Frontier-Based RTDP: A New Approach to Solving the Robotic Adversarial Coverage Problem

Roi Yehoshua, Noa Agmon and Gal A. Kaminka *
Computer Science Department
Bar Ilan University, Israel
{yehoshr1,agmon,galk}@cs.biu.ac.il

ABSTRACT

Area coverage is an important problem in robotics, where one or more robots are required to visit all points in a given area. In this paper we consider a recently introduced version of the problem, *adversarial coverage*, in which the covering robot operates in an environment that contains threats that might stop it. The objective is to cover the target area as quickly as possible, while minimizing the probability that the robot will be stopped before completing the coverage. We first model this problem as a Markov Decision Process (MDP), and show that finding an optimal policy of the MDP also provides an optimal solution to this problem. Since the state space of the MDP is exponential in the size of the target area's map, we use real-time dynamic programming (RTDP), a well-known heuristic search algorithm for solving MDPs with large state spaces. Although RTDP achieves faster convergence than value iteration on this problem, practically it cannot handle maps with sizes larger than 7×7. Hence, we introduce the use of frontiers, states that separate the covered regions in the search space from those uncovered, into RTDP. Frontier-Based RTDP (FBRTDP) converges orders of magnitude faster than RTDP, and obtains significant improvement over the state-of-the-art solution for the adversarial coverage problem.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Algorithms, Theory

Keywords

Mobile robot coverage, adversarial coverage, motion and path planning, Markov decision process, real-time dynamic programming

*Supported in part by ISF Grant #1511/12. Thanks to K. Ushi.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

1. INTRODUCTION

There are many real-life applications that require a robot, or a group of robots, to cover an area. For example, a vacuum cleaning robot that needs to clean an entire room [5], intrusion detection, mine cleaning [11] and search-and-rescue missions.

Most previous studies of the coverage problem dealt with non-adversarial settings, where nothing in the environment is hindering the robot's task. However, on many occasions, robots and autonomous agents need to perform coverage missions in hazardous environments, such as operations in nuclear power plants, exploration of Mars, demining and search and rescue in the battlefield.

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, as introduced in [15]. Each point in the area is associated with a probability of the robot being stopped at that point. The objective of the robot is to cover the *entire* target area as quickly as possible while maximizing its own safety. We will refer to this problem as the *adversarial coverage problem*. Here 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 its movement.

In this paper, we show how the adversarial coverage problem can be modeled as a Markov Decision Process (MDP). MDPs are a natural choice for implementing a solution to this problem because they explicitly represent costs and uncertainty in results of actions, as well as doing lookahead to examine the potential consequences of sequences of actions. We show that, given an appropriate definition of the MDP, finding an optimal policy for the model also provides an optimal solution to the adversarial coverage problem.

Since the state space of the MDP is exponential in the size of the target area's map, using classical (synchronous) value iteration techniques to find the optimal policy is possible only for small-sized maps. Therefore, in order to handle larger maps, we use real-time dynamic programming (RTDP), which is a well-known heuristic search algorithm for solving MDPs with intractably large state space [1]. Although RTDP achieves faster convergence than value iteration on this problem, practically it cannot handle maps with sizes larger than 7×7. This is because the search graph representing the problem is highly connected, and thus RTDP trials often get trapped in loops (moving repeatedly between the same states).

Hence, we introduce the use of frontiers, states that separate the covered regions in the search space from those

uncovered, into RTDP. In each step of the trial, Frontier-Based RTDP (FBRTDP) searches for a path with minimum expected cost from the current state to a new frontier state, choosing outcomes of this path stochastically according to their probability. FBRTDP avoids getting trapped in loops by advancing towards a new frontier in each step. The use of frontiers speeds up the convergence of RTDP quite dramatically, while retaining its focus and anytime behavior. Finally, we show that FBRTDP attains significant improvement over the state-of-the-art solution to the problem [16], in terms of both the coverage time and probability to complete the coverage.

2. RELATED WORK

The problem of robot coverage has been extensively discussed in the literature (see [7] for a recent exhaustive survey). Grid-based coverage methods, such as we utilize here, use a representation of the environment decomposed into a collection of uniform grid cells, e.g., [6], [10].

Other optimization problems related to adversarial coverage include the Canadian Traveller Problem (CTP) [14], in which the objective is to find a shortest path between two nodes in a partially-observable graph, where some edges may be non-traversable. In contrast, here the graph is fully-observable and the agent must visit every node in the graph (some of them may stop the robot). MDPs have been shown to be useful in solving CTP in certain types of graphs [12].

The offline adversarial coverage problem was formally defined in a recent study [15]. There the authors proposed a simplistic heuristic algorithm that generates a coverage path which tries to minimize a cost function, that takes into account both the survivability of the robot and the coverage path length. The heuristic algorithm worked only for obstacle-free areas, and without any guarantees. In a follow-up paper [16] they have addressed a more specific version of the problem, namely, finding the safest coverage path. They suggested two heuristic algorithms to solve this problem with some theoretical guarantees. However, they could handle only one level of threats, i.e., the environment could contain either safe or dangerous areas. In contrast, here we suggest a model that can find optimal (or near optimal) coverage paths that meet any desired risk and time levels in environments that can contain any number of threat levels, in addition to obstacles.

