Multi-Robot Repeated Boundary Coverage Under Uncertainty

Pooyan Fazli, Alireza Davoodi, Alan K. Mackworth
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada V6T 1Z4
Email: {pooyanf,davoodi,mack}@cs.ubc.ca

***Abstract*—This paper describes work in progress addressing the problem of repeated coverage by a team of robots of the boundaries of both a target area and the obstacles inside it. Events are generated randomly on the boundaries and may have different importance weights. In addition, boundaries of the area and the obstacles are heterogeneous, in that events might appear with varying probabilities on different parts of the boundary. The goal is to maximize the reward by detecting the maximum number of events, weighted by their importance, in minimum time. The reward a robot receives for detecting an event depends on how early the event is detected. To this end, a *Markov Decision Process (MDP)* formalism is used to model the coverage problem and capture the uncertainties in the scenario. The performance of the algorithm proposed to solve the *MDP* will be compared with two static algorithms on the basis of the total reward gained during a repeated boundary coverage mission.**

I. Introduction

Multi-Robot Boundary Coverage is a challenging problem with different applications such as surveillance and monitoring, cleaning, intrusion detection, facility inspection, and so on. In this task, a team of robots cooperatively visits (observes or sweeps) the boundaries of a target area and the obstacles inside it. The goal is to build efficient paths for all the robots which jointly ensure that every point on the boundaries is visited by at least one of the robots. The *Boundary Coverage* is a variant of the *Area Coverage* [3], [8]–[10] problem, in that, the aim is to cover just the boundaries, not the entire area.

There are two classes of boundary coverage problems:

- ***Single Coverage:*** The aim is to cover the boundary until all its accessible points of interests have been visited at least once, while minimizing the time, the distance traversed by the robots, or the number of visits to the points [4], [19].
- ***Repeated Coverage:*** The goal is to cover all the accessible points of interest on the boundary repeatedly over time, while maximizing the frequency of visiting points on the boundary, minimizing the weighted average event detection time, minimizing the sum/maximum length of the paths/tours generated for the robots, or balancing the workload distribution among the robots. Visiting the points on the boundary can be performed with uniform or non-uniform frequency, depending on the priorities of different parts of the boundary[1] [1], [2].

II. Problem Definition and Preliminaries

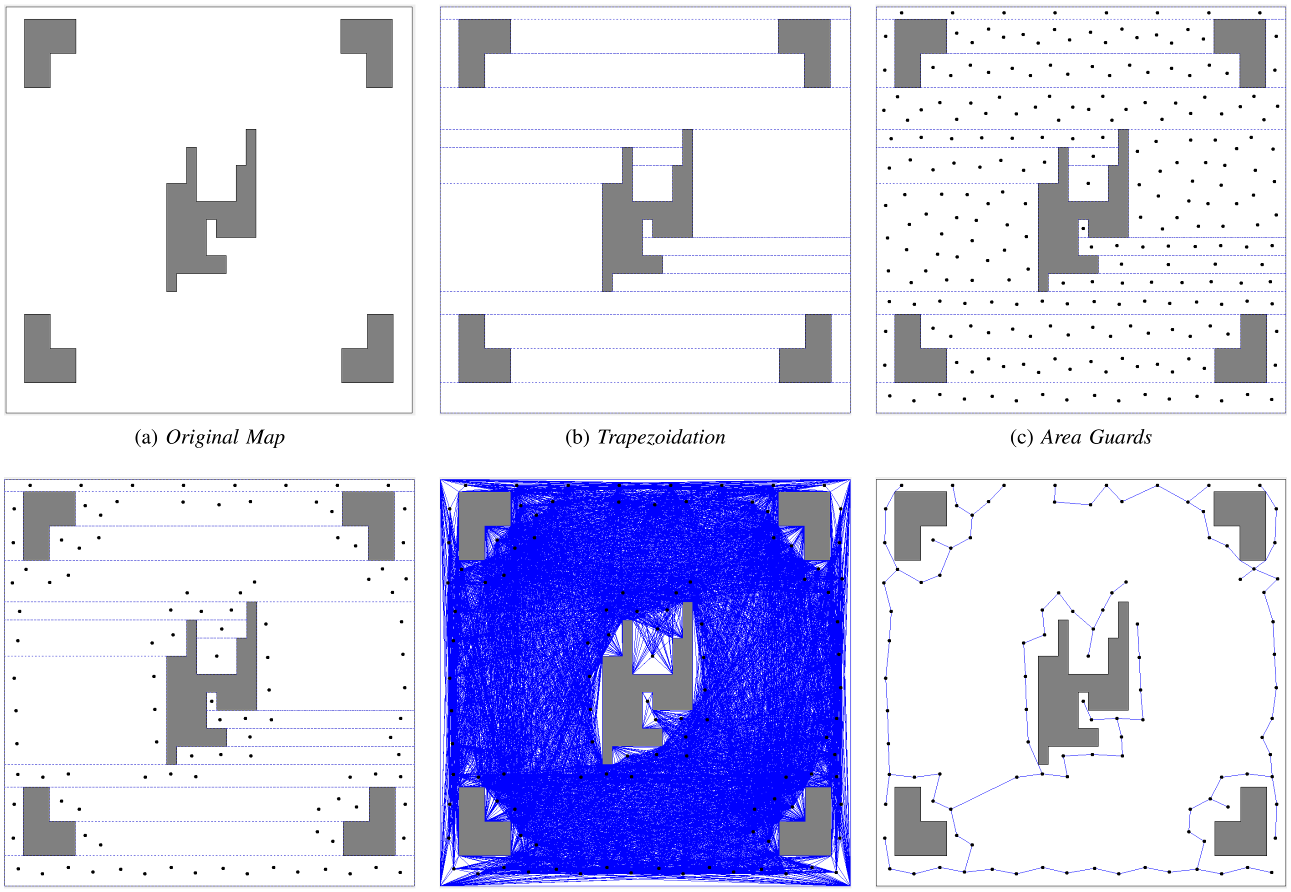
In this paper, we address the *Multi-Robot Repeated Boundary Coverage* problem with the following specifications:

