# Robotic Adversarial Coverage: Introduction and Preliminary Results

Roi Yehoshua, Noa Agmon and Gal A. Kaminka
Computer Science Department
Bar Ilan University, Israel
roiyeho@gmail.com, agmon@cs.biu.ac.il, galk@cs.biu.ac.il

*Abstract*— **This paper discusses the problem of generating efficient coverage paths for a mobile robot in an adversarial environment, where threats exist that might stop the robot. First, we formally define the problem of adversarial coverage, and present optimization criteria used for evaluation of coverage algorithms in adversarial environments. We then present a coverage area planning algorithm based on a map of the probable threats. The algorithm tries to minimize the total risk involved in covering the target area while taking into account coverage time constrains. The algorithm is based on incrementally extending the coverage path to the nearest safe cells while allowing the robot to repeat its steps. By allowing the robot to visit each cell in the target area more than once, the accumulated risk can be reduced at the expense of extending the coverage time. We show the effectiveness of this algorithm in extensive experiments.**

## I. INTRODUCTION

The general problem of covering an area by single or multi robot systems is a fundamental problem in robotics. Besides its theoretical interest, it has important applications in various domains, from automatic floor cleaning and coating in facilities, such as supermarkets [4] and train stations [9], to humanitarian missions such as search and rescue and field demining [8].

The optimal coverage problem can be formulated as a generalization of the Traveling Salesperson Problem (TSP) for a continuous domain, and thus is $\mathcal{NP}$-hard [1]. It is therefore natural to seek efficient algorithms for the coverage problem.

Previous efforts on the coverage problem have focused mainly on achieving a covering path with minimal coverage time under various conditions and assumptions (see Choset [2] for a comprehensive survey). However, all these works were intended for non-adversarial settings, where nothing, outside of the environment itself, is hindering the robot's task. Often, however, robots and autonomous vehicles need to perform coverage missions in hazardous environments, such as operations in nuclear power plants, exploration of Mars, and surveillance of enemy forces in the battle field. Even domestic coverage missions such as floor cleaning might pose various risks to the robot's safety.

Hence, our work addresses the problem of planning for a robot whose task is to cover a given terrain without being detected or damaged by an adversary. Each point in the area is associated with a probability of harming the robot at that point and the probabilities can vary from one point to another. The objective of the robot is to complete the given mission—to cover the specified area—as quickly as possible while maximizing its own safety. We will refer to this problem as the general *adversarial coverage problem*. In this paper we discuss the offline version of this problem, in which the map of threats is given in advance, therefore the coverage path of the robot can be determined prior to the execution of the coverage algorithm.

In the following sections we formally define the offline adversarial coverage problem and propose an algorithm for solving it heuristically. The algorithm tries to optimize both the survivability of the robot and the coverage time given some predefined objective function. The algorithm we propose has a polynomial time complexity in the number of locations to be covered. We evaluate this algorithm in systematic experiments and show its effectiveness.

## II. RELATED WORK

The problem of single and multi-robot coverage has been extensively discussed in the literature and many approaches to coverage path planning have been developed.

In the single robot case, an optimal coverage path means finding a non-redundant path, i.e., a path that visits each cell in the target area exactly once. Gabriely and Rimon [5] have introduced the Spanning Tree Coverage (STC) algorithm that provides optimal coverage path in a uniform grid based terrain. Their algorithm subdivides the area into disjoint cells and then follows a spanning tree of the graph induced by the cells, while covering every point precisely once.

The idea was broadened for a multi-robot system by Hazon and Kaminka in [6] in the family of Multi Robot Spanning Tree Coverage (MSTC) algorithms. Their solution, along with decreasing the total coverage time, achieved robustness in the sense that as long as one robot works properly, the coverage of the terrain is guaranteed. They have also shown that in multi robot teams redundancy might be necessary for more efficiency. We also accept redundant coverage, but for safety.

