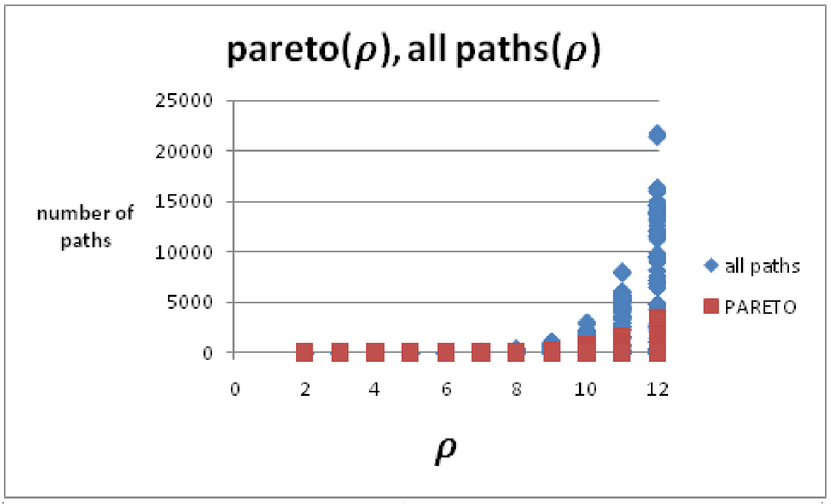


Figure 11: *Number of paths and Pareto-paths as a function of* ρ.

of ρ be considered and thus a greater number of paths. For computational reasons this justifies choosing the final positions for the reorganization that minimize the distance the robots need to travel.

All of the above results suggest that using the Pareto-optimization is efficient in decreasing the input size in practice. As for the practicability of the method by means of time complexity, we present results regarding the computation time. Figure 13 presents the measured time of computing the Pareto-paths in compared to the time to compute the total paths, and the influence of ρ on the computation time. Figure 14 presents the influence of n, k on the computation time of the Pareto-paths, in additional settings: fixed ρ, t: $\rho = 11$, $t = \frac{3}{4}d_{k-1}$ and variable values for n, k: $n = 20 \ldots 60$, $k = 3 \ldots \frac{n}{2}$. We used 1.9Ghz Intel $i7$ CPU with 4GB memory for the experiments. The graph in Figure 13 shows that the time grows exponentially as ρ grows. This is due to the fact that the number of paths and Pareto-paths grows exponentially. The graph in Figure 14 shows that as k increases the computation time increases linearly. As n increases for a fixed k the computation time decreases. This is due to the fact that d_k increases and thus $dist$ increases. The greater $dist$ is, for a fixed ρ, the fewer the number of possible paths, and thus less computation time. The method used to compute the Pareto-paths was brute-force, with an improvement of initially eliminating paths that duplicate another path's contribution. Clearly, there can be ways to improve this computation, but even with a slightly improved brute-force, the time measured is

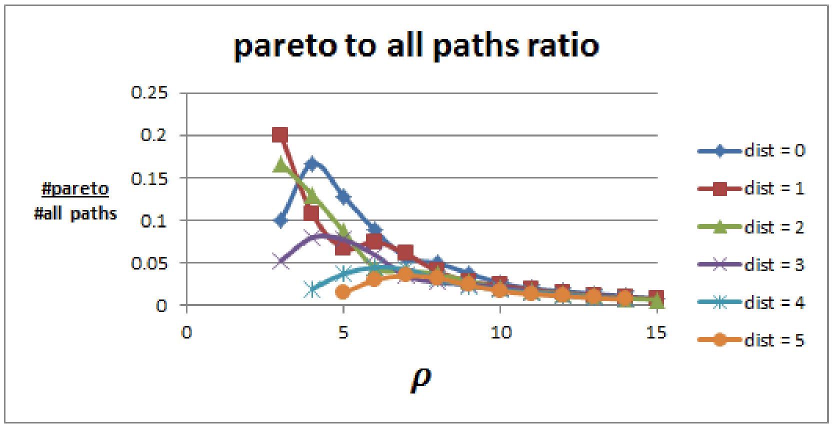


Figure 12: *Exponential drop in the ratio between the number of Pareto-paths and all paths, with respect to ρ. A smaller $dist$ yields a higher ratio. Only $k = 3, d_k = 3, t = 2$ is shown in the figure as different cases cannot be distinguished in this type of graph.*

still small in practice. It is important to note that the time invested in generating the Pareto-paths is negligible compared to value of removing enough paths to allow the solvers to generate a solution. In other words: if one does not reduce the number of paths by using Pareto-paths (i.e., significantly improve the space complexity), the solvers are unable to solve the problem. Hence we pay in time to reduce the space, and enable the problem solving.