Perhaps the simplest algorithm for solving MDPs is value iteration, which solves for an optimal policy on the full state space. However, in many realistic problems, such as the one we discuss here, only a small fraction of the state space is relevant to the problem of reaching a goal state from a fixed start state s. There are different heuristic algorithms for solving MDPs, including offline and realtime methods. Real-Time Dynamic Programming [1] and its variants (such as Labeled RTDP [4]) have been shown to outperform other heuristic search methods for solving MDPs, such as AO*/LAO*, on several benchmark problems [4]. The advantage of real-time heuristic methods is that states that are more likely to be visited in the search graph (as defined by the probability function) are updated more frequently, which leads the algorithm to focus on updating states which are more relevant to the problem solving. In our case, states that are more likely to be visited by the robot represent coverage paths that encounter a smaller number of threats, thus updating those states more frequently can focus the search for the optimal coverage path.

3. ADVERSARIAL COVERAGE PROBLEM DEFINITION

We are given a map of a target area T, which is decomposed into a regular square grid with n cells. We assume that T can be decomposed into a regular square grid with n cells. There are two types of cells: free cells and cells that are occupied by obstacles. Some of the free cells contain threats. Each free cell c_i is associated with a threat probability p_i $(0 \leq p_i < 1)$, which measures the likelihood that a threat in that cell will stop the robot. We assume the robot can move continuously, in the four basic directions (up/down, left/right), and can locate itself within the work-area to within a specific cell. The robot's task is to plan a path through T such that every accessible free cell in T (including the threat points) is visited by the robot at least once.

The objective is to find a coverage path of T that maximizes the probability of covering the entire area and also minimizes the coverage time. Clearly, there is a tradeoff between these two objectives: trying to minimize the risk involved in the coverage path could mean 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.

We now formally define this objective function. First, we 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 not stopped when it follows the path A. The probability that the robot is able to complete this path is:

$$P(S_A) = \prod_{i \in (a_1,...,a_m)} (1 - p_i) \quad (1)$$

In order to take into account both the accumulated risk and the coverage time, we define the following cost function:

$$f(A) = -\alpha \cdot P(S_A) + \beta \cdot |A| \quad (2)$$

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] Therefore, we wish to find a coverage path A that minimizes the cost function $f(A)$, i.e., $f(A) \leq f(B)$ for all possible coverage paths B.

The problem of finding a minimum time coverage path ($\alpha = 0$) is equivalent to finding a Hamiltonian walk in a graph, which is known to be $\mathcal{NP}$-complete [13]. Finding a coverage path with maximum probability to complete ($\beta = 0$) has also been shown to be $\mathcal{NP}$-complete [16]. By reduction, the problem in the general case of $\alpha, \beta \geq 0$ is $\mathcal{NP}$-complete as well.

4. MDP MODELING

We now formulate the adversarial coverage problem as an undiscounted stochastic shortest-path problem [3]. Stochastic shortest path (SSP) problems are a subclass of MDPs, and they are given by:

[1] The travel time is uniform along the grid, thus coverage time is measured directly by the number of steps.

S1. a discrete and finite state space S
S2. an initial state $s_0 \in S$
S3. a set $G \subseteq S$ of goal states
S4. Actions $A(s) \subseteq A$ applicable in each state $s \in S$
S5. Transition probabilities $P(s'|s,a)$ for $s \in S, a \in A(s)$
S6. Positive action costs $C_a(s,s') > 0$
S7. Fully observable states

Let us denote the SSP that represents the adversarial coverage problem by $\mathcal{M}$. We now describe each of $\mathcal{M}$'s components.

States. The set of states in our model contains all possible configurations of the environment's coverage status and the robot's location. A coverage status of the environment is represented by a boolean matrix that indicates for each cell in the grid if it has already been visited by the robot or not. The state captures all relevant information from the history of the robot's movements, thus it satisfies the Markovian property.

The initial state s_0 is the state in which the robot is located at the starting cell of the coverage path, and this is the only cell marked as visited. The goal states G are the states in which all the grid cells are covered. In addition, one of the states in S is defined as a dead state, denoted by s_d, which represents the situation where the robot was stopped by a threat. This dead-end state requires special attention, as discussed in section 4.1. The goal states, as well as s_d, are termination states, i.e., taking any action in them causes a self-transition with probability 1.

Actions. There are only four possible actions the robot can perform - go up, down, left or right. There could be fewer than four actions applicable in a given state, depending on the number of obstacles that surround the cell in which the robot currently resides.

The Transition Function. The transition function describes the probability that the robot will be able to move from its current location to the next location on its coverage path. More specifically, if the current state is s, in which the robot is located in cell c_i, and the robot executes an action a that leads it to cell c_j, then we distinguish between two possible cases:

Case 1. c_j is a safe cell, i.e., $p_j = 0$. In this case there is only one possible outcome for (s,a). The outcome is a new state s', in which c_j is added to the environment's coverage status and the robot's location is changed to cell c_j. The transition probability in this case is $P_a(s'|s) = 1$.

Case 2. c_j is a dangerous cell. In this case there are two possible successor states to (s,a). The first one, s', represents the possibility that the robot will be able to enter the new cell c_j. In s', c_j is added to the environment's coverage status and the robot's location is changed to cell c_j. The transition probability to s' is $P_a(s'|s) = 1 - p_j$, where p_j denotes the probability that the threat in cell c_j will stop the robot. The second successor state is s_d, which represents the possibility that the robot will be stopped by a threat in the cell c_j. The transition probability to this state is $P_a(s_d|s) = p_j$.

Note that the probabilities on all the outgoing transitions of each state/action pair sum to 1.

The Cost Function. We define a uniform fixed cost $C_a(s,s') = 1$ for actions that lead the robot to a safe cell.

