Reactive Motion Planning in Uncertain Environments via Mutual Information Policies

Ryan A. MacDonald and Stephen L. Smith

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo ON, Canada
{ryan.macdonald,stephen.smith}@uwaterloo.ca

Abstract. This paper addresses path planning with real-time reaction to environmental uncertainty. The environment is represented as a graph and is uncertain in that the edges of the graph are unknown to the robot a priori. Instead, the robots prior information consists of a distribution over candidate edge sets. At each vertex, the robot can take a measurement to determine the presence or absence of an edge. Within this model, the Reactive Planning Problem (RPP) provides the robot with a start location and a goal location and asks it to compute a policy that minimizes the expected travel and observation cost. In contrast to computing paths that maximize the probability of success, we focus on complete policies (i.e., policies that produce a path, or determine no such path exists). We prove that the RPP is NP-Hard and provide a suboptimal, but computationally efficient, solution. This solution, based on mutual information, returns a complete policy and a bound on the gap between the policy's expected cost and the optimal. Finally, simulations are run on a flexible factory scenario to demonstrate the scalability of the proposed approach.

Keywords: Motion and Path Planning, Planning under Uncertainty, Mutual Information

1 Introduction

Robot motion planning under uncertainty is typically concerned with uncertainty in the robot's state within an environment and/or uncertainty in the outcome of a selected action on the robot's state [1–5]. In this work, we consider motion planning with uncertainty in the set of *motion* actions that a robot has access to at a given state. This problem arises in scenarios where the robot is given a set of possible locations for obstacles in an environment. The obstacles restrict the set of motions available to the robot at each point in the environment. By taking sensor measurements, the robot can narrow down the set of feasible obstacle locations and thus the motion actions it has available. Our goal is to compute motion and sensing policies prior to robot deployment that enable the robot to efficiently navigate in such environments. In this paper, we focus on the task of moving from a start location to a goal location while minimizing the expected action cost. The challenge in this problem is that future costs (for obtaining information and moving between locations) are dependent on the information the

robot has obtained thus far. We present conditions where exploration is no longer helpful. When these conditions are met, the robot should exploit the known motion action set to reach the goal. We also develop a policy that provides constant time lookup for the next action given the outcomes of prior observations. This allows for implementation on robots where on-board computational resources are limited at deployment, or in which high-speed motion is required.

Related Work: In robotics, there are several effective methods for dealing with uncertainty. Point-to-point motion is addressed in [6] using persistent paths, which maximize the probability of success. However, if the computed path is obstructed, the robot ends in failure. To avoid failure, Partially Observable Markov Decision Processes (POMDPs) can be used to compute reactive motion policies. A POMDP selects actions based on partially observed states, but the computation of policies is in general a PSPACE-Complete problem [7]. In our work, we are interested in cases where the environment has a very large state space; for these cases, the POMDP's scalability becomes a barrier to use [8]. To avoid the computational complexity, algorithms like A* and D* lite allow for replanning during execution [9, 10]. These re-computations may present a bottleneck in real-time performance. Informative path planning is studied in several works [1–4, 11], all of which provide methods for real-time reaction to information within the environment. Research in this area focuses on tasks ranging from underwater inspection [3] to maximizing information from start to goal [1]. These works plan prior to deployment of the robot and react to new information collected by the robot, but their possible actions are known prior. In contrast, we consider cases where information may not be attainable until the robot has explored parts of the environment, which is not captured in this prior work.

In operations research, a closely related problem is planning with recourse and the Canadian Travelers Problem (CTP). Planning with recourse in [12] provides possible obstacle locations, but assumes obstacles will never make movement from the start to goal impossible. The CTP is PSPACE-Complete [13] and does not include prior information on obstacles. To make the problem more tractable, [14] provides possible obstacle locations in their problem R-SSPPR and seeks to minimize the expected cost between start and goal. In contrast to this work, R-SSPPR uses move actions, which can always be taken (i.e., a path to goal always exists), that have costs sensed for no cost. In transportation research, [15] presents an integer linear program to solve a route blocked problem, but they select a primary path and only switch to a secondary path when the primary fails. Our work wishes to minimize the expected cost to get to the goal or realize the goal is unreachable.

Our work leverages the concept of mutual information within discrete environments. Mutual Information is widely used to develop efficient sub-optimal solutions for gaining information in planning [5,11,16,17]. In particular, a mutual information gradient controller is presented in [5], where multiple robots search for targets and avoid hazards. The authors present a discretized probabilistic model where targets and hazards may exist and focus on positions of failure. In

contrast, our work considers unknown environments where locations may not be reachable until the robot acquires a certain level of environmental awareness.

