Distributed Multi-Robot Area Patrolling in Adversarial Environments

Tauhidul Alam[*] Matthew Edwards[†] Leonardo Bobadilla[*] Dylan Shell[†]

ABSTRACT

Multi-robot patrolling is the problem of repeatedly visiting a group of regions of interest in an environment with a group of robots to prevent intrusion. Early works proposed deterministic patrolling algorithms which could be learned by an adversary observing them over time. More recent works provide non-deterministic patrolling schemes, but these are limited to perimeter patrolling and require coordination and/or synchronization. Here we investigate the problem of finding robust and scalable strategies for multi-robot patrolling under an adversarial environment. We present algorithms to find different distributed strategies for a patroller in the form of Markov chains which use convex optimization to minimize the average commute time for an environment, a subset of the environment, or a specific region of an environment. Additionally, we use these strategies in a game theoretical setup to form a payoff matrix to obtain an optimal mixed strategy for patroller. Our results show the scalability and applicability of our approach in different types of environments. Despite the lack of synchronization and coordination among patrollers, our approach performs competitively compared to existing methods.

1. INTRODUCTION

Patrolling is the problem of repeatedly visiting a group of locations in an environment. This problem has applications in such areas as environmental monitoring, infrastructure surveillance, and border security. In the multi-agent version patrolling is performed by multiple robots working together to ensure the safety of the area under surveillance. In its adversarial setting, one or more adversaries attempt to penetrate the environment being patrolled. Patrolling schemes can be further categorized into perimeter patrolling or area patrolling.

Multi-robot patrolling has been investigated in many different studies. Initial research proposed deterministic approaches based on the optimization of the frequency of visits to the regions in the environment [3, 4, 5, 6]. A survey of multi-agent patrolling strategies can be found in [1] where strategies are evaluated based on robot perception, communication, coordination, and decision-making capabilities. However, an adversary can easily penetrate the perimeter or area if a deterministic patrolling is used. For example, if a patrolling strategy ensures that a region around a perimeter is visited every 20 seconds and it takes an adversary 15 seconds to breech the perimeter, then the adversary is guaranteed success by attacking just after the region is visited [8].

Motivated by the presence of adversaries and the need for unpredictability, Agmon et al. [8] presents perimeter patrolling strategies based on Markov chain models that maximize the probability of detecting an adversary [8, 12, 14]. In this line of research, only cyclic graphs were considered, and the robots have to be synchronized and tightly-coupled, limiting deployments of these strategies practically. Moreover, all robots have to be placed in known locations and with equal separation distances in the environment. Sak et al. [7] consider an empirical non-deterministic approach for general graphs rather than a perimeter. However, only experimental results were presented without formalization or algorithmic details.

The problem of patrolling in the presence of adversaries can also be formulated as a *Bayesian Stackelberg Game*. In this game theoretical setting, the patroller first commits to an optimal mixed strategy generated through DOBSS (Decomposed Optimal Bayesian Stackelberg Solver) [11] where each strategy is a path in a fully connected graph. Once the patroller has decided its mixed strategy, the adversary may use knowledge of the mixed strategy to choose a region to attack. One drawback of this approach is that the set of strategies (paths) should be chosen *a priori* from a large set of possible paths and the particular connectivity of the graph is not studied in detailed.

Also related to our work is the problem of minimizing the effective resistance of a graph through convex optimization. This problem finds its original motivation in electrical networks where the solution is a measure of how well "connected" the network is [9]. The concept of effective resistance can be applied to Markov chains where minimizing the resistance between vertices on a graph corresponds to minimizing the commute times between vertices. With this application, robots patrolling on a graph with Markov chains would be able to quickly travel between every pair of vertices.

The purpose of our work is threefold. First, we would like to extend current ideas into more general class of graphs. The previous perimeter ideas can be extended to general graphs but this will need first finding a Hamiltonian cycle which is an NP-complete problem. Second, we would like to remove the need for communication, synchronization, and known initial placement of the randomized patrolling strategies. This will allow patrolling algorithms to be implemented with simple robots, in a decentralized fashion, and in communication denied environments. Third, we would like to adapt the game theoretical setup [11] to use Markov chains instead of deterministic strategies and to be applied to different graphs.

