Artificial Intelligence 311 (2022) 103769

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Multi-robot adversarial patrolling strategies via lattice paths ☆

Jan Buermann *,[1], Jie Zhang [2]

University of Southampton, Southampton SO17 1BJ, UK

ARTICLE INFO

Article history:
Received 29 December 2020
Received in revised form 25 July 2022
Accepted 28 July 2022
Available online 2 August 2022

Keywords:
Multi-robot systems
Robots in adversarial settings

ABSTRACT

In full-knowledge multi-robot adversarial patrolling, a group of robots has to detect an adversary who knows the robots' strategy. The adversary can easily take advantage of any deterministic patrolling strategy, which necessitates the employment of a randomised strategy. While the Markov decision process has been the dominant methodology in computing the penetration detection probabilities on polylines, we apply *enumerative combinatorics* to characterise the penetration detection probabilities for four penetration configurations. It allows us to provide the closed formulae of these probabilities and facilitates characterising optimal random defence strategies. Comparing to iteratively updating the Markov transition matrices, we empirically show that our method reduces the runtime by up to several hours. This allows us extensive simulations on the two dominant robot movement types for patrolling a perimeter showing that a movement with direction is up to 0.4 more likely to detect an adversary. Therefore, our approach greatly benefits the theoretical and empirical analysis of optimal patrolling strategies with extendability to more complicated attacks and other structured environments.

1. Introduction

In multi-robot adversarial patrolling, a defender is trying to use a number of robots to detect an *adversary*'s penetration attempts into important areas or at important points. It is a well-established problem with numerous security applications including crime prevention [2], stopping piracy [3], defending critical infrastructure [3], and protecting animals, natural reserves, or the environment [4]. In these settings, robots provide a cheap, mobile and more reliable addition to humans in monitoring vast open areas [2,5,2]. Adversarial patrolling has many characteristics and has been considered with different approaches; we focus on finding optimal memoryless random strategies for detecting penetration attempts in polyline graphs against full-knowledge adversaries. Previous work on this determined optimal random strategies using Markov chains to model the robots' movement. In contrast, we present a new technique in analysing these random strategies using lattice paths. Applying this technique reduces the runtime to determine optimal strategies by up to several hours, as well as allowing a theoretical analysis of the strategy space. The significantly reduced runtime is not only beneficial in itself but it allows us to perform a large scale empirical comparison of the main two movement types when patrolling a perimeter.

☆ A preliminary version of this work appeared in the Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020) [1].

* Corresponding author.
E-mail addresses: J.Buermann@soton.ac.uk (J. Buermann), Jie.Zhang@soton.ac.uk (J. Zhang).

[1] This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) doctoral training grant EP/M508147/1.
[2] Jie Zhang was supported by a Leverhulme Trust Research Project Grant (2021 – 2024).

https://doi.org/10.1016/j.artint.2022.103769
0004-3702/

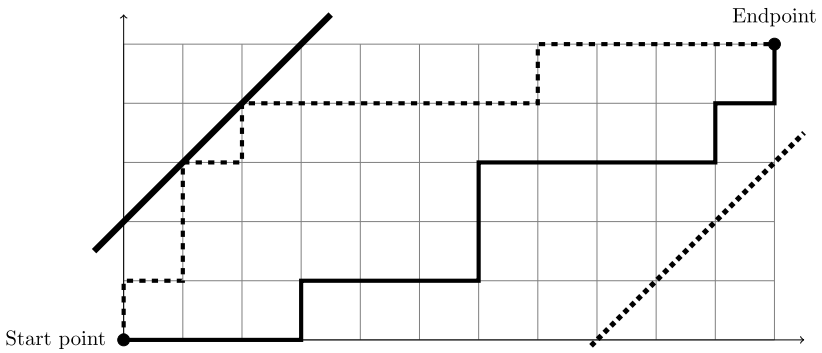


Fig. 1. Examples of a $\mathbb{Z}^2$ lattice and two lattice paths. Both lattice paths start in $(0, 0)$ and end in $(12, 6)$. Moreover, both lattice paths use only horizontal and vertical unit steps. The two vertical lines represent path restrictions. The lower right dotted line might be a restriction which lattice paths are not allowed to touch or cross. The vertical line to the left might be boundary a path has to touch. For example, the dashed line touches the line twice.

Optimal memoryless random strategies on polylines are a notable setting within adversarial patrolling for four reasons. Firstly, polylines, graphs where the nodes form either a line (open polyline) or a circle (closed polyline), represent an important real-world environment. They are useful to model the patrolling of either a fence or perimeter. Moreover, despite their discrete nature they can be used to model continuous environments [6]. Secondly, random strategies play an important role in adversarial patrolling since only these can guarantee that full-knowledge adversaries, i.e. those with knowledge of the defender's strategy, can be detected or caught [7]. The level of the adversary's knowledge is important in adversarial patrolling since it addresses the defender's lack of knowledge of the adversary's capabilities, as well as the adversary's reasoning for choosing a location for their penetration attempt. Therefore, it forms the basis of determining the right strategy, and further informs on the possibility to detect penetration attempts in general. Thirdly, *memoryless* strategies, where the next step of a robot depends only on the position of the robot and not on any previous locations, perform well in comparison to longer histories, i.e. where the next step depends on a number of previous locations [6,7] Memoryless strategies are also said to satisfy the *Markov property*. Finally, in comparison to the usually intractable problems of security scenarios in general, and patrolling in particular, the considered setting allows tractable results [6,8,9].

Addressing the computation of random strategies in this setting, the current approach uses a black-box algorithm with significant time and space complexity. The probability of detecting an adversary's penetration attempt depends on the possible movements of the robots and the paths the robots walk using a particular strategy. Based on this, optimal random strategies can be determined with a two-step process. Firstly, the probability functions that describe the robots' movements are determined. Secondly, the resulting probability functions are used to determine an optimal strategy. For the first step, the current approach is to algorithmically determine the probability functions using Markov chain models [6,10,9,11]. This approach has two problems. Firstly, the algorithm has to be repeated for every problem instance since the probability functions depend on the size of the polyline as well as the penetration time, i.e. the time the adversary requires to breach the perimeter or fence. Secondly, the algorithm that calculates the probability functions is a black box and does not allow comprehension of the strategy space. This is despite the fact that prevailing classes of graphs like polylines allow succinct expression of the number of possible paths.

We leverage this possibility to succinctly express the number of possible paths in order to analytically express the probability functions. We achieve this via a novel approach, modelling all possible paths of a robot's random strategy as lattice paths. A lattice path is a path along points in an Euclidean space (see also Fig. 1). Lattice path problems aim to count the number of lattice paths given a start and end point of the paths, allowed step directions, coordinates restricting the paths, and further path features. Lattice paths have been studied for a long time and are, despite apparent simplicity, powerful as well as having applications to many problems including physical systems, encoding, probability and statistics [12]. The counting via lattice paths directly allows us to explicitly state the probability functions.

Our main contribution is the modelling of the robots' paths as lattice paths (see Section 3.1 and 3.2). For two movement types, and open and closed polylines, we describe the characteristics that influence the different paths the robots can walk and how this can be translated into lattice path characteristics. Using the lattice path representation, we count the number of paths which immediately allow us to explicitly state the probability of detecting the adversary at different nodes. This removes the previously necessary first step of calculating probability functions, such that the remaining (previously second) step of solving the system of equations for an optimal strategy can be done efficiently. We show empirically that this reduces the runtime significantly by up to several hours to a fraction of the previously required runtime (see Section 6.1). Additionally, we show that explicit probability functions provide further advantages by allowing structural analyses. We perform such structural analyses for one movement type which allows us to reduce the runtime even further (see Section 5). Finally, the fractional runtime facilitates experiments at a much larger scale and thus allows an extensive analysis of optimal strategies (see Section 6.2). The analysis allows us to empirically show that on the closed polyline an optimal strategy of a robot that has to turn to change direction has a significantly increased probability of up to 0.4 of detecting an adversary

compared to that of a robot that can freely go in either direction (omnidirectional). Additionally, we highlight the optimal strategies for omnidirectional movement on an open polyline.

In the remainder of this paper we discuss related work (Section 1.1); formally introduce the model (Section 2); discuss the algorithm to calculate an optimal strategy (Section 4); and we provide and discuss our results.

1.1. Related work

Multi-robot patrolling is an area of ongoing research with substantial attention in the past decade [13]. The area's four main modelling dimensions are: the objective, the type of adversary, the environment and the approach. Objectives are broadly divided between regular patrolling and adversarial patrolling [13]. In regular patrolling the aim is to optimise a frequency related goal; for example, minimising travel time or maximising the visits of important points [14]. Contrastingly, the aim of adversarial patrolling is to defend against an adversary, by detecting [15] or handling [9] penetration events.

The adversarial model largely focuses on the adversary's knowledge about the robot's strategy, varying from no information (random strategies), ability to learn, and possession of full knowledge [16,15,13]. For example, the learning adversary in the work of Sak et al. [7] collects information on the robot's visits and uses this data to predict safe attack times. In comparison, we assume that the adversary has full knowledge of the defenders' strategy; this could be interpreted as the adversary has had enough time to learn the strategy. Such a full-knowledge adversary or an adversary that can learn can only be detected or caught using random strategies [6].

The environment in patrolling settings is generally assumed to be discrete [4] or the underlying continuous environment is discretised. For example, Elmaliach et al. [14] consider a 2D environment which they divide into cells. Similarly, the nodes in Sea et al. [17] represent coordinates on a plane. Discrete environments might be general graphs [18,15], which represent important points, or polylines [6,19], which represent fences or perimeters. Furthermore, Agmon et al. [6] show that any continuous fence or perimeter can be represented as a graph with nodes that represent sections of equal travel time. We use the same technique and thus our results, for discrete polylines, can be applied to cover continuous environments as well.

Multi-robot adversarial patrolling also overlaps with the game-theoretic area of security games [9,15]. This area is dominated by leader-follower or Stackelberg games [9] in which the leader, the defender, makes a move, i.e. commits to a strategy; then the attacker can respond with a move, i.e. a penetration attempt. The overall goal is to find equilibria that are good for the defender.

However, general graphs and game-theoretic approaches are mostly either NP-hard [8] or, due to their non-linear constraints and the application of non-linear optimisation, assumed to be NP-hard [9]. For example, Chevaleyre [18] and Basilico et al. [15] prove that finding an optimal strategy is NP-hard. Consequentially, many works use experimental evaluation of heuristics [18–20,15,21,17,13]. These heuristic approaches are in contrast to our work of finding specific optimal solutions for specific graphs and movement patterns.

A more closely related game is the patrolling game introduced by Alpern et al. [22]. They assume the defender and attacker are playing a zero-sum game whose structure is similar to the one in our work. The game is played on a graph and the attacker takes a specified number of steps to attack one of the nodes. In comparison to our work, their aim is to determine or bound the probability that the attacker is detected. Moreover, in contrast with our assumption of a full-knowledge adversary, they consider different strategies of the attacker. These strategies include: uniform probability over the nodes as well as strategies that consider nodes based on specific features of the graph or the movement of the robots. More recently, Alpern et al. [23] determined a patrolling strategy for the patrolling game on a line graph and a penetration time of 2. Their setting is different to ours with regards to the types of strategies and the knowledge of the adversary. Their defender chooses mixed strategies of paths and the attacker chooses a strategy first, whereas in our case we consider memoryless walks, the defender sets a strategy first and the attacker knows that strategy.

Our work is close to the works of Agmon et al. [6], and subsequent papers addressing different aspects like coordinated attacks [10], sequential attacks [9] and deception [11]. Their aim is to find optimal polynomial-time patrol strategies for specific graphs and specific goals maximising the minimal probability of the adversary being detected. They use algorithmic approaches, mainly the construction and evaluation of Markov transition matrices, to find the probability of the robots detecting the adversary [6] or the number of paths a robot can walk [9]. In contrast, we give explicit terms and functions for the number of paths and the probability of detecting the adversary which removes the necessity for the respective algorithms. Moreover, our approach has a much faster runtime and thus allows us a much larger empirical analysis instead of providing a few examples.

2. Preliminaries

We consider the setting of multi-robot patrolling around a perimeter or along a fence. The homogeneous robots are used by a defender who wants to detect penetration attempts by an adversary. A penetration attempt is the presence of an adversary with the intent of passing the perimeter or fence. In this work, we focus on a setting with one robot since Agmon et al. established that a single-robot model can be extended to a multi-robot setting by placing the robots equidistant to each other and synchronising their movements [6].

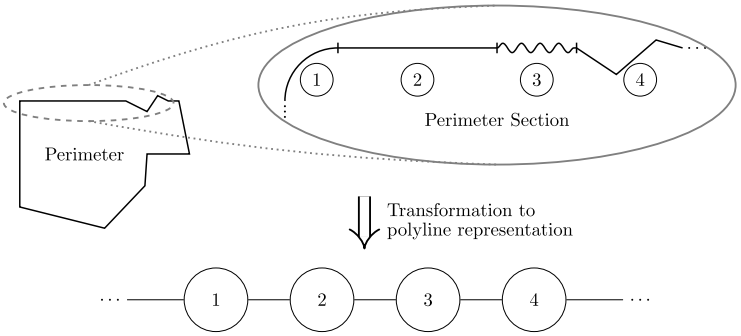


Fig. 2. Illustration of the transformation from a continuous perimeter to the polyline representation. The left polygon (marked with 'Perimeter') represents a continuous perimeter patrolling path. The top part is a magnification of a section of the perimeter linking the perimeter's segmentation to the polyline. Segment 1 represents a bend which requires the robot to slow down to change direction. Segment 2 is a straight stretch with good visibility allowing the robot to cover more ground than in other segments. Segment 3 represents an uneven surface where the robot needs longer to drive and scan. Segment 4 may be a part where the robot has to slow down to avoid obstacles. The resulting polyline graph (bottom) represents the segments with fixed traversal and scan time.