For actions that lead the robot to a cell c_j with threat probability p_j, we define different costs for the two possible outcomes of this action. If the robot was not hit by the threat, the cost of the transition to the next state s' is defined as $C_a(s,s') = \frac{1}{1-p_j}$. Otherwise, the transition cost to the dead-end state s_d is defined as $C_a(s,s_d) = -D \cdot \frac{\log(1-p_j)}{p_j}$, where $D \geq 0$ is a fixed penalty assigned to reaching the dead state.

The value of the penalty D should be set according to the desired balance between the risk and the coverage time (α/β). For example, setting $D = 0$ should make the model find the shortest coverage path, while setting $D = \infty$ should make it find the safest coverage path. For the in-between cases, to help us calibrate the value of D, we will define that when $\alpha = \beta$, i.e., when the risk and the coverage time factors have equal importance, the penalty on making a move to a dangerous cell (with a minimum threat probability) will be equal to making one step in the grid. In particular, if p_{min} is the minimum threat probability, then D is set to:

$$D = -\frac{\alpha}{\beta} \cdot \frac{1}{\log(1-p_{min})} \tag{3}$$

Note that this is the only place in the model where the risk and the time factors are combined together.

This concludes the definition of the model $\mathcal{M}$ representing the adversarial coverage problem. To demonstrate the model, let us consider the following simple grid (cells are numbered 1 to 2 from top to bottom and left to right, the numbers in the cells indicate the threat probabilities p_i):

0	0
0.4	0.2

Assume that the robot starts the coverage at cell $(1,1)$ and then moves right to cell $(1,2)$. Let us denote the current state of the environment and the robot by s_1. Figure 1 shows the graph describing the possible transitions from s_1. Circular nodes of the graph represent states of the MDP and the rectangular nodes represent actions. Inside each state node there is a description of the coverage status of the environment and the robot's position (marked by 'R'). Edges from actions to states are annotated with transition probabilities and costs.

As can be seen in the graph, there are two possible actions in state s_1: going left (a_1) and going down (a_2). Moving left to the safe cell $(1,1)$ (i.e., choosing action a_1) has only one possible outcome with probability 1. In the resultant state s_2, the coverage status of the environment does not change, only the robot's location. On the other hand, going down to the dangerous cell $(2,2)$ (i.e., choosing action a_2) leads to two possible outcomes. If the move succeeds, i.e., the robot is not stopped by the threat, then the state changes to s_3, in which the robot is located at cell $(2,2)$ and this cell is added to the coverage status of the environment. The cost of this transition is: $\frac{1}{1-0.2} = 1.25$. However, if the move fails, then the robot moves to the dead-end state s_d. The probability of the transition to s_d is 0.2, which is the threat probability in cell $(2,2)$. The cost of this transition is: $-D \cdot \frac{\log(1-0.2)}{0.2} = 1.116D$.

The solution of an MDP takes the form of a *policy* π mapping states s into actions $a \in A(s)$. The value function of a policy π, V^π, represents the expected cost incurred from following policy π from any given state s in S, i.e.:

$$V^\pi(s) = E\Big[\sum_{t=0}^{\infty} C_{a_t}(s_t, s_{t+1})\Big] \tag{4}$$

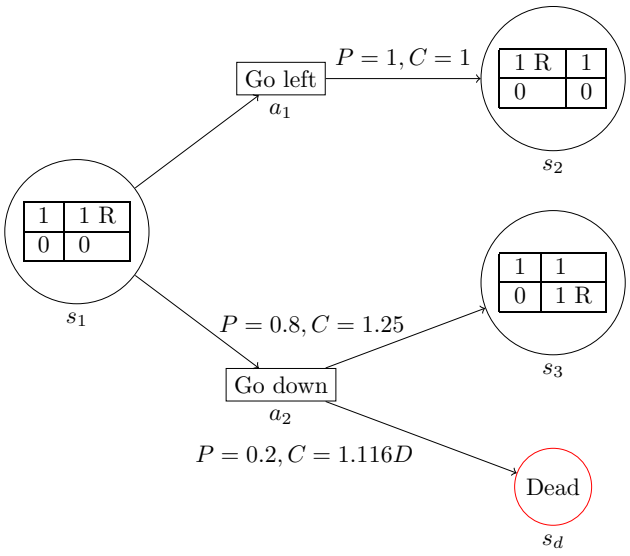

Figure 1: An example for a state in the MDP and its outgoing transitions. Edges from actions to states are annotated with transition probabilities and costs.

where $s_0 = s$, and $a_t = \pi(s_t)$ is the action taken at time step t and causes a transition from state s_t to state s_{t+1}.

An *optimal policy* is a policy π^* that has a minimum expected cost for all possible initial states. Such a policy is guaranteed to exist if the following assumption holds [3]:

S8. The goal is reachable from every state with non-zero probability.

In $\mathcal{M}$, all states except for the dead-end state satisfy this assumption (since all threat probabilities are less than 1). We discuss how to treat the dead-end state in section 4.1.

An optimal value function, denoted by $V^*(s)$, assigns to each state its value according to an optimal policy π^*, and satisfies the following fixed point equation, also known as Bellman's optimality equation [2]:

$$V^*(s) = \min_{a \in A(s)} \sum_{s' \in S} P(s'|s,a)\big[C_a(s,s') + V^*(s')\big] \tag{5}$$

Value iteration (VI) is a standard dynamic programming method for solving MDPs based on Eq. (5). VI algorithms start with an initial guess for V_0 and repeatedly update so that V gets closer to V^*.

We now prove the correctness of the model $\mathcal{M}$.