Contributions: The contributions of this paper are fourfold. First, we introduce the Reactive Planning Problem (RPP) and prove it is NP-Hard. Second, we provide properties that allow for a compact representation of a RPP policy. Third, we present an efficient algorithm for a sub-optimal solution to RPP that utilizes mutual information to guide exploration and uses an estimation of the cost-to-go for exploitation. Fourth, we provide a method to bound the gap between the expected cost of our policy and that of the optimal.

Organization: Section 2 introduces background terminology from graph theory and the Informative Path Planning problem. The Reactive Planning Problem is defined in Section 3 along with the environmental and robotic models. Section 4 provides problem properties as well as proof of computational complexity. Several properties from Section 4 are then expanded as a base for scalable policy generation in Section 5. Section 6 provides simulation results.

2 Background

2.1 Graph Terminology

A directed graph G is defined by the pair $G = (\mathcal{V}, \mathcal{E})$ and a cost function $c : \mathcal{E} \to \mathbb{R}$. The set $\mathcal{V}$ with $n = |\mathcal{V}|$ is the set of vertices that are connected by the set of edges $\mathcal{E}$, and $c(e)$ gives the cost of traversing an edge $e \in \mathcal{E}$. A path P in a graph is defined by a sequence of vertices $v_1, \ldots, v_k$ that satisfies $(v_i, v_{i+1}) \in \mathcal{E}$ for all $i \in \{1, \ldots, k-1\}$ with cost of traversal defined by $c(P) = \sum_{i=1}^{k-1} c(v_i, v_{i+1})$. With some abuse of notation for $v, w \in \mathcal{V}$, $c(v, w)$ refers to the minimum cost of a path from v to w. Given a graph $G = (\mathcal{V}, \mathcal{E})$, the subgraph $G_E = (V, E)$ is induced by $E \subseteq \mathcal{E}$ with $V \subseteq \mathcal{V}$ given by the endpoints of E.

An edge $e = (v, u) \in E$ is said to be *incident* with vertices v and u. As the graph is directed, e is outgoing at v and incoming at u. Therefore, e is *incident-in* to u and is *incident-out* to v with the set of edges *incident-out* to v, $I_v \subseteq \mathcal{E}$.

2.2 Informative Path Planning Review

In [11], the Informative Path Planning (IPP) problem under noiseless observation is defined as a tuple (X, d, H, ρ, O, Z, r). A robot starts at r and can visit the set of sensing locations X. The cost of travel between these locations is $d(x, y)$ for $x, y \in X$. There is a finite set of hypotheses H, which has a probability mass function ρ, and a set of observations O, which are sensed with $Z(x, h, o)$ for $x \in X$, $h \in H$ and $o \in O$. The function Z returns 1 when o agrees with h and 0 otherwise. The problem then asks to minimize the expected cost of identifying the correct hypothesis. An optimal policy can be encoded as a binary tree where nodes contain sensing information and the outgoing edges are selected via the sensing outcome. From [11], IPP is NP-hard as it contains the optimal decision tree problem [18] as a special case. We will use IPP to prove our problem is NP-Hard (decision form NP-Complete).

3 Problem Definition

We consider a single robot in a discrete environment. The robot and environment models are defined using a directed and doubly weighted graph $G = (\mathcal{V}, \mathcal{E}, c, \mu)$ where $c(e) \in \mathbb{R}_{\geq 0}$ defines the robot's cost for traversing $e \in \mathcal{E}$ and $\mu(e) \in \mathbb{R}_{\geq 0}$ defines robot's cost for sensing if edge e is obstructed. The robot knows the vertex it occupies, but does not know which edges leaving that vertex are free to traverse (that is, which edges are obstructed by obstacles). If the robot is unsure of an edge's state, it must first inspect the edge and incur the sensing cost.

3.1 Environmental Model

The unknown environment is one of m subgraphs of G, denoted $G_1, \ldots, G_m$, and we refer to the indices of these subgraphs as *environmental states* with *environmental state space* $\mathbb{N}_m = \{1, \ldots, m\}$. Each subgraph G_i is induced by a subset $E_i \subseteq \mathcal{E}$ for $i \in \mathbb{N}_m$. The robot is given the set of possible edge subsets $\mathcal{S} = \{E_1, \ldots, E_m\}$ along with a probability mass function capturing the likelihood of each subgraph. We encode the probability as a random variable X that takes values from $\mathbb{N}_m$. Given a random draw x from X, the edge subset E_x induces the *realization* $G_x = (V_x, E_x, c, \mu)$; the robot must operate in G_x without knowing x.