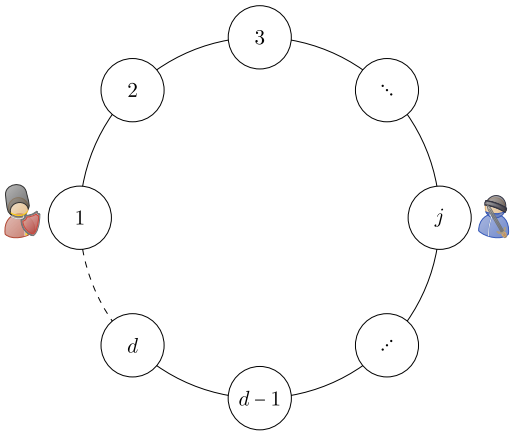


Fig. 3. The polyline graph with d segments. The dashed line indicates the edge that is present in the perimeter/circle but not in the fence/line. The guard with the shield represents the robot positioned in segment 1 and the other person (wearing a jacket and a knit cap, and holding a sword) represents the adversary penetrating in segment j.

The perimeter or fence the robot is patrolling is represented by an open or closed polyline with $d \in \mathbb{N}$ nodes called *segments* (see Fig. 3). The nodes are numbered 1 to d such that node i is adjacent to node $i-1$ for $i \in \{2, \ldots, d\}$ as in Fig. 3. We derive such a representation from a continuous perimeter or fence by dividing it into segments as depicted in Fig. 2. The division is done such that there is a fixed time which allows the robot to traverse and use its sensors to scan for penetration attempts in any one segment [6]. Hence, a segment might correspond to longer or shorter stretches of the perimeter or fence, based on a difference in traversal/scanning time, e.g. different types of terrain. Thus, irrespective of the actual environment, we have a discrete polyline representing either a perimeter or a fence.

1. **Perimeter/Circle**: A perimeter is modelled as a closed polyline, i.e. a graph that is one connected component where all nodes have degree two. Fig. 3 with the dashed line being an edge represents this. Due to the shape we also call this environment a circle.
2. **Fence/Line**: A fence is modelled as an open polyline, i.e. a graph that is one connected component where all nodes except two have degree two and the remaining two nodes have degree one. Fig. 3 without the dashed line represents this. Due to the shape we also call this environment a line.

The discrete environment also allows us to assume a discrete time model where a robot needs one time step to go from one segment to an adjacent segment [6].

This work considers memoryless robots where the next step the robot takes relies only on its current location. On the perimeter or fence the memoryless robot shall detect a penetration attempt by a full-knowledge adversary. A penetration attempt falls into one of the segments $1 \leq j \leq d$ and requires a penetration time of t time steps, i.e. a multiple of the time a robot needs to traverse any segment. The defender does not know the target segment j nor the time at which the adversary starts the penetration attempt. The only information known to the defender is the time t the adversary needs for

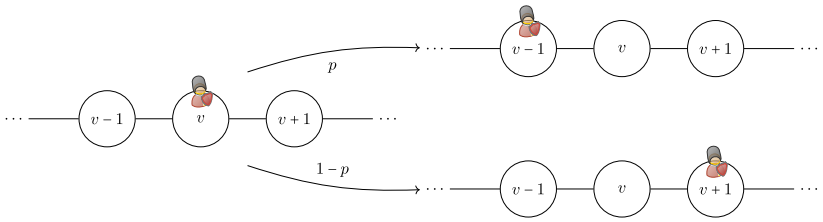


Fig. 4. The omnidirectional robot's movement. If the robot (represented by the guard with shield) is in a segment v then with a probability of p it moves into segment $v-1$ to the left in the next time step and with a probability of $1-p$ it moves into segment $v+1$ on the right.

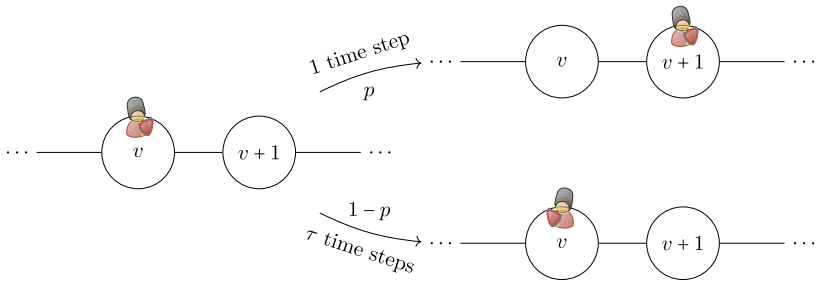


Fig. 5. The directional robot's movement. If the robot (represented by the guard with shield) is in a segment v then with a probability of p it moves into segment $v+1$ to the right in the next time step and with a probability of $1-p$ it turns around in segment v requiring τ time steps. The case where the robot faces left and would walk ahead into segment $v-1$ is analogue.

a successful penetration. A penetration attempt is detected if a robot reaches the segment targeted by the adversary within the t time steps.

We consider a single penetration attempt and we set the start time of this penetration attempt to be time 0. Accordingly, the important time horizon is from time 0 to at most the end of the penetration attempt (time t). The robot's actions outside of this time horizon are irrelevant since an adversary's detection by a memoryless robot depends only on the robot's location with respect to the targeted segment j and the robot's movements during the penetration attempt.

We also limit the range of penetration times t, similar to previous works, to those where the robot can reach the adversary in time but is not guaranteed to detect the adversary. This means we exclude those instances where the penetration time is so short that a robot from its current position cannot reach all segments. In this case the adversary would choose a segment that is not reachable and avoid detection, e.g. in a circle of size 6 with 2 time steps the robot can reach segments 2 and 3 clockwise, and segments 4 and 5 anticlockwise but cannot reach segment 6. Simultaneously, we exclude those instances where the penetration time is so long that the robot can simply continue to walk in one direction (including turning around and changing direction at the end of the fence in the case of the fence) and reach all segments at least once during the penetration time. In this case the adversary has no chance of a successful penetration attempt, e.g. with a penetration time of 5 on the same circle of size 6 the robot could reach every segment by just walking clockwise. Hence, we consider penetration times where the robot is able to reach any segment but could never reach all within the penetration time.

From a top down view, the polyline restricts the robot's movement to two directions, clockwise and anticlockwise, or right and left. We differentiate between two movement types based on if the robot can freely move to either adjacent segment in one time step, or if the robot has a direction and needs to turn around first to reach a segment in opposite direction [6]. This models robots with different modes of movement and sensors, e.g. driving with a forward facing camera or movement on rails scanning in all directions.

1. **Omnidirectional**: The omnidirectional movement allows a robot to freely walk into either adjacent segment in one time step. In a circle this is the next segment in either the clockwise or anticlockwise direction and on a line it is the segment to the left or the right of the current segment (see Fig. 4).
2. **Directional**: The robot has a direction pointing along the axis of the polyline in one or the other direction. The robot can walk into the segment ahead (in terms of its direction) of the current segment in one time step but needs to turn around to change direction which takes $\tau \geq 1$ time steps (see Fig. 5).

Having established the environment and the movement, next we examine the strategy space and define optimal strategies. Firstly, the robot has to act randomly (due to the adversary's full knowledge of the robots' strategy) and any random strategy is a random walk. Hence, in line with the fact that there are two possible actions, e.g. clockwise and anticlockwise, and since we are considering memoryless strategies, the random walk is a Bernoulli process. Thus, any strategy can be described by one parameter $p \in [0,1]$ and any motion is with probability p one of the actions and with probability $1-p$ the other action. For example, with probability p the robot goes clockwise and with probability $1-p$ the robot

goes anticlockwise. Based on this strategy space, an optimal strategy is a value of p that maximises the probability of detecting the adversary. However, again, since the adversary has full knowledge, they will choose the segment j with the lowest probability of the robot reaching it in time. Thus, to optimise p we need to maximise the probability of the segment with minimal probability of the robot detecting the adversary within the given time. More formally, we seek $p \in \arg\max_{p' \in [0,1]} \min_{j \in [d]} Pr(j', j, t, d, p')$ where $Pr(j' j, t, d, p')$ is the probability that on a graph of d segments a robot in segment j' using strategy parameter p' detects, within the t time steps, the adversary in segment j. This is also referred to as *minmax* approach [6].

Finally, the probability functions $Pr(j' j, t, d, p')$ comprise of the probabilities of the individual paths the robot can take. Since this depends on all paths a robot can walk, we start with defining paths as well as distance. Then, utilising the path definition, we can define the probability functions. In brief, a path is a sequence of consecutive segments representing the robots' clockwise, anticlockwise, left, right, forward or turn movements.

Definition 1 *(Path, Right/Clockwise Step, Left/Anticlockwise Step).* A *path* of length l is a sequence of consecutive segments, i.e. a vector $(s_1, s_2, \ldots, s_l) \in [d]^l$ where every pair of segments (s_i, s_{i+1}) for $1 \leq i < l$ are adjacent in the graph. For a path, a *step to the right/clockwise step* or a *step to the left/anticlockwise step* is a tuple (s_{i-1}, s_i) where the second segment is right of/next to in the clockwise direction or left of/next to in the anticlockwise direction of the first segment, respectively.

Definition 2 *(Distance $dist(\cdot, \cdot)$).* The distance $dist(j, j')$ between two segments j and j' is the number of edges of a shortest path. Thus, $dist(j, j')$ is $|j - j'|$ if we consider a line or a circle with paths in the clockwise direction, and $d - |j - j'|$ if we consider a circle with paths in the anticlockwise direction. For a circle, the respective direction will be clear from context.

Since the random walk is a Bernoulli process, describing the probability of one path is straightforward. The outcome set Ω consists of all paths of length t from the start segment and the probability measure P_W is the obvious Bernoulli process measure.

Definition 3 *(Probability of a Path).* The *probability of a path* $\omega \in \Omega$ is $P_W(\omega) = p^Y \cdot (1-p)^X$ where X and Y are the number of actions depending on the movement type.

Detecting the adversary means that within the penetration time t there is a time step in which the robot is in the segment of the adversary's penetration attempt. In other words, if the adversary attempts to penetrate at segment $j \in [d]$ and the robot walks a path that contains j, the robot will detect the adversary. Hence, we can define the probability measure of detecting the adversary $Pr(j', j, t, d, p)$ as the probability of reaching the adversary:

$$Pr(j', j, t, d, p) := P(\{\omega | \omega \in \Omega \wedge j \in \omega\}) = \sum_{\omega \in \Omega : j \in \omega} P_W(\omega).$$

We note with regards to the extension to multiple robots that the probability of one robot detecting the adversary is independent of another robot detecting the adversary. Hence, the probability of any robot detecting the adversary can be calculated using the addition law of probability.

3. Determining the probability functions

Our main theoretical contribution is the modelling of the robots' path using lattice paths, allowing us to explicitly state the probability of catching the adversary. For this we have four settings: omnidirectional movement in a circle; directional movement in a circle; omnidirectional movement on a line; and finally, directional movement on a line. For all four settings determining the probability functions is slightly different. However, the general steps are the same and the models developed have some overlap. Hence, we use the omnidirectional movement in a circle to introduce the general procedure and the modelling. This encompasses the respective types of paths as well as the lattice path representation, and how the counting of paths yields the probability functions.

3.1. Omnidirectional movement in a circle

For an omnidirectional robot in a circle specifically we conclude with proving that the probability of the robot reaching a specified segment j within the t time steps is as follows (where $C(\cdot, \cdot)$ is as defined in Theorem 2).

Theorem 1. *The probability that an omnidirectional robot in segment* 1 *using strategy parameter* p *detects an adversary in segment* j *within a horizon of* t *on a perimeter of size* d *is*

$$Pr(1, j, t, d, p) = \sum_{i=0}^{\left\lfloor \frac{t-d+j-1}{2} \right\rfloor} C(d-j+i, i) \cdot p^{d+i-j+1} \cdot (1-p)^i + \sum_{i=0}^{\left\lfloor \frac{t-j+1}{2} \right\rfloor} C(j-2+i, i) \cdot p^i \cdot (1-p)^{i+j-1}.$$

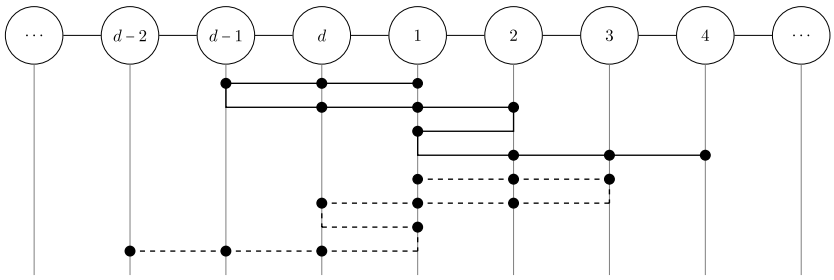


Fig. 6. Example of a path and its mirrored path. The solid path starts in segment 1 and the target segment is segment 4. Transforming every step from this path into a step in the opposite direction results in the dashed path. This path starts in 1 and the target segment is segment $d-(4-2)=d-2$ as stated by Lemma 1.