Randomized strategies based on Markov chains are used in our work for several reasons: 1) These will make it harder for an adversary to successfully complete an attack due to the unpredictability of the strategies; 2) A randomized motion can be easily imple-

[*]Tauhidul Alam and Leonardo Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL, USA. `talam005,bobadilla@cs.fiu.edu`

[†]Matthew Edwards and Dylan A. Shell are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. `medwards92, dshell@cse.tamu.edu`

This work was supported by the NSF via Awards CNS-#1263124, IIS-#1302393, and the CTE-Montague Scholar fund.

mented in a mobile robot, since its communication, sensing, and computation requirements are minimal; and 3) Efficient algorithms can calculate Markov chains with desired properties [9].

The contributions of the paper are as follows:

- We present algorithms that do not require communication, are based on convex optimization, can scale well, and can also be applied to any type of environment represented as a graph.
- We present a game theoretical approach to patrolling where the set of strategies are Markov chains. We also calculate the payoffs of each strategy and present approaches to generate the optimal mixed strategy for patroller and the optimal strategy for the adversary.

The remainder of the paper is organized as follows: Section II presents the problem formulation; Section III introduces algorithms to find decentralized and distributed strategies for multi-robot patrolling and a game theoretical approach for finding optimal strategies; Section IV presents experimental results of our approach; and Section V discusses our conclusions and future directions for research.

2. PROBLEM FORMULATION

We define the patrolling environment as an *undirected graph*, $G = (V,E)$ with $|V| = n$, $|E| = m$. Each vertex ($v \in V$) corresponds to a region, and each edge ($e \in E$) corresponds to a connection between two regions in the environment. Let M be a discrete time Markov chain on the graph with M_{ij}, the probability of transitioning between vertex i and vertex j. In this form, the patrollers need to know which node they are at in the graph when they arrive there to use the right weighting for outgoing edges. Once the robot has determined that it is at vertex k, it selects the k-th column from M, and uses this to weight its random choice of next vertex to visit. This is weaker than having to know your (x,y) locations at all times: indeed, given an initial location, a way of distinguishing the outgoing edges (*e.g.*, in clockwise order), and a sensor which indicates to a robot that it has arrived at a vertex, no other localization information is required.

The worst case scenario for patrollers is when they do not know where an adversary is going to attack and they are not certain about their initial positions in the graph. In this case, it is better to find a strategy for patrollers that minimizes the average commute or hitting time between every pair of vertices. The *hitting time*, H_{ij}, is the (random) time taken to reach vertex j starting from vertex i using the Markov chain. The *commute time*, C_{ij}, is the time it takes to travel from vertex i to vertex j and back using the Markov chain, $C_{ij} = H_{ij} + H_{ji}$. The average hitting time, $\bar{H}$, and average commute time, $\bar{C}$, for a Markov chain in a graph are the times averaged over all pairs of vertices. In this context, the patrolling problem will be defined as:

Problem 1: Finding randomized patrolling strategies
Given the graph, $G = (V,E)$, find distributed strategies that minimize average commute time ($\bar{C}$) or average hitting time ($\bar{H}$): 1) for all vertices in V; 2) for a clique $v_1...v_d \in V$, $d \leq n$; and 3) to a particular vertex $v \in V$

We also explicitly model an adversary who (1) knows all possible strategies that a patroller can choose, (2) has full knowledge of the environment, and (3) is able to optimally choose the vertex to attack in graph G. This scenario is known as a *Bayesian Stackelberg Game* where the two players of the game are the patroller and the adversary. In our proposed game theoretical setup, each patroller's strategy is a Markov Chain. The adversary also has a set of strategies to attack in any of the n vertices. The game is formulated as follows:

- A nonempty, finite set called the set of patrolling strategies $\mathscr{M} = \{M_1,...,M_k\}$ where k is the number of Markov chains.
- A nonempty, finite set called the set of adversary strategies V. Each $v \in V$ is a vertex in the graph.
- A function $P : \mathscr{M} \times V \longrightarrow \mathbb{R} \cup \{\infty\}$ called the payoff matrix for the patroller.
- A function $Q : \mathscr{M} \times V \longrightarrow \mathbb{R} \cup \{\infty\}$ called the payoff matrix for the adversary.

To calculate both payoff matrices, we need the following values:

- d_i^{pat} : value of goods in region/vertex i to the patroller.
- d_i^{adv} : value of goods in region/vertex i to the adversary.
- c_i^{pat} : reward to the patroller of catching the adversary in the i-th region (or equivalently the i-th vertex).
- c_i^{adv} : cost to the adversary of getting caught in the i-th region (or vertex).
- p_i : the probability that the patroller will catch the adversary at the i-th vertex/region of the environment.

