Stochastic Patrolling in Adversarial Settings

Ahmad Bilal Asghar Stephen L. Smith

***Abstract*— In this paper, we consider a patrolling problem in adversarial environments where intruders use the information about a patrolling path to increase chances of successful attacks. We use Markov chains to design randomized patrolling paths on graphs. We present four different intruder models, each of which use information about the patrolling path in a different manner. We characterize the expected rewards for those intruder models as a function of the Markov chain that is being used for patrolling. We show that minimizing the reward functions is a non-convex optimization problem. We propose a pattern search based algorithm to determine a locally optimal patrolling strategy. We also show that for a certain type of intruder, a deterministic patrolling policy given by an orienteering tour of the graph is the optimal patrolling strategy.**

I. Introduction

Consider a marketplace where a security agent is patrolling at night to prevent thefts. The thief may have more incentive to rob a jewelry shop than a grocery shop. If the patrolling path of the agent is predictable, a potential thief can study the path, and time the theft to avoid being detected. As a simple example, assume that the patrolling agent takes a round of the market and visits each shop once every 45 minutes. If the thief requires half an hour to steal, he can commit the crime without being detected. Moreover, if the patrolling agent visits the grocery shop as often as the jewelry shop, the thief would make an attempt at the more valuable location. So, in such a case, randomizing the patrolling path can increase the surveillance quality of the area. Another relevant example is in border patrol [1], [2] where an intruder can time its crossing of the border in order to infiltrate unnoticed. We study the effects of randomizing the patrolling path in the presence of such intruders and attempt to design the paths that optimize the surveillance quality for a given environment.

Related work: There is a substantial amount of work related to design of patrolling policies for surveillance of environments. For deterministic paths, there is a breadth of work considering both single and multiple robots [3], [4], [5], [6]. In [7] the authors discuss deterministic monitoring strategies to minimize the weighted latency of the path. The path visits the vertices with higher weight more often. Our work looks at a similar problem, but considers intelligent intruders and thus we consider non-deterministic paths.

Srivastava *et al.* [8] advocate random patrolling paths by showing that it is hard to find deterministic strategies that satisfy the surveillance criterion of visiting vertices proportional to their importance. In [9] the authors consider stochastic surveillance paths so that the intruders cannot easily exploit the predictability of a deterministic path. We extend this notion and look at ways in which an intelligent intruder can learn a randomized path, and utilize this information to launch attacks. In [1] the authors look at intelligent intruders in a perimeter patrolling problem. Our problem is different in the sense that we deal with environments represented as graphs, for which border patrol is a special case.

Markov chains are often used to model random paths on a graph [8], [9], [10]. For example, in [10] the authors use semidefinite programming to minimize the mean first passage time from one state of the Markov chain to any other state, called Kemeny constant [11]. They also use the Markov chains on weighted graphs, where the probability of traversing an edge is independent of the weight of that edge. This formulation is more useful in the case of surveillance since the environment is usually represented as a weighted graph and the Markov chain is then defined on that graph. This *doubly weighted graph* formulation is not studied much in the literature to the best of our knowledge. We study the problem for both weighted and unweighted graphs.

Boyd *et al.* [12] design a Markov chain with fastest mixing time, which approaches the steady state distribution as quickly as possible. In [13] the authors use a finite set of Markov chains to formulate the problem as a *Bayesian Stackelberg Game*.

In [14] the authors empirically show that randomized paths can decrease the probability of successful intrusions. Instead of using Markov chains, the propose methods for randomization such as randomly interchanging the order of vertices in each cycle of a TSP tour. They consider three different types of intruders based on the attacking strategy. We also present different attack models for the intruders and analyze their performance. The approach of presenting intruder models for attacks on the environment is similar to that used in computer security (for example, [15] on encryption or [16] on wireless sensor networks).

Contributions: We formally define four intruder models. Each intruder model requires different capabilities and knowledge from the intruder. The expected reward for each of these intruders is derived that serves as the objective function to find a patrolling policy. The expected reward of the intruder is a non-convex and non-smooth function of the patrolling policy, and we propose a pattern search based algorithm to find the local minimum of the expected reward and present experimental results. We also show that for a certain intelligent intruder, a deterministic patrolling policy is optimal. Proofs are omitted due to space limitations and will appear in full elsewhere.