Considering the movement of one robot, without loss of generality, the robot is placed in segment 1. This can be done since the segment numbering is arbitrary and we can always shift the segments for several robots according to their position. Furthermore, we use an arbitrary but fixed $j \in [d]$ as the specified target segment. Moreover, the probability of a path $P_W = p^Y \cdot (1-p)^X$ (see also Definition 3) is determined by the number of clockwise steps X and the number of anticlockwise steps Y. Finally, the penetration time range for this setting is $\lfloor d/2 \rfloor \leq t \leq d-2$. It is easy to see that for a small t, i.e. $t < \lfloor d/2 \rfloor$, the robot cannot reach the adversary in segment $\lfloor d/2 \rfloor + 1$. Contrarily, for large t, i.e. $t \geq d-1$, the robot can deterministically walk in clockwise direction around the circle and reach every segment in $d-2$ steps. Therefore, the robot would always detect the adversary.

We prove Theorem 1 in Section 3.1.6 by determining the number of valid paths and partitioning them, and by using lattice path modelling to count the number of paths in one partition. In Section 3.1.4 we show how one group of paths can be modelled using lattice paths (see also Definition 6 for a formal definition of lattice paths) and how many paths there are (see Theorem 2). We provide the proof for this number of paths in one group in Section 3.1.5. The groups are determined by the possible number of steps away from and towards the target segment j which gives a freedom of movement described in Section 3.1.3. Preceding these results we establish two results that reduce the number of sub-settings we have to consider. Firstly, the symmetry of the circle means that we only have to consider reaching a segment in one direction and the other directions follow similarly (see Section 3.1.1 and Lemma 1). Secondly, by the definition of detection, i.e. the adversary is detected if the robot reaches the segment of the penetration attempt, we only need to consider paths that end in j and are of maximal length t (see Section 3.1.2 and Observation 2).

3.1.1. Path symmetry

We start with reducing the sub-settings we have to consider. We observe that, by virtue of a circle, for every path reaching some segment in clockwise direction there is a path that mirrors it, i.e. switches clockwise and anticlockwise steps, and therefore reaches a segment with the same distance to the robot's location in anticlockwise direction (see also Fig. 6). More formally presented in the following definition, we call the latter path the mirrored path of the former path.

Definition 4 *(Mirrored Path).* For a path of length l let $(s_1, \ldots, s_{l-1})$ be a sequence of clockwise and anticlockwise steps, i.e. $s_i \in \{\textit{clockwise}, \textit{anticlockwise}\}$ for $i \in [l-1]$. The *mirrored path* is the path whose clockwise and anticlockwise step sequence is $(s'_1, \ldots, s'_{l-1})$ with $s'_i = \textit{clockwise}$ if and only if $s_i = \textit{anticlockwise}$ for all $i \in [l-1]$.

By the definition, we can directly observe the number of clockwise and anticlockwise steps of a mirrored path.

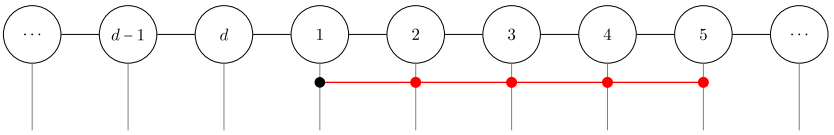
Observation 1. *For a path with X clockwise steps and Y anticlockwise steps, the mirrored path has Y clockwise steps and X anticlockwise steps.*

This symmetry means that a set of paths and the set of their mirrored paths have to have the same cardinality.

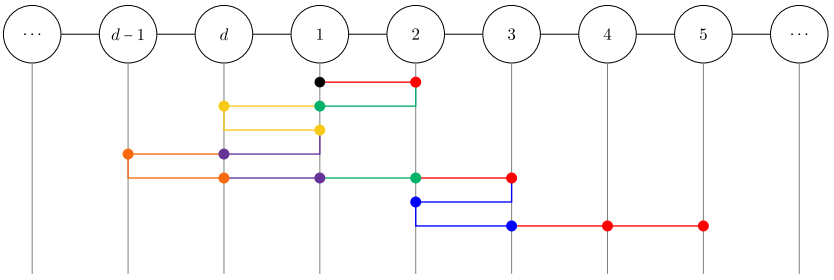
Lemma 1. *Moving clockwise into segment $j \in \{2, \ldots, \lceil d/2 \rceil\}$ is mirrored by moving anticlockwise into segment $d-(j-2) \in \{\lceil d/2 \rceil + 1, \ldots, d\}$ and vice versa.*

Proof. Let P be the set of all paths that start in segment 1 and end in segment $j \in \{2, \ldots, \lceil d/2 \rceil\}$ which was reached in the clockwise direction. Overall, these paths all walk $j-1 = X-Y$ steps in the clockwise direction. However, every path in P has a mirrored path which walks $X-Y$ steps in the anticlockwise direction into segment $d-(X-Y-1) = d-(j-2)$. The opposite follows similarly. □

Consequently, we can focus on one direction and the other direction follows.



(a) In the direct path each step of the robot is towards the target segment (in this case clockwise).



(b) Besides the direct path the robot can perform further steps which results in alternate paths. The depicted path is one possibility with 10 extra steps. It can be observed, that the direct steps are still present and that all other steps come in pairs.

Fig. 7. Example of the freedom of movement for a path where a robot is in segment 1 and the target segment is 5. The two figures show the direct path (Figure a) and a path with extra steps (Figure b). (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

3.1.2. Restriction of necessary paths

In addition to the previous section, we can further limit the number of paths we have to consider by observing that any path passing j is covered by a sub-path that ends in j. This is based on the fact that any path passing through j detects the adversary (see Section 2). Accordingly, we divide all paths into sets of paths which share a common prefix, i.e. all paths that are identical from segment 1 up to the first occurrence of j.

Let ω' be such a prefix that ends in j and has length $\ell \leq t$. The probability that the robot walks any path of length t with this prefix is $P_W(\omega') \cdot \sum_{i=0}^{t-\ell} p^i (1-p)^{t-\ell}$. We observe that the sum of the suffixes in this expression is one. This observation also makes sense from the probability argument that they are the space of all possible paths of length $t - \ell$. Consequently, the probability that the robot walks any path of the set is equivalent to the probability of the common prefix ω'. Hence, we can conclude that to determine the probability of detecting the adversary we only need to consider the prefixes (paths ending in j).

Observation 2. *It suffices to consider the subset of paths which end in segment j, without having passed it before, and are at most of length t.*

3.1.3. Freedom of movement

In preparation of the lattice path modelling we start with partitioning the robot's paths. As a result of the previous two sections, we can focus on the paths of maximal length t in one direction that end in j. We start with observing that the robot needs a (distance dependent) number of steps to reach segment j but otherwise has a freedom to walk as many *additional steps* as the difference between t and the distance to j allows (see Fig. 7). Based on the further observation that for a fixed amount of additional steps the probability is the same, we partition the paths based on this number of additional steps.

In more detail, as stated in the preliminaries we consider the start time to be 0 and the robot has up to t time steps to reach the segment of the adversary's penetration attempt (see also Section 2). Therefore, the robot can walk up to t steps before the adversary succeeds without detection. Within these t steps the robot needs to reach j at a distance of $dist := dist(1, j)$ (this distance is $j - 1$ for the clockwise direction and $d - j + 1$ for the anticlockwise direction). Hence, besides the $dist$ moves to reach j the robot has $t - dist$ additional steps. Their number is greater than or equal to zero since we know $t \geq dist$ (see Section 2). We refer to the steps that reach j as the direct path (see also Fig. 7a) and any step in the same direction as towards j and any step in the opposite direction as away from j.

Definition 5 *(Direct Path/Towards/Away).* The direct path is a path that has exactly $dist$ steps and ends in j. In the case that this is in clockwise direction, any step in the anticlockwise direction is away from the target segment and any step in the clockwise direction is towards the target segment. The converse is true for the anticlockwise direction.

Considering the direct path as the starting point, we observe that any step away from j has to be countered by a step towards j at some point. For example, if, as in Fig. 7, the robot reaches a segment in the clockwise direction, any step in the anticlockwise direction (away) has to be countered by a step in the clockwise direction (towards).

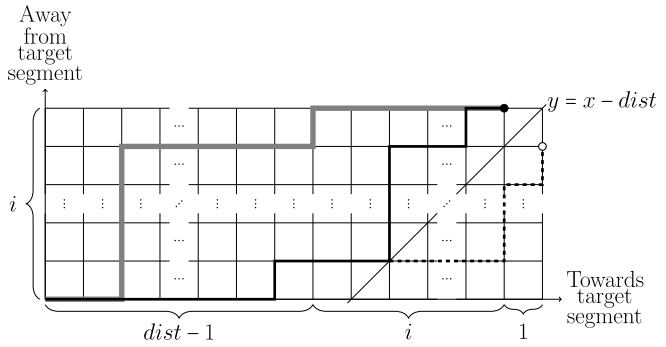


Fig. 8. The lattice that represents the movement pattern. Steps along the x-axis are steps in the anticlockwise direction and steps along the y-axis are steps in the clockwise direction. The start is the bottom left corner and the finish is the top right corner. The disk (filled circle) and the circle are the end points of the valid and the invalid paths, respectively. The thicker grey line is a valid path whereas the solid black path is an illegal path that touches the line $y = x - dist$. The dashed path shows how reflecting the solid black path leads to the circle.

Observation 3. *The robot may walk up to* $\lfloor (t - dist)/2 \rfloor$ *steps away from* j *which have to be countered by the same number of steps towards* j.

Hence, by Observation 3 and 2 for every path length of $dist + 2i$ with $i \in \{0, 1, \ldots \lfloor t - dist/2 \rfloor\}$ we have a different amount of steps and their placement determines the robot's path.

3.1.4. Modelling as lattice paths

We can use lattice path modelling to count the number of paths in one partition, as established in the previous section. Taking these collectively over all partitions gives us the number of all paths. The overall result, the number of paths for fixed d, j and i, is as stated in the following theorem. Since the number matches an entry of Catalan's triangle [24][3] we use this as the notation for the number.

Theorem 2. *The number of paths of exact length* $dist + 2i$ *is* $C(dist - 1 + i, i)$ *with* $C(n, k) := \binom{n+k}{k} - \binom{n+k}{k-1}$.

In order to establish this result, we define the lattice path setting (see Fig. 8) as follows. The lattice is the $\mathbb{N}^2$ lattice and the robot's initial position corresponds to $(0, 0)$, the bottom left corner. Starting there, we allow the following lattice paths.

Definition 6 *(Lattice Path).* A *lattice path* is a sequence of horizontal and vertical unit steps beginning at $(0, 0)$ and ending at a specified point.

The horizontal direction represents movement towards the target and the vertical direction represents movement away from the target. Hence, a point $(x, y) \in \mathbb{N}^2$ corresponds to the robot being in segment $y - x + 1$ if $x \leq y$ or in segment $y - x + d + 1$ if the opposite holds. The fact that we consider only paths that end in j is incorporated by including a boundary line $y = x - dist$ which no path is allowed to cross.

Finally, while a robot's path that ends in j after $dist + 2i$ steps corresponds to a lattice path that ends in $(dist + i, i)$, for technical reasons, we will consider lattice paths to $(dist - 1 + i, i)$. The number of paths is not affected by this. Logically, every path of the robot that ends in j after $dist + 2i$ steps must end in $j - 1$ after $dist - 1 + i$ steps. Graphically, considering Fig. 8, the only way to reach $(dist + i, i)$ without crossing the line is by passing through $(dist - 1 + i, i)$.

3.1.5. Counting the lattice paths

The lattice path modelling of the previous section immediately allows us to show that the number of lattice paths from $(0, 0)$ to $(dist - 1 + i, i)$ amounts to the number stated in Theorem 2.

Lemma 2. *The number of lattice paths from* $(0, 0)$ *to* $(dist - 1 + i, i)$ *not crossing line* $y = x - dist$ *is* $C(dist - 1 + i, i)$.

Proof. The number of unrestricted lattice paths (not restricted by a line) from $(0, 0)$ to (a, b) is $\binom{a+b}{b}$ (see Theorem 10.3.1 in *Lattice Path Enumeration* [12]).

For the number of lattice paths restricted by a line, we can count these by using the general ballot theorem [25] or André's reflection principle [12]. The reflection principle describes a procedure where a path's horizontal steps after the

[3] Catalan's triangle generalises the Catalan numbers which in turn appear in recursive counting problems [12].

first touch point become vertical steps and vice versa. More importantly, according to the principle, any such reflected path ends in $(dist + i, i - 1)$ (see Fig. 8). Hence, using above binomial coefficient for the total number as well as the number of invalid lattice paths (path that touch or cross the line), the total number of lattice paths is

$$\binom{dist - 1 + 2i}{i} - \binom{dist - 1 + 2i}{i - 1} = \frac{dist}{i}\binom{dist - 1 + 2i}{i - 1}.$$

This is equivalent to $C(dist - 1 + i, i)$. □

The number of the robot's paths follows immediately.

Proof of Theorem 2. By the correspondence between the robot's paths and the lattice paths (see Section 3.1.4), Lemma 2 directly implies the claim. □

3.1.6. Probability of detecting the adversary

Finally, using Theorem 2 we prove that the probability functions stated in Theorem 1 express the probability that a robot reaches and thus detects an adversary.