Other related robotic tasks in adversarial environments have been actively studied in the robotics community. For example, the problem of path planning in uncertain and adversarial environments has been investigated in [7] and [10]. The patrol problem, where a multi-robot team needs to patrol around a closed area with the existence of an adversary attempting to penetrate into the area, has been discussed in [3]. The patrol problem resembles the coverage problem in the sense that both require the robot or group of robots to visit all points in the given terrain. However, while coverage seeks to minimize the number of visits to each point (ideally, visiting it only once), patrolling seeks to maximize it (while still visiting all points).

The adversarial coverage problem discussed in this paper is unique in the sense that the adversarial nature of the environment is expressed in the possibility of physically harming the robot (rather than harming the utility of a robotic team, as in [6]). It has an intrinsic complexity that is not typical to the other problems mentioned, since it presents a delicate tradeoff between minimizing the accumulated risk and minimizing the total coverage time. Trying to minimize the risk involved in the coverage path means making some redundant steps, which in turn can make the coverage path longer, and thus increase the risk involved, as well as increase the coverage time.

## III. The Adversarial Coverage Problem Formulation

The adversarial coverage problem can be defined as follows. We are given a target area $T$. The robot's task is to plan a path through $T$ such that every point in $T$ is visited by the robot at least once. However, some points in $T$ have probability of containing threats, which may stop the robot. Thus the robot must plan its coverage path such that it takes into account the likelihood of completing it (i.e., its survivability).

In particular, given $T$, three questions may be asked:

1) What is the minimum coverage time for $T$, and at what survivability?
2) What is the maximum survivability for $T$, and at what coverage time?
3) Given specified levels of survivability and coverage time, what is the optimal coverage path?

In this paper we focus on a specific instantiation of adversarial coverage. We assume that the target region $T$ is decomposed into a regular square grid with $n$ cells, whose size equals the size of the robot. We also restrict the robot to move only in four compass directions. For each cell $i$, we are given an associated threat probability $p_i$, which measures the likelihood that a threat is present in a point within the cell $i$. If a cell contains a threat and the robot visits this cell, then it stops and cannot continue its path.

Now let us denote the coverage path followed by the robot by $A = (a_1, a_2, ..., a_m)$. Note that $m \geq n$, i.e., the number of cells in the coverage path might be greater than the number of cells in the target area, since the robot is allowed to repeat its steps. We define the event $S_A$ as the event that the robot is neutralized when it follows the path $A = (a_1, a_2, ..., a_m)$. Denote the complement of event $E$ by $\overline{E}$.

The probability $P(\overline{S_A})$ that the robot is able to complete its coverage path $A = (a_1, ..., a_m)$ can be expressed as:

$$P(\overline{S_A}) = \prod_{i \in (a_1, \ldots, a_m)} (1 - p_i) \tag{1}$$

Now, given a coverage path $A$, let us denote the number of unexplored cells visited by the robot until it is neutralized by $C_A$. We will define the **survivability** of the robot as $E(C_A)$, i.e., as the expected number of unexplored cells that it visits while following the coverage path $A$. The term expected coverage will be used interchangeably with survivability throughout this paper.

To compute $E(C_A)$, we need to consider the sequence of unexplored cells discovered along the coverage path $A$. Let us denote this sequence by $(b_1, ..., b_n)$. Note that the total number of cells in this sequence is exactly the number of cells in the grid ($n$). For each cell in the sequence $b_i$, we will denote the sub-path in $A$ that leads from the origin cell $a_1$ to it by $g_i$. Then, under the threat probability function $p$, the survivability of the robot following the coverage path $A$ can be expressed as:

$$E(C_A) = \sum_{i \in (b_1, \ldots, b_n)} \prod_{j \in g_i} (1 - p_j) \tag{2}$$

For example, let us consider the following simple grid, which is composed of 4 cells: $a_{11}, a_{12}, a_{21}$ and $a_{22}$, with the probabilities for danger $p_{ij}$ specified in each cell.