8.3. Discussion

In this work we focused on a Markovian patrolling policy with a history length of ρ, which is an optimal Markovian policy. The challenge is that the input size is exponential in ρ, which is the largest input size of all policies. In addition we examined policies with smaller values of the history length, denoted by $|h|$, and presented two models that address this problem.

It is surprising that both models are unsuccessful, because we had expected that if we could apply a policy with $|h| = \rho$, then we could apply all policies with $|h| < \rho$. However, in all previous work with similar formulations, only very small values of history lengths were able to run in practicality, and in some problems only $|h| = 1$ was possible. Higher values could not be tested due to either space or time issues. This makes it interesting to understand the distinction between the case where $|h| = \rho$ and cases with smaller values of $|h|$.

First, we analyze the size of the problem. In order to confront the input size of paths of length ρ we applied a Pareto-optimization to the paths. This

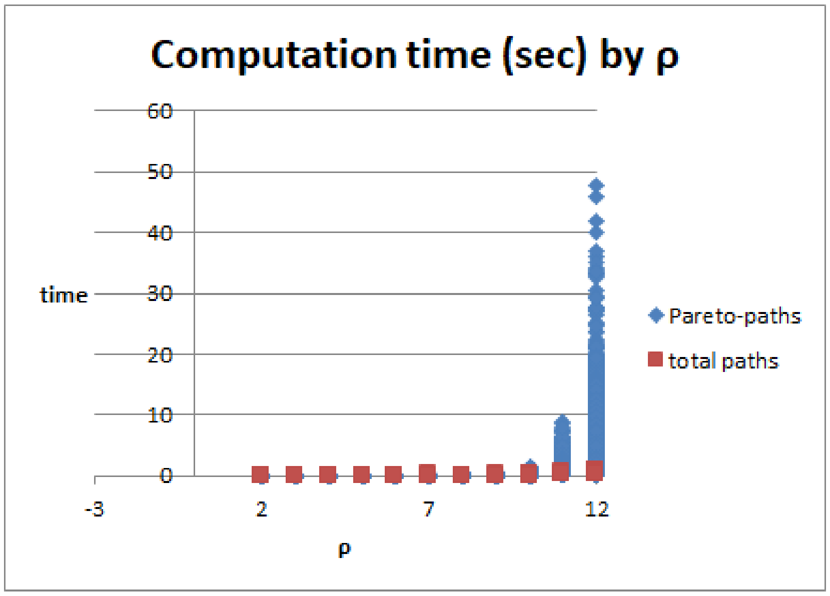


Figure 13: *Demonstrating the time to extract the Pareto paths from the total number of paths. The time it takes to compute the Pareto paths grows exponentially with* ρ*.*

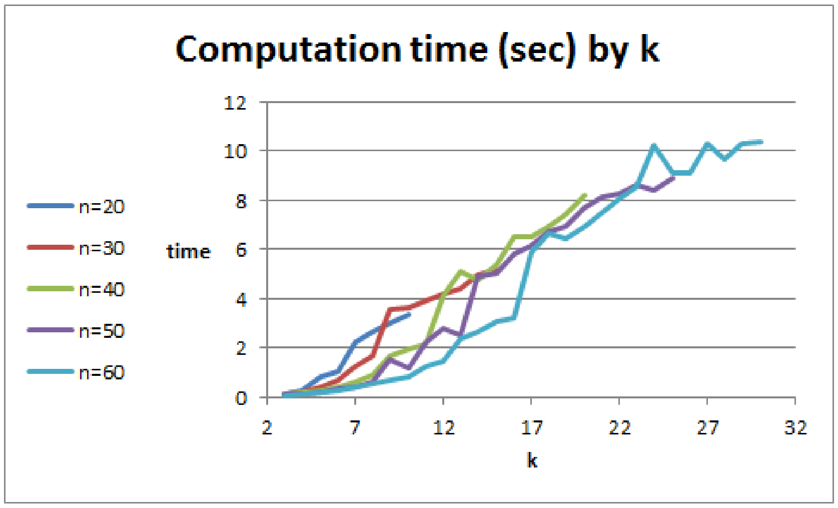


Figure 14: *Demonstrating the time to extract the Pareto paths from the total number of paths. The time it takes to compute the Pareto paths grows exponentially with* ρ*.*