A patroller's reward must consider the factor c_i^{pat} for capturing the adversary (with probability p_i) and also the value lost when the adversary is not captured (probability $1 - p_i$). Conversely, the adversary is pays cost c_i^{adv} (with probability p_i) but gains d_i^{adv} : with $1 - p_i$. Additionally, p_i also depends on the i-th hitting time of the Markov chain for each patrolling strategy. Given these definitions, we are interested in the following problem:

Problem 2: Generating the optimal strategies
Given the payoff matrices, P and Q, the set of patroller strategies, $\mathscr{M}$, the set of strategies for adversary, V, find the optimal mixed strategy for the patroller and the optimal strategy for the adversary.

3. METHODS

In this section, we present algorithms to generate patrolling strategies and a game theoretical approach to obtain optimal strategies.

3.1 Distributed Patrolling Strategies

As mentioned before, we consider patrolling strategies in a graph as Markov chains $\mathscr{M}$. Gosh et al. [9] define the effective resistance between two vertices i and j in a graph as R_{ij}. They also define the nonnegative conductance on edge l as g_l which is a weight that can be assigned to an edge of a graph. R_{ij} is small when there are many paths between vertices i and j with high edge weights and is large when there are fewer paths between vertices i and j with low edge weights. In [9], the *total effective resistance, R_{tot}* is the sum of the effective resistances between all pairs of vertices,

$$R_{tot} = \frac{1}{2} \sum_{i,j=1}^{n} R_{ij} = \sum_{i<j} R_{ij} \tag{1}$$

R_{tot} is related to the average commute time ($\bar{C}$) of the Markov chain.

In this work, an environment with small total effective resistance corresponds to a Markov chain with small hitting or commute times between vertices, and a large total effective resistance corresponds to a Markov chain with large hitting or commute times between at least some pairs of vertices. In [9], a convex optimization method is proposed for minimizing the total effective resistance of the graph by allocating a fixed total conductance among the edges:

$$\begin{array}{ll} \text{minimize} & R_{tot} \\ \text{subject to} & \mathbf{1}^T g = 1, \quad g \geq 0 \end{array} \tag{2}$$

The optimization variable is $g \in R^m$, the vector of edge conductances. In our patrolling strategies, the total *effective resistance minimization problem* (ERMP) is equivalent to the problem of selecting weights on edges to minimize the commute (or hitting) time between vertices. Once R_{ij} is considered as distances among the vertices, the ERMP is the problem of allocating the edge weights to a graph to make the graph small in terms of average distance between vertices.

The relationship between commute time, C_{ij}, and effective resistance, R_{ij}, between vertex i and j is the following [9]:

$$C_{ij} = (\mathbf{1}^T g) R_{ij}$$

Using this relationship, we present our first patrolling strategy of *minimization of average commute time* (MACT) on a graph following equation (2).

In the second patrolling strategy, we have also extended the MACT problem for a subset of vertices or clique. This extension ensures a higher probability of travelling along the edges within the subset of vertices while still allowing for travel to the remaining vertices of the graph. For example, the edges $e = \{shortestPath(i,j), shortestPath(j,k), shortestPath(k,i)\}$, where $e \in E$, are given priority. This means that the edges along this shortest cycle will be optimized such that the edge weights will consist of the majority of the sum of all edge weights. We extend the ERMP as follows:

$$\begin{array}{ll} \text{minimize} & R_{tot} \\ \text{subject to} & \mathbf{1}^T g = 1, \qquad g \geq 0 \\ & e = \text{shortestCycle}(s), \\ & \sum_e w(e) \geq t \end{array} \tag{3}$$

In this problem, the variable e represents an edge along the shortest cycle between the vertices in subset, s, of V. The condition $\sum_e w(e) \geq t$ ensures that the sum of the edge weights of $w(e) \in g$ will be greater than or equal to a threshold that represents the percentage of edge weights allocated to the cycle.