Theorem 1. *(correctness) The optimal policy of the MDP $\mathcal{M}$ represents an optimal solution to the adversarial coverage problem.*

Proof. By definition, the optimal policy π^* of MDP $\mathcal{M}$ minimizes the expected cost:

$$E\Big[\sum_{t=0}^{\infty} C_{a_t}(s_t, s_{t+1})\Big] \tag{6}$$

where s_0 is the initial state, and $a_t = \pi^*(s_t)$ is the action taken at time step t according to the policy π^*.

The sequence of actions $(a_0, a_1, a_2, ...)$ taken by the optimal policy must eventually lead to a goal state. This is due to the fact that all action costs are positive (except for actions in the goal states), thus if the sequence of actions never reaches a goal, then the expected cost of the optimal policy becomes infinite, which violates assumption S8 that states that there must be at least one policy that reaches a goal state from any state.

Now denote by n the number of state transitions needed for the optimal policy to reach a goal state from the initial state s_0. Then the expected cost can be written as:

$$E\Big[\sum_{t=0}^{n-1} C_{a_t}(s_t, s_{t+1})\Big] \tag{7}$$

where s_n is a goal state.

From the linearity of expectation, we get:

$$E\Big[\sum_{t=0}^{n-1} C_{a_t}(s_t, s_{t+1})\Big] = \sum_{t=0}^{n-1} E\big[C_{a_t}(s_t, s_{t+1})\big] \tag{8}$$

Now let us denote by $(c_1, ..., c_n)$ the sequence of cells that were visited by the robot following the actions in $(a_0, ..., a_{n-1})$, and their threat probabilities by $(p_1, ..., p_n)$. According to the cost function $C_a(s, s')$ in the MDP model, the expected cost of moving to cell c_j is:

$$\begin{aligned} E[C(c_j)] &= (1 - p_j) \cdot \frac{1}{1 - p_j} + p_j \cdot \Big[- D \cdot \frac{\log(1 - p_j)}{p_j} \Big] \\ &= 1 - D \log(1 - p_j) \end{aligned} \tag{9}$$

The resultant expression is also true for the expected cost of moving to a safe cell c_j, since in that case $p_j = 0$, thus $E[C(c_j)]$ becomes equal to 1.

Thus, the sum in Eq. (8) becomes:

$$\sum_{t=0}^{n-1} E\big[C_{a_t}(s_t, s_{t+1})\big] = n - D \sum_{j=1}^{n} \log(1 - p_j) \tag{10}$$

The result is a sum of two expressions - the first is determined by the coverage path length (n) and the second is determined by the accumulated risk that was taken by the robot along the path. It is trivial to verify that if there are two coverage paths with the same accumulated risk but with different lengths, then the optimal policy will prefer the shorter one. We now prove that if there are two coverage paths with the same length but with different accumulated risks, then the optimal policy will prefer the safer path. Let us denote the sequence of cells visited along the first coverage path by $(u_1, ..., u_l)$ and the sequence of cells visited along the second path by $(v_1, ..., v_m)$. Let us assume that the first path is safer than the second, i.e., it has a greater probability to complete. Thus, we can write:

$$\prod_{i \in (u_1, ..., u_l)} (1 - p_i) \geq \prod_{j \in (v_1, ..., v_m)} (1 - p_j) \tag{11}$$

Since the logarithm is a monotonically increasing function of its argument, the above expression is equivalent to:

$$\sum_{i \in (u_1, ..., u_l)} \log(1 - p_i) \geq \sum_{j \in (v_1, ..., v_m)} \log(1 - p_j) \tag{12}$$

If we multiply both sides by $-D$ and add n (the path length), we get:

$$n - D \sum_{i \in (u_1,\ldots,u_l)} \log(1-p_i) \leq n - D \sum_{j \in (v_1,\ldots,v_m)} \log(1-p_j) \tag{13}$$

Note that the expressions on both sides of Eq. (13) are similar to the expression on the right-hand side of Eq. (10). Thus, we can conclude that the expected cost of a policy that generates the first coverage path is lower than the expected cost of a policy that generates the second one. Therefore, the optimal policy of MDP $\mathcal{M}$ is guaranteed to produce the safer coverage path.

In the in-between cases, where we want to find an optimal coverage path that takes into account both the coverage time and the accumulated risk, we can adjust the penalty D according to the desired levels of risk and the time. The optimal policy will then produce a coverage path that minimizes the expected cost defined in Eq. (10), which is dependent on D. The higher the penalty D is set, the more safer coverage paths will be favored over shorter ones. □

4.1 MDPs with Dead Ends

Researchers have realized that allowing dead ends in goal-oriented MDPs could break the existing methods for solving them (e.g., [9]). In MDPs with dead ends the objective of finding a policy that minimizes the expected cost of reaching the goal becomes ill-defined, since it implicitly assumes that for at least one policy the cost incurred by all of the policy's trajectories is finite.

This problem can be resolved by assigning a finite positive penalty D for visiting a dead end, and augmenting the action set A of the MDP with a special action a' that causes a transition from the dead end to the goal with probability 1. This MDP now satisfies assumption S8, since reaching the goal with certainty is possible from every state. However, this solution comes with a caveat - it may cause non-dead-end states that lie on potential paths to a dead end to have higher costs than the dead ends themselves. As a consequence, the optimal policy may prefer getting into a dead end rather than reaching the goal. Kolobov et al. [8] suggest resolving this issue by capping the cost of each state by D. They use the following modified Bellman equation:

$$V^{\pi}(s) = \min\Big\{D, \min_{a \in A(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)\big[C_a(s,s') + V^{\pi}(s')\big]\Big\} \tag{14}$$

They show that MDPs with dead ends can be solved with VI that uses Eq. (14) for updates. Moreover, all heuristic search algorithms for solving SSPs (such as RTDP) and their guarantees apply to this type of MDPs if they use Eq. (14) in lieu of Bellman's update.