Note that if every edge subset is possible, $m = 2^{|\mathcal{E}|}$, then the absence or presence of an edge cannot imply the absence or presence of other edges. In this paper, we focus on cases where $m \ll 2^{|\mathcal{E}|}$, and thus observing one edge allows the robot to infer the state of other edges. This is motivated in Section 4.2 by the space complexity required for a control policy.

Example 1. To illustrate the problem, consider Fig. 1 as a simplified model of a building. A robot is tasked with delivering a package to room Q. The robot starts outside the building at S, which has two main entrances A and B. Both A and B will be locked in an emergency. From entrance A, there are two paths to Q. However, at times, one of these paths is obstructed. From entrance B, there is a single path. For this problem, we use $m = 3$ *environmental states* since they directly affect the completion of the robot's task. In state 1, no dashed edges exist; in state 2, all edges except (A, Q) exist; and in state 3, all edges exist.

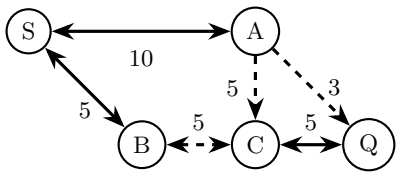

Fig. 1. Building model. Travel costs are shown on edges. Dashed edges are uncertain.

Remark 1. Given G_i for any $i \in \mathbb{N}_m$, the connected component containing the start must be strongly connected.

3.2 Robot Model

When the robot is located at vertex v, it can gather information on the environment by inspecting *incident-out* edges at v. This process is defined by the following model.

Definition 1 (Robot Observations). *Given a graph G and a vertex $v \in \mathcal{V}$ with $e \in I_v$, a robot observation is a mapping $\gamma : \mathcal{E} \to \{0, 1\}$ where $\gamma(e) = 1$ if $e \in E_x$ and $\gamma(e) = 0$ otherwise. The robot must occupy v to attain $\gamma(e)$.*

When we say the robot travels to an *observation* e, we mean it travels to v such that $(v, u) = e \in \mathcal{E}$. Given the robot is at v, the set I_v represents all possible *observations*.

Observations allow the robot to rule out *environmental states*. If an *observation* indicates an edge e does not exist in G_x, the robot knows that all edge subsets that contain e cannot be correct.

Definition 2 (Consistent). *Given a set of observed edges O, an edge subset E is consistent with O if and only if for every $e \in O$, $\gamma(e) = 1$ for $e \in E$ and $\gamma(e) = 0$ for $e \notin E$.*

Given a set of *observed* edges O, we define $Y \subseteq \mathbb{N}_m$ to be the set of *environmental states consistent* with O. To avoid collisions with an obstacle in the environment, we impose the restriction that an edge e can be traversed only if $\mathbb{P}(e|Y) = (\sum_{j \in Y} \mathbb{P}(X = j))^{-1} \sum_{i \in Y \mid e \in E_i} \mathbb{P}(X = i) = 1$. If an edge's existence is uncertain then the robot must first *observe* the edge and incur the corresponding *observation* cost, before proceeding across the edge.

3.3 Policy Space

The robot state is characterized by the set of *environmental states* Y that are *consistent* with its *observations* and the vertex v it occupies. Thus, the robot state space is $(2^{\mathbb{N}_m}, \mathcal{V})$. At each state (Y, v), the robot selects an action $a = (e, d)$ from $A_v = I_v \times \{0, 1\}$ to *observe* an edge $(d = 0)$ or to move over an edge $(d = 1)$ unless it terminates denoted by $\emptyset$. A policy maps the robot state space to the set of actions, $\pi : (2^{\mathbb{N}_m}, \mathcal{V}) \to (\cup_{v \in \mathcal{V}} A_v) \cup \emptyset$. Our safety constraint allows $d = 1$ for $e \in I_v$ only if $\mathbb{P}(e|Y) = 1$.