| | |
|---|---|
| 0 | 0.1 |
| 0.2 | 0.5 |

Assume that the initial location of the robot is in cell $a_{11}$. Then the coverage path with minimal risk in this case is $A = (a_{11}, a_{12}, a_{11}, a_{21}, a_{22})$. The sequence of unexplored cells discovered along this path is $(a_{11}, a_{12}, a_{21}, a_{22})$. Thus, the survivability of the robot following path $A$ is:

$$\begin{aligned} E(C_A) &= 1 + 1 \cdot 0.9 + 1 \cdot 0.9 \cdot 1 \cdot 0.8 \\ &\quad +1 \cdot 0.9 \cdot 1 \cdot 0.8 \cdot 0.5 \\ &= 1 + 0.9 + 0.72 + 0.36 = 2.98 \end{aligned}$$

However, the shortest coverage paths are $A_1 = (a_{11}, a_{12}, a_{22}, a_{21})$ and $A_2 = (a_{11}, a_{21}, a_{22}, a_{12})$. Clearly, path $A_1$ is safer than $A_2$. The survivability of the robot following path $A_1$ is:

$$\begin{aligned} E(C_{A_1}) &= 1 + 1 \cdot 0.9 + 1 \cdot 0.9 \cdot 0.5 + 1 \cdot 0.9 \cdot 0.5 \cdot 0.8 \\ &= 1 + 0.9 + 0.45 + 0.36 = 2.71 \end{aligned}$$

Therefore, by making one additional step, the robot is able to raise its expected number of covered cells from 2.71 to 2.98.

In order to help us answer the questions raised in the beginning of this section, we will define the following weighted cost function that takes both the survivability and the coverage time factors into consideration. For a given coverage path $A$, define

$$f(A) = -\alpha \cdot E(C_A) + \beta \cdot |A| \tag{3}$$

where $\alpha, \beta \geq 0$, and $|A|$ is the number of the steps the robot needs to take in order to complete the coverage path.[1] Then the problem is to find a coverage path $A$ that minimizes the cost function $f(A)$, i.e., $f(A) \leq f(B)$ for all coverage paths $B$.

[1]For now, we treat all steps, in any location, to have uniform cost, and coverage time is measured directly by the number of steps.

When $\alpha = 0$, the objective translates to finding a minimum time coverage path regardless of the risk involved. Achieving this objective will provide an answer to our first question. When $\beta = 0$, the objective translates to finding a coverage path with a minimal risk, without a limit on the path length. Achieving this objective will provide an answer to our second question. Lastly, setting fixed levels for $\alpha$ and $\beta$ will provide an answer to our third question.

## IV. Adversarial Coverage Path Algorithm

In this section we describe an adversarial coverage path construction algorithm, Build_Coverage_Path. The algorithm also creates a spanning tree, connecting all the cells in the given area, which serves as the skeleton of the coverage path. The tree is created gradually starting from the initial position of the robot, such that in each cycle one new cell is added to the tree. The cell is chosen in a way that minimizes the probability of the robot being damaged along the path $(s, ..., t)$ leading from the robot's current position $s$ to the new cell's position $t$, while keeping it as shortest as possible (taking both $\alpha$ and $\beta$ terms defined in the objective function into consideration).

For that purpose, we use Dijkstra's algorithm for finding a minimal path between the robot's current position and all the possible neighbors of the cells the robot has already visited until now. The following subsection describes how to define a suitable edge weighting that will represent both the risk involved in traversing each edge and the time cost of making the transition.

### A. *Defining Graph Weights*

In each cycle of the algorithm, we need to find a path $(s, ..., t)$ from the robot's current position to a new cell in the grid that minimizes the risk of visiting each cell along the path, i.e., we need to find a path from cell $s$ to cell $t$ such that

$$\prod_{i\in(s,a_2,...,t)} (1-p_i) \geq \prod_{j\in(s,b_2,...,t)} (1-p_j) \qquad (4)$$

for all paths $(s, b_2, ..., t)$ leading from $s$ to $t$.