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (abasghar@uwaterloo.ca; stephen.smith@uwaterloo.ca)

A. Background

We use homogeneous Markov chains to model random paths on graphs. The transition matrix P of a Markov chain gives the transition probabilities between states of the Markov chain. We say that state i *communicates* with state j, if $p_{ij}^{(m)} > 0$ for some $m > 0$. The states i and j *intercommunicate* with each other if i communicates with j and j communicates with i. A set of states C is called a *communicating class* if states i and j intercommunicate with each other for $i, j \in C$ and no state in C communicates with any state not in C. We will only be considering the Markov chains with a single communicating class in this paper.

The ORIENTEERING PROBLEM [17] is a variant of the TRAVELING SALESMAN PROBLEM. The input to the problem is a weighted graph $G = (V, E, W)$ and a time limit T_{max}. Each vertex $i \in V$ has an associated score ψ_i. The problem is to find a closed walk of length not exceeding T_{max} which maximizes the total score obtained by visiting the vertices.

II. PROBLEM STATEMENT

Consider a surveillance agent monitoring an environment to detect possible attacks on the areas of the environment. We represent the environment by an undirected weighted graph $G = (V, E, W)$ where the set of vertices $V = \{1, 2, \ldots, n\}$ represents the areas to be monitored and the edges represent the routes along which the agent can travel. The weight on the edge $\{i, j\}$ is given by w_{ij}. Each vertex i has an importance ϕ_i representing the intruder's reward for launching a successful attack on i. The length of attack is $l > 0$, and an attack is *successful* if the surveillance agent does not visit the attacked vertex for the duration of the attack. We use first order Markov chains to define random patrolling paths on the graphs. The transition matrix of the Markov chain $P = [p_{ij}]$ governs the random path of the agent. The probability p_{ij} of going from vertex i to vertex j is independent of the history of the path. We call the transition matrix P the *patrolling policy* since it completely characterizes the agent's patrolling path.

The probability of successful attack depends on the method of attack used by the intruder. The *intruder model* defines the procedure by which the intruder decides the time and vertex of attack. The model may or may not use information the intruder has gathered about the agent's patrolling path. Given an intruder model, our goal is to find a patrolling policy to minimize the intruder's expected reward.

> **Problem Statement:** Given a graph $G = (V, E, W)$ and the intruder model, determine a patrolling policy P that minimizes the expected reward of the intruder.

In the following section we define four intruder models and we derive expressions for the probability of successful attacks on the vertices of the graph.

III. INTRUDER MODELS

This section discusses some of the intuitive policies that an intruder can follow to launch attacks on the environment. We assume that there is only one attack on the environment at a time. Moreover, the intruder does not travel on the graph and appears only at the location of the attack for the duration of the attack. An intruder model will define the following two parameters:

- *Vertex of attack*: The intruder chooses the vertex of attack according to some probability distribution defined in the model. The probability of launching an attack on vertex i denoted as $\mathbb{P}[\texttt{attack} = i]$, where $\sum_{i \in V} \mathbb{P}[\texttt{attack} = i] = 1$
- *Time of attack*: This defines the vertex occupied by the patrolling agent when the attack is launched. The intruder may wait until the agent reaches a certain vertex before launching an attack. In general, this will be a function of the vertex of attack.

To find the probability of an attack being successful, we need to define the notions of *first visit time* and the *success matrix*, which are defined in the following sections.

A. Intruder Model Preliminaries: First Visit Time

The Markov chain transition matrix P can be used to determine the probability of launching a successful attack on vertex j when the patrolling agent is at vertex i. For a given Markov chain, we can calculate the probability of going from state i to state j in exactly k steps. In [18] the probability of first visit to state $j \in V$ from starting state $i \in V$ in k transitions is given for the Markov chains where each transition takes same amount of time. Hence, that result is valid for Markov chains on unweighted graphs. We extend these results for graphs with edge weights where each transition can take a different amount of time. Unweighted graphs can model the environments in the form of grid graphs, whereas weighted graphs capture the more general problem since the travel times between different nodes of the environment can be different.