Given a start and goal $s, g \in \mathcal{V}$, the *environmental state space* $\mathbb{N}_m$ is partitioned into $Y_{\text{goal}} = \{i \in \mathbb{N}_m | c(s, g)$ calculated on G_i is finite$\}$ and $Y_{\text{no goal}}$ otherwise. We restrict policies to satisfy the following definition:

Definition 3 (Complete Policy). *A policy π is complete if for any realization it produces a sequence of actions that reach the goal (i.e., a state (Y, g) with $Y \subseteq Y_{\text{goal}}$) or that determine no path exists (i.e., a state (Z, v) with $Z \subseteq Y_{\text{no goal}}$).*

Consider the function $f : ((2^{\mathbb{N}_m}, \mathcal{V}), (\cup_{v \in \mathcal{V}} A_v) \cup \emptyset) \to (2^{\mathbb{N}_m}, \mathcal{V})$ where f updates Y after $d = 0$ and updates v after $d = 1$. Given $i \in \mathbb{N}_m$, a *complete* policy emits a sequence of states and actions $r_1, a_1, r_2, a_2, \ldots, a_T$ where $a_T = \emptyset$. The cost of this sequence is the sum of movement costs $(d = 1)$ and *observation* costs $(d = 0)$ defined by $\text{cost}(\pi|X = i) = \sum_{j=1}^{T-1} \mu(e_j)(1 - d_j) + c(e_j)d_j$.

Remark 2. Note that the domain of the policy has $\mathcal{O}(n2^m)$ states. In the following section we will derive properties that enable a more compact representation.

3.4 The Reactive Planning Problem

The expected cost of a *complete* policy π is found by taking the expectation over the *environmental states*,

$$\mathbb{E}_X(\pi) = \sum_{i \in \mathbb{N}_m} \text{cost}(\pi | X = i)\mathbb{P}(X = i) \; . \tag{1}$$

Problem 1 (Reactive Planning Problem, RPP). Given a graph G, start and goal vertices $s, g \in \mathcal{V}$ and a set of edge subsets $\mathcal{S}$ with corresponding random variable X that has a known probability mass function, find a *complete* policy π that minimizes $\mathbb{E}_X(\pi)$ over induced subgraph G_x for random draw x from X.

4 Properties and Complexity of Reactive Planning

In this section, we establish several properties of robot actions that enable us to efficiently represent *complete* policies along with the complexity of the RPP.

4.1 Action Properties

As the robot moves along a path P in G_x, it gathers *observations* $O_v \subseteq I_v \cup \emptyset$ for all $v \in P$, where $\emptyset$ is used to denote that no observation is taken at v. We define this sequence of actions to be an *observed path*.

Definition 4 (Observed Path). *Given a path $P = v_1, \ldots, v_k$ with observations O_v for all $v \in P$, the observed path is the sequence $O_P = O_{v_1}, \ldots, O_{v_k}$.*

The cost of an *observed path* can be found as the sum of travel costs and *observation* costs along the path: $\text{cost}(O_P) = c(P) + \mu(O_P)$. The robot's understanding of G_x, namely Y, is based on the *observed path* beginning at a starting vertex s. Two important subgraphs can be formed within this understanding.

Definition 5 (Known Subgraph). *Given a set of environmental states Y, the graph $\overline{G} = (\overline{V}, \overline{E}, c, \mu)$ induced by $\overline{E} = \{e | \mathbb{P}(e|Y) = 1\}$ is the known subgraph.*

Definition 6 (Consistent Subgraph). *Given a set of environmental states Y, the graph $\underline{G} = (\underline{V}, \underline{E}, c, \mu)$ induced by $\underline{E} = \{e | \mathbb{P}(e|Y) > 0\}$ is the consistent subgraph.*

The *known subgraph* includes only edges that are sure to exist, while the *consistent subgraph* includes all edges that may still exist. If an *observation* provides new information, then it partitions Y as follows.

Definition 7 (Constructive Observation). *Given environmental states Y, a constructive observation o updates Y to $Y_1 = \{i \in Y \mid o \in E_i \text{ for } \gamma(o) = 1\}$, and $Y_0 = Y \setminus Y_1$ such that $Y_0, Y_1 \neq \emptyset$.*

An *observed path* can be broken into smaller sections called *legs* that start at one *constructive observation* and end at the next *constructive observation*.

Definition 8 (Leg). *Given an observed path O_P, a leg is a subpath of P, namely $v'_1, v'_2, \ldots, v'_y$ such that $v_{i+j} = v'_j$ for $1 \leq j \leq y$ and $0 \leq i \leq k - y$, where the only non-empty observations are $O_{v'_1}$ and $O_{v'_y}$.*

A *leg* can be thought of as a meta-edge between *constructive observations*. Since the robot can move only on edges which are known to exist, a *leg* is composed only of edges which are understood to exist after the *leg's* first *observation* set $O_{v'_1}$. Therefore, a *leg* is a sequence of move actions that join *construction observation* actions.