- The environment is a simple polygon including rectilinear or non-rectilinear polygonal obstacles.
- The 2D map of the environment is known *a priori*.
- An arbitrary number of homogeneous robots is involved in the coverage mission. The robots are assumed to move at the same unit speed.
- The robots are equipped with a panoramic visual sensor with limited visual range. The sensors are ideal and without noise, that is, they guarantee the detection of an event occurring within the visual range of the robot.
- The events are generated randomly and might occur on any part of the boundaries.
- The events can have different types, and each event type has its own importance weight.

Definition 1. *Event Type: m* types of events might happen in the environment which are of interest to the robots. The set of all event types is represented by $E = \{E_1, E_2, ..., E_m\}$. Similarly, an event of type E_i is denoted as e_i.

Definition 2. *Event Importance:* A degree of importance is defined for each event type E. The importance of an event type E is given by $weight(E)$. It is assumed that $weight(E) \in (0,1]$ such that 1 is the highest degree of importance. The importance can also be referred to as the *priority*, meaning that an event of higher

[1]In this paper, we use the terms 'Coverage' and 'Repeated Coverage' interchangeably.


(a) *Original Map* (b) *Trapezoidation* (c) *Area Guards*

(d) *Boundary Guards* (e) *Visibility Graph* (f) *Boundary Graph*

Fig. 1: *Sequential Stages of Building the Boundary Graph*

importance should have higher priority of being detected.

- The reward a robot receives for detecting an event depends on how early the event is detected. At each time step after the event occurrence, the detection reward of the event is decreased by a multiplicative discount factor.
- The boundaries are heterogeneous, in that, events of one type might appear with varying probabilities on different parts of the boundary.

Two types of approaches are proposed to handle the problem: (1) *Uninformed Boundary Coverage* and (2) *Informed Boundary Coverage* algorithms.

Uninformed Boundary Coverage, consisting of two static algorithms, ignores the uncertainties in the coverage mission. In this approach, the robots presume that the events are all of the same importance value and the boundaries are homogeneous. On the other hand, *Informed Boundary Coverage* is primarily based on the *Markov Decision Process (MDP)* formalism in which the robots try to cooperatively maximize the reward function by detecting the maximum number of events, weighted by their importance values, in minimum time. To this end, the robots *learn* the expected reward of visiting a state in the target area at each time step, and based on that, they *plan* to select the best possible path to visit the most promising state at the time in the target area.

The performance of the proposed approaches will be evaluated on the basis of the total reward gained during a finite repeated boundary coverage mission.