Algorithm 1 MACTtowardsVertices(A, b)

```
Input: $A, b(\cdot)$ {Incidence matrix and edge costs of graph}
Output: $\mathscr{M} = \{M_1, .., M_n\}$ {Markov chains for each vertex}
 1: $\mathscr{M} \leftarrow 0$
 2: for $i = 1$ to $n$ do
 3:     $M(n \times n) \leftarrow 0$
 4:     for $j = 1$ to $n$ do
 5:         if $j == i$ then
 6:             for all edges $e_{jb} \in A$ do
 7:                 $M_{jb} \leftarrow M_{jb} + 1$
 8:             end for
 9:             continue
10:         end if
11:         $k \leftarrow kShortestPaths(b(\cdot), j, i)$
12:         for all edges $e_{ab} \in k$ do
13:             $M_{ab} \leftarrow M_{ab} + 1$
14:         end for
15:     end for
16:     normalize $M$
17:     $\mathscr{M} \leftarrow \mathscr{M} \cup M$
18: end for
19: return $\mathscr{M}$
```

Let A be an $n \times m$ incidence matrix of a graph $G = (V, E)$ where $n = |V|$ and $m = |E|$. Suppose edge l connects vertices i and j. We define $A_{il} = 1$, $A_{jl} = -1$, and all other elements 0. Let $b : E \rightarrow \mathscr{R}^+$ denote the distance associated with each edge in graph G. We present an algorithm for MACT towards each vertex.

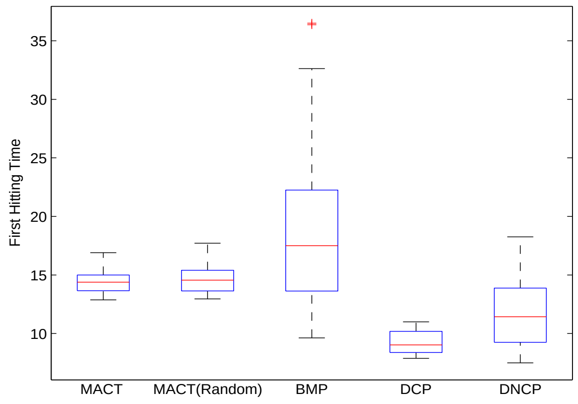

Figure 1: Comparison result of our MACT method with uniform and random robot placement as well as existing three methods (e.g. BMP, DCP, DNCP) [8] for patrolling. Each line represents the maximum, minimum first hitting time and each box represents the median along with the mean hitting time in the middle.

Algorithm 1 generates a set of n Markov chains, $\mathscr{M}$, that minimizes the commute time towards every vertex $v \in V$ for the graph G. For every Markov chain, the k shortest paths from v_j to the target vertex v_i are found (line 11). The k shortest paths are found using Yen's algorithm [15], which has a worst case runtime of $O(n^2)$. After the k shortest paths are found, every edge transition found within the k paths is incremented in the Markov chain that is being generated (lines 12-14). When $v_j = v_i$, all possible edge transitions from the target vertex v_j to adjacent vertices are incremented (lines 5-8). After all paths have been explored for a Markov chain, M is normalized and is added to the set of Markov chains $\mathscr{M}$.

3.2 Game Theoretical Approach

The game theoretical approach has two stages; the first stage calculates payoff matrices, the second stage generates optimal strategies.

Payoff Matrices Calculation:

The payoff matrices for patroller and adversary, P and Q, are calculated from the set of Markov chains, $\mathscr{M}$. We calculate n Markov chains from Algorithm 1 and we also have two other Markov chains from MACT for all vertices and MACT for a subset of vertices or clique. Now we have $n+2$ Markov chains in the set of Markov chains, $\mathscr{M}$.

In Algorithm 2, we present a procedure to calculate the payoff matrices. The algorithm assigns uniform values of goods (d_i^{pat}, d_i^{adv}) in each vertex for both the patroller and the adversary. It calculates the hitting time matrix, H, (line 6) for each Markov chain. The mean hitting time vector (lines 8-10), mht, is calculated using the function $HTofPreferredVertices$. This function takes the vertices involved in the Markov chain optimization as follows: 1) In the case of MACT strategy, it calculates the mean of the hitting times for all vertices; 2) In the case of MACT for a subset of vertices or clique, it takes mean hitting time for the subset of vertices; and 3) In the case of MACT toward each vertex, it takes the hitting time towards that particular vertex. In lines 16-24, it assigns the values to the variables needed to calculate the payoff matrices. In lines

25-28, it calculates the payoff matrices using these variables.