method can also be applied to sub-paths of lengths smaller than ρ, but with less effectiveness. To show this we define dominance over sub-paths, which adds an additional constraint to the paths dominance criterion, and in order to compute Pareto sub-paths of length h we need to consider buffered sub-paths of length $2t+h-1$. In order to determine dominance between two sub-paths μ_1, μ_2, requiring $I(s,\mu_1)[t_a] \geq I(s,\mu_2)[t_a]$ for every s, t_a is not sufficient. Assume that sub-paths μ_0, μ_3 exist such that when concatenating the sub-paths to create $\mu_0\ \mu_1\ \mu_3$, $\mu_0\ \mu_2\ \mu_3$ then $I(s,\mu_0\ \mu_1\ \mu_3)[t_a] \leq I(s,\mu_0\ \mu_2\ \mu_3)[t_a]$ for some s, t_a. Hence, for the path $\mu_0\ \mu_2\ \mu_3$ it is not profitable to consider $\mu_0\ \mu_1\ \mu_3$ instead. Consequently we cannot replace every occurrence of μ_2 with μ_1.

Definition 8.3. *A sub-path μ_1 dominates μ_2 if $I(s,\mu_0\ \mu_1\ \mu_3)[t_a]$ $\geq I(s,\mu_0\ \mu_2\ \mu_3)[t_a]$, for every $s \in S$, $0 \leq t_a \leq h+t-1$ and every μ_0, μ_3, where μ_0, μ_3 are sub-paths of length $t-1$.*

According to this definition, for any path that contains a dominated sub-path we can replace it with its dominant sub-path, and receive a dominant path.

Namely, when Pareto-optimizing history paths (i.e., sub-paths), more paths remain (created by concatenating sub-paths) than if the Pareto-optimization would be applied to all paths, rather than the sub-paths. This causes the method to be less effective. Even though the number of sub-paths is smaller than the number of paths, in the optimization problem we consider the sub-paths over time, and with different starting points. Accordingly their quantity is multiplied by $V \cdot T$.

The second issue is the complexity of the optimization problem. When $|h| = \rho$, non-linear constraints result from overlaps between the robots' coverage and using multiple ρs. When $|h| < \rho$ substantially more non-linear constraints are added, which influence the ability of a solver to produce results.

9. Conclusions

We examined the problem of defending against a sequential attack in a knowledgeable adversarial environment. In this case the robots are required to respond to a penetration attempt, and therefore a knowledgeable adversary can exploit the influence of an attack in order to coordinate a second attack. We focused on optimizing the reorganization phase after the first penetration attempt is detected and one robot is extracted to handle it. In this phase the robots reorganize in order to achieve optimal behavior for $k-1$ robots as established by Agmon *et al.* [1]. Moreover, this method is applicative in scenarios of robot faults which require the remaining robots to reorganize, for this case the faulty robot still needs to maintain some of its sensing ability or additional sensing elements need to be present. We set the foundations for the models and distinguished between the cases where the second penetration is bounded in time, and when it is not. Considering a bounded period of penetration attempts enable the allocation of a single reorganization time, as large as needed. We presented a polynomial time algorithm for optimal coverage, divided into cases, which guarantees *maxmin*ppd of at least $\frac{1}{4}$ during the reorganization phase.

When the second penetration attempt time is unknown, we formulated a non-linear optimization problem. We considered policy with history and presented a novel approach that uses a full history policy, the FULL model, even when formulations with smaller history policies fail to yield results. We proved that a single reorganization time is not sufficient, and that the reorganization time must be randomized in order to avoid vulnerability points. In order to determine the range of ρs to consider, we showed that from some value considering greater ρs has no significant impact on the ppd. We validated this by conducting experiments which also show that using the minimal length of range of ρs required to cover all vulnerabilities yields a ppd that is greater than 70% of the steady state. Achieving a ppd that is greater than the steady state has no effect, as the adversary would then choose to penetrate at the steady state. Another interesting result is that when randomizing over multiple ρs, using the deterministic approach with a high probability yields higher ppd values. 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 ρs we are able to avoid the vulnerability points that are created by the deterministic approach.