III. Environment Modeling

The *Uninformed Boundary Coverage* and *Informed Boundary Coverage* algorithms both require that a roadmap is built within the target area, capturing the connectivity of the free space close to the boundaries while taking into account the limited visual range of the robots. To this end, a graph-based representation called the *Boundary Graph* is constructed on the target area. The *Boundary Graph* provides a roadmap for the robots, enabling them to move throughout the environment to monitor the boundaries of the area and the obstacles. In order to construct the roadmap, a sufficient number of control points, called the *boundary guards*, are placed within the environment, considering the limited visual range of the robots.

A. Locating Guards with Limited Visual Range

In our problem definition, we presume the robots are equipped with panoramic cameras with a 360° field of view. However, the cameras' visual range is limited. The proposed approach initially locates a set of *area guards* required to visually cover an entire area. The term *guard* is taken from the *Art Gallery Problem* [17]. These static *area guards* are control points that can jointly cover the whole environment while satisfying the limited visual range constraint of the robots. In other words, if we had as many robots as the number of *guards*, and each robot was stationed on a *guard*, the entire area would be covered visually by the robots.

To locate the *guards*, the algorithm decomposes the initial target area, a 2D simple polygon with static obstacles, into a collection of convex polygons using a *Trapezoidal Decomposition* method [20], and then applies a post-processing approach to eliminate as many trapezoids as possible (Figure 1b). The post-processing step is more effective in cluttered areas, and since the number of *guards* located by the algorithm is directly correlated to the number of trapezoids, fewer trapezoids will result in fewer guards.

At the next step, a divide-and-conquer method similar to that in [12] is used to successively subdivide each of the resulting convex polygons (trapezoids) into smaller convex sub-polygons until each of them can be covered visually by one *guard* (Figure 1c).

Since, in the current problem, we are interested in monitoring just the boundaries, not all the computed *area guards* are necessary. So, all the *guards* whose visual area does not intersect the boundaries are removed from the set of *area guards*. Figure 1d illustrates the *boundary guards* computed on the sample environment.

B. Building the Graph

Once the boundary guards are located in the target area, a graph called the *Visibility Graph (VG)* [13] is constructed on the guards and the corners of the obstacles (Figure 1e). In order to build the *Visibility Graph*, any two points of interest (a boundary guard or an obstacle corner) which are mutually visible are connected by an edge. Two points are mutually visible if the edge connecting them does not intersect any obstacles in the environment.

C. Boundary Graph

Algorithm 1 describes the steps of the construction of the *Boundary Graph* on a given environment. The input of the algorithm is the *Visibility Graph* made on the map of the area.

The method starts by using the *Floyd-Warshall* algorithm to find the set $MD = \{(c_{ij}, v_i, v_j) | v_i, v_j \in V_{vis}\}$ of minimum distances, c_{ij}, and the set $SP = \{(r_{ij}, v_i, v_j) | v_i, v_j \in V_{vis}\}$ of shortest paths, r_{ij}, between any pair of vertices v_i and v_j of the input graph *(line 4)*.

The minimum value of all the minimum distances in *MD* is then selected provided that both the endpoints of the corresponding shortest path in *SP* belong to the set of boundary guards, *SG*, computed in section III-A *(line 5)*. The chosen path *(line 6)*, including all its nodes and edges, forms the initial component of the *Boundary Graph (line 7)*.

Algorithm 1: Boundary Graph

```
Input:
Graph $G_{vis}(V_{vis}, E_{vis})$, where $V_{vis} = SG \bigcup P$   /* VG */
$SG = \{g_1, g_2, ...., g_m\}$   /* Boundary Guards */
$P = \{p_1, p_2, ...., p_n\}$   /* Endpoints of Obstacles */

Output:
$G_{boundary}(V_{boundary}, E_{boundary})$ where $V_{boundary} = SG \bigcup \widetilde{P}$,
$\widetilde{P} \subset P$   /* Boundary Graph */

1  begin
2      $V_{boundary} \longleftarrow \phi$
3      $E_{boundary} \longleftarrow \phi$
4      $(MD, SP) \longleftarrow FloydWarshall(G_{vis})$
5      $(i,j) \longleftarrow \underset{(i,j)}{\arg\min}\{c_{ij} | (c_{ij}, v_i, v_j) \in MD \ \& \ v_i, v_j \in SG\}$
6      $r_{ij} \longleftarrow GetCorrespondingShortestPath(i,j)$
7      $G_{boundary}(V_{boundary}, E_{boundary}) \longleftarrow$
       $InitialBoundaryGraph(r_{ij})$
8      while $\neg$ all the guards added do
9          $g \longleftarrow FindClosestGuardTo(G_{boundary})$
10         $Expand(G_{boundary}, g)$
11     end
12     return $G_{boundary}(V_{boundary}, E_{boundary})$
13 end
```