We assume that the weights on the edges of the graphs are positive integers. Given a graph $G = (V, E, W)$ and the transition matrix P, the probability mass function of the time taken for the first visit from state i to state j is given by

$$F_k(i,j) = \begin{cases} p_{ij}(w_{ij} = 1), & k = 1 \\ \sum_{h \neq j} P_{ih} F_{k-w_{ih}}(h,j) + p_{ij}(w_{ij} = k), & k \geq 2 \end{cases} \quad (1)$$

where $(w_{ij} = 1)$ is a truth statement which evaluates to 1 or 0, and $F_k(i,j) = 0$ for non positive values of k.

Given i and j, $F_k(i,j)$ gives a probability mass function over the random variable K which is the amount of time/steps taken in hitting the state j for the first time starting from i. In the subsequent sections, we will refer to the three dimensional matrix F as the *first visit matrix*.

B. Intruder Model Preliminaries: Success Matrix

We will now use matrix F to calculate the probability of successful attacks by the intruder. An attack on a particular location is successful if the patrolling agent does not visit that location for the entire duration of the attack. So, for a

specified length of attack l, the probability of attack being successful on j given the patrolling agent's position is i at the time of launching the attack can be simply calculated as

$$\mathbb{P}[\texttt{success at } j|i] = 1 - \sum_{k=1}^{l} F_k(i,j)$$

which is the probability of not visiting the state j in l consecutive steps from the state i. For notational convenience, we define the *success matrix* S, whose element s_{ij} gives the probability of a successful attack at vertex j of the graph if the agent is at vertex i at the time the attack is launched. We can write S as

$$S = J - \sum_{k=1}^{l} F_k \tag{2}$$

where J is a matrix of all ones. The matrix S may or may not be used by the intruders to decide on the location and the time of the attack. The following sections present some intruder models.

C. Intruder Model 1: Naïve Attacks

The naïve intruder model represents an intruder that does not learn or use any information about the patrolling path of the agent. The time and vertex of attack for this model are decided as follows.

Vertex of attack: Whenever an attack is to be launched, the vertex of attack is chosen depending on the importance of that vertex. So, the probability of attacking vertex j is

$$\mathbb{P}[\texttt{attack} = j] = \frac{\phi_j}{\sum_i \phi_i}$$

Time of attack: In this model, the intruder launches the attack randomly and independent of the agent's location. The location of the agent at the time of attack is a random variable, and the probability that the agent is at location i when the attack arrives is given by the frequency with which agent visits location i, which is π_i where π is the stationary distribution of the Markov chain being followed by the agent. Using the expression $\mathbb{P}[\texttt{agent at } i]$ for the probability of the agent being at i when the attack is launched, we can write

$$\mathbb{P}[\texttt{agent at } i] = \pi_i.$$

Objective function: Using the time and vertex of attack, the reward function for this intruder model can be written as

$$\begin{aligned} f(P) &= \sum_{j=1}^{n} \phi_j \mathbb{P}[\texttt{success at } j]\, \mathbb{P}[\texttt{attack} = j] \\ &= \sum_{j=1}^{n} \phi_j \mathbb{P}[\texttt{success at } j]\, \phi_j \end{aligned}$$

assuming ϕ is normalized. Then using Bayes theorem,

$$\begin{aligned} f(P) &= \sum_{j=1}^{n} \phi_j^2 \sum_{i=1}^{n} \mathbb{P}[\texttt{success at } j|i]\, \mathbb{P}[\texttt{agent at } i] \\ &= \sum_{j=1}^{n} \phi_j^2 \sum_{i=1}^{n} s_{ij}\pi_i = \pi^\top S \phi^2 \end{aligned}$$

where π and ϕ are vectors. The reward function is a function of the transition matrix P, since the matrix S and the vector π are functions of P.

D. Intruder Model 2: Deterministic Intruder

The deterministic intruder model is motivated by the intruder that attacks a vertex immediately after the patrolling agent departs that vertex. This is an effective strategy if the patrolling agent is following a cyclic tour of the graph.