5. REAL-TIME DYNAMIC PROGRAMMING

RTDP [1] is a heuristic-search DP algorithm for solving non-deterministic planning problems with full observability. In relation to other dynamic programming methods, RTDP has two benefits. First, it is focused, namely it updates only states that are encountered in the search and thus relevant to the problem solving. Second, it has a good anytime behavior, i.e., it produces good policies fast and these policies improve smoothly with time.

RTDP works by repeated trials or runs (see algorithm 1). Each trial starts at the initial state s_0 and ends in a goal state or a dead-end state. At each step, action selection is greedy based on the current value function, and outcome selection is stochastic according to the distribution of possible successor states given the chosen action. The values $V(s)$ of the visited states are updated along the way, using Bellman's equation (Eq. (5)). The initial values of $V(s)$ are given by an heuristic function $h(s)$.

After the termination condition is met, a coverage path is built by following the greedy policy from the starting state to a goal state. In this final phase, we never enter a dead-end state; whenever the robot visits a threat point, the outcome that represents its survival of the threat is chosen deterministically. This way we guarantee that a complete coverage path is created, i.e., a path that covers all the cells in the target area. In contrast, RTDP trials may be terminated before the entire area is covered. This helps the algorithm focus on updating states which the robot has more chance to reach.

From [1], it is known that under conditions S1-S8 for SSPs, if the initial value function is admissible, i.e., $h(s) \leq V^*(s)$ for every state s, then repeated RTDP trials eventually yield optimal values $V(s) = V^*(s)$ over all relevant states (states that can be reached by at least one optimal policy). In our experiments we have used the admissible heuristic function $h \equiv 0$.

6. FRONTIER-BASED RTDP

A good heuristic can lead to faster convergence of RTDP. However, choosing a good admissible heuristic function is often a non-trivial task. Moreover, in a huge state space such as we have here, finding a good heuristic function may not be enough. Initial results from our empirical evaluation have indicated that one of the main reasons for the slow convergence of RTDP is that its trials a waste considerable amount of time moving back and forth between already visited states. For example, let us examine states s_1 and s_2 from the search subgraph depicted in Figure 1. Since both cells $(1,1)$ and $(1,2)$ are safe, an RTDP trial would travel back and forth between states s_1 and s_2 until the estimated cost of the repeated transition between them becomes higher than the cost of moving to one of the dangerous cells $(2,1)$ or $(2,2)$. Clearly, these repeated transitions cannot be part of the trajectory followed by the optimal policy (they only increase the cost of the path to the goal), and thus should be eliminated from the search.

Frontier-Based RTDP (algorithm 2) avoids such fruitless cyclic returns in the search graph, by maintaining a list of *frontier* states, defined as states that separate the covered regions of the search space from those uncovered. Each time a new state is encountered by an RTDP trial, it goes over all its possible successors, and adds to the frontier list all the unvisited successors that are not already in this list. A state is taken out of the frontier list once it is visited by the trial.

At each step of the trial, FBRTDP examines all the possible paths from the current state to one of the frontier states, and chooses the path with the minimum expected cost according to the current value function. To allow a transition from any given state to a frontier state, we extend the set of actions A in the MDP model with the following definition.

Algorithm 1 Real_Time_Dynamic_Programming

Input: a grid G, a starting cell c_0, a termination criterion ϵ
Output: a coverage path P that covers all reachable cells in G from c_0

function RTDP(s_0) // *s_0 is the initial state*
  while $\max_{s \in visited}$ RESIDUAL(s) $> \epsilon$ **do**
    RTDPTRIAL(s_0)
  return BUILDCOVERAGEPATH(s_0)

function RTDPTRIAL(s) // *Execute one trial of RTDP*
  while not GOAL(s) and $s \neq s_d$ **do**
    // *Pick best action and update hash*
    $a \leftarrow$ GREEDYACTION(s)
    UPDATE(s, a)
    // *Stochastically simulate next state*
    $s \leftarrow$ CHOOSENEXTSTATE(a)

function GOAL(s)
  return if all reachable cells from c_0 are covered in s

function INITSTATE(s) // *Implicitly called the first time each state s is touched*
  $s.V \leftarrow h(s)$

function GREEDYACTION(s)
  return $\arg\min_{a \in A(s)}$ QVALUE(s, a)

function QVALUE(s, a)
  return $\sum_{s' \in \mathcal{S}} P(s'|s,a)\big[C_a(s,s') + s'.V\big]$

function UPDATE(s, a)
  $s.V \leftarrow$ QVALUE(s, a)

function CHOOSENEXTSTATE(s, a)
  Choose s' with probability $P(s'|s,a)$
  return s'

function RESIDUAL(s)
  $a \leftarrow$ GREEDYACTION(s)
  return $|s.V -$ QVALUE(s, a)$|$

function BUILDCOVERAGEPATH(s)
  Create a new coverage path P
  Add starting cell c_0 to P
  while not GOAL(s) **do**
    $a \leftarrow$ GREEDYACTION(s)
    Make the robot move according to action a
    Add the cell c where the robot is located to P
    // *Deterministically simulate next state*
    $s \leftarrow s$ with cell c marked as visited and robot's location is at c
  return P

Definition 1. ***Composite action*** *$\hat{a}$ is an action that consists of a sequence of actions $(a_1, ..., a_n)$ from A.*

The possible outcomes of a composite action consist of all the states that could be reached by an RTDP trial following the sequence $(a_1, ..., a_n)$.