Next, among all the guards that have not yet been added to the graph, the algorithm finds the closest guard to the current component *(line 9)*, merging the corresponding shortest path with it *(line 10)*. Following the same process, the algorithm keeps expanding the *Boundary Graph* until there are no more boundary guards to be added to the graph *(lines 8-11)*. The resultant graph is the final *Boundary Graph (line 12)*. The nodes of the *Boundary Graph* includes all the boundary guards *(SG)* and the subset of the obstacles' nodes *($\widetilde{P} \subset P$)*, collectively referred to as *Points of Interests ($PoI = SG \bigcup \widetilde{P}$)*. Traversing the *Boundary Graph* guarantees complete coverage of the boundaries given the limited visual range of the robots.

Figure 1f illustrates the *Boundary Graph* built on the *Visibility Graph* of figure 1e.

D. Boundary Segmentation

The boundaries of the area and the obstacles are divided into identical length *segments*, each of which is small enough to be completely visible by a guard, and such that the probability of event occurrence is uniform along the *segment*.

Definition 3. *Visual Area of a Guard (VA_g):* The *visual area* of a guard, VA_g, is the set of all the *segments* which are visible to the guard g, *i.e.* $VA_g = \{seg_g^1, seg_g^2, ..., .seg_g^p\}$.

Definition 4. *Shared Segment:* A *shared segment* is common to the *visual area* of two or more guards.

Assumption 1. *The events occurring within the visual area of a guard are detected only when the robot is located on the guard.*

IV. Uninformed Boundary Coverage

Uninformed Boundary Coverage ignores the presence of uncertainties in the coverage mission. In this approach, the robots assume that the events are all of the same importance value and the boundaries are homogeneous. We suggest two algorithms for *Uninformed Boundary Coverage*: (1) the *Cyclic Boundary Coverage* and (2) the *Cluster-based Boundary Coverage* algorithms.

A. Cyclic Boundary Coverage

In *Cyclic Boundary Coverage (Algorithm 2)*, a tour is constructed on the *Boundary Graph* using the *Chained Lin-Kernighan* algorithm.

Chained Lin-Kernighan (CLK), a modification of the *Lin-Kernighan* algorithm [14], is generally considered to be one of the best heuristic methods for generating *optimal* or *near-optimal* solutions for the *Euclidean Traveling Salesman Problem* [6]. Given the distance between each pair of a finite number of nodes in a *complete* graph, the *Travelling Salesman Problem (TSP)* is to find the shortest tour passing through all the nodes exactly once and returning to the starting node [5].

This *Lin-Kernighan* algorithm, a *local search* algorithm [11], is a generalization of the *k-opt* algorithm [7]. A *k-opt* algorithm explores all the *TSP* tours which can be obtained by removing k edges from the original tour and adding k different edges, such that the resulting tour is feasible. In order to improve the efficiency, Lin and Kernighan introduce a variable *k-opt* algorithm, which adaptively decides at each iteration what value of k to use [14]. Given the computation time limit, the process is repeated by generating new initial tours and applying the *Lin-Kernighan* algorithm to possibly find a tour shorter than the best one thus far. Martin *et. al* [15], [16] suggest that instead of repeatedly starting from new tours, which is inefficient, the alternative is to perturb the *Lin-Kernighan* tour, and then reapply the algorithm. If this leads to a shorter tour, then discard the old tour, and start with the new one. Otherwise, continue with the old tour and perturb it again.