Algorithm 2 PayoffMatrixCalculation($\mathscr{M}$)

```
Input: $\mathscr{M} = \{M_1,..,M_{n+2}\}$ {Markov chains for all patrolling strategies}
Output: $P,Q$ {Payoff matrix for patroller and adversary}
 1: for $i = 1$ to $n$ do
 2:     $d_i^{pat} \leftarrow 1/n$
 3:     $d_i^{adv} \leftarrow 1/n$
 4: end for
 5: for $i = 1$ to $|\mathscr{M}|$ do
 6:     $H \leftarrow hittingTimes(M_i)$
 7:     $mht \leftarrow 0$
 8:     for $j = 1$ to $n$ do
 9:         $mht_j \leftarrow HTofPreferredVertices(H)$
10:     end for
11:     $asc \leftarrow sortAscending(mht)$
12:     $desc \leftarrow sortDescending(mht)$
13:     $c^{pat} \leftarrow 1$
14:     $c^{adv} \leftarrow 1$
15:     $hh \leftarrow n$
16:     for $k = 1$ to $n$ do
17:         $c_{asc_k}^{pat} \leftarrow c_{asc_k}^{pat} hh$
18:         $c_{desc_k}^{adv} \leftarrow c_{desc_k}^{adv} hh$
19:         $hh \leftarrow hh - 1$
20:     end for
21:     $p \leftarrow 1$
22:     for $k = 2$ to $n$ do
23:         $p_k \leftarrow 1/2^{k-2}$
24:     end for
25:     for $k = 1$ to $n$ do
26:         $P_{ik} \leftarrow p_k c_k^{pat} + (1 - p_k) d_k^{pat}$
27:         $Q_{ik} \leftarrow p_k c_k^{adv} + (1 - p_k) d_k^{adv}$
28:     end for
29: end for
30: return $P,Q$
```

Optimal Strategies Generation:

The probability distribution of choosing the strategies for the patroller can be represented as an m-dimensional vector,

$$w = (w_1, w_2, ..., w_m) \tag{4}$$

Equation (5) should satisfy: 1) $w_i \geq 0$ for all $i \in 1...m$, and 2) $w_1 + w_2 + ... + w_m = 1$. The value w_i is the proportion of the patroller choosing strategy w_i.

Similarly z represents all the possible strategies for the adversary as a n-dimensional vector,

$$z = (z_1, z_2, ..., z_n) \tag{5}$$

Equation (6) should satisfy: 1) $z_i \in \{0, 1\}$ for all $i \in 1...n$, and 2) $z_1 + z_2 + ... + z_n = 1$. Since the adversary can attack any region $n \in |V|$ in the environment, the adversary's strategies, z_i, are strategies for attacking each region separately, and only one strategy can be chosen.

DOBSS generates the optimal mixed strategy for the patroller while considering an optimal adversarial response for all patrolling strategies [11]. It considers reward-maximizing strategies for patroller and cost-minimizing strategy for adversary.

4. EXPERIMENTAL RESULTS

4.1 Decentralized Patrolling Results:

We have compared our MACT method with the three patrolling methods proposed by Agmon et al. [8]. In our approach, we place patrollers on the graph either randomly or an equal distance from each other (uniform) around the graph; Agmon et al's three methods, BMP, DCP, DNCP, all require that patrollers are placed an equal distance from each other on the graph. The patrollers are mobile robots such as UAVs or wheeled robots. In Figure 1, we show the mean first hitting time of MACT with both uniform and random placement of patrollers compared to their three methods on a graph of 64 vertices and 2016 edges, (K_{64} : 2016). Since their methods require a cycle graph to be tested on, we have tested their methods on a cycle graph of 64 vertices, (C_{64}), which is assumed to be equivalent to the Hamiltonian cycle of our original graph, K_{64}. In our test, we have simulated four patrollers so the distance between two consecutive robots, d, is 16. We also consider the number of state transitions it takes to penetrate the environment, t, as 12 units. Though our approach does not use synchronization and communication among robots, it still surpasses the mean hitting time of one of their methods and performance does not degrade much compared to other two.

We compare more closely our MACT method with DCP and DNCP methods using varying number of patrollers on the graphs mentioned above (K_{64} and C_{64}). The result of this comparison is shown in Table 1. It shows that the more patrollers are present on the graph, the closer the first hitting times of DCP and DNCP's methods approach the data from our approach.