In the adversarial coverage case, any composite action has only two possible outcomes: reaching the destination cell of the final action in the sequence $(a_1, ..., a_n)$ or entering the dead state. The probabilities of these outcomes depend on the threat probabilities of the cells $(c_1, ..., c_n)$ encountered along the path taken by the robot following the actions in $(a_1, ..., a_n)$. More specifically, the probability of the first outcome, in which the robot is able to visit all the cells $(c_1, ..., c_n)$ without being hit by a threat, is:

$$P(s'|s, \hat{a}) = \prod_{i=1}^{n}(1 - p_i) \tag{15}$$

whereas the probability of the second outcome, in which the robot is stopped by a threat along the path, is complementary to the probability of the first outcome, i.e.,

$$P(s_d|s, \hat{a}) = 1 - P(s'|s, \hat{a}) \tag{16}$$

More generally, to compute the probability $P(s'|s, \hat{a})$ of each outcome of a composite action $\hat{a}$, we need to add up the probabilities of all the possible paths from the current state s to the destination state s' of that outcome.

We now define the cost of a composite action $C_{\hat{a}}(s, s')$ as the sum of the costs of all its primitive actions $(a_1, ..., a_n)$. If we denote by $(s_0, ..., s_n)$ the set of states visited by the RTDP trial following the actions $(a_1, ..., a_n)$, starting from the current state s_0, then the cost of $\hat{a}$ is:

$$C_{\hat{a}}(s_0, s_n) = \sum_{i=1}^{n} C_{a_i}(s_{i-1}, s_i) \tag{17}$$

By linearity of expectation, the expected cost of a composite action $\hat{a}$ is the sum of the expected costs of all its primitive actions, i.e.,

$$E\big[C_{\hat{a}}(s_0, s_n)\big] = \sum_{i=1}^{n} E\big[C_{a_i}(s_{i-1}, s_i)\big] \tag{18}$$

In order to find a path with minimal expected cost from the current state to a frontier state, at each step of the trial we build a subgraph of the search space that consists of the visited states so far and the frontier states. Then, we execute Dijkstra's shortest paths algorithm on this subgraph, where the weight w_{ij} of the edge connecting states s_i and s_j is defined as the expected cost of the action leading from state s_i to s_j, i.e., $w_{ij} = E\big[C_a(s_i, s_j)\big]$.

Dijkstra's algorithm finds paths with minimum expected costs between the current state s_0 and all the frontier states in this subgraph. The next action chosen by FBRTDP is the composite action that leads from the current state s_0 to the frontier state with minimum expected cost path from s_0.

Additionally, one can exploit domain-specific knowledge to narrow down the set of frontier states and thus prune more irrelevant states from the search. Specifically, in the adversarial coverage case, we consider only states in which the robot reaches an unvisited cell in the map as frontier states. For instance, let us examine state s_2 from Figure 1. Although this state has not been encountered in the search before, we can treat it as a non-frontier state, since in this state the robot returns to an already visited cell $(1, 1)$. Thus, the possible successor frontier states of s_1 that should be considered are the states in which the robot reaches one of the unexplored cells $(2, 1)$ or $(2, 2)$.

We now prove that FBRTDP has the same optimal convergence guarantees as RTDP.

Theorem 2. *Under conditions S1-S8, if the initial value function is admissible, repeated FBRTDP trials eventually yield optimal values $V(s) = V^*(s)$ along every optimal path from the initial state to a goal state.*

Proof. The main idea of the proof is to show that it is enough to consider the paths to frontier states from any given state in order to reach the optimal value function.

The first observation is that FBRTDP preserves the non-overestimating property of h when visiting a state and updating its value. Let us denote by $F(s)$ the set of frontier

Algorithm 2 Frontier_Based_RTDP

Data structures: *frontier* - set of frontier states
visited - set of states already visited by the current trial

function FBRTDP(s_0) // *s_0 is the initial state*
  while $\max_{s \in visited}$ RESIDUAL(s) $> \epsilon$ **do**
    visited $\leftarrow \{s_0\}$
    frontier $\leftarrow$ all successors of s_0
    FBRTDPTRIAL(s_0)

function FBRTDPTRIAL(s) // *Execute one trial*
  while not GOAL(s) and $s \neq s_d$ **do**
    // *Pick best composite action and update hash*
    $\hat{a} \leftarrow$ GREEDYCOMPOSITEACTION(s)
    UPDATE($s, \hat{a}$)
    // *Stochastically simulate next state*
    $s \leftarrow$ CHOOSENEXTSTATE($\hat{a}$)
    if $s \notin$ *visited* **then**
      Add s to *visited*
      UPDATEFRONTIER(s)

function GREEDYCOMPOSITEACTION(s)
  Build a graph G that consists of the states in *visited* $\cup$ *frontier* and its edge weights defined as the expected costs of the state transitions
  Run Dijkstra on the graph G starting from s
  Find a frontier f with minimum cost path from s
  Let $\hat{a} = (a_1, ..., a_n)$ be the sequence of actions leading from s to f on the minimum cost path
  return $\hat{a}$

function QVALUE($s, \hat{a}$)
  return $\sum_{s' \in S} P(s'|s, \hat{a})\left[C_{\hat{a}}(s, s') + s'.V\right]$

function UPDATE($s, \hat{a}$)
  $s.V \leftarrow$ QVALUE($s, \hat{a}$)

function CHOOSENEXTSTATE($s, \hat{a}$)
  Choose s' with probability $P(s'|s, \hat{a})$
  return s'

function UPDATEFRONTIER(s)
  Remove s from *frontier*
  for every successor state s' of s **do**
    if $s' \notin$ *visited* and $s' \notin$ *frontier* **then**
      Add s' to *frontier*

states that can be reached from a given state s. Assuming that the h values of $F(s)$ do not overestimate the expected cost to reach the goal, then after adding paths with minimum expected cost from s to each of these frontiers, the minimum of the resulting values cannot overestimate the expected cost to the goal from the given state.