We will first show how to convert this problem to a problem of finding a shortest path. Since the logarithm is a monotonically increasing function of its argument, the above expression is equivalent to:

$$\sum_{i\in(s,a_2,...,t)} (-log(1-p_i)) \leq \sum_{j\in(s,b_2,...,t)} (-log(1-p_j)) \quad (5)$$

Now, define a directed graph $G = (V, E)$, where the set of nodes $V$ is the set of cells in the target area. Define $w_{ij}$, the weight of the edge $(v_i, v_j)$, as follows:

$$w_{ij} = \begin{cases} -log(1-p_j) & \text{if cells i and j are adjacent} \\ \infty & \text{otherwise} \end{cases} \qquad (6)$$

By defining the weights of the edges as in Eq. (6), we can convert the minimum risk path problem into the problem of finding a shortest path.

Since we want to consider also the path length, we will include two constant terms in the link weight that are dependant on $\alpha$ and $\beta$ given in the objective function. So the weight of the edge $(v_i, v_j)$ becomes:

$$w_{ij} = \begin{cases} -c_\alpha \cdot \alpha \cdot log(1-p_j) + c_\beta \cdot \beta \\ \text{if cells i and j are adjacent} \\ \infty \qquad \text{otherwise} \end{cases} \qquad (7)$$

The meaning of this formula is that the weight of each edge in the graph becomes lower as the probability to survive traversing this edge is higher (this is the term defined by $-\alpha \cdot log(1-p_j)$) and the cost of making one step in the grid is lower (this is the term defined by $\beta$).

In section V we first determine the terms $c_\alpha$ and $c_\beta$ used by the algorithm. Then we describe extensive simulations of Build_Coverage_Path with our chosen $c_\alpha$ and $c_\beta$.

### B. *Building a Coverage Path*

The pseudocode for the algorithm is given in the following procedure.

```
Algorithm Build_Coverage_Path($G$)
Input: Graph $G = (V, E)$, where $V$ is the set of nodes representing the cells in the grid and $E$ is the set of edges connecting each cell in the grid to its neighbors with the weight as defined in formula (7).
 1) Let $T \leftarrow \emptyset$ be the spanning tree
 2) Let $\pi \leftarrow \emptyset$ be the coverage path
 3) Let $H \leftarrow \emptyset$ be the subgraph of $G$ that contains all the neighbors of nodes in $\pi$
 4) Let $s$ be the node representing the initial location of the robot
 5) Add node $s$ to $T$, $\pi$ and $H$
 6) Mark $s$ as visited
 7) While not all the cells in the grid are covered
      a) Add the neighbors of $s$ that are marked as not visited to $H$
      b) Compute Dijkstra (shortest path) from $s$ to each node in the subgraph $H$
      c) Let $v$ be the node in $H$ with the shortest distance from $s$ that was not visited yet
      d) Add $v$ and the edge connecting it to the tree to $T$
      e) Add the path $s \rightsquigarrow v$ to $\pi$
      f) Mark $v$ as visited
      g) $s \leftarrow v$
 8) return $T$ and $\pi$
```

The idea behind the algorithm is that it gradually builds a spanning tree of uncovered cells that it discovers, while keeping track of the path used to discover those cells. The spanning tree is built by a depth-first-like procedure: Scan for uncovered neighboring cells (Line 7a), find a shortest path from the robot's current position to one of those cells (Lines 7b-c), build a tree edge to it (Line 7d) and continue with this cell (Line 7g).

The time complexity of Build_Coverage_Path algorithm is $O(V^2 lgV)$, since the graph is sparse ($|E| = O(|V|)$), thus running Dijkstra (shortest path) on the entire graph takes $O(VlgV)$, and the algorithm runs Dijkstra $|V|$ times.

## V. Evaluation

We evaluate the algorithm empirically, to illustrate the effects of its parameters, and to understand the scope of its success. In subsection *A*, we use a specific, randomly-generated map, to illustrate the operation of the algorithm under various parameter settings. In subsections *B-D* we report on the statistical analysis based on multiple randomly-generated maps with various parameters, such as map size, cost values, etc.

### A. Description of the World

We considered a target area consisting of $20 \times 20$ square cells. For half of the grid cells $p_i = 0$. For the other half, we generated random numbers with a uniform distribution in the range $[0.0, 0.25]$ and assigned them to $p_i$. The origin is cell $(1, 1)$, which is always a safe cell ($p_{11} = 0$).