Table 1: Comparison of average first hitting time of our approach, MACT with uniform and random robot placement and existing two methods (DCP, DNCP) for 30 experiments

No. of Patrollers	Uniform MACT	Random MACT	DCP	DNCP
2 (d=32, t=24)	30.35	30.58	17.29	20.85
4 (d=16, t=12)	14.22	14.70	9.26	11.74
8 (d=8, t=6)	7.11	7.30	5.90	6.90
16 (d=4, t=3)	3.21	3.40	3.1	3.16

We have also tested our methods on different types of graphs, including line, tree, mesh, complete, and randomly generated graphs. Figure 2 shows the edge weight allocation on different graphs [16] for our MACT method. In each graph, the thickness of each edge corresponds to the optimal edge weight value, or the probability of that edge being chosen by a patroller. For example, a wider edge connection between two vertices represents a high probability of that edge being chosen for travel, and vice versa.

4.2 Game Theoretical Optimal Strategies Result:

We have tested DOBSS with a small graph [16] consisting of eight vertices and thirteen edges. The patroller has ten patrolling strategies available: eight which minimize the average commute time towards each of eight vertices, one that minimizes the average commute time towards a subset of vertices or clique, and one that minimizes the average commute time over all vertices. The resulting graphs for all 10 patrolling strategies are shown in Fig 3.

As an illustration, payoff matrices of a small graph are shown in Table 2, where the values of the payoff matrices for patroller and adversary, P and Q, for ten patrolling strategies are calculated using Algorithm 2.

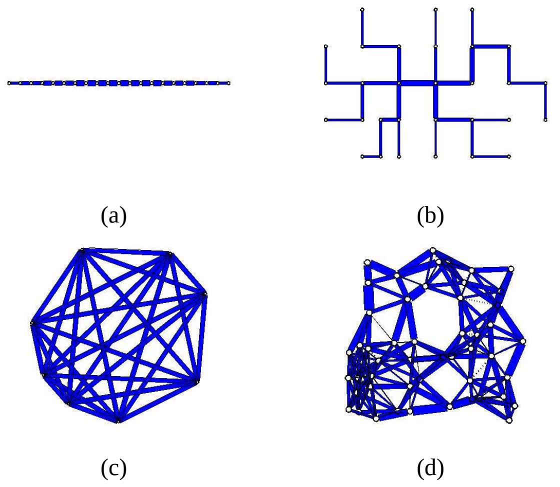

Figure 2: Different types of Graph for MACT method: a) Edge weight allocation on a line graph of 20 vertices; b) Edge weight allocation on a tree with 30 vertices; c) Edge weight allocation on a complete graph, K_8; d) Edge weight allocation on a randomly generated graph with 50 vertices and 200 edges.

DOBSS produces the optimal mixed strategy as shown in Table 3. The patroller will patrol the graph with an optimal mixed strategy consisting of strategies 7 and 9. Here the cost-minimizing strategy for the adversary generates an optimal response for attacking vertex 7 (i.e. $z_7 = 1, z_i = 0, i \neq 7$ for all $i \in 1..n$).

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced algorithms for finding distributed patrolling strategies for multiple robots based on Markov chains which minimize the average commute time towards (1) a specific vertex, (2) a subset of vertices, and (3) the average commute time for all pairs of vertices in the graph. Methods (1) and (2) use convex optimization and method (3) seeks shortest path in graphs. Several interesting directions are left for future work as described in further detail below.