Proof of Theorem 1. The probability functions follow by combining the statements in this section. Firstly, Observation 1 and Lemma 1 imply that the results hold for both directions. Secondly, as stated in Section 3.1.3 by Observations 2 and 3 we can take the sum of paths of all partitions to get the number of all paths. Finally, Theorem 2 gives the number of paths for one partition and Definition 3 gives the probability of a single path. □

3.2. Remaining settings

In Section 3.1 we present the general approach which provides the probability functions using omnidirectional movement in a circle. The probability functions in the other three settings (directional movement in a circle; omnidirectional movement on a line and directional movement on a line) can be found similarly by modelling the robot's paths as lattice paths. Modelling of these remaining settings share a large number of similarities with the first setting; hence, we focus on the important differences. We present the modelling and the probabilities for the settings of the directional movement in a circle and the omnidirectional movement on a line. For the directional movement on a line we simply present the probability functions since no further modelling is required.

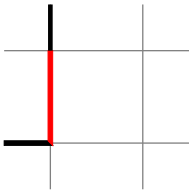
3.2.1. Directional movement in a circle

For the case of a robot with directional movement in a circle we show that the probability depends on three types of additional motions which give us the following functions (where C_{DMC} is as in Definition 8).

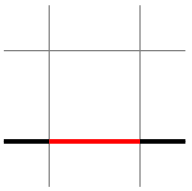
Theorem 3. *The probability of a directional robot detecting an adversary in segment j on a perimeter of size d is*

$$\begin{aligned} Pr(1, j, t, d, p) = &\sum_{i=0}^{\left\lfloor \frac{t-d+j-2}{2\cdot\tau} \right\rfloor} \sum_{m=0}^{\min\left\{i, \left\lfloor \frac{t-d+j-2}{2\cdot\tau+2} \right\rfloor\right\}} \sum_{k=0}^{\left\lfloor \frac{t-d+j-2-2i-2m}{2} \right\rfloor} \\ &C_{DMC}(d - j + 1, k, i, m) \\ &\cdot p^{d-j+1+2k+2m} \cdot (1-p)^{2m+2(i-m)+1} \\ &+ \sum_{i=0}^{\left\lfloor \frac{t-j}{2\cdot\tau} \right\rfloor} \sum_{m=0}^{\min\left\{i, \left\lfloor \frac{t-j}{2\cdot\tau+2} \right\rfloor\right\}} \sum_{k=0}^{\left\lfloor \frac{t-j-2i-2m}{2} \right\rfloor} C_{DMC}(j - 1, k, i, m) \\ &\cdot p^{j-1+2k+2m} \cdot (1-p)^{2m+2(i-m)}. \end{aligned}$$

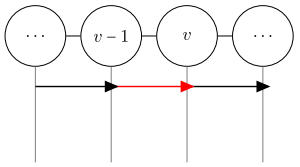
The difference to the omnidirectional movement is that a path and its probability $P_W = p^X \cdot (1-p)^Y$ is not solely dependent on the number of steps X, but also on the number of directional changes Y. It is especially with these directional changes, which do not directly correspond to steps, that require additional modelling. We incorporate the turning of the robot into the lattice path model by separating the additional motions (the actions available from the difference between t and the distance to j) into three different types. The first motion type is called additional steps. This motion is the same as in the omnidirectional case; the additional steps are all steps that are not directly required to reach the target segment. The other two types reflect the changes in direction and are called turn and spin.



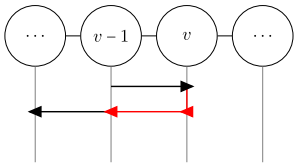
(a) Examples of a change in direction. The path changes direction upwards (red step).



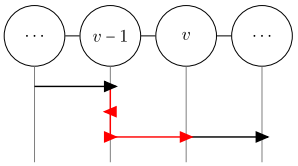
(b) Examples of a path without a change in direction. The path simply continues.



(c) Examples of a path without change in direction.



(d) Example of a turn. After reaching segment v the robot changes direction and continues in the new direction.



(e) Examples of a spin. In segment $v-1$ the robot changes direction twice before continuing in the original direction.

Fig. 9. Illustration of robot's paths and lattice paths with and without directional changes. Figures (a) and (b) are sections of a lattice path. Figures (c), (d) and (e) show the robot's paths on the graph. At the top are the segments of the polyline. The robot's movement is indicated as arrows on the grey lines under the segments. The arrowheads on the grey lines represent the positioning and the direction of the robot, whereas the lines represent the connecting steps of the robot. The images highlight how turns and steps can be modelled as lattice path features but spins have to be considered separately. The simple walk in one direction in Figure (c) would simply be a lattice path that does not change direction as in Figure (b). Likewise, the turn in Figure (d) can be represented as a turn in a lattice path as shown in Figure (a). However, the spin in Figure (e) would appear in a lattice path in the same way as a path without any directional change (see also Figure (b)). (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

Definition 7 *(Turn/Spin).* A *turn* is a change of direction followed by a step into the new direction and a *spin* are two consecutive changes of direction.

We separate changes in direction into turns and spins since turns can be represented in the lattice, as presented in Section 3.1.4. Similar to additional steps, they can be included in the lattice path model as changes in the direction since they are followed by a step (see also Fig. 9). This is not the case for spins. Nevertheless, since a spin can be performed at any point of a path they can be factored in after accounting for steps and turns.

Similar to the steps in Section 3.1, the majority of the results in this section help to establish the number of paths in one of the partitions of all paths. A partition in this setting is based on the number of additional steps, turns and spins. Hence, after establishing the partitioning for this case, we prove that the number of paths for one partition is as follows.

Definition 8 *(C_{DMC}).* Let the number C_{DMC} be defined as follows (with *DMC* referring to directional movement in a circle):

$$C_{DMC}(dist, k, i, m) = \binom{dist + 2k + 2m + i - m - 1 - 1}{i - m} \cdot \left(\binom{dist + k + m - 1}{m} \binom{k + m}{m} - \binom{dist + k + m}{m} \binom{k + m - 1}{m} \right).$$

Lemma 3. *The number of paths from $(0,0)$ to $(dist + k + m - 1, k + m)$ with $2m$ turns, $2(i-m)$ spins, $2k$ additional steps, and not crossing line $y = x - dist$ is $C_{DMC}(dist, k, i, m)$.*

We prove Lemma 3 after first explaining how turns can be counted as changes of direction in the lattice path. This will allow us to prove that the second factor in Definition 8 corresponds to the pairs of turns and additional steps. The proof combines this with factoring in the pairs of spins (first factor in Definition 8).

Before we elaborate on this, we start with the partitioning. We indicate a partition for this case with three numbers: the number of changes of direction, the number of turns and the number of additional steps. Similar to the omnidirectional case we have $t - dist$ time steps for the robot to perform these motions in addition to reaching the target segment. Also as in the previous setting we can observe that all types of motions come in pairs. Both moving or turning away from the target has to be reversed at some point (see also Section 3.1.3 and Observation 3).

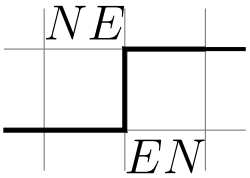


Fig. 10. A section of a lattice path that contains both types of turns: an EN- and a NE-turn. As indicated, the first turn is an EN-turn where a horizontal step is followed by a vertical step. This is followed by a NE-turn where a vertical step is followed by a horizontal step.

Observation 4. *Every additional motion a robot makes has to be countered by the same type of motion in the opposite direction at some point. The only exception is that reaching a segment in the anticlockwise direction needs an additional turn, derived from the assumption that the initial direction taken is clockwise.*

We determine the numbers for the partitions starting with the changes in direction. Since a turn takes τ steps, the robot can perform at most $\left\lfloor \frac{(t-dist)}{(2\cdot\tau)} \right\rfloor$ many pairs of changes in direction. Similar to the previous setting (see Section 3.1.3), the robot may perform any fixed number of these $i \in \left\{0, 1, \ldots, \left\lfloor \frac{(t-dist)}{(2\cdot\tau)} \right\rfloor\right\}$.

Of these i turn and spin pairs, we denote the number of pairs of turns with m. This number must be within the range of all changes of direction. It also has to be within the total number of possible turns including the two accompanying steps. Hence, m ranges from 0 to $\min\left\{i, \left\lfloor \frac{(t-dist)}{(2\cdot\tau+2)} \right\rfloor\right\}$. Consequently, a set number of turns imply a fixed number of $i-m$ pairs of spins. Finally, the remaining moves can be used for $k \in \left\{0, \cdots, \left\lfloor \frac{(t-dist-2i-2m)}{2} \right\rfloor\right\}$ additional steps. Altogether, this gives us a partition where we can count the paths for fixed i, m and k.

Within one partition we use lattice path modelling to count the number of paths for given additional steps k and turns m. We achieve this by extending the lattice path modelling introduced in Section 3.1.4 to factor in the turns. They can be represented as either a *north east turn* or an *east north turn* (see Fig. 10):

Definition 9 *(NE-turn/EN-turn).* A point on a lattice path is a *north east turn (NE-turn)* if it is the end point of a vertical step and the starting point of a horizontal step. Similarly, a point is an *east north turn (EN-turn)* if it is the end point of a horizontal step and the starting point of a vertical step.

We factor the number of turn pairs m by the number of NE-turns since those represent the total number of turns.

Lemma 4. *A robot's path with $2m$ turns is a lattice path with m NE-turns.*

Proof. Every turn of the robot shows up as a NE-turn or an EN-turn (see the thick grey path in Fig. 8 with two turns of either type), with two exceptions. The first exception is walking vertically from $(0, 0)$ which is a turn of direction (equivalent to EN-turn), and the second exception is walking vertically into the target segment. The latter step of walking into the target segment is not possible in our setting (compare Observation 2). Consequently, the lattice paths with m NE-turns represent all path with $2m$ turns. □

Using this we are able to prove the number of paths in one partition.

Proof of Lemma 3. We count the number of paths for $2k$ additional steps, $2i$ turns and spins and $2m$ turns. The term is comprised of two factors: the number of paths, given turns and additional steps, as well as all the ways to include the spins.

Firstly, we count all paths with m NE-turns, given the number of steps and turns (see Lemma 4). The number of paths from $(0, 0)$ to (a, b) with ℓ NE-turns is $\binom{a}{\ell}\binom{b}{\ell}$. This result is a special case of Equation 10.120 in Section 10.14 of *Lattice Path Enumeration* [12]. Hence, we can use this to determine the number of paths from $(0, 0)$ to $(dist - 1 + k + m, k + m)$. Moreover, as in Lemma 2, we get all valid paths by subtracting the paths that cross the line using the reflection principle. Together, this gives us the second factor in Definition 8.

Secondly, we have to include the spins to get the overall result. These can be performed in every segment along a path which means we can include them by distributing the $i-m$ pairs of spins over the $dist-1+2k+2m$ positions of a path. This is equivalent to putting indistinguishable balls in distinguishable boxes. Counting those combinations is an elementary result in enumerative combinatorics (see e.g. *Enumerative Combinatorics* [26]). Specifically, the number of r indistinguishable balls into s distinguishable boxes, allowing multiple balls per box, is $\binom{s+r-1}{r}$. Applying this gives us the first factor in Definition 8 and the product of the two factors gives us the overall result. □

Finally, the probability functions are determined by summing up over the partitions given by i, m and k similar to Theorem 1.

Proof of Theorem 3. For both directions i, m and k determine the number of changes in direction, the number of pairs of turns, and finally the number of pairs of additional steps, respectively. This depends on the distance which, as before, is different for the clockwise and the anticlockwise direction. As previously discussed, all possible combinations result in different numbers of paths. Hence, considering all combinations for both directions yields the sums.

For both directions we have to determine, for a given i, m and k combination, the probability of a path. In summary, the power of p is determined by the number of steps and the power of $1-p$ is the number of changes in direction. Hence, we have to add up these respective numbers. For both directions, the steps are determined by the distance, the pairs of additional steps, as well as the pairs of steps after turns. Therefore, we have $j-1+2k+2m$ for the clockwise direction and $d-j+1+2k+2m$ for the anticlockwise direction. Changes in direction are determined by the pairs of turns and the pairs of spins. Hence, we have $2m+2(i-m)$ for both directions; this is also true for the anticlockwise case where there is an additional change in direction under the assumption that the robot starts facing clockwise.

Finally, Lemma 3 gives the number of lattice paths for a given partition representing the number of paths the robot can walk. □

3.2.2. Omnidirectional movement on a line

Next we are considering the setting of the robot patrolling a line. The main difference from the circle are the defined ends of the line. At the ends the robot has to turn around, thus changing the probability and breaking the clockwise-anticlockwise symmetry. Hence, the probability here depends on the distance to the target segment and the distance to the end of the line. This also means that the probability is different for every pair consisting of the position of the robot and the position of the penetration attempt. Considering all these points gives us the following probability functions for this setting (where C_{OML} is as in Definition 11).