The order in which *observations* can be visited depends on *observations* to date. The following definition provides a property of an optimal *complete* policy that can react to the environment without re-computation of that policy.

Definition 9 (Reachable). *Given a known subgraph $\overline{G}$ and a vertex v, an observation o is reachable from v if there exists a path from v to o in $\overline{G}$.*

The following result ties the notion of *reachability* to that of *legs* between *constructive observation*.

Lemma 1. *Consider two consecutive constructive observations o_1 and o_2 on a path P. Let Y be the environmental state after o_1. Then, in the known subgraph $\overline{G}$ defined by Y, observation o_2 is reachable from o_1.*

Proof. After $o_1 = (v_1, u_1)$ the understanding of the environment Y is fixed until it gains new information at $o_2 = (v_2, u_2)$. The robot can only select move actions for edges that are known to exist. $\overline{G}$, defined by Y, contains only edges that do not need to be *observed* before traversal; therefore, the robot can only reach o_2 if there exists a path from v_1 to v_2 in $\overline{G}$. □

4.2 Control Policy Properties

We now show how a *complete* policy can be efficiently represented by a binary tree $\pi = (N, L)$. The nodes N are tuples (Y, o) where Y corresponds to the possible *environmental states* prior to *constructive observation* o. The edges are defined by *legs* L between *constructive observations*. Every non-leaf node $(Y, o) \in N$ must have one *leg incident-in* and two *legs incident-out*. The two *legs incident-out* to (Y, o) are *incident-in* to $(Y_0, o_0), (Y_1, o_1) \in N$ corresponding to $\gamma(o) = 0$ and $\gamma(o) = 1$ respectively where $Y = Y_0 \sqcup Y_1$ and $Y_0, Y_1 \neq \emptyset$. This allows real-time reaction in every possible *environmental state* by Lemma 1.

Lemma 2. *A complete policy can be encoded as a binary tree using $\mathcal{O}(nm+m^2)$ space with n the number of G's vertices and m the number of edge subsets.*

Proof. A *constructive observation* has a worst-case partition of Y_0 and Y_1 where one *environmental state* is ruled out (i.e., $|Y_0|$ or $|Y_1|$ is 1). In this case, the robot will need to make $m-1$ *constructive observations* nodes of size $\mathcal{O}(m)$. We know each *leg* connecting these *observations* will visit at most n vertices. Therefore, the policy can be stored as a lookup table of size $\mathcal{O}(nm + m^2)$. □

Remark 3. Note that the policy size scales linearly with m which motivates $m \ll 2^{|\mathcal{E}|}$. A POMDP with nm states and a MDP with $n2^m$ states can be encoded for the RPP, but for our cases this is still very large.

Continuing Example 1, suppose the robot travels to B and *observes* the edge from B to C (o_1) to find $\gamma(o_1) = 0$. The robot knows the *environment state* is 1, and thus it has reached the no goal terminal state. Alternatively, if $\gamma(o_1) = 1$, the robot then travels from B to C to Q and delivers the package. Fig. 2 displays this policy. Let *observation* costs be 0. We can specialize Eq. 1 for *observed paths* labelled $O_{P|X=i}$ for each $i \in \mathbb{N}_m$. The expected cost of a policy π is

$$\mathbb{E}_X(\pi) = \sum_{i \in \mathbb{N}_m} \mathbb{P}(X = i)\text{cost}(O_{P|X=i}) \,. \tag{2}$$

If the PMF of X is $\{0.05, 0.5, 0.45\}$, then the policy in Fig. 2 using Eq. 2 renders expected cost of 14.5. Note: Q is reached without fully knowing x.

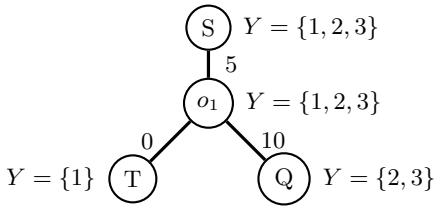

Fig. 2. Weights are the costs of L. T indicates no goal terminal state.

Remark 4. The policy in Fig. 2 satisfies both the *reachability* condition in Lemma 1 and the *constructive observation* property.