In the maps we used 5 different shades of purple to represent the following intervals of $p_i$: $[0, 0.05), [0.05, 0.1), [0.1, 0.15), [0.15, 0.2)$ and $[0.2, 0.25)$. Darker shades represent higher values of $p_i$ (more dangerous areas). See Figure 1 for an example of the world.

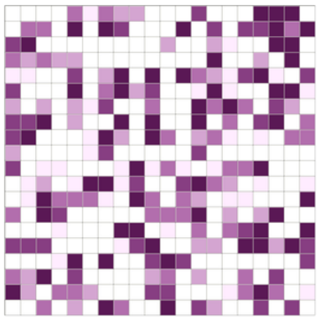

Fig. 1. An example of the world. Darker regions represent more dangerous areas.

First, we wanted to test the algorithm when the coverage risk is not penalized and only the coverage time is taken into account (i.e., $\alpha = 0$). As expected, in this case the robot walks around the area in straight lines until it covers the whole area. Running the algorithm on a randomly selected map resulted in a path with expected coverage of 3.65% and total length of 399 steps. The spanning tree for this path can be seen on the left side of Figure 2.

Next, we wanted to test the algorithm when the coverage time is not penalized and only the coverage risk is taken into account (i.e., $\beta = 0$). In this case, in each cycle the robot is trying to find the safest cell on the board to continue its path from. Running the algorithm on the same map resulted in a path with expected coverage of 39.32% and total length of 2012 steps, i.e., on average the robot visits each cell in the grid about 5 times. The spanning tree for this path in the same world can be seen on the right side of Figure 2.

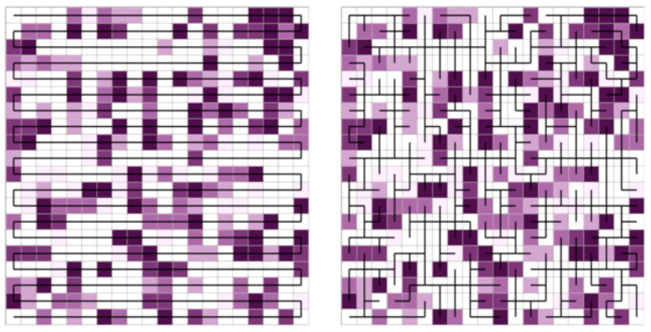

Fig. 2. Left - spanning tree when the risk is not penalized ($\alpha = 0$). Right - spanning tree when the coverage time is not penalized ($\beta = 0$).

### B. Determining $c_\alpha$ and $c_\beta$

Now, we discuss the general case where $\alpha > 0$ and $\beta > 0$. We first need to determine the optimal coefficients $c_\alpha$ and $c_\beta$ for finding a minimal cost value. Let's start with a specific case where $\alpha = 1$ and $\beta = 0.1$, i.e., when the cost function is

$$f(A) = -E(C_A) + 0.1 \cdot |A|$$

Figure 3 shows the cost function value $f(A)$ for different values of $c_\alpha$ and $c_\beta$ in the interval $[0, 1]$. The results are averaged on 30 randomized maps.

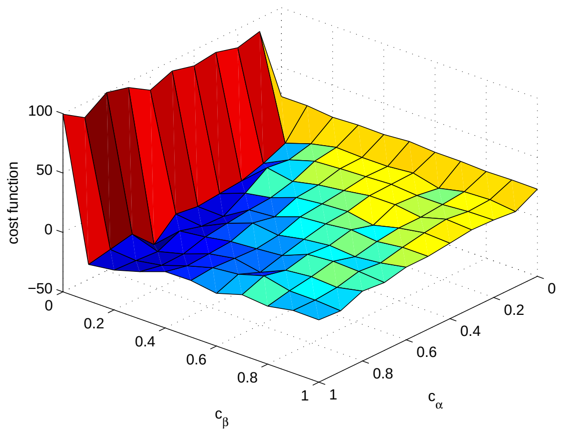

Fig. 3. Cost function values for $\alpha = 1$ and $\beta = 0.1$

Generally speaking, we can see that for a given $c_\alpha$, the cost function decreases as $c_\beta$ increases, until it reaches a minimum value somewhere around $c_\beta \approx 0.2c_\alpha$, and then it increases again.