Theorem 4. *The probability of an omnidirectional robot detecting an adversary in segment j on a line of size d is*

$$Pr(j', j, t, d, p) = \sum_{i=0}^{\left\lfloor \frac{t-|j-j'|}{2} \right\rfloor} \Bigg[C_{OML}(j, j', 1, i, 0) \cdot p^i \cdot (1-p)^{|j-j'|+i} \sum_{k=1}^{i-|1-j'|+1} C_{OML}(j, j', 1, i, k) \cdot p^i \cdot (1-p)^{|j-j'|+i-k} \Bigg]$$

if segment j' is left of segment j and

$$Pr(j', j, t, d, p) = \sum_{i=0}^{\left\lfloor \frac{t-|j-j'|}{2} \right\rfloor} \Bigg[C_{OML}(j, j', d, i, 0) \cdot p^{|j-j'|+i} \cdot (1-p)^i \sum_{k=1}^{i-|d-j'|+1} C_{OML}(j, j', d, i, k) \cdot p^{|j-j'|+i-k} \cdot (1-p)^i \Bigg]$$

if segment j' is right of segment j. If the robot is in one of the end segments $\hat{j} \in \{1, d\}$ the probability is

$$Pr(\hat{j}, j, t, d, p) = \begin{cases} Pr(2, j, t-1, d, p) & \text{if } \hat{j} = 1 \\ Pr(d-1, j, t-1, d, p) & \text{if } \hat{j} = d. \end{cases}$$

Since the robot has to turn around at the end, its motion is now split into X right steps with probability $(1-p)$, Y left steps with probability p, and Z steps turning around in the end segments; this implies the probability measure $P_W(\omega) = p^Y \cdot (1-p)^X \cdot 1^Z$ for one path. This difference in probability between a step and reaching the end of the line means that for the partitioning of the robot's paths and the lattice path modelling, we have to consider the number of times the robot reaches the end. The latter will be done by introducing a second line (besides the one introduced in the modelling in Section 3.1) representing the end of the line (see Fig. 11).

Before we extend the lattice path modelling, we establish a symmetry which allow us to focus on one setting with arbitrary but fixed locations of the robot j' and the penetration attempt j. This symmetry is based on the fact that any setting where the target segment is left of the robot's initial segment is similar to a setting where the target segment is right of the robot's initial segment. The requirement for this is the following: the distance between the robot's initial segment and the other end of the line, as well as between the robot's initial segment and the target segment, has to be the same between the two settings. We only have to consider one end of the line since we still only have to take into account paths that end in the target segment (see also Section 2 and Observation 2) and therefore the robot can only reach one end of the line without detecting the adversary. The following lemma formalises this and thus allows us to focus on the case where the initial segment j' is left of the target segment j, i.e. $j' < j$, and the other case follows.

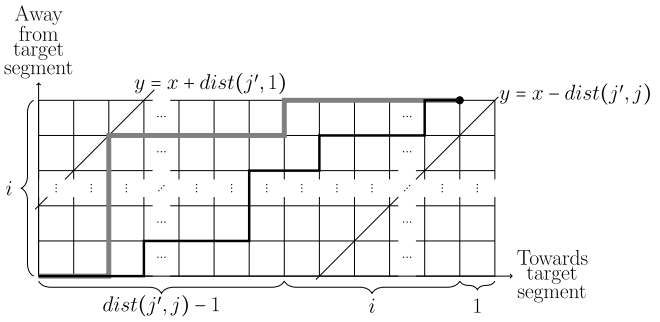


Fig. 11. The lattice for the line setting. The general lattice is like the one in Fig. 8. Also similar, the line $y = x - dist(j', j)$ is to exclude paths not just ending in segment j. The newly introduced second line $y = x - dist(j', 1)$ represents the end of the polyline. In comparison to the other restricting line, a path may touch the line, like in the case of the thicker grey line, but may not cross it. The black path is a valid path that touches neither line.

Lemma 5. *The number of paths from segment j' to segment j with $j' < j$ is the same as the number of paths from a segment $d - dist(1, j')$ to segment $d - dist(1, j)$ with $dist(j', j) := |j - j'|$.*

Next we establish the partitions for one fixed j' and j and how to count the number of paths within one partition. As before any path can have $i \in \left\{0, 1, \ldots, \left\lfloor \frac{(t - dist(j', j))}{2} \right\rfloor\right\}$ additional moves. In addition, we denote with k the number of times the end of the line is reached. The range of k is limited by the distances and the penetration time $0 \leq k \leq \left\lfloor \frac{(t - dist(1, j') - dist(1, j))}{2} \right\rfloor$.

Hence, fixed j, j', i and k determine one partition. Modelling one partition as a lattice path setting utilises the same lattice as in the setting of the circle (see Section 3.1.4 and Fig. 8). This includes the line $y = x - dist(j', j)$ representing that segment j is entered last. This holds with only a minor difference of perspective in that the point $(0, 0)$ now corresponds to j'. The main difference is the second line $y = x + dist(j', 1)$ which has been introduced to represent the end of the polyline. This line is the focus of the additional results in this section. We establish how to count the number of lattice paths that reach the end of the line k times and do not touch the line representing segment j. Altogether, this allows us to establish the number of paths for one partition in this setting, allowing us to prove Theorem 4.

In order to count the number of paths touching (not intersecting) the line $y = x + dist(j', 1)$ exactly k times, we first equate it to a setting where the line has been touched only once. We show this correspondence via a bijection between the two settings which adapts a bijection used by Spivey [27] (see also Fig. 12).

Lemma 6. *There is an explicit bijection between paths from $(0, 0)$ to (a, b) touching the line $y = x + s$ exactly k times and paths from $(0, 0)$ to $(a, b - k + 1)$ touching the line exactly once.*

Proof. A path from the first group is transformed to a path in the second group by removing any vertical steps that end on the line until only one touch is left, starting with the last touch.

For the inverse we take any path from the second group. Starting from the last point where the path touches or intersects $y = x + s - 1$ and we add into the path one vertical step. We repeat this until the path ends in (a, b). This procedure will always produce a path in the first group since at least one touch is guaranteed, through touching the line $y = x + s$. Moreover, shifting the path by one vertical step results in a new intersection of the path and $y = x + s - 1$. By always choosing the last touch or intersection this procedure reverses the mapping. □

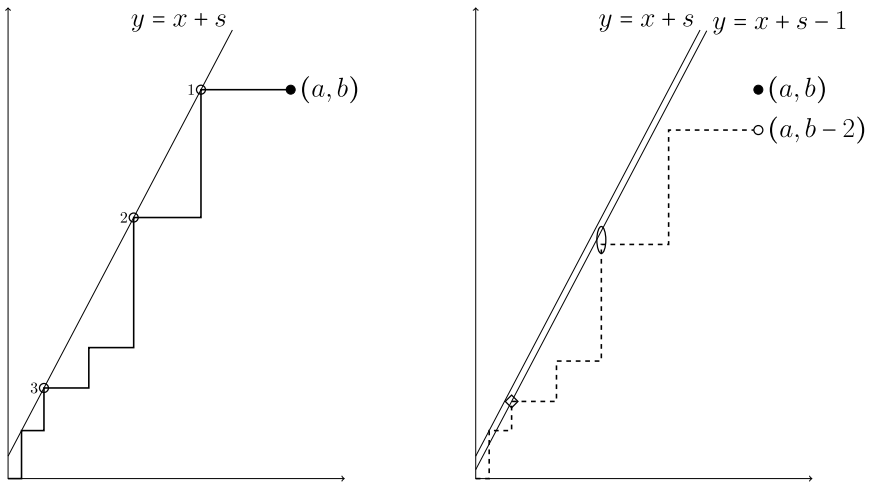
This lemma makes it easy for us to count lattice paths as we have done previously. The equivalent paths with only one touch can be split into two path sections: a path from the origin to a point on the line and a path from that point to an end point. Hence, we can count paths that do not touch any line for the two path sections individually. Furthermore, summing up over all possible midpoints gives us the total number of paths.

Lemma 7. *The number of paths from $(0, 0)$ to (a, b) touching the line $y = x + s$ exactly k times is*

$$\sum_{m=0}^{b-s-k+1} |(0, 0) \to (m, m + s)| \cdot |(m, m + s) \to (a, b - k)|,$$

where $|(u, v) \to (x, y)|$ denotes the number of paths from (u, v) to (x, y).

Proof. By Lemma 6, for any path touching the line k times there is a path touching it once in the first touch point. Hence, if we sum up the paths over every possible point on the line, we get the total number of paths. The first point is at $(0, s)$ and the last point is at $(b - k - s + 1, b - k + 1)$. Hence, there are $b - s - k + 1$ such points. □



(a) A path in the first group which starts at $(0,0)$, ends at (a,b) and touches the line 4 times. The numbered circles indicate the order in which the touches are processed in the bijection.

(b) A path in the second group which starts at $(0,0)$, ends at $(a,b-3)$ and touches the line once. The diamond indicates the last contact with the line $y = x + s - 1$.

Fig. 12. Illustration of the bijection in Lemma 6. Beginning with the last contact (touch 1, Figure a), we remove one vertical step at the intersection of the path and the line. Repeating this for all touch/contact points, with the exception of the first (1, 2 and 3), results in the dashed path (Figure b). For the inverse, we add vertical steps where the dashed path touches the line $y = x + s - 1$. Beginning with the last (diamond in Figure b), adding a step will restore contact with the line $y = x + s$ and add a new touch with the line $y = x + s - 1$ (at the oval). Repeating this process until the path ends in (a,b) again completes the bijection.

What remains is to establish the actual numbers of paths in the sum of this lemma. We cannot rely on previous results for this since we need to count lattice paths between two lines rather than avoiding one. Nevertheless, as before we can rely on standard lattice path results. We denote the number of paths between two lines with D_{OML} (with *OML* referring to omnidirectional movement on a line) and its correctness is given by the following lemma which is Theorem 10.3.4 in *Lattice Path Enumeration* [12].

Definition 10 *(D_{OML}).* Let the number D_{OML} be defined as follows:

$$\begin{aligned} &D_{OML}(a,b,c,d,e,g) \\ &= \sum_{\ell=1}^{\lfloor (e-g+1)/2 \rfloor} \frac{4}{e-g+2} \left(2 \cdot \cos\left(\frac{\pi \ell}{e-g+2}\right)\right)^{c+d-a-b} \cdot \sin\left(\frac{\pi \ell (a-b+e+1)}{e-g+2}\right) \cdot \sin\left(\frac{\pi \ell (c-d+e+1)}{e-g+2}\right). \end{aligned}$$

Lemma 8. *The number of paths from (a,b) to (c,d) neither crossing the line $y = x + e$ nor crossing the line $y = x + g$ is $D_{OML}(a,b,c,d,e,g)$.*

Using D_{OML} we can make the expressions in Lemma 7 explicit. We define the explicit sum as C_{OML}; following the definition, we prove that this is the number of robot's paths of one partition.

Definition 11 *(C_{OML}).* Let the number C_{OML} be defined as follows:

$$\begin{aligned} &C_{OML}(j, j', \hat{j}, i, k) \\ &= \sum_{m=0}^{i-k-|\hat{j}-j'|+1} D_{OML}(0,0,m,m+|\hat{j}-j'|-1,|\hat{j}-j'|-1,1-|j-j'|) \\ &\qquad \cdot D_{OML}(m+1,m+|\hat{j}-j'|,|j-j'|+i-1,i-k, \\ &\qquad\qquad |\hat{j}-j'|-1,1-|j-j'|) \end{aligned}$$

for $k > 0$ and otherwise

$$C_{OML}(j, j', \hat{j}, i, 0)$$
$$= D_{OML}(0, 0, |j - j'| + i - 1, i, |\hat{j} - j'| - 1, 1 - |j - j'|).$$

Lemma 9. *There are $C_{OML}(j, j', \hat{j}, i, k)$ paths for a robot in segment j' walking to segment j which reaches segment $\hat{j}$ exactly k times using i additional steps.*

Proof. The statement follows from replacing the variables in Lemma 8 to fit to our lattice path modelling of the robot's path. It is clear that a and b are zero at the start point, hence we continue with the lines defined by e and g. We substitute e for the distance to the end of the line $\hat{j}$ which gives us $y = x + |\hat{j} - j'| - 1$. Likewise, we substitute g for the distance to segment j which gives us $y = x - |j - j'| + 1$.

Next, we address making contact once with the line representing $\hat{j}$ if k is greater than zero. Since Lemma 8 gives us the number of paths between two lines, we have two points representing the contact. The point $(m, m + |\hat{j}| - j'| - 1)$ is the end point of a path reaching the line and the point $(m + 1, m + |\hat{j}| - j'|)$ is the starting point of a path leaving the line.

Finally, the endpoint, i.e. (c, d), is either $(|j - j'| + i - 1, i - k)$ if k is greater than zero or $(|j - j'| + i - 1, i)$. □

Finally, the proof of the number of steps in a partition allows us to prove the correctness of this setting's probability functions.

Proof of Theorem 4. We focus on the case where the robot is left of the target segment, since the other case follows by symmetry (see Lemma 5).

To sum up over all partitions we need to consider all combinations of additional steps and times the robot reaches the end of the line. The first sum ranges over the possible number of additional step pairs. For one fixed i we then sum up the case where the paths do not reach the end segment, i.e. $k = 0$, and all possible cases where $k \geq 1$.

Regarding the probabilities, all steps to the left are one step from the additional steps pairs, i.e. the power of p is i. However, the right steps are every other step of the additional step pairs and the steps required to reach the target segment j. We only have to subtract the steps at the end of the line from $\hat{j}$ to the adjacent segment as this happens with certainty, i.e. the power of $1 - p$ is $|j - j'| + i - k$.

Finally, by Lemma 9, D_{OML} (see Definition 11) is the number of paths per partition. □

3.2.3. Directional movement on a line

To determine the probability functions for the last setting of a directional robot defending the fence we require not further extension of the modelling. Applying the ideas and approaches of the other three settings allows us to simply state the number of paths and the probability functions.