4.3 Computational Complexity

Consider this slight variation to the Reactive Planning Problem:

Problem 2 (Probable World Problem, PWP). Given a graph G, a start vertex $s \in \mathcal{V}$ and a set of edge subsets $\mathcal{S}$ with corresponding random variable X that has a known probability mass function, find a policy π that minimizes $\mathbb{E}_X(\pi)$ and identifies induced subgraph G_x for a random draw x from X.

Proposition 1. *The Probable World Problem is NP-Hard.*

Proof. Consider the tuple (X, d, H, ρ, O, Z, r) that defines an instance of IPP from Section 2.2. We will reduce IPP to PWP. Create a graph of vertices $\mathcal{V} = A \cup B$ where A mirrors X and B mirrors O. Let $r = s$. Create an edge subset E_h for every $h \in H$. In every E_h, connect A with edges of cost defined by d. For each $a \in A$ and $b \in B$, add an edge from a to b for subset E_h only if $Z(a, h, b) = 1$. Let random variable X's PMF be in line with ρ. Set $\mu((a, b)) = 0$ for all *observations*. Consider a solution S for PWP. Change each visited vertex of A to X and each *constructive observation* to respective elements of O for a solution S'. The *legs* of S contain no verticies of B as B has no path to *constructive observations*. Given S identifies random draw x, S' identifies true h. Given IPP (perfect sensing) is NP-Hard [11], PWP must be NP-Hard. □

Theorem 1. *The Reactive Planning Problem is NP-Hard.*

Proof. We will prove this result by reducing PWP to RPP. Consider an instance of PWP. Given the graph for PWP, add a set of vertices Q with $|Q| = m$, an intermediary vertex h and a goal vertex g. Connect every $v \in \mathcal{V}$ to h with 0 cost for all $E \in \mathcal{S}$. Let α be the maximum of all traversal and *observation* costs. We can upper bound the expected cost of any optimal policy with $\alpha(mn + m^2)$ by Lemma 2. Connect h bidirectionally with each $q \in Q$ with traversal cost of U for all $E \in \mathcal{S}$ such that $(1 - \mathbb{P}(X = y))U \gg \alpha(mn + m^2)$ where $E_y \in \mathcal{S}$ is the most probable edge subset. Add an edge to $E_i \in \mathcal{S}$ from $q_i \in Q$ to g with cost of 0. In other words, there will only ever be one edge from Q to g, and it is always different for each subset. This new problem is in the form of RPP.

Suppose, by way of contradiction, there existed a solution to this RPP without solving the original PWP. This would imply there were at least two *environmental states* Y *consistent* with the *observations* of an *observed path* (starting at s) of the policy before attempting to reach g. This policy would move the robot to $q_i \in Q$ and *observe* the edge from q_i to g for $i \in Y$ (only exists in E_i). The policy must react to $\gamma((q_i, g)) = 0$. The resulting expected cost is at least $p_i U + (1 - p_i)2U$. Given $(1 - p_i)U \gg \alpha(mn + m^2)$, there exists a policy that can do better as $\alpha(mn + m^2)$ upper bounds the expected cost of an optimal policy providing a contradiction. This shows RPP solves PWP. Given PWP is NP-Hard by Proposition 1, RPP is NP-Hard. □

Remark 5. In the decision version of RPP we are given a budget B and asked to find a *complete* policy with expected cost less than or equal to B. From Lemma 2, it is straightforward to see that the decision version is in NP, and thus is NP-Complete. [14] provides a similar result for R-PPSSR.

5 Scalable Policy Generation

The Reactive Planning Problem seeks information to reach the goal. The robot explores until it is beneficial to exploit the *observed* information and move to the goal. To address this, we maximize weighted mutual information (explore) and establish a condition to prune *observations* (exploit).

5.1 Exploration

Consider RPP. By Lemma 1, information can only be collected at the set of *reachable observations*. To select which *constructive observation* is benificial, we maximize mutual information extended from [11, 16, 17].