Table 2: Payoff Matrices for Small Graph

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
P_1	8.0000	1.5938	0.1543	0.1318	3.5625	0.3672	0.7344	0.2148
Q_1	1.0000	0.8438	0.2324	0.1865	1.0625	0.4297	0.6094	0.3086
P_2	0.2148	8.0000	1.5938	0.7344	0.1543	0.1318	0.3672	3.5625
Q_2	0.3086	1.0000	0.8438	0.6094	0.2324	0.1865	0.4297	1.0625
P_3	0.2148	1.5938	8.0000	3.5625	0.1543	0.1318	0.7344	0.3672
Q_3	0.3086	0.8438	1.0000	1.0625	0.2324	0.1865	0.6094	0.4297
P_4	0.3672	0.7344	3.5625	8.0000	0.1543	0.1318	0.2148	1.5938
Q_4	0.4297	0.6094	1.0625	1.0000	0.2324	0.1865	0.3086	0.8438
P_5	0.7344	0.3672	0.1543	0.1318	8.0000	3.5625	1.5938	0.2148
Q_5	0.6094	0.4297	0.2324	0.1865	1.0000	1.0625	0.8438	0.3086
P_6	0.3672	0.1543	0.7344	0.1318	3.5625	8.0000	1.5938	0.2148
Q_6	0.4297	0.2324	0.6094	0.1865	1.0625	1.0000	0.8438	0.3086
P_7	0.1318	1.5938	0.7344	0.1543	0.2148	0.3672	8.0000	3.5625
Q_7	0.1865	0.8438	0.6094	0.2324	0.3086	0.4297	1.0000	1.0625
P_8	0.2148	1.5938	0.3672	3.5625	0.1318	0.1543	0.7344	8.0000
Q_8	0.3086	0.8438	0.4297	1.0625	0.1865	0.2324	0.6094	1.0000
P_{MACT}	0.2148	3.5625	0.7344	0.1543	1.5938	0.1318	8.0000	0.3672
Q_{MACT}	0.3086	1.0625	0.6094	0.2324	0.8438	0.1865	1.0000	0.4297
P_{Clique}	0.7344	3.5625	0.1318	0.2148	1.5938	0.1543	0.3672	8.0000
Q_{Clique}	0.6094	1.0625	0.1865	0.3086	0.8438	0.2324	0.4297	1.0000

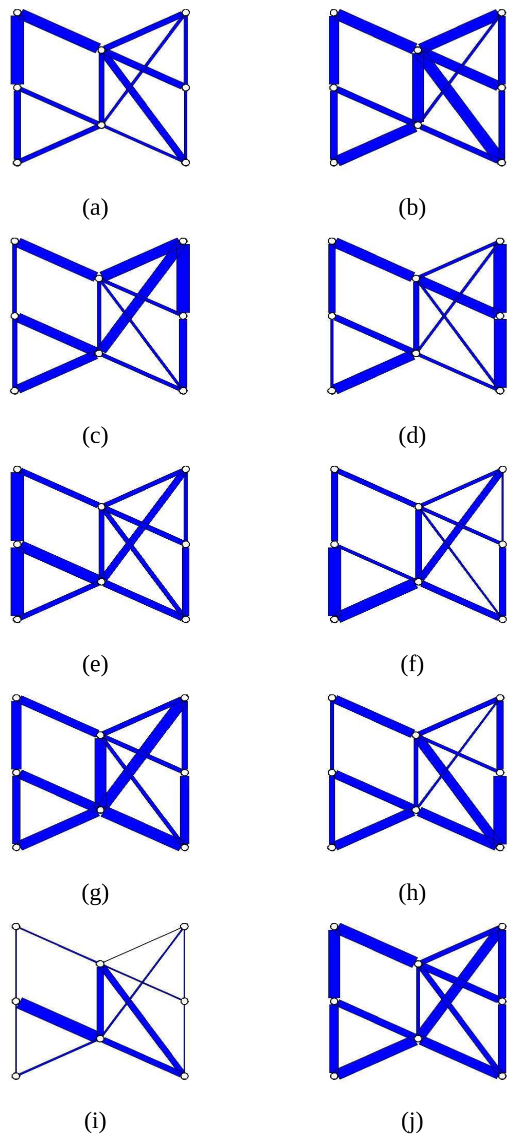

Figure 3: Edge weight allocation for ten patrolling strategies of a small graph: a)-h) Edge weight allocation for minimizing average commute time towards vertex 1 to vertex 8 respectively; i) Edge weight allocation for minimizing average commute time preferring a clique or subset of vertices, 2,5,8; j) Edge weight allocation for minimizing average commute time over all vertices.

Table 3: Optimal Mixed Strategy Result for Small Graph

Patrolling Strategy No., M_i	Proportion of using Strategy w_i
1	0
2	0
3	0
4	0
5	0
6	0
7	0.9012
8	0
9	0.0988
10	0

One of the goals of our work was to remove the communication and localization requirements in [8] and extend the class of graphs to which Markov chain based strategies can be applied. Even though we have used simple Markov chain based strategies for patrollers, one surprising result of our work seems to indicate that our performance is better than BMP, despite the lack of communication and synchronization when compared to that algorithm. Further experimental tests and analytical tools are going to be used to quantify the differences in performance between our approach and existing methods.

Another immediate line of future work is the implementation of our ideas in physical deployments. Since our approaches do not require that we solve metric localization,[1]communication, or synchronization problems, and the motions of the patrollers follow simple Markov chains, so they can be implemented in inexpensive robotic platforms. In order to apply our methods, we can take a 2D or 3D workspace with obstacles, create a grid where each element of the grid is a state in the Markov chain, and choose the appropriate neighborhoods for transitions.