Vertex of attack: The vertex of attack is chosen just like the naïve policy, so

$$\mathbb{P}[\texttt{attack} = j] = \frac{\phi_j}{\sum_i \phi_i}.$$

Time of attack: The intruder waits and launches the attack on vertex i as soon as the patrolling agent leaves vertex i on its path. Hence,

$$\mathbb{P}[\texttt{agent at } i] = \begin{cases} 1 & \text{attack is on vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Objective function: The reward function for such an intruder will be

$$f(P) = \sum_{i=1}^{n} \phi_i^2 s_{ii}.$$

If the patrolling path of the agent is a TSP tour and the length of the tour is greater than the length of attack, then this attack policy is optimal in terms of choosing the time of attack.

E. Intruder Model 3: Intelligent Intruder with Assigned Locations

Let us consider an intruder that has observed the patrolling path of the agent and wants to use the learned information to increase the chances of a successful attack. The models presented in this and the next section address such intruders. We will assume that the intruder has observed the surveillance path long enough to have modeled the transition matrix P of the patrolling Markov chain.

Vertex of attack: According to this model, the vertex to attack is selected based on its importance like the previous two models. So, the intruder does not use the patrolling path information to choose the vertex of attack. Hence,

$$\mathbb{P}[\texttt{attack} = j] = \frac{\phi_j}{\sum_i \phi_i}.$$

This can be realized in a scenario where the intruder has been pre-assigned the locations that it has to attack, and the intruder then uses the information about the patrolling path to choose the time of attack for those locations.

Time of attack: Since the intruder has access to the transition matrix P, it can calculate the success matrix S (2) and use it to figure out the best possible time to attack. The intruder maximizes its chances of a successful attack at vertex j, so the location of the agent at the time of attack is given by

$$\mathbb{P}[\texttt{agent at } i] = \begin{cases} 1, & i = \arg\max_x s_{xj}. \\ 0, & \text{otherwise} \end{cases}$$

Hence, the intruder looks at the column of S corresponding to the vertex it has to attack and the index of the maximum element in that column gives the vertex at which the agent should be when that attack is launched. The intruder waits until the agent reaches vertex i to launch the attack at j.

Objective function: The expected reward for the intruder following this model will be

$$f(P) = \sum_{j=1}^{n} \phi_j^2 \max_i\{s_{ij}\} = \max(S)\phi^2$$

where $\max(S)$ is a row vector with j^{th} element being the maximum entry in the j^{th} column of matrix S.

F. Intruder Model 4: Intelligent Intruder

Unlike the previous model, the intruder has not been assigned the locations to attack, hence this model uses P to choose the vertex of attack as well.

Vertex of attack: The maximum expected reward the intruder can obtain is by attacking the best possible vertex at the best possible time. The vertex with the maximum expected reward is given by

$$j = \arg\max_y \left\{\phi_y \max_i\{s_{iy}\}\right\}.$$

Here $\max_i\{s_{iy}\}$ gives the highest probability of attack on vertex y. So, $\phi_y \max_i\{s_{iy}\}$ gives the expected reward for attacking the vertex y and the vertex to attack j is chosen as the vertex that gives the most expected reward.

Time of attack: The best time to attack vertex j is given by the index of the maximum entry in the j^{th} column of S. The position of agent at the time of attack is then given by

$$i = \arg\max_x s_{xj}.$$

Objective function: So, the maximum expected reward for the intruder is

$$f(P) = \max_j \left\{\phi_j \max_i\{s_{ij}\}\right\} = \max_{i,j}\{\phi_j s_{ij}\}$$

Among the intruder models that decide the time and vertex of attack based on the patrolling policy P only, this model is optimal in the sense that it maximizes the intruder's reward. We now look at computing transition matrices that minimize the intruder's expected reward.

IV. Patrolling Policy

Given one of the intruder models, our task is to find the patrolling policy P that minimizes the expected reward $f(P)$ for that intruder. In this section, $f(P)$ will generally refer to the reward function independent of the intruder model unless a model is specified.

A. Non-Convexity of the Problem

The reward function $f(P)$ for all of the intruders is not a convex function in general. The Hessian of s_{ij} is not necessarily positive semi definite and hence s_{ij} is not a convex function of P making $f(P)$ non convex. This can be observed by considering the Hessian of s_{11} for length of attack $l = 2$ as an example.

Hence we cannot use convex programming to find the global optimal patrolling policy P. The following result shows that the optimal patrolling policy for the intruder model 3 is the solution to the orienteering tour problem of the graph.