Let X_Y encode the probability distribution over environments given a set of *consistent environmental states* Y: The pmf of X_Y is $\mathbb{P}(X_Y = i) = \frac{\mathbb{P}(X=i)}{\sum_{j \in Y} \mathbb{P}(X=j)}$ for each $i \in Y$. Mutual information is the difference between entropy of X_Y and conditional entropy of X_Y given *observation* $o \in R_v$ where R_v is the set of *reachable constructive observations* from (Y, v). Formally,

$$MI(X_Y, o) = H(X_Y) - H(X_Y|o)\,. \tag{3}$$

The entropy $H(X_Y)$ does not depend on o; therefore, this problem can be reduced to minimization of conditional entropy,

$$H(X_Y|o) = -\sum_{j=0}^{1} \mathbb{P}(\gamma(o) = j) \sum_{i \in Y} \mathbb{P}(X_Y = i|\gamma(o) = j) \log\left(\mathbb{P}(X_Y = i|\gamma(o) = j)\right).$$

5.2 Exploitation

The robot must be able to decide when it has collected enough information. We begin with the following inequality from the principle of optimality,

$$c_G(v, g) \leq c_G(v, u) + \mu(o) + c_G(u, g) \quad (u, w) = o \in R_v\,, \tag{4}$$

where the subscript on the cost function c indicates the realization of the environment in which the cost is calculated.

Intuitively, making a measurement and going to the goal is at least as expensive as going straight to the goal in G. The cost calculated in G often performs poorly as an under-estimator for G_x. To address this, a new cost-to-go function is calculated as an expectation over the possible *environmental states* Y. The expected cost-to-go,

$$\mathbb{C}_Y(u, g) = \sum_{i \in Y} c_{G_i}(u, g) \mathbb{P}(X_Y = i)\,, \tag{5}$$

is found for every vertex $u \in \mathcal{V}$. To calculate $c_{G_i}(u, g)$, the edges are flipped in each G_i and a shortest path algorithm is run from g to all other $u \in V_i$. If $c_i(u, g)$ is infinite, we set such costs to zero as the robot will not travel any further (i.e., no goal terminal state).

Eq. 4 is augmented to include the current environmental understanding and the expected cost-to-go. The pruning condition can be written as

$$c_{\overline{G}}(v, g) \leq c_{\overline{G}}(v, u) + \mu(o) + \mathbb{C}_Y(u, g) \quad (u, w) = o \in R_v\,. \tag{6}$$

If going straight to the goal is less expensive than gaining information and going to the goal, the information should not be collected.

Lemma 3. *Given a robot state r with constructive observations R_v, if all $o \in R_v$ satisfy Eq. 6, the robot should move to the goal.*

Proof. Consider $c_{\overline{G}}(v, g) = \infty$. This implies there is no known path to goal. No *observation* satisfies Eq. 6, so this trivially holds. Now, consider the case where enough information has been gathered to $r = (Y, v)$ for $c_{\overline{G}}(v, g) < \infty$. If all $o \in R_v$ satisfy Eq. 6, the known cost of making any *observation* and the expected cost-to-go is more than the known cost to complete the task. Thus, the robot should move to the goal. □

Lemma 4. *The expected cost-to-go from the start, $\mathbb{C}_{\mathbb{N}_m}(s, g)$, forms a lower bound on the expected cost of any policy π.*

Proof. Consider any two *environmental states* $i, j \in \mathbb{N}_m$. If G_i and/or G_j do not have paths to the goal, the robot must identify the *environmental state* and return no goal terminal state. To do this, the robot uses an *observed path* to gain the information. The cost of such a path is at least 0. The expected cost-to-go for these cases is always 0. Suppose G_i and G_j can both reach the goal. There is at least one *leg* the robot must travel for both G_i and G_j. The expected cost-to-go selects the optimal paths independently. Therefore, the expected cost of the *observed paths* from π for i and for j can never be less than the expected cost-to-go, even if the robot acts optimally otherwise. □

Remark 6. We find this bound performs well in practice (See Section 6). Although, the bound performs poorly when $\mathbb{P}(X \in Y_{\text{no goal}})$ and *observation* costs are large relative to $\mathbb{P}(X \in Y_{\text{goal}})$ and travel costs respectively since they are unaccounted for in $\mathbb{C}_{\mathbb{N}_m}(s, g)$.

5.3 Combining Exploration and Exploitation

Information gain and motion to goal can be combined as a function of the exploration metric $H(X_Y|o)$ and the exploitation metric $c_{\overline{G}}(v, u) + \mu(o) + \mathbb{C}_Y(u, g)$.

Weighted conditional entropy [19] is a well-studied method for combining entropy with a second metric, and thus in the following presentation and simulations we take the product of the two metrics and select *observations* satisfying

$$o_{\min} = \underset{(u,w)=o \in R_v}{\operatorname{argmin}} \; (c_{\overline{G}}(v, u) + \mu(o) + \mathbb{C}_Y(u, g)) H\left(X_Y|o\right) \; . \tag{7}$$

The product of the exploitation and exploration terms is non-negative. A value of 0 is achieved when o fully determines Y, or when traversal, *observation*, and the expected cost-to-go all have 0 cost.