For the number of paths, denoted with D_{DML} (with *DML* referring to directional movement on a line), we rely on established results from lattice path literature. We need another expression for this setting as we need to count all paths between two lines, given a number of NE-turns. The following lemma, that formally states this, is Theorem 10.14.3 in *Lattice Path Enumeration* [12].

Definition 12 *(D_{DML}).* Let the number D_{DML} be defined as follows:

$$D_{DML}(a, b, c, d, e, g, \ell)$$
$$= \sum_{k \in \mathbb{Z}} \left(\binom{c - a - k(e - g)}{\ell + k} \binom{d - b + k(e - g)}{\ell - k} - \binom{c - b - k(e - g) + g - 1}{\ell + k} \binom{d - a + k(e - g) - g + 1}{\ell - k} \right).$$

Lemma 10. *Let $a + t \geq b \geq a + s$ and $c + t \geq d \geq c + s$. The number of all paths from (a, b) to (c, d) neither crossing the line $y = x + t$ nor crossing the line $y = x + s$ with exactly ℓ NE-turns is given by $D_{DML}(a, b, c, d, t, s, \ell)$.*

Using this and the distribution of spins as established in the proof of Lemma 3, we can define C_{DML} as the number of possible paths of the robot. The proof is omitted since it is similar to the one of Lemma 9.

Definition 13 *(C_{DML}).* Let the number C_{DML} be defined as follows:

$$C_{DML}(j, j', \hat{j}, i, k, \ell)$$
$$= \binom{|j - j'| + 2k + 2m + i - m - 1}{i - m}$$
$$\cdot \sum_{m=0}^{i-k-|\hat{j}-j'|+1} \sum_{r=0}^{\ell} D_{DML}(0, 0, m, m + |\hat{j} - j'| - 1, |\hat{j} - j'| - 1, 1 - |j - j'|, r)$$

$$\cdot D_{DML}(m+1, m+|\hat{j}-j'|, |j-j'|+i-1, i-k, |\hat{j}-j'|-1, 1-|j-j'|, r)$$

if $k > 0$ and otherwise

$$C_{DML}(j, j', \hat{j}, i, k, \ell) = \binom{|j-j'|+2k+2m+i-m-1}{i-m} \cdot \sum_{r=0}^{\ell} D_{DML}(0, 0, |j-j'|+i-1, i, |\hat{j}-j'|-1, 1-|j-j'|, r).$$

Lemma 11. *The number of paths from the initial segment j' to segment j with the respective end segment $\hat{j} \in \{1, d\}$, i and k is $C_{DML}(j, j', \hat{j}, i, k, \ell)$.*

Finally, we state the probability function in the following theorem. The proof is omitted since it follows similarly to Theorem 3 and 4.

Theorem 5. *The probability of a directional robot detecting an adversary in segment j on a line of size d is*

$$Pr(j', j, t, d, p) = \sum_{i=0}^{\left\lfloor \frac{t-\delta-|j-j'|}{2\tau} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{t-\delta-|1-j'|-|1-j|-2}{2} \right\rfloor} \sum_{m=0}^{\min\left\{i, \left\lfloor \frac{t-\delta-|j-j'|}{2\cdot\tau+2} \right\rfloor\right\}} \sum_{\ell=m}^{\left\lfloor \frac{t-\delta-|j-j'|-2i-2m}{2} \right\rfloor} C_{DML}(j, j', \hat{j}, i, k, \ell) \cdot p^{|j-j'|+2k+2m} \cdot (1-p)^{2m+2(i-m)+\delta}$$

where δ is 1 if the robot is facing towards j and 0 if the robot is facing towards an end segment $\hat{j} \in \{1, d\}$. The end segment $\hat{j}$ is 1 if j' is left of segment j and d if segment j' is right of segment j.

4. Calculation of optimal strategies

In the remainder of this work we focus on the calculation of optimal strategies. For this we firstly describe the implementation to calculate an optimal strategy according to the minmax approach. Utilising our explicit probability functions, the remaining part of this is the previous second step of the two-step process (see Section 1). We also briefly discuss the algorithm of Agmon et al. [6] to determine probability functions. This shall facilitate the runtime comparison in Section 6.1.

Algorithm 1 FINDP.

Require: $d, t, j', Pr(j', j, d, t, p)$
$p_{opt} = 0$
$i_{opt} = 1$
for $1 \le i \le d-1$ **do**
  Compute local maxima p_{max} for $Pr(j', i, t, d, p)$ in the range $(0, 1)$
  for $j \in \{1, \ldots, d-1\} \backslash \{i\}$ **do**
    Compute intersection p_j of $Pr(j', i, t, d, p)$ and $Pr(j', j, t, d, p)$ in the range $(0, 1)$
    if $Pr(j', i, t, d, p_j) > Pr(j', i, t, d, p_{max})$ and $Pr(j', i, t, d, p_j) \le Pr(j', k, t, d, p_j)$ for all $k \in \{1, \ldots d-1\}$ **then**
      $p_{opt} = p_j$
      $i_{opt} = i$
    end if
    if $Pr(j', i, t, d, p_{max}) > Pr(j', i, t, d, p_i)$ and $Pr(j', i, t, d, p_j) \le Pr(j', k, t, d, p_j)$ for all $k \in \{1, \ldots d-1\}$ **then**
      $p_{opt} = p_{max}$
      $i_{opt} = i$
    end if
  end for
end for
return $(p_{opt}, Pr(j', i_{opt}, t, d, p_{opt}))$

4.1. Calculating the optimal solution

The probability functions are all we need to calculate an optimal strategy according to the minmax approach. By the approach's definition (see Section 2) an optimal strategy parameter p has to be a point-wise minimum over all segment's probability functions. Such a point-wise minimum over a set of functions is also referred to as the lower envelope. However, instead of checking the entire lower envelope, it is sufficient to consider the local maxima of the probability functions and the intersection of all pairs of functions (see Lemma 6 of Agmon et al. [6]). Hence, to find an optimal value for p we have to determine and iterate over all local maxima and intersections of the probability functions. Agmon et al. [6] devised the algorithm FindP for this purpose. We have adapted the algorithm to our notation and we present it in Algorithm 1.

Finding local maxima and intersections requires solving equations for their roots. We have implemented FindP in Python using Mathematica 12.1 [28] for efficient and accurate root-finding. Finally, we also want to state here some requirements for the calculation of the probabilities, based on the functions presented in Section 3.1 and 3.2. A careful consideration of the functions shows that there are some edge cases at $p = 1$ and $p = 0$ which are not mathematically defined. However, for all edge cases it makes sense to define them to be the limit of the respective functions, which is always well-defined.

4.2. Markov chain approach

The Markov chain approach, devised by Agmon et al. [6], is applied via their FindFunc algorithm. We give a brief overview of the algorithm but refer to their work for details. The first step of the algorithm is to create a transition matrix, based on Markov chain modelling of the respective setting (depending on the environment and the robot's movement type). The terminal states of the Markov chain represent the detection of the adversary in the segments and the t-step transition matrix represent the system's state after t time steps. The algorithm calculates the t-step transition matrix and extracts the probability functions. For every segment the probability function is an entry in the matrix that corresponds to reaching a terminal state from the respective segment.

We have implemented FindFunc in python using the SymPy library [29] to calculate the transition matrix and the probability functions. The calculated probability functions are identical to the probability functions presented in this work.

5. Segment types/reducing the search space

As we have introduced (see Section 1), having explicit probability also allows the possibility to analyse these functions and achieve further improvements. We demonstrate this by showing that for the omnidirectional movement in a circle, some probability functions can be ignored when executing FundP. The reason for this is that some of the functions in Theorem 1 cannot affect the lower envelope. Ultimately, this allows us to reduce the runtime of FindP for shorter penetration times (see Section 6.1).

The functions that can be excluded correspond to all segments, with the exception of two, that the robot can reach only in one direction. The two segments that are not excluded are: the segment that is furthest away in the clockwise direction, which cannot be reached in the anticlockwise direction; and the segment that is furthest away in the anticlockwise direction, which cannot be reached in the clockwise direction. Intuitively, among the segments reachable in only one direction, these two segments are the least likely to be reached by the robot. Consequently, only the probability of these two segments can affect the lower envelope and all other segments closer to segment 1 can be ignored. This can be seen in Fig. 13b where the probabilities of segments 3 and 9 are smaller than the probabilities of segments 2 and 10, respectively.

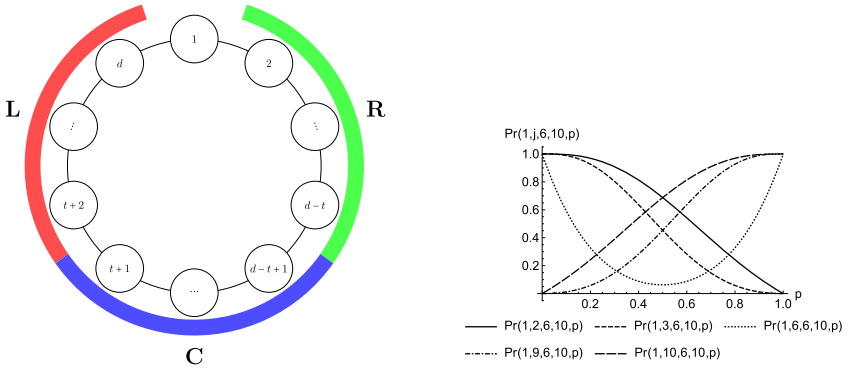
We are interested in the segments where only one of the two sums of $Pr(1, j, t, d, p)$ is non-zero. Dividing probability functions based on whether one or both sums are non-zero results in three types of functions (see Fig. 13b). Depending on p, the graph could be the following: the probability function increases; the probability function decreases; or the probability function decreases initially and then increases again. Segments of the first two types are those that are reachable in only one direction and segments of the third type are those that are reachable in both directions. We denote these three types of segments with R, L and C (see also Fig. 13a).

Definition 14 *(R/L/C Segment).* A segment $j \in [d]$ has one of three types. The type is determined by which of these three cases it satisfies:

R Segments $j \leq d - t$,
L Segments $j \geq t + 2$,
C Segments $d - t + 1 \leq j \leq t + 1$.

Using this definition we can formally state our claim that all but one probability function from each of the L and R segments can be excluded.

Theorem 6. *In each of the two types of L and R segments, the segments adjacent to a C segment have a lower probability than any other segment of the same type. More formally,* $Pr(1, j, t, d, p) < Pr(1, j', t, d, p)$ *for all* $p \in [0, 1]$ *for segments with*



(a) The segment types indicated in a circle. Segments that satisfy the condition for R segments are green and segments that satisfy the condition for L segments are red. The C segments indicated in blue are those where both conditions are true at the same time.

(b) Plot of a selection of probability functions for $d = 10$ and $t = 6$. $P(1, 2, 6)$ and $P(1, 3, 6)$ are examples of R segments, $P(1, 9, 6)$ and $P(1, 10, 6)$ are examples of L segments, and $P(1, 6, 6)$ is an example of a C segment.

Fig. 13. An illustration of the location and the probability functions of the segment types of Definition 14. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

- $j = d - t$ *and* $j' < d - t$ *for* R *segments or*
- $j = t + 2$ *and* $j' > t + 2$ *for* L *segments.*

Prior to proving the theorem, we first prove that the definition describes three distinct groups of segments.

Lemma 12. *The segment types are mutually exclusive.*

Proof. It is easy to see that type C segments are separate from type R and L.

For R and L segments, if they were overlapping that would mean that $t < \frac{d}{2}$. However, this is outside the range of penetration times we consider. Hence, R and L segments are distinct. □

Having established that the types are well defined, we prove our claim.

Proof of Theorem 6. Before we begin, we can focus on one type as the other follows through symmetry (see also Section 3.1.1). Hence, we focus on L type segments and, for sake of contradiction, assume the opposite of the claim is true, i.e. there exists $p \in [0, 1]$ such that

$$Pr(1, t + 2, t, d, p) \geq Pr(1, j, t, d, p) \tag{1}$$

with $j \in \{t + 3, \ldots, d\}$.

We begin with rewriting Equation (1). The left hand side is equivalent to

$$p^{d-j+1} \cdot p^{j-t-2} \cdot \left(\sum_{i=1}^{\left\lfloor \frac{2t-d+1}{2} \right\rfloor} C(d - t + i - 2, i) \cdot p^i \cdot (1 - p)^i + C(d - t - 2, 0) \right)$$

and the right hand side is equivalent to

$$p^{d-j+1} \cdot \sum_{i=1}^{\left\lfloor \frac{t-d+j-1}{2} \right\rfloor} C(d - j + i, i) \cdot p^i \cdot (1 - p)^i + C(d - j, 0).$$

This makes it easy to see that we can express Equation (1) as a bound on the probability of the robot walking the distance between the segments $t + 2$ and j.

$$p^{j-t-2} \geq \frac{\sum_{i=1}^{\left\lfloor \frac{t-d+j-1}{2} \right\rfloor} C(d-j+i,i) \cdot p^i \cdot (1-p)^i + 1}{\sum_{i=1}^{\left\lfloor \frac{2t-d+1}{2} \right\rfloor} C(d-t+i-2,i) \cdot p^i \cdot (1-p)^i + 1} \tag{2}$$

In order to arrive at a contradiction, we show that the right-hand side of Equation (2) is larger than 1. We prove this by considering the difference in magnitude of the numerator and the denominator using two observations. Firstly, the numerator has more terms since $j \geq t+2$ implies $t-d+j-1$ is strictly greater than $2t-d+1$. Secondly, the binomial coefficients given by $C(\cdot,\cdot)$ in the numerator are also bigger.