We now define the value $V(s)$ of a state s to be *consistent* with the frontier states that can be reached from it, if $V(s) = \min_{s' \in F(s)} \left[E[C(s, s')] + V(s')\right]$, where $E[C(s, s')]$ is the expected cost of the optimal path from s to s'.

Now, assume the converse of the theorem, that after an infinite number of trials, there exists a state along an optimal path from the initial state to a goal whose value is not optimal. Assuming that h of all goal states is zero, if the value of any state along any path to a goal state is not optimal, then some frontier state along the same path must be inconsistent. This follows formally by induction on the distance from the goal.

If there exists a frontier state whose value is inconsistent, then there must exist at least one such state in an arbitrary ordering of the states. Call such a state x. By assumption, x lies along an optimal path from the initial state s to a goal state. In addition, since all the h values are non-overestimating and this property is preserved by FBRTDP, the values of all the states along the optimal path from s to x, are less than or equal to their optimal values. This ensures that state x will eventually be visited by FBRTDP. When it is, its value will become consistent with the frontier states that can be reached from it, thus violating the assumption that it is the least inconsistent frontier state in some ordering. Therefore, the value of every state along an optimal path from the initial state to a goal state must eventually reach its optimal value. □

7. EMPIRICAL EVALUATION

In this section we evaluate FBRTDP in relation to RTDP and three other algorithms: VI, the standard dynamic programming algorithm, LRTDP (Labeled RTDP) [4], and GAC (Greedy Adversarial Coverage), the state-of-the-art solution to the adversarial coverage problem as described in [16]. We use a specific map to illustrate the operation of the algorithms and we also report on the statistical analysis of their behavior based on multiple randomly generated maps with varying parameters.

Figure 2 shows an example for the optimal safest coverage path found by VI on a map of size 7×7. The map contains 25% threat points with 5 different threat probabilities between 0.006 and 0.03, and 30% obstacles. Obstacles are represented by black cells, safe cells are colored white and dangerous cells are represented by 5 different shades of purple. Darker shades represent higher values of p_i (more dangerous areas). The number of visits to each cell along the coverage path is indicated within that cell. The termination criterion was set to $\epsilon = 0.1$.

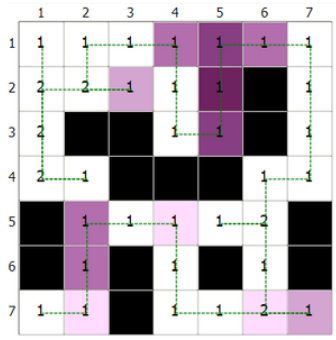

Figure 2: An optimal coverage path generated from a Value Iteration run.

As can be seen, the coverage path revisits only a single threat point, which has the lowest threat probability. The other five revisits are to safe cells. This coverage path is optimal, since any coverage path of this map must revisit at least one threat point (there is a threat point located next to the two lower corners and the robot must get in and out of at least one of these corners in order to complete the coverage). The probability to complete the coverage path generated by VI was 49.5%, and its total length was 40. Running GAC on the same map generated a coverage path with 44.71% probability to complete (containing 4 revisits to threat points) and total length of 72. RTDP, LRTDP and FBRTDP converged to the same optimal solution as VI on this map, albeit in a much shorter time. The curves in Figure 3 display the evolution of the expected cost to

the goal as a function of time for the different algorithms. FBRTDP shows the best profile, converging to the optimal policy in only 0.429 seconds, while RTDP, LRTDP and VI converge to the optimal policy in 541, 530, and 803 seconds, respectively.

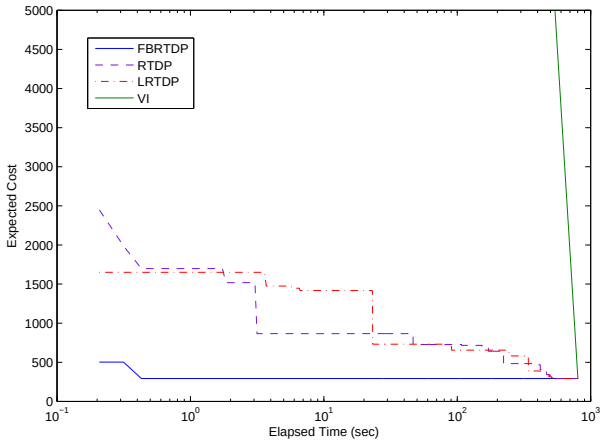

Figure 3: Expected cost to the goal vs. time for VI, RTDP, LRTDP and FBRTDP. The time axis is plotted on a logarithmic scale.

For map sizes larger than 7×7, the MDP's state space exceeded the available memory, thus VI could not be executed for such maps. Moreover, in such maps, RTDP's and LRTDP's convergence was very slow (it took a few hours for them to converge on maps of size 8×8).

Therefore, for large-sized maps we have analyzed the performance of only FBRTDP and GAC. Figure 4 shows the completion probability and the path length obtained by both algorithms for varying α/β ratios between 0.001 and 1000. The results are averaged on 30 random maps. In all experiments we have used map sizes of 20×20, the ratio of obstacles was 30%, the ratio of threats was 30% and the number of threat levels was 5. The locations of the threat points and the obstacles were randomly chosen. Note that for maps of this size, FBRTDP did not converge (i.e., the residual didn't get below ϵ) in a reasonable amount of time, thus we halted it after 1000 trials.