Finally, 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 we showed through extensive simulations that it is considerably smaller in practice.

In addition, we discussed two other models of the optimization problem to address varied history lengths (and not only the full history). In these problem settings the models failed to yield results using four different solvers. The first model, the BGA-extension, even though polynomial in theory (and exponential only in the history size), is too large in practice. The second model, HISTORY-T, allows consideration of the history of every size, with a space complexity of the number of histories only of a length of t. However, solvers failed to solve instances with this model (MINOS, filter, SNOPT, Ipopt). It would be interesting to apply this model to different problems without as many constraints (such as reorganization constraints). Though we concentrated on two adversaries, the extension to multiple adversaries is straightforward.

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. Another interesting venue would be 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). We would also like to examine the effect of the robot's inspection time, as we believe it might lead to a more robust patrol against multiple attacks. An additional direction for future work would be to handle other sequential attack adversarial models, for example having more than two penetration attempts, other timing constraints on the penetration time, and more.

Acknowledgments

This research has been supported in part by the Ministry of Science and Technology, Israel and the Japan Science and Technology Agency (JST), Japan, and by ISF grant 1337/15.

References

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

[2] N. Basilico, N. Gatti, F. Amigoni, Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder, Artificial intelligence 184 (2012) 78–123.

[3] B. Bošanskỳ, V. Lisỳ, M. Jakob, M. Pěchouček, Computing time-dependent policies for patrolling games with mobile targets, in: The Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011, pp. 989–996.

[4] P. Fazli, A. Mackworth, Multi-robot repeated boundary coverage under uncertainty, in: IEEE International Conference on Robotics and Biometrics (ROBIO), 2012, pp. 2167–2174.

[5] P. Villacorta, D. Pelta, Exploiting adversarial uncertainty in robotic patrolling: A simulation-based analysis, Advances in Computational Intelligence (2012) 529–538.

[6] N. Basilico, N. Gatti, 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), 2009, pp. 57–64.

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

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

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

[10] N. Agmon, G. Kaminka, S. Kraus, Multi-robot fence patrol in adversarial domains, in: Proceedings of the Tenth Conference on Intelligent Autonomous Systems (IAS), 2008, pp. 193–201.

[11] N. Basilico, N. Gatti, F. Villa, Asynchronous multi-robot patrolling against intrusions in arbitrary topologies., in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI), p. 1224-1229, 2010.

[12] Y. Vorobeychik, B. An, M. Tambe, S. Singh, Computing solutions in infinite-horizon discounted adversarial patrolling games, in: Proc. 24th International Conference on Automated Planning and Scheduling (ICAPS 2014), 2014.

[13] Y. Elmaliach, A. Shiloni, 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), 2008, pp. 63–70.

[14] A. Machado, G. Ramalho, J. Zucker, A. Drogoul, Multi-agent patrolling: An empirical analysis of alternative architectures, in: Multi-agent Based Simulations (MABS), 2003, pp. 155–170.

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

[16] N. Agmon, V. Sadov, G. A. Kaminka, 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), 2008, pp. 55–62.

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

[18] F. M. D. Fave, E. A. Shieh, M. Jain, A. X. Jiang, H. Rosoff, M. Tambe, J. P. Sullivan, Efficient solutions for joint activity based security games: fast algorithms, results and a field experiment on a transit system, Autonomous Agents and Multi-Agent Systems 29 (2015) 787–820.

[19] A. Kučera, T. Lamser, Regular strategies and strategy improvement: Efficient tools for solving large patrolling problems, in: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, 2016, pp. 1171–1179.

[20] E. Sless, N. Agmon, S. Kraus, Multi-robot adversarial patrolling: Facing coordinated attacks, in: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1093–1100.

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

[22] N. Basilico, G. De Nittis, N. Gatti, Adversarial patrolling with spatially uncertain alarm signals, Artificial Intelligence 246 (2017) 220–257.

[23] G. H. Golub, C. F. Van Loan, Matrix computations, 3rd Edition, volume 3, JHU Press, 1996.

[24] H. N. Jahnke, Cauchys cours danalyse. an annotated translation by robert e. bradley, and c. edward sandifer. sources and studies in the history of mathematics and physical sciences. dordrecht (springer), 2009.