B. Deterministic Patrolling Policy for Intelligent Intruder with Assigned Locations

The motivation behind randomizing the patrolling path is that a deterministic could be exploited by the intruder. However, if the intruder is following intruder model 3, a deterministic path minimizes the expected reward. Informally, that is because the intruder has to attack all the vertices according to their importance, and if the patrolling agent just visits the 'most important' vertices within time l, the intruder can never successfully attack those vertices. The following proposition formally presents the result.

Proposition IV.1 (Deterministic Patrolling Policy for Intruder Model 3)**.** *The solution to the* ORIENTEERING PROBLEM *for the graph $G = (V, E, W)$ with score on vertex $i \in V$ given by $\psi_i = \phi_i^2$, and the time limit $T_{max} = l$ minimizes the expected reward for the intruder following intruder model* 3 *(intelligent intruder with assigned locations).*

The proof of the proposition is omitted due to space constraints, but the idea behind the proof is to consider a vertex r outside the orienteering tour and show that any patrolling policy that visits r cannot decrease the reward function.

Proposition IV.1 shows that a deterministic strategy minimizes the expected reward of intruder model 3. In general, finding the optimal policy P for patrolling given an intruder model is a non convex optimization problem. Gradient descent methods can be used to find a locally optimal patrolling policy but the runtime of calculating the gradient of the objective function is $O(n^5 l)$ where n is the number of vertices in the graph and l is length of attack. In the following section we propose a gradient-free method based on pattern search for computing locally optimal patrolling policies.

C. Locally Optimal Markov Chains

Consider the space of stochastic matrices $P \in \mathbb{R}^{n\times n}$ with $p_{ij} \geq 0$ for all $i, j \in \{1, \ldots, n\}$ and $\sum_{i=1}^{n} p_{ij} = 1$ for all $j \in \{1, \ldots, n\}$. Given an objective $f(P)$, and an initial point P, or goal is to move in a descent direction from P. To do this, we propose Algorithm 1, which iteratively searches a set of candidate descent directions using pattern search [19]. At each iteration of the method, a set of search directions are chosen and the function is evaluated at a given step length along these direction. As soon as a better point is found, it is

chosen as the current point and the method proceeds to the next iteration. Let $\mathcal{D}_k$ denote the set of possible directions to chose from at the k^{th} iteration and let the step length at iteration k is given by γ_k.

Algorithm 1: LOACALLYOPTIMALCHAIN

Input: Graph $G = (V, E, W)$; intruder's reward function $f(P)$; algorithm parameters γ_T, δ and θ
Output: A locally optimal transition matrix

Pick a transition matrix P for the graph G and initial step length γ_o
for $k = 0, 1, 2, \ldots$ **do**
  if $\gamma_k < \gamma_T$ **then** Stop
  if $f(P_k + \gamma_k d_k) < f(P_k) - \rho(\gamma_k)$ *for some* $d_k \in \mathcal{D}_k$ **then**
    $P_{k+1} \leftarrow P_k + \gamma_k d_k$
    $\gamma_{k+1} \leftarrow \delta\gamma_k$
  else
    $P_{k+1} \leftarrow P_k$
    $\gamma_{k+1} \leftarrow \theta\gamma_k$

In the algorithm, $\delta > 1$ is used to increase step length for the next iteration if a decrease direction is found. The parameter $\theta < 1$ decreases the step length if no decrease direction is found in the said iteration. If the step length becomes less than a set value γ_T, the algorithm terminates. The sufficient decrease function $\rho(\gamma_k)$ is chosen as $M\gamma_k^{3/2}$ where M is a constant, which, from [19], guarantees convergence.

Search Directions for Transition Matrices: We require a set of directions that has the property that if the gradient of the function is not zero, then at least one of the directions is a direction of descent. So, a proposed set is the set of all unit directions. In our case, the optimization is constrained to the set of stochastic matrices, and thus some of the unit directions may not be valid. To address this, we define the set $\mathcal{D}_k$ as follows. For each row i of the transition matrix, we have a simplex in $\mathbb{R}^{q_i}$ dimensions where q_i is the number of neighbors of vertex i. We can define $q_i - 1$ mutually perpendicular directions on the simplex for each i and use these as the possible directions. They are guaranteed to include a descent direction if one exists. In our implementation, we also include some random directions in $\mathcal{D}_k$ to speed the descent process.