Whilst the first observation stands for itself, we will prove the second one. First, for brevity, we introduce some notation. We denote the distance between the two segments $t+2$ and j with $k := j-t-2$. Additionally, we define three products which we use to substitute the binomial coefficients: $\alpha := \prod_{\ell=1}^{k}(d-j+2i-k+\ell)$, $\beta := \prod_{\ell=1}^{k}(d-j+i-k+\ell)$ and $\gamma := \prod_{\ell=1}^{k}(d-j+i-k+\ell+1)$.

Using this notation we show that $C(d-j+i,i)$ (in the numerator) is greater than $C(d-t+i-2,i)$ (in the denominator). Firstly we use α, β and γ to rewrite $C(d-j+i,i)$ as

$$\alpha\left(\frac{(d-j+2i-k)!}{i!(d-j+i-k)!}\cdot\frac{1}{\beta} - \frac{(d-j+2i-k)!}{(i-1)!(d-j+i-k+1)!}\cdot\frac{1}{\gamma}\right).$$

In this term, we can bound $\frac{1}{\gamma}$ with $\frac{1}{\beta}$ since β is evidently strictly less than γ. Hence, the whole term is greater than $\frac{\alpha}{\beta}\cdot C(d-t+i-2,i)$. Moreover, α is at least as big as β which means the whole term is at least as big as $C(d-t+i-2,i)$, concluding the second observation.

The two observations together particularly imply that the right-hand side of Equation (2) is bigger than 1. However, the left-hand side of Equation (2) cannot be bigger than 1 which immediately contradicts our assumption. □

Consequentially, it suffices to consider the segments of type C as well as segment $d-t$ and $t+2$. The combinations of these segments together may be significantly less than all d^2 pairs of functions.

Lemma 13. *To find an optimal p, the probability functions of $2t-d+3$ segments have to be considered for a given d and t.*

Proof. By Theorem 6 only segments $d-t$ and $t+2$ out of all L and R segments can affect the lower envelope. Hence, we can exclude all other L and R segments. Specifically, from the L segments we can exclude the $d-t-2$ segments from segment $t+3$ to segment d. Similarly, from the R segments we can exclude the $d-t-1$ segments from segment 2 to segment $d-t-1$. Subtracting both numbers from the total number of segments yields the claim. □

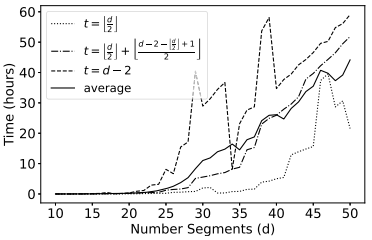
6. Experimental optimal strategies and runtime

We firstly highlight, experimentally, the runtime reduction from using our approach which is substantial in allowing further analysis. We show that, with our results, the runtime is greatly reduced (see Section 6.1). In turn, the expedited calculation allows us to extensively investigate optimal strategies for both movement types in a circle and highlight optimal strategies of an omnidirectional robot on a line (see Section 6.2). Notably, we conclude that an optimal strategy for the directional movement performs vastly better than an optimal strategy for the omnidirectional movement.

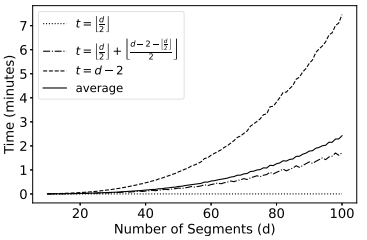
6.1. Experimental runtime

We demonstrate the vast reduction in runtime by setting into context the calculated runtimes of FindFunc (see also Section 4.2) and FindP (see also Section 4.1). Previously, the runtime for finding an optimal strategy would be the runtime of FindFunc plus the runtime of FindP. As previously mentioned, only FindP has to be executed with our approach since we have the explicit probability functions. All runtimes were calculated by running the algorithms on the Iridis 5 Compute Cluster at the University of Southampton. Each execution of the respective algorithms ran on one core of an Intel Xeon E5-2670 processor of a node with 192 GB of DDR4 memory.

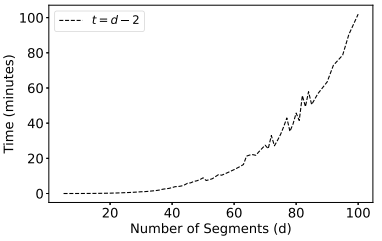
To give an overview of the runtimes which is dependent on the penetration time t, we have selected three different values of t and the average over all t for a fixed d. The three chosen values are the following: the shortest possible time of $\left\lfloor \frac{d}{2} \right\rfloor$, the longest possible time of $d-2$, and finally the mid range time of $\left\lfloor \frac{d}{2} \right\rfloor + \left\lfloor \frac{d-2-\left\lfloor \frac{d}{2} \right\rfloor+1}{2} \right\rfloor$. We present the runtime for these four values for increasing values of d for an omnidirectional robot in a circle.



(a) The runtime (in hours) of FindFunc, i.e. to calculate the probability functions, for a robot with omnidirectional movement.



(b) The runtime (in minutes) of FindP, i.e. to calculate an optimal p, for a robot with omnidirectional movement.



(c) The runtime (in minutes) of FindP, i.e. to calculate an optimal p, for a robot with directional movement.

Fig. 14. The runtime for finding the probability functions and an optimal p in a circle with respect to the number of segments.

While the runtime of FindFunc is not as smooth as that of FindP, it is clear that the runtime of FindP is several times smaller (see Fig. 14). The runtime of FindFunc as depicted in Fig. 14a, despite a number of outliers, increases polynomially and the rate of increase appears to diverge slowly. At $d = 50$ the average runtime reaches 44:09 hours and the runtime for the shortest penetration time is 21:33 hours. In comparison, the runtime of FindP, as shown in Fig. 14b, is in the range of minutes rather than hours. For $d = 100$ the runtime for the shortest penetration time is 24 seconds and for the mid range penetration time 1:24 minutes. The runtime for the longest penetration time is evidently slower, but is still only 7:27 minutes at $d = 100$. Moreover, the runtime for the average penetration time is closer to the runtime of the mid range penetration time and takes only 2:25 minutes at $d = 100$.

Clearly, the calculation for shorter penetration times benefits from the results of Section 5. The runtime for these penetration times is markedly faster in comparison to the runtime for the longest penetration time. Another factor of improvement is that we can calculate the first and second derivative of the probability functions prior to execution. These do not have to be determined during the execution of the algorithm which benefits the runtime of all penetration times. Altogether, calculation of an optimal p can be performed significantly faster as the execution of only FindP attains a shorter runtime in the range of several hours.

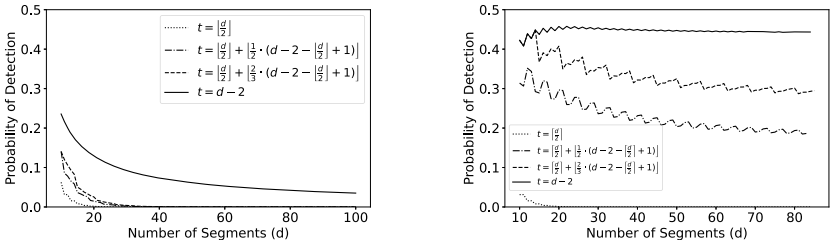
In addition, Fig. 14c shows the runtime for the directional movement in a circle. While the runtime is around 15 times the runtime for the omnidirectional movement for 100 segments, it is still several orders of magnitude smaller than the time it would take to determine the functions.

6.2. Experimental analysis of optimal strategies

Lastly, we utilise the massively reduced runtime shown in the previous section to extensively examine the probability to detect the adversary as well as the underlying optimal values of p.

6.2.1. Omnidirectional movement vs directional movement

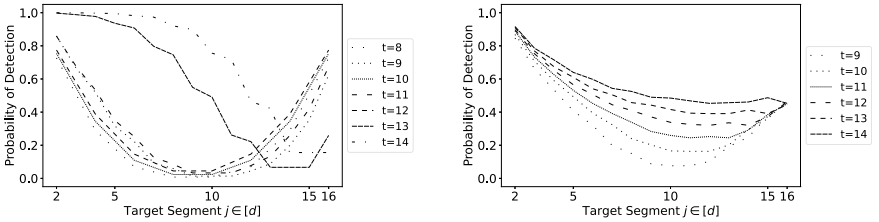
One of our main observations is that in a circle the directional movement is superior to the omnidirectional movement. Fig. 15a and 15b show the probability of detecting the adversary with respect to the number of segments d for a robot with omnidirectional movement and a robot with directional movement, respectively. For both figures, we plot the probability, for an increasing number of segments d, for four different penetration times (t) from the range of all possible penetration times. Those four values are: the shortest penetration time ($\left\lfloor \frac{d}{2} \right\rfloor / \left\lceil \frac{d}{2} \right\rceil$), the longest penetration time ($d-2$), the value at half and the value at two thirds of the range of all penetration times for the respective number of segments. All results in this section assume a turnaround time of one ($\tau = 1$). Longer turnaround times have only a small effect on the results as we discuss below (see Section 6.2.2).



(a) The probability of detecting the adversary by a robot with omnidirectional movement in a circle.

(b) The probability of detecting the adversary by a robot with directional movement in a circle.

Fig. 15. The probability of detecting the adversary in a circle for an optimal p.



(a) The probability for a robot with omnidirectional movement.

(b) The probability for a robot with directional movement.

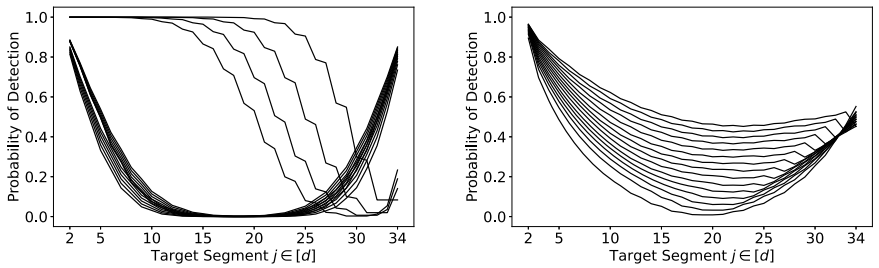
Fig. 16. The probability of detecting the adversary under an optimal p in a circle of size $d = 16$ for increasing t.

Fig. 15a shows that an omnidirectional robot's probability of detecting the adversary decreases quickly. For the two thirds penetration time, it is already below 0.05 for a circle with 16 segments and below 0.01 for a circle of 25 segments. Even the probability for the longest penetration time reaches only 0.235 for a circle with 10 segments and falls below 0.1 for a circle with 28 segments. In contrast, a directional robot's chance to detect the adversary is greater than 0.3 for half the range of penetration times for a circle with 10 segments (see Fig. 15b). Its probability peaks at 0.457 for a circle with 20 segments. Moreover, the detection probability decreases considerably slower. For a circle with 80 segments the probability is still more than 0.18 for half of the penetration times. For the upper third of the penetration times the probability is still more than 0.301. Lastly, at the maximal penetration time the probability is more than 0.444. In comparison, for the omnidirectional movement the probability is already below 0.18 for a circle with 14 segments.

Considering the detection probabilities more broadly, they appear to show a decreased reduction in detection probability. The decrease (measured at the bottom peaks in Fig. 15b) drops from 0.0228 to 0.0015 and from 0.0175 to 0.0014 for two thirds and half of the penetration time ranges, respectively. In addition to this, we can observe a cyclic pattern in Fig. 15b which we assume comes from three factors. Firstly, some cyclic behaviour comes from the fact that most movements have to come in pairs (see also Section 3.1.3). Secondly, in the anticlockwise direction, there is an offset of one as the robot has to turn around first (see Observation 4). Thirdly, the first two factors might be amplified by the way we determine the penetration time at half and two thirds of the range of penetration times.

The stark difference in detection probability can also be seen when considering the detection probability with respect to the penetration time. For a directional robot the probability increases visibly with penetration time, as can be seen in Fig. 16b. In contrast, for an omnidirectional robot the probability increases only slightly for all but the longest penetration time. This trend does not change for circles with more segments. As a further example of this, Fig. 17 shows the probabilities for a 34-segments circle in comparison to the 16-segment circle considered in Fig. 16.

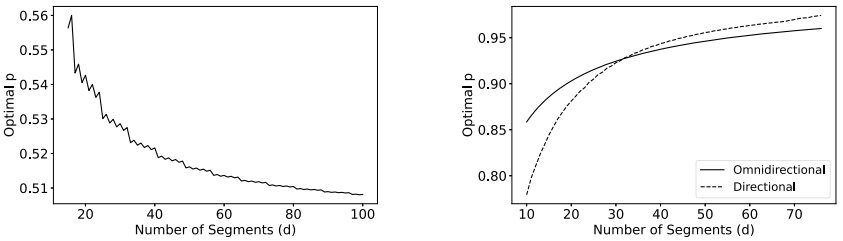
Finally, based on optimal strategies we infer that the directional movement has higher probabilities of detecting the adversary because it tends to walk in one direction. An omnidirectional robot only applies such a strategy for longer penetration times. For these penetration times, two optimal p values diverge towards 0 and 1, respectively. In contrast, for the majority of penetration times the robot has almost equal probability for going clockwise or anticlockwise. Fig. 19a shows this for a circle with 34 segments where for all but the longest four penetration times the value of p tends towards 0.5. However, the detection probability for those is remarkably higher than those tending towards 0.5. We can observe this by comparing the parabolic functions (with a minimum towards the segment furthest away from the robot's initial position) to the functions which generally decrease in Fig. 16a or 17a. Both trends become more pronounced with an increase in the number of segments as Fig. 18a and 18b show. In these figures we focus on the longest penetration time before the strategy switches to a preference of one direction (e.g. time 28 for a circle with 34 segments) and the longest penetration time, respectively.