The global minimum cost value is achieved when $c_\alpha = 0.7$ and $c_\beta = 0.1$. For these coefficients, the robot achieves an expected coverage percentage of 28.21% while keeping the total coverage length at 836 steps on average, i.e., by visiting each cell about twice than it should have, the robot is able to increase its expected coverage percentage from 3.65% to 28.21%.

The spanning tree generated for the example world using the coefficients $c_\alpha = 0.7$ and $c_\beta = 0.1$ is shown on the left side of Figure 4. In this case the expected coverage percentage was $35.06\%$ and the total path length was $885$.

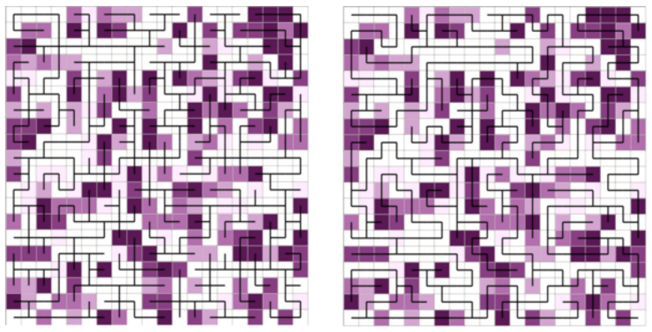

Fig. 4. Left - spanning tree for $\alpha = 1$ and $\beta = 0.1$. Right - spanning tree for $\alpha = 0.1$ and $\beta = 0.1$.

### C. *Experimental Results,* $c_\alpha = 0.7, c_\beta = 0.1$

Next, we wanted to test the algorithm's performance on different values of $\alpha$ and $\beta$. We used $\alpha$ values in the range [0, 1] and $\beta$ values in the range [0, 0.2]. In all the experiments, we set the coefficient values to $c_\alpha = 0.7, c_\beta = 0.1$. We repeated the experiments for 30 randomized maps.

Figure 5 shows the expected coverage percentage for different values of $\alpha$ and $\beta$. As expected, for a given level of $\alpha$, as the value of $\beta$ gets higher, the robot is willing to take more risky moves in order to shorten the coverage path, thus the expected coverage percentage is lower. Conversely, for a given level of $\beta$, as the value of $\alpha$ gets higher, the robot is taking more cautious steps, thus the expected coverage percentage is greater. For a fixed ratio between $\alpha$ and $\beta$ we get similar coverage percentages.

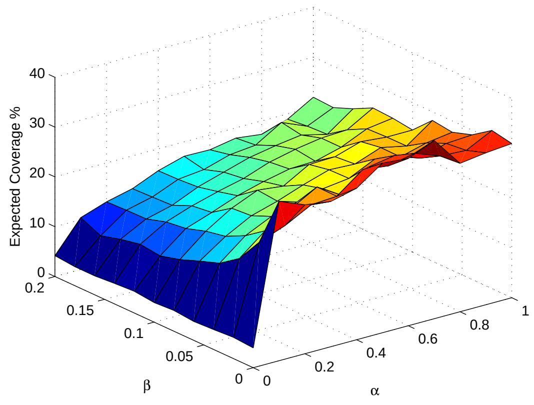

Fig. 5. Expected coverage percentage for different values of $\alpha$ and $\beta$

Figure 6 shows the coverage path length for different values of $\alpha$ and $\beta$. For a given level of $\alpha$, as the value of $\beta$ gets higher, the time factor gets more significance and thus the coverage path becomes shorter at the expense of increasing the risk. Conversely, for a given level of $\beta$, as the value of $\alpha$ gets higher, the risk factor gets more significance and thus the robot makes more redundant steps in order to reduce the risk involved, which makes the coverage path longer.