The input of the *Chained Lin-Kernighan* algorithm needs to be a complete graph. To this end, the *Boundary Graph* is made complete *(line 2)* by adding edges from the original *VG* graph, when there does not exist an edge between two nodes in the *Boundary Graph*. If there is not an edge between the two nodes in the original graph either, a *virtual edge* is added to the *Boundary Graph* to connect the two nodes. The weights of these edges are set to the length of the shortest path

Algorithm 2: Cyclic Boundary Coverage

```
Input:
  $G_{vis}(V_{vis}, E_{vis})$, where $V_{vis} = SG \bigcup P$  /* VG */
  $SG = \{g_1, g_2, ...., g_m\}$  /* Boundary Guards */
  $P = \{p_1, p_2, ...., p_n\}$  /* Endpoints of Obstacles */
  $G_{boundary}(V_{boundary}, E_{boundary})$ : the Boundary Graph
  $|R|$: Number of Robots

Output:
  A tour, dTour, distributed among the robots, passing
  through all the nodes (Points of Interests) of the
  Boundary Graph

1 begin
2     $CG_{Boundary} \longleftarrow CompleteGraph(G_{boundary}, G_{vis})$
3     $tour \longleftarrow BuildTour(CG_{Boundary}, CLK)$
4     $dTour \longleftarrow DistributeRobots(tour, |R|)$
5     return $dTour$
6 end
```

between the two nodes in the original *VG* graph. The *Chained Lin-Kernighan* algorithm then finds the shortest tour passing through all the nodes of the *Boundary graph*, returning to the start node *(line 3)*. The robots are then distributed equidistantly along the tour *(line 4)* and move repeatedly around it in the same direction.

B. Cluster-based Boundary Coverage

The *Cluster-based Boundary Coverage* algorithm (*Algorithm 3)*, uses the *k-Means* clustering algorithm to divide the guards into $|R|$ disjoint clusters. The initial centroids are found as follows: the endpoints of the longest path in the original *VG* graph are selected as the starting points of the first two centroids, such that the endpoints belong to the set of guards, *SG*. For the next centroid, a guard in *SG* is selected such that it maximizes the minimum distance from the starting points of the first two centroids. Similarly, for the next centroid, a guard is selected that maximizes the minimum distance from the starting points of the other three centroids. This continues until $|R|$ initial centroids are found for the $|R|$ clusters of the guards. In the next iterations, since the computed centroids may not lie on the nodes of the *Boundary Graph*, they are matched to the closest guard in the environment *(line 2)*.

Distance from the centroids is determined based on the distance in the original *VG* graph rather than the Euclidean distance. Having built the $|R|$ clusters on the guards *(line 3)*, we connect each pair of guards in each cluster if they have a corresponding edge in the *Boundary Graph (line 5)*. Thereafter, we do a connectivity test on all the clusters, meaning that each pair of guards in each cluster should be connected through a path. For this purpose, we first find the disconnected components within the cluster *(line 6)* and then

Algorithm 3: Cluster-based Boundary Coverage

```
Input:
$G_{vis}(V_{vis}, E_{vis})$, where $V_{vis} = SG \bigcup P$  /* VG */
$SG = \{g_1, g_2, ...., g_m\}$  /* Boundary Guards */
$P = \{p_1, p_2, ...., p_n\}$  /* Endpoints of Obstacles */
$G_{boundary}(V_{boundary}, E_{boundary})$ : the Boundary Graph
$|R|$: Number of Robots

Output:
A set of $|R|$ tours, $Tours = \{T_1, T_2, \ldots, T_{|R|}\}$ where
$\bigcup_{i=1}^{|R|} V_{T_i} = SG$, $SG$ is the set of guards of the Boundary
Graph and $V_{T_i}$ is the set of guards of the tour $T_i$

1  begin
2      $initialCentroids \longleftarrow FindInitialCentroids(G_{vis}, |R|)$
3      $Tours \longleftarrow kMeans(G_{vis}, |R|, initialCentroids)$
4      foreach $T_i \in Tours$ do
5          $ConnectGuards(T_i, G_{boundary})$
6          $disconnectedComponents \longleftarrow$
           $FindDisconnectedComponents(T_i)$
7          $MST \longleftarrow$
           $BuildMST(G_{vis}, disconnectedComponents)$
8          $T_i \longleftarrow T_i + MST$
9          $T_i \longleftarrow BuildTour(T_i, CLK)$
10     end
11     return $Tours$
12 end
```

compute a *Minimum Spanning Tree* on them based on the edges of the original *VG* graph *(line 7)*. Finally, we add the *Minimum Spanning Tree*'s corresponding edges and nodes to the cluster *(line 8)*, and the *Chained Lin-Kernighan* algorithm is used to build a tour on it *(line 9)*. The tour is then assigned to a robot, and the robot repeatedly traverses the tour.