As can be seen, the coverage path length increases as the risk factor α become more dominant in both algorithms. The probability to complete increases until the ratio α/β is around 5 and then it starts to decrease. This is due to the fact that when α is too high, the algorithms try to avoid visiting higher-level threats as much as possible, which makes them revisit lower-level threats more times, thus the coverage path gets longer and riskier.

In all experiments, FBRTDP consistently outperforms the greedy algorithm in terms of both the completion probability and the path length. On average, FBRTDP achieves about 1% increase in the robot's survivability and 5% decrease in the path length compared to GAC (which is statistically significant; one-tailed t-test $p = 7.43 \cdot 10^{-16}$). As can be seen from the graph, changing the ratio between the risk and the time factors (α/β) from 0 to ∞, under the given map settings, can change the survivability probability by only 1.5% in both algorithms. Thus, a 1% increase in the robot's survivability is quite dramatic. The absolute difference between the algorithms' results depends upon map settings. For example, when the threats ratio was decreased to 25%, FBRTDP attained a robot's survivability probability which was 3% higher than GAC.

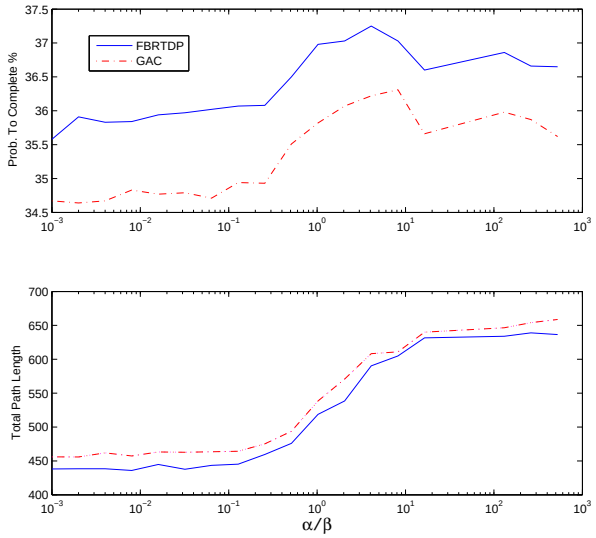

Figure 4: Probability to complete the coverage and total path length for different risk and time levels. x axis is plotted on a logarithmic scale.

On the down-side, FBRTDP's average running time was significantly higher than GAC's (179 seconds in FBRTDP, 0.154 seconds in GAC). This difference is caused by the high number of FBRTDP trials that was needed in order to reach convergence. However, FBRTDP typically outperforms GAC after a small number of trials (an FBRTDP trial follows a greedy policy which resembles the greedy behavior of GAC).

8. CONCLUSIONS AND FUTURE WORK

We have described how to model the robotic adversarial coverage problem as an MDP. We have shown how the model can be used to find an optimal solution to the problem on small-sized maps, and obtain significant improvement over the state-of-the-art solution for larger maps. To the best of our knowledge, this is the first time that MDPs have been used to represent problems in the robotic coverage field.

We have also introduced FBRTDP, a new improvement to RTDP, which maintains a list of frontier states and extends the set of actions that can be used by the model. FBRTDP provides significant speedup, allows RTDP to solve the adversarial coverage problem on much larger maps, and has the same optimal convergence guarantees as RTDP.

In the future we plan to use the MDP model to handle other variants of the adversarial coverage problem, such as a variant in which threats may cause time delays instead of completely stopping the robot. We also intend to evaluate FBRTDP on other planning problems and compare its performance to other heuristic algorithms for solving MDPs.

9. REFERENCES

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[2] R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[3] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1 and 2. Athena Scientific, 1995.

[4] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proc. of ICAPS*, volume 3, pages 12–21, 2003.

[5] J. Colegrave and A. Branch. A case study of autonomous household vacuum cleaner. *AIAA/NASA CIRFFSS*, page 107, 1994.

[6] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry*, 24(3):197–224, 2003.

[7] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[8] A. Kolobov, Mausam, and D. Weld. A theory of goal-oriented mdps with dead ends. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pages 438–447, 2012.

[9] I. Little and S. Thiebaux. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*, 2007.

[10] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet. A solution to vicinity problem of obstacles in complete coverage path planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA-02)*, volume 1, pages 612–617, 2002.

[11] J. D. Nicoud and M. K. Habib. The pemex-b autonomous demining robot: perception and navigation strategies. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 'Human Robot Interaction and Cooperative Robots'*, volume 1, pages 419–424, 1995.

[12] E. Nikolova and D. R. Karger. Route planning under uncertainty: The canadian traveller problem. In *Proc. of the Twenty-Third Conference on Artificial Intelligence (AAAI-08)*, pages 969–974, 2008.

[13] T. Nishizeki, T. Asano, and T. Watanabe. An approximation algorithm for the hamiltonian walk problem on maximal planar graphs. *Discrete applied mathematics*, 5(2):211–222, 1983.

[14] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Automata, Languages and Programming*, pages 610–620. Springer, 1989.

[15] R. Yehoshua, N. Agmon, and G. A. Kaminka. Robotic adversarial coverage: Introduction and preliminary results. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-13)*, pages 6000–6005, 2013.

[16] R. Yehoshua, N. Agmon, and G. A. Kaminka. Safest path adversarial coverage. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-14)*, pages 3027–3032, 2014.