Properties of Algorithm 1: With the choice of search directions given above, Algorithm 1 can be applied to locally optimize a Markov Chain for each of the four intruder models. The algorithm has the following properties:

(i) The cost $f(P)$ monotonically decreases at each iteration,
(ii) The algorithm terminates in finite time, and
(iii) The worst case run-time per iteration is $O(n^5 l)$ since there are $O(n^2)$ search directions and the runtime for calculating the function is $O(n^3 l)$ for a general graph.

The algorithm moves to the next iteration as soon as a better point is found and hence each iteration is less expensive than that of gradient descent. On the other hand, while more computationally expensive per iteration, the gradient descent method may require fewer iterations to converge. We used both the gradient descent and pattern search based method to compute patrolling policies on a variety of graphs and found the later to be much faster in practice.

V. SIMULATION RESULTS

We performed the optimization of the expected reward for a given intruder using Algorithm 1 on some instances of the problem. This section presents and examines the results of those experiments.

A. Comparison with other policies

In our first experiment, we ran Algorithm 1 on small instances of the graph for different starting points and compared the results with some other patrolling policies. The intruder model chosen was the 'intelligent intruder' given in Section III-F. The sample instances of the graph were created randomly following a procedure similar to [14]. The graph was constructed using $n = 10$ Euclidean points in a plane of size $\mathrm{MAX} = 10$. Each point was at least $\mathrm{MIN} = 3$ units of distance apart from the others. Each vertex was connected to k of its nearest points where k was chosen randomly between $\mathrm{NBR}_{\mathtt{min}} = 3$ and $\mathrm{NBR}_{\mathtt{max}} = 5$. By a probability of $P_{\mathtt{conn}} = 0.1$ each point was connected to some other random point as well. The Euclidean distance between the connected points was calculated and then rounded off to give integer weights on the edges. The wights on the vertices of the graph were randomly picked to be between 1 and 3. The length of the attack was taken to be the length of the minimum spanning tree of the graph.

The experiment was performed for 40 different graphs, and for each of the graph instance, we compared the performance of the policy (transition matrix) found by the Algorithm 1 to the following two policies.

(i) Minimum Kemeny Constant Policy: The policy given in [10] of finding the transition matrix with minimum Kemeny constant.
(ii) Uniform Policy: The transition matrix P is chosen such that the entry p_{ij} in each row i of P is $1/q_i$ where q_i is the number of neighbors of i^{th} vertex.

Algorithm 1 found a better policy for all of the 40 instances since it particularly minimizes the objective function. The comparison for 5 of those instances is given in Table V-A. The expected reward is normalized such that it is 1 for the locally optimal policy.

B. Patrolling an indoor environment using unweighted graph

For the next experiment we consider the intruder model presented in Section III-E. Here the intruder cannot choose the vertex to attack but can choose the time to launch the attack. We consider an unweighted grid graph which can be used for patrolling in indoor environments. The graph has $n = 52$ vertices and is shown in Figure 1. The weights on the vertices of the graph are 1 for all the vertices.

Kemeny Constant	Uniform	Algorithm 1
1.42	1.56	1
1.21	1.43	1
1.36	1.41	1
1.32	1.34	1
1.56	1.86	1

TABLE I

THE EXPECTED REWARD OF THE INTRUDER FOR DIFFERENT POLICIES.

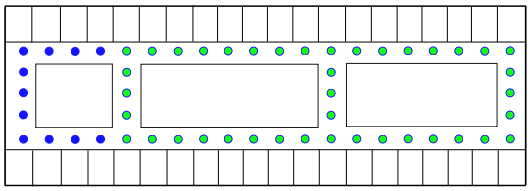

Fig. 1. The grid graph representing an indoor environment. The green vertices represent the orienteering tour of length 44.