V. Informed Boundary Coverage

Informed Boundary Coverage is primarily based on the *Markov Decision Process (MDP)* formalism in which the robots try to cooperatively maximize the reward by detecting the maximum number of events in the minimum time, considering the importance value of the events. To this end, the robots *learn* the expected reward of visiting a state in the target area at each time step, and based on that, they *plan* to select the best possible path to visit the most promising state at the time in the area.

The algorithm starts by decomposing the *Boundary Graph* into as many clusters as there are robots in the environment using the *k-Means* algorithm, similar to the process discussed before in Section IV-B. Each robot then traverses the cluster assigned to it according to the policy being learned. In this approach, the robots do not need to communicate about every one of the guards they visit. They only update each other about the guards with one or more *shared segments*, and those *shared segments* belong to more than one robot. When a robot, R_i, visits a guard having a *segment* in common with another guard assigned to robot R_j, it notifies R_j about the visit and the events detected in the *shared segment*.

Definition 5. *Time of Last Visit (TLV):* Each robot keeps track of the time of the last visit to its guards, and to the guards of the other robots with some *segments* shared with one of the robot's guards. If $\{g_1, g_2, ..., g_p\}$ is the set of guards monitored by a robot, then for each g_i, TLV_{g_i} represents the last time the guard g_i was visited.

Assumption 2. *The robots are aware of the types of the events occurring on the boundaries and their importance weights.*

The *Multi-Robot Repeated Boundary Coverage* problem is formulated as a tuple *(S,A,ST,STR)* where:

- $S = SG \times TLV$ is the set of states, where SG is the set of guards and TLV is the set of last visits to the guards.
- A is the set of actions available for a robot in each state. An action is defined as moving from one guard to another. So in each state, the robot might have one or more actions available.
- ST is the state transition function which is deterministic, that is, it guarantees reaching the target state chosen by the robot when the action is performed.
- STR is the state reward, which is equal to the sum of the discounted importance of the detected events at the state. If $t(e)$ is the time interval between starting event e and the detection time, the STR is formulated as:

  $$STR(g) = \sum_{seg_g \in VA_g} \sum_{E_i \in E} \sum_{e_i \in E_i} weight(e_i) \times \gamma^{t(e_i)}$$

  Once a robot arrives at a guard g, it can detect all the events occurring within the VA_g, the *visual area* of the guard g. If a robot were aware of the starting time of the events, it would receive the *STR*. It is assumed that the reward a robot receives for an event depends on how early the event is detected. At each time step after the event occurrence, the detection reward of the event is multiplied by a discount factor of $\gamma = 0.95$.

Definition 6. *Policy:* A policy, $\pi : S \rightarrow A$ at each state determines what action should be performed next by the robot.

A. Learning

If the robots had knowledge of the probability of occurrence of the different events in each state as well as the starting time of the events, they would be able to utilize the *STR* to find a policy which maximizes the total reward of the boundary coverage mission. But since this information is not available to the robots, the *STR* is estimated by the sum of the *Expected*

Segment Reward (ESR) of the *segments* comprising the state:

$$STR(g) \simeq \sum_{seg_g \in VA_g}(ESR(seg_g, t))$$

Expected Segment Reward (ESR) is defined to represent the expected reward of seg_g at time t. The *ESR* can be calculated using the sum of the discounted importance of the events occurred between the last visit, TLV_g, and the current visit time, t, to the *segment's* corresponding guard, g:

$$ESR(seg_g, t) = \sum_{E_i \in E} \sum_{e_i \in E_i} (1 + \gamma^1 + \gamma^2 + ... + \gamma^{t-TLV_g}) \times PSE(e_i, seg_g) \times weight(E_i).$$

where γ is the reward discount factor. We assume that for every time step after an event occurs without being detected, the event detection reward is discounted by γ. Furthermore, the *Probability of Segment Event (PSE)* is defined for each event type $E_i \in E$ and each *segment* seg_g, $g \in SG$, to indicate the probability of events of type E_i occurring within the seg_g at each time step.