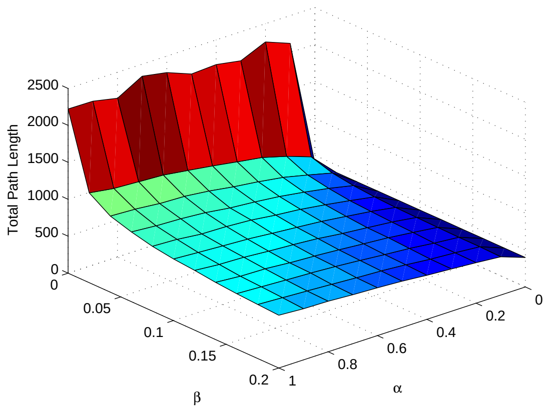

Fig. 6. Total path length for different values of $\alpha$ and $\beta$

Examining the spanning trees for various $\alpha$ and $\beta$ values supports this observation. On the left side of Figure 4 we have seen the spanning tree for the example world when $\alpha = 1$ and $\beta = 0.1$. The right side of the same figure shows the spanning tree for the same map when $\alpha = 0.1$ and $\beta = 0.1$. In this case, the expected coverage percentage was $6.5\%$ and the total path length was $550$.

As can be seen, the tree on the left side contains more twists and turns than the tree on the right side. For example, the area with safe cells in the upper center region (including the cells in rows 3 and 4 from column 6 to column 13) is contiguous in the spanning tree for $\alpha = 0.1$, as the robot is willing to move through the dangerous cell $(3, 11)$, while the same area is split in the spanning tree for $\alpha = 1$, as the robot is trying to avoid cell $(3, 11)$ in its first pass through this area.

### D. *Testing on different terrains*

Next, we wanted to test the behavior of the algorithm under various terrain conditions. We start by examining the effect of the safe cells ratio on the coverage paths found by the algorithm (in previous experiments a safe cell ratio of $0.5$ was used). Figure 7 shows the expected coverage percentage and the coverage path length as a function of the safe cells ratio. In all the runs we set $\alpha = 1$ and $\beta = 0.1$ and used the coefficients $c_\alpha = 0.7$ and $c_\beta = 0.1$. As before, the results are averaged on 30 randomized maps.

As can be seen in the figure, the expected coverage percentage graph is monotonically increasing. The increasing slope of the graph is low at the beginning, until the safe cells ratio reaches the level of approximately $0.65$. At that level, the graph starts to behave like a linear graph. This is due to the fact that when there is a high percentage of safe cells in the grid, the robot is able to cover most of the safe cells before attempting to cover the dangerous ones, thus there is a linear match between the coverage percentage and the safe cells percentage above a certain level of safe cells in the grid.

However, the path length graph increases until it reaches a maximum level when the safe cells ratio is approximately $0.65$, and then it starts to decrease. This can be explained by the fact that above a certain level of safe cells in the grid, the robot is able to increase its coverage ability while reducing the number of redundant steps that it makes along the path, and thus the coverage path gets shorter.

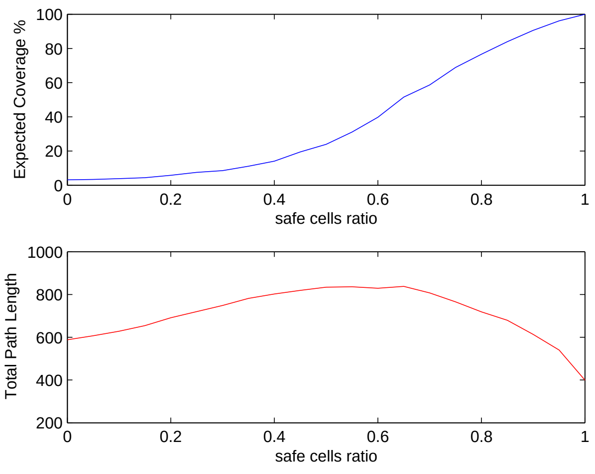

Fig. 7. Expected coverage percentage and coverage path length for different safe cells ratios

Next, we wanted to examine the behavior of the algorithm on various coverage area sizes. We ran the algorithm on various grid sizes in the range between $10 \times 10$ and $40 \times 40$. All the other parameter settings remained the same. Figure 8 shows the expected number of covered cells and the coverage path length as a function of the area size.