We also consider a game theoretical setup where the adversary knows the strategies of the patroller (given as a set of Markov chains) and responds optimally. We have proposed a set of payoff matrices for representing the game and find the optimal mixed strategy for patrollers through DOBSS. In the game theoretical experiment, for the 8 vertices and 13 edges example in Figure 3, we have considered as strategies for the patroller Markov chains that minimize commute times to each of the vertex, a Markov chain that minimizes to a clique, and one that minimizes the average commute time for all vertices. Interestingly, the optimal mixed strategy chosen was a Markov chain that minimizes the commute time to the vertex under attack v_7 and a Markov chain that minimizes average commute time for all vertices. This validates our hypothesis of minimizing average commute time for all vertices as a "good" strategy for the patroller and shows that it is also important for the patroller to guard the vertex under attack. In future work, we plan to test our game theoretical setup in larger graphs and extend it to multiple adversaries.

We plan to incorporate a less rational adversary in our framework. In our current setup, the adversary has complete knowledge of the environment and is assumed to be perfectly rational. However, real world situations will involve adversaries that are less knowledgeable and rational and more unpredictable. Because of this, we wish to extend our ideas to use a more realistic adversarial model to help test the true effectiveness of the patrolling policies.

[1]Recall from the discussion on page 2, the robots need only localize to a vertex, *i.e.*, topologically.

6. REFERENCES

[1] D. Portugal, and R. Rocha, "A Survey on Multi-robot Patrolling Schemes," in *Proceedings of Doctoral Conference on Computing, Electrical and Industrial Systems*, Lisbon, Portugal, 2011.

[2] William J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[3] Y. Chevaleyre, F. Sempé, and G. Ramalho, "A Theoretical Analysis of Multi-Agent Patrolling Strategies," in *Proceedings of the third International on Autonomous Agents and Multiagent Systems*, pp. 1524-1525, August 2004.

[4] A. Almeida, G. Ramalho, H. Santana, P. A. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre, "Recent Advances on Multi-agent Patrolling," in *Proceedings of the Brazilian Symposium on Artificial Intelligence*, vol. 3171, pp. 474-483, Springer 2004.

[5] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping task," in *Proceedings of the International Conference on Robotics and Automation*, pp. 1724-1729, May 2006.

[6] Y. Elmaliach, N. Agmon, and G. A. Kaminka, "A multi-robot area petrol under frequency constraints," in *Proceedings of the International Conference on Robotics and Automation*, pp. 385-390, May 2007.

[7] T. Sak, J. Wainer, and S. K. Goldenstein, "Probabilistic Multiagent Patrolling," in *Proceedings of the Brazilian Symposium on Artificial Intelligence*, vol. 5249, pp. 124-133, Springer 2008.

[8] N. Agmon, S. Kraus, and G. A. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *Proceedings of the International Conference on Robotics and Automation*, pp. 2339-2345, May 2008.

[9] A. Ghosh, S. Boyd, and A. Saberi, "Minimizing effective resistance of a graph," in *Society for Industrial and Applied Mathematics Review*, vol. 50, no. 1, pp. 37-66, 2008.

[10] P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordóñez, S. Kraus, "An efficient heuristic approach for security against multiple adversaries," in *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, May 2007.

[11] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, S. Kraus. "Playing Games for Security: An efficient exact algorithm for solving bayesian stackelberg games," in *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*, pp. 895-902, 2008.

[12] N. Agmon, "On events in multi-robot patrol in adversarial environments," in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 591-598, 2010.

[13] D. Portugal and R. Rocha, "MSP Algorithm: Multi-robot Patrolling Based on Territory Allocation Using Balanced Graph Partitioning," in *Proceedings of Symposium on Applied Computing*, pp. 1271-1276, March 2010.

[14] N. Agmon, G. A. Kaminka, and S. Kraus, "Multi-robot adversarial patrolling: facing a full-knowledge opponent," *Journal of Artificial Intelligence Research*, vol. 42, no. 1, pp. 887-916, September 2011.

[15] Yen's Algorithm (September 2014) [Online]. Available:http://en.wikipedia.org/wiki/Yen.

[16] Convex Optimization in Matlab (September 2014) [Online]. Available: http://cvxr.com/cvx/examples/.