The *Expected Segment Reward* can be reformulated as below:

$$ESR(seg_g, t) = (1 + \gamma^1 + \gamma^2 + ... + \gamma^{t-TLV_g}) \times \sum_{E_i \in E} \sum_{e_i \in E_i} PSE(e_i, seg_g) \times weight(E_i)$$

In the above formula, $\sum_{E_i \in E} \sum_{e_i \in E_i} PSE(e_i, seg_g) \times weight(E_i)$ is called the *Potential Segment Reward (PSR)*, indicating the potential reward of all the events at seg_g, per time step, and is represented by $PSR(seg_g)$. If a robot knows the *PSR* of the events at each *segment* of the *visual area* of the guards, it can calculate the *ESR* for any arbitrary time t.

To this end, a learning procedure for estimating the *PSR* gradually updates its initial value. In the initialization step, we assume that all the events have the same probability of occurrence at each *segment*. Therefore, all the *PSR*s are initialized to 1. When a robot arrives at a guard g, it can detect whether an event has occurred at any of the *segments* belonging to VA_g. If there is at least one event occurring in the seg_g, the $PSR(seg_g)$ is updated using the following formula:

$$PSR(seg_g) = (1 - \alpha) \times PSR(seg_g) + \alpha \times \frac{\sum_{E_i \in E} \sum_{e_i \in E_i}(weight(e_i))}{t - TLV_g}$$

where α is the learning rate set to 0.9 and t is the time of the visit to g. This formula gives more weight to the new information than that given to the past information. The robot performs the updating process for all the event types and all the *segments* of the guards.

On the contrary, if no event occurs during the time period between the last visit, TLV_g, and the current visit time, t, the *PSR* is updated to reflect the new fact. To this end, the current value of the *PSR* is multiplied by the discount factor of $\beta = 0.9$:

$$PSR(seg_g) = PSR(seg_g) \times \beta$$

This helps to gradually discard the effects of the *segments* in which events occur rarely.

In summary, the *PSR* is updated once a robot visits a guard. As already mentioned, the *PSR* represents the potential reward of each *segment* per time step. Now, we can use the *PSR* to calculate the *ESR* using the following procedure:

At the beginning, the *ESR*s of all the *segments* are initialized to zero. Then, at each time step, if the robot has yet to arrive at a guard, the value of *ESR* is updated using the following equation:

$$ESR(seg_g, t) = \gamma \times ESR(seg_g, t-1) + PSR(seg_g), \ seg_g \in VA_g \ and \ g \in SG$$

If the robot arrives at g, it detects all the events and consequently:

$$\forall seg_g \in VA_g, \ ESR(seg_g, t) = 0.$$

This updating process continues during the boundary coverage operation.

B. Planning

Once a robot arrives at a guard and detects all the events which might have occurred at the *segments* of the guard, it selects the next action to perform. As already mentioned, the action in the boundary coverage operation is defined as moving from one guard to another. At each state, the robot considers all the *shortest paths* to all the other guards it can move to. Note that, since we divide the *Boundary Graph* among the robots, each robot can only move to the guards assigned to it. For each path, $path(g_i, g_j)$, where g_i is the current guard and g_j is the target guard, *Path Reward (PR)* is defined as the reward the robot receives when moving from the guard g_i to the guard g_j. The path from g_i to g_j includes zero or more intermediate guards and can be represented as:

$$path(g_i, g_j) = [g_i, g_{i+1}, g_{i+2}, ..., g_{j-1}, g_j].$$

Given the speed of the robot, the arrival time at each of the guards on the path can be estimated. Hence, the robot can have an estimate of the $ESR(seg_g, t_g)$ for each *segment* of the guard g, in which t_g is the arrival time to the guard g. For such a path, the *PR* is calculated as below:

$$PR(path(g_i,g_j)) = \sum_{g \in path(g_i,g_j)} \sum_{seg_g \in VA_g} ESR(seg_g,t_g).$$

When calculating the *PR*, the robot should take into account the visits to the *shared segment* by the other robots. Moreover, as long as a robot does not receive a message from another robot regarding a visit to a *shared segment*, the robot assumes that the *shared segment* has not already been visited and will not be visited by any other robot.