For all area sizes, the observed length of the coverage path is approximately twice the number of cells in the grid. The expected number of cells covered by the robot is monotonically increasing until it reaches a plateau around $140$ cells. This number is assumed to be determined by the maximum number of unsafe cells the robot is able to cover without being catastrophically damaged.

## VI. Conclusions and Future Work

In this work we have discussed the adversarial coverage problem and its various aspects. First, we have suggested optimization criteria for the evaluation of coverage algorithms in adversarial environments. These criteria take into account both the survivability of the robot and the total coverage time. Next, we described an algorithm for finding a coverage path that tries to minimize a predefined cost function which is based on these criteria. We have conducted systematic experiments with our implementation in order to measure the algorithm's effectiveness. The results show that the algorithm works well in different environments and world sizes. Lastly, we have examined the coverage paths generated by the algorithm and compared their structures.

There are several areas we plan to pursue in future work. First, we would like to consider other types of terrains, such

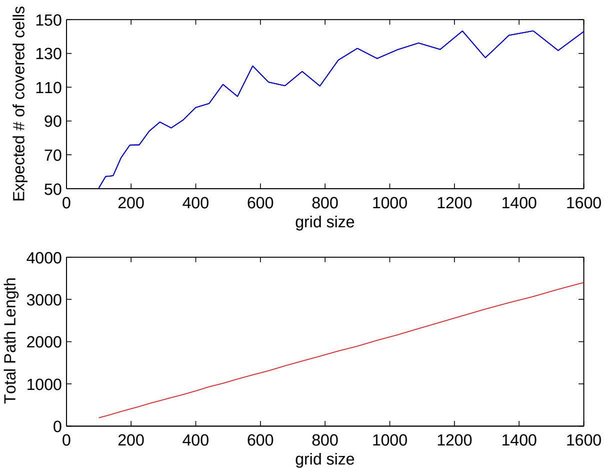

Fig. 8. Expected number of covered cells and coverage path length for different grid sizes

as terrains with obstacles and non-rectangular areas. Second, we would like to investigate theoretically different scenarios of the problem and search for algorithms that can solve it with proven optimality bounds. In addition, we are interested in extending the algorithm to handle online coverage, in which the coverage has to be completed without the use of a map or any a-priori knowledge of the area. Finally, we would like to extend the algorithm for multi-robot systems. Using multiple robots for coverage has the potential for more efficient coverage and greater robustness; even if one robot is totally damaged, others may take over its coverage subtask.

## References

[1] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55:197-218, 1994.

[2] H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113-126, 2001.

[3] Y. Elmaliach, N. Agmon and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 385-390, April 2007.

[4] H. Enders, W. Feiten, and G. Lawitzky. Field test of navigation system: Autonomous cleaning in supermarkets. In *IEEE Int. Conf. on Robotics and Automation*, pages 1779-1781, 1998.

[5] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77-98, 2001.

[6] N. Hazon and G. A. Kaminka. Redundancy, efficiency and robustness in multi-robot coverage. In *Proceedings of the IEEE International Conference on Robotice and Automation(ICRA)*, pages 735-741, 2005.

[7] M. Likhachev, A. Stentz. Goal Directed Navigation with Uncertainty in Adversary Locations. *IEEE/RSJ International Conference on Intelligent Robot Systems (IROS-07)*, pages 4127-4134, 2007.

[8] J. D. Nicoud and M. K. Habib. The pemex autonomous demining robot: Perception and navigation strategies. In *IEEE/RSJ International Conference on Intelligent Robot Systems*, pages 1:419-424, 1995.

[9] H. Yaguchi. Robot introduction to cleaning work in the east japan railway co. *Adv. Robotics*, 10 (4):403-414, 1996.

[10] M. Zabarankin, S. Uryasev and P. Pardalos. Optimal risk path algorithms. Applied Optimization, 66:273-296, 2002.