In the Figure 2, we have plotted the performance of pattern search on the grid graph for length of attack $l = 44$. The starting point for the Algorithm 1 was a random Markov chain, and it can be observed that the expected reward for the intruder was almost 1 for that chain. It means that the intruder was always able to find a suitable time to attack for any of the vertices of the graph. Algorithm 1 decreased the expected reward of the intruder to 0.278. Notice that since we are considering the intruder model 3, the orienteering tour of the graph with time limit 44 is the optimal solution. The green vertices of the graph in Figure 1 represent the vertices in the orienteering tour. If the agent follows that deterministic path, the intruder can attack the blue vertices with 100% success rate and it can never successfully attack the green vertices. So the expected reward for the intruder will be $11/52 = 0.2115$.

VI. Conclusions and Future Work

In this paper we discussed the patrolling problem in adversarial environments and proposed patrolling policies for different intruder models. We showed that for a certain type of intruder, the orienteering tour of the graph is the optimal patrolling policy. We proposed a pattern search based algorithm to find the locally optimal policies and presented the experimental results.

We assumed that the intruder has perfectly modeled the Markov chain being followed by the agent. The learning of Markov chain can be incorporated in the models to make them more realistic. Moreover, designing patrolling paths for multiple patrolling agents is also an interesting problem for future work.

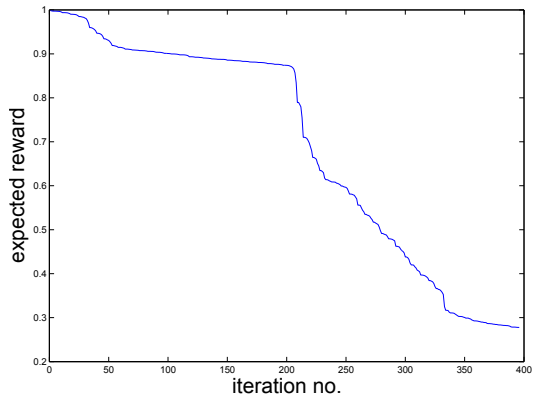

Fig. 2. The performance of Algorithm 1 on grid graph with $l = 44$.

References

[1] N. Agmon, S. Kraus, and G. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 2339–2345.

[2] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, "Cooperative perimeter surveillance with a team of mobile robots under communication constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5067–5072.

[3] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[4] D. Portugal and R. P. Rocha, "Multi-robot patrolling algorithms: examining performance and scalability," *Advanced Robotics*, vol. 27, no. 5, pp. 325–336, 2013.

[5] J. Yu, S. Karaman, and D. Rus, "Persistent monitoring of events with stochastic arrivals at multiple stations," in *IEEE International Conference on Robotics and Automation*, Hong Kong, May 2014, pp. 5758–5765.

[6] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Beijing, China, Sep. 2004, pp. 302–308.

[7] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014.

[8] K. Srivastava, D. Stipanovic, and M. Spong, "On a stochastic robotic surveillance problem," in *IEEE Conference on Decision and Control and Chinese Control Conference*, Dec 2009, pp. 8567–8574.

[9] J. Grace and J. Baillieul, "Stochastic strategies for autonomous robotic surveillance," in *IEEE Conference on Decision and Control and European Control Conference*, Dec 2005, pp. 2200–2205.

[10] R. Patel, P. Agharkar, and F. Bullo, "Robotic surveillance and markov chains with minimal weighted kemeny constant," *IEEE Transactions on Automatic Control*, vol. 60, no. 12, pp. 3156–3167, Dec 2015.

[11] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. Springer, 1976.

[12] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[13] T. Alam, M. Edwards, L. Bobadilla, and D. Shell, "Distributed multi-robot area patrolling in adversarial environments," in *International Workshop on Robotic Sensor Networks*, Seattle, WA, USA, Apr. 2015.

[14] T. Sak, J. Wainer, and S. K. Goldenstein, "Probabilistic multiagent patrolling," in *19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, ser. SBIA '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 124–133.

[15] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[16] Z. Benenson, E. Blaß, and F. C. Freiling, "Attacker models for wireless sensor networks (angreifermodelle für drahtlose sensornetze)," *it - Information Technology*, vol. 52, no. 6, pp. 320–324, 2010.

[17] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[18] E. Cinlar, *Introduction to Stochastic Processes*, ser. Dover Books on Mathematics Series. Dover Publications, Incorporated, 2013.

[19] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.