Next, for each path, the *Average Path Reward (APR)* is calculated using the following formula:

$$APR(path(g_i,g_j)) = \frac{PR(path(g_i,g_j))}{t_{g_j} - TLV_{g_i}}$$

where g_j is the target guard on the path, and t_{g_j} is the arrival time to the guard g_j. The robot will select a path with the maximum *Average Path Reward* to traverse next.

VI. Proposed Experiments

We have developed a simulator to test the algorithms in different scenarios. The simulator can support different numbers of robots in the workspace, different visual ranges for the robots, and varying degrees of clutter in the environment. A random map generator was also developed as a part of the simulator which extends a library [18] to build rectilinear or non-rectilinear polygons with free form polygonal obstacles within the space. Maps can have different numbers of nodes and percentages of clutter.

We aim to compare the *Informed Boundary Coverage* with the two variants of the *Uninformed Boundary Coverage* algorithm in terms of the total reward received by the team for detecting the events. To this end, we consider three types of environments in the experiments: sparse ($0-25\%$ cluttered), semi-cluttered ($25-50\%$ cluttered), cluttered ($50-75\%$ cluttered). Ten different maps are used in the experiments for each of the three environment types (30 in total). The clutter percentage of an environment is the ratio of the area of the obstacles to the whole target area (*i.e.* obstacles + free space). The experiments are conducted using $1,2,3,\ldots,15$ robots. 5 different event types are also used in the experiments. Finally, the effect of change in the robots' visual range on the performance of the boundary coverage algorithms is investigated.

VII. Future Work

In the *Informed Boundary Coverage* approach, we considered a static decomposition of the *Boundary Graph* among the robots. However, this is not always a reasonable approach, as it is possible that all the events are generated in just one cluster, and so just one robot will be in charge of handling all the events. In other words, for the other robots, there are no events occurring in their clusters. To cope with this issue, we are investigating a dynamic approach to decompose the *Boundary Graph* among the robots, considering the location and the weight of the events occurring on the boundary.

References

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

[2] N. Agmon, S. Kraus, G. A. Kaminka, and V. Sadov, "Adversarial uncertainty in multi-robot patrol," in *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2009, pp. 1811–1817.

[3] 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, SBIA*, 2004, pp. 474–483.

[4] P. Amstutz, N. Correll, and A. Martinoli, "Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm," *Annals Mathematics Artificial Intelligence*, vol. 52, no. 2-4, pp. 307–333, 2008.

[5] D. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007.

[6] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *INFORMS Journal on Computing*, vol. 15, pp. 82–92, January 2003.

[7] B. Chandra, H. Karloff, and C. Tovey, "New results on the old k-opt algorithm for the traveling salesman problem," *SIAM Journal on Computing*, vol. 28, pp. 1998–2029, August 1999.

[8] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[9] P. Fazli, A. Davoodi, P. Pasquier, and A. K. Mackworth, "Multi-robot area coverage with limited visibility," in *Proceedings of The International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2010, pp. 1501–1502.

[10] ——, "Complete and robust cooperative robot area coverage with limited range," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2010, pp. 5577–5582.

[11] H. Hoos and T. Sttzle, *Stochastic Local Search: Foundations & Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[12] G. D. Kazazakis and A. A. Argyros, "Fast positioning of limited visibility guards for inspection of 2D workspaces," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2002, pp. 2843–2848.

[13] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[14] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[15] O. Martin, S. Otto, and E. Felten, "Large-step markov chains for the traveling salesman problem," *Complex Systems*, vol. 5, no. 3, pp. 299–326, 1991.

[16] O. Martin, S. W. Otto, and E. W. Felten, "Large-step markov chains for the TSP incorporating local search heuristics," *Operations Research Letters*, vol. 11, pp. 219–224, 1992.

[17] J. O'Rourke, *Art gallery theorems and algorithms*. New York, NY, USA: Oxford University Press, Inc., 1987.

[18] A. P. Tomás and A. L. Bajuelos, "Quadratic-time linear-space algorithms for generating orthogonal polygons with a given number of vertices," in *Proceedings of the International Conference on Computational Science and Its Applications, ICCSA*, 2004, pp. 117–126.

[19] K. Williams and J. Burdick, "Multi-robot boundary coverage with plan revision," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, 2006, pp. 1716–1723.

[20] B. Zalik and G. J. Clapworthy, "A universal trapezoidation algorithm for planar polygons," *Computers & Graphics*, vol. 23, no. 3, pp. 353 – 363, 1999.