(a) The probability for a robot with omnidirectional movement.

(b) The probability for a robot with directional movement.

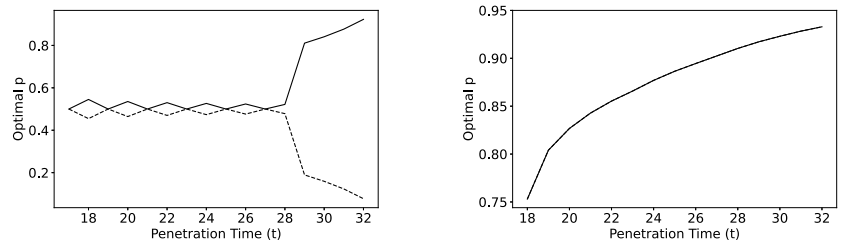
Fig. 17. The probability of detecting the adversary under an optimal p in a circle of size $d = 34$ for increasing t.



(a) For the omnidirectional movement, the optimal p of the longest penetration time before the strategy switches to preferring on direction

(b) The optimal p for the longest penetration time for both the omnidirectional and the directional movement.

Fig. 18. The value of the optimal p for selected penetration times with respect to an increasing number of segments for the directional and omnidirectional movement in a circle.



(a) The optimal p for a robot with omnidirectional movement ($17 \leq t \leq 32$).

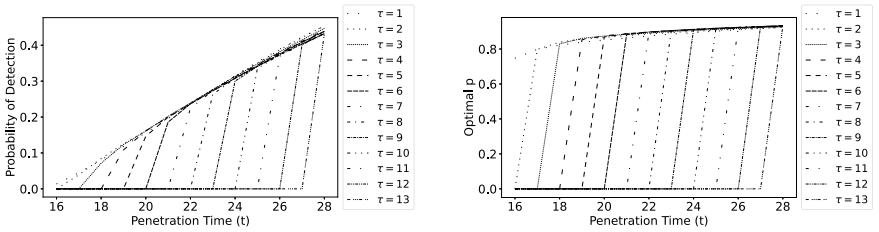
(b) The optimal p for a robot with directional movement ($18 \leq t \leq 32$).

Fig. 19. The optimal p for a circle of size $d = 34$ for all possible penetration times. The solid and the dashed line show one or two optimal p values.

For the directional robot in comparison, the robot has a general preference for walking forwards. The probability of this also increases with longer penetration times (see Fig. 19b) and more segments (see Fig. 18b). Moreover, for circles with 26 or more segments the directional robot's preference for one direction is higher than the preference of the omnidirectional robot (see Fig. 18b). Ultimately, the directional robot turning around makes the previously backwards direction now the forwards direction, thus the evidently preferred direction. This grants a benefit to the directional robot. In comparison, even for the longer penetration times, the omnidirectional robot will always be drawn towards one direction.

6.2.2. Longer turnaround times

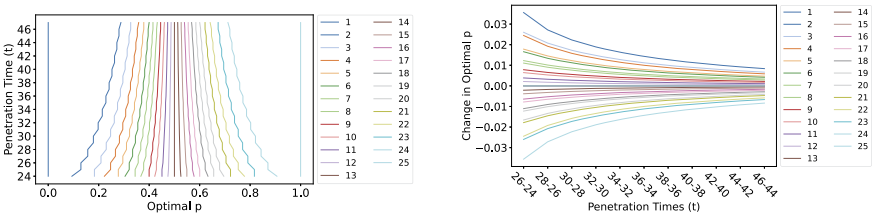
Increasing the turnaround time τ changes the optimal parameter p as well as the probability of detection only slightly (see Fig. 20a). The only pronounced difference is that the range of penetration times where the robot can reach all segments shrinks. We highlight this for a circle with 30 segments. In Fig. 20a we can see how every increase of the turnaround time by 1 reduces the range of penetration times by 1. Whereas, the difference in detection probability between all turnaround times is within 0.02272 for a penetration time of 28. This is the biggest difference in detection probability across all penetration times (not taking into account of penetration times where the probability drops to zero). The same pattern (reduction in the range of penetration times) can be seen in the optimal strategy values (see Fig. 20b) The strategies are close to identical since the optimal strategy parameter over all penetration times is within 0.02403.



(a) The probability of detection for each τ.

(b) The value of the optimal p for each τ.

Fig. 20. The detection probability and the optimal p value for all possible turnaround times (τ) with respect to the range of penetration times, for a robot with directional movement in a circle with 30 segments.



(a) The value of optimal p for all segments and penetration times. The step pattern from the additional moves being in pairs and requiring an increase of penetration time of 2 is clearly visible.

(b) The rate of change of the optimal p for triplets of penetration times that form a step in the optimal p over the penetration times.

Fig. 21. The value and rate of change of the optimal p for the omnidirectional movement on a line with 25 segments. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

6.2.3. Optimal strategies on a line

In addition to the results previously discussed, we briefly show the different patterns of optimal omnidirectional strategies on a line. As discussed in Section 3.2.2 there is a different optimal strategy value for each segment. We highlight this for a line with 25 segments (see Fig. 21). As Fig. 21a shows, the optimal p tends towards 0.5 for all segments. The closer a segment is to the middle of the line the closer it is to 0.5. In case of an odd number of segments, this culminates in a middle segment with an optimal strategy of exactly 0.5.

A further interesting observation is that on a line, the detection probability has a pairing of segments. This is in addition to the cyclic change of the detection probability, based on the fact that additional motions come in pairs. Fig. 21a shows this cyclic change which we already observed in a circle. Smoothing out this effect as in Fig. 21b demonstrates how almost all segments' optimal strategy values change in pairs of a segment with an odd index and a segment with an even index. The only exception are the two end segments where the robot has to turn around.

7. Conclusion

We use lattice path techniques to model and determine the number of possible paths a robot can take on a perimeter or a fence with two different movement patterns. This allows us to explicitly state the probability of detecting an adversary. The probability functions were previously determined using Markov chain based polynomial-time black box algorithms [6]. By omitting the probability function calculation step for every instance, the calculation of optimal detection strategies is reduced by a range of several minutes up to over 50 hours for a circle with 50 segments. The reduction in runtime is aided by showing how analysing the probability functions can be used to further significantly reduce the runtime. We show this for the case of omnidirectional movement on a perimeter. Finally, the runtime reduction allowed us to run extensive experiments showing optimal strategies and penetration detection probabilities thereunder for a large set of instances. A prominent part of this is that a directional robot has a higher probability of detecting the adversary. Such a directional robot has a detection probability of up to 0.457 for a circle with 20 segments and of at least 0.18 for half of the instances for a circle with 80 segments. These detection probabilities are distinctly more than the almost negligible optimal detection probability of an omnidirectional robot.

The techniques and results invite future work by showing how calculations can be simplified and further insights can be gained. The modelling has the potential to simplify the calculation in all similar scenarios [10,9]. Furthermore, it would be interesting to apply the techniques to other structured environments like grid graphs and longer histories. Finally, more specific to the continuation of this work, while high degree polynomials allow no general algebraic solution [30], it would

be interesting to analyse further patterns. For example, using the apparent monotonicity in probability function intersections to further reduce the system of equations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to express their gratitude to Lara E. Buermann for her constructive criticism of the manuscript. The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

[1] J. Buermann, J. Zhang, Multi-robot adversarial patrolling strategies via lattice paths, in: Proc. of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020), IJCAI, Japan, 2020, pp. 4213–4219.
[2] B. An, M. Tambe, A. Sinha, Stackelberg Security Games (SSG) basics and application overview, in: A.E. Abbas, M. Tambe, D. von Winterfeldt (Eds.), Improving Homeland Security Decisions, Cambridge University Press, Cambridge, 2017, pp. 485–507.
[3] G. Oliva, R. Setola, M. Tesei, A Stackelberg game-theoretical approach to maritime counter-piracy, IEEE Syst. J. 13 (1) (2019) 982–993, https://doi.org/10.1109/JSYST.2018.2795892.
[4] N. Basilico, G. De Nittis, N. Gatti, Adversarial patrolling with spatially uncertain alarm signals, Artif. Intell. 246 (2017) 220–257, https://doi.org/10.1016/j.artint.2017.02.007.
[5] A. Rosenfeld, O. Maksimov, S. Kraus, Optimal cruiser-drone traffic enforcement under energy limitation, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, vol. 2018-July, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3848–3855.
[6] N. Agmon, G.A. Kaminka, S. Kraus, Multi-robot adversarial patrolling: facing a full-knowledge opponent, J. Artif. Intell. Res. 42 (2011) 887–916, https://doi.org/10.1613/jair.3365, arXiv:1401.3903.
[7] T. Sak, J. Wainer, S.K. Goldenstein, Probabilistic multiagent patrolling, in: G. Zaverucha, A.L. da Costa (Eds.), Advances in Artificial Intelligence - SBIA 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 124–133.
[8] M. Jain, B. An, M. Tambe, Security games applied to real-world: research contributions and challenges, in: S. Jajodia, A.K. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, X.S. Wang (Eds.), Moving Target Defense II, Springer, New York, NY, USA, 2013, pp. 15–39.
[9] E. Sless Lin, N. Agmon, S. Kraus, Multi-robot adversarial patrolling: handling sequential attacks, Artif. Intell. 274 (2019) 1–25, https://doi.org/10.1016/j.artint.2019.02.004.
[10] E. Sless, N. Agmon, S. Kraus, Multi-robot adversarial patrolling: facing coordinated attacks, in: 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014, vol. 2, 2014, pp. 1093–1100.
[11] N. Talmor, N. Agmon, On the power and limitations of deception in multi-robot adversarial patrolling, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, pp. 430–436.
[12] C. Krattenthaler, Lattice path enumeration, in: M. Bóna (Ed.), Handbook of Enumerative Combinatorics, 1st edition, in: Discrete Mathematics and Its Applications, CRC Press, Boca Raton-London-New York, 2015, pp. 589–678 (Ch. 10).
[13] L. Huang, M. Zhou, K. Hao, E. Hou, A survey of multi-robot regular and adversarial patrolling, IEEE/CAA J. Autom. Sin. 6 (4) (2019) 894–903, https://doi.org/10.1109/JAS.2019.1911537.
[14] Y. Elmaliach, N. Agmon, G.A. Kaminka, Multi-robot area patrol under frequency constraints, Ann. Math. Artif. Intell. 57 (3) (2009) 293–320, https://doi.org/10.1007/s10472-010-9193-y.
[15] N. Basilico, N. Gatti, F. Amigoni, Patrolling security games: definition and algorithms for solving large instances with single patroller and single intruder, Artif. Intell. 184–185 (2012) 78–123, https://doi.org/10.1016/j.artint.2012.03.003.
[16] N. Agmon, V. Sadov, G.A. Kaminka, S. Kraus, The impact of adversarial knowledge on adversarial planning in perimeter patrol, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 55–62.
[17] V. Sea, A. Sugiyama, T. Sugawara, Frequency-based multi-agent patrolling model and its area partitioning solution method for balanced workload, in: Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer International Publishing, Cham, 2018, pp. 530–545.
[18] Y. Chevaleyre, Theoretical analysis of the multi-agent patrolling problem, in: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004, 2004, pp. 302–308.
[19] Y. Elmaliach, A. Shiloni, G.A. Kaminka, A realistic model of frequency-based multi-robot polyline patrolling, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, vol. 1, 2008, pp. 63–70.
[20] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, F. Amigoni, Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings, in: Proceedings - 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2009, vol. 2, 2009, pp. 557–564.
[21] J.B. Clempner, A continuous-time Markov Stackelberg security game approach for reasoning about real patrol strategies, Int. J. Control 91 (11) (2018) 2494–2510, https://doi.org/10.1080/00207179.2017.1371853.
[22] S. Alpern, A. Morton, K. Papadaki, Patrolling games, Oper. Res. 59 (5) (2011) 1246–1257, https://doi.org/10.1287/opre.1110.0983.
[23] S. Alpern, T. Lidbetter, K. Papadaki, Optimizing periodic patrols against short attacks on the line and other networks, Eur. J. Oper. Res. 273 (3) (2019) 1065–1073, https://doi.org/10.1016/j.ejor.2018.08.050.
[24] The OEIS Foundation Inc., On-Line Encyclopedia of Integer Sequences - Catalan's triangle, 2020.
[25] I.P. Goulden, L.G. Serrano, Maintaining the spirit of the reflection principle when the boundary has arbitrary integer slope, J. Comb. Theory, Ser. A 104 (2) (2003) 317–326, https://doi.org/10.1016/j.jcta.2003.09.004.
[26] R.P. Stanley, Enumerative Combinatorics, vol. 1, 2nd edition, Cambridge University Press, New York, NY, USA, 2011.
[27] M.Z. Spivey, Enumerating lattice paths touching or crossing the diagonal at a given number of lattice points, Electron. J. Comb. 19 (3) (2012) 1–6.
[28] Wolfram Research, Wolfram Mathematica, 2020.
[29] SymPy Development Team, SymPy, 2021.
[30] F. Neri, Linear Algebra for Computational Sciences and Engineering, Springer International Publishing, Cham, 2016.