5.4 Algorithm

Algorithm 1 minimizes every *leg* based on Eq. 7. It calls $Reachable(G, \mathcal{S}, (Y, v))$ which computes the minimum path lengths $d[u]$ from v to all other vertices u in the *known subgraph* $\overline{G}$. The set R_v is formed from edges that may or may not exist which render finite path cost from v, i.e. $R_v = \{(u, w) = e | \exists i, j \in Y \text{ s.t. } e \in E_i, e \notin E_j, d[u] \neq \infty\}$ with their corresponding distances D_v.

Algorithm 1: RPP Minimization of Conditional Entropy Policy

Data: Graph G, edge subsets $\mathcal{S}$, vertices s & g, states $\mathbb{N}_m$, probabilities p
Result: policy π for RPP and expected cost lower bound L

1 Compute $c_{G_i}(g, v)$ for all $v \in V$ and $i \in \mathbb{N}_m$;
2 Let Q contain only $(\mathbb{N}_m, s)$;
3 **while** Q *not empty* **do**
4 Remove (Y, v) from Q;
5 d = dijkstra($\underline{G}, v$);
6 **if** $d[g] = \infty$ **then**
7 Mark π, at v for Y, no goal terminal state;
8 **else**
9 Compute $(R_v, D_v) = Reachable(G, \mathcal{S}, (Y, v))$;
10 Remove elements of R_v that satisfy Eq. 6;
11 **if** $|R_v| = 0$ **then**
12 Add *leg* from v to g, marked goal terminal state, to π;
13 **else**
14 Let $o = (u', w')$ be the minimum of Eq. 7;
15 Add *leg* from v to u' and node (Y, o) to π;
16 Add $(Y_{\gamma(o)=0}, u')$ and $(Y_{\gamma(o)=1}, u')$ to Q;
17 Let $L = \mathbb{C}_{\mathbb{N}_m}(s, g)$;
18 Return π and L;

Remark 7. The runtime is dominated by the m calls to Dijkstra's Algorithm, which gives $\mathcal{O}(m(|\mathcal{V}|+|\mathcal{E}|)\log|\mathcal{V}|)$ (priority queue implemented as a binary heap).

The biased cost from Eq. 7 and the pruning condition from Eq. 6 complement each other to provide incentive toward the goal. The biased cost encourages *observation* selection closer to the goal. Once enough information is gained, the pruning condition removes information that is not important for the task. When the pruning condition removes all *observations* from R_v, the robot makes its way to the goal.

Remark 8. The policy π, returned by Algorithm 1, is independent of the order states are removed from Q in line 4 (potential for parallel computation).

To avoid learning the *environmental state* when it is impossible to reach the goal, the *consistent subgraph* $\underline{G}$ is checked for a path to goal. If the distance from the current location to g is infinite, there is no point continuing.

Theorem 2. *Algorithm 1 returns a complete policy.*

Proof. Suppose by contradiction, Algorithm 1 did not return a *complete* policy. This would imply either it terminates at (Y, g) for $Y \subseteq Y_{\text{no goal}}$ (false positive) or it terminates a (Z, v) for $v \neq g$ and $\exists z \in Z$ s.t. $z \in Y_{\text{goal}}$ (false negative).

False positive: Algorithm 1 must have directed the robot to travel on an edge that does not exist because $Y \subseteq Y_{\text{no goal}}$. Since the environment does not have

a path to goal, $\underline{G}$ for $Y \subseteq Y_{\text{no goal}}$ will not have a path to goal. Line 6 directly catches this case.

False negative: Algorithm 1 would not be able to find a path in $\underline{G}$, but since the *environmental state* z is still possible, $\underline{G}$ for Z will have a path to goal. This contradicts the fact that π is marked no goal terminal state in Line 7. □

6 Simulation Results

In this section we provide simulation results on a large scale practical example and on randomly generated environments. Tests where run on a single Intel Core i7-6700 at 3.4GHz.

6.1 Flexible Factory

Flexible factories often spend considerable downtime between contracts due to changes in infrastructure and machinery. Consider Fig. 3 as a simple flexible factory that produces D items per hour. We are interested in knowing if the robot can move this volume. The dashed vertices indicate areas that require heavy use. For clarity, in Table 1 the column labelled "Vertex Obstructions" indicates the properties of the environment obstruction. For instance, in region 0 (vertices labelled 0) up to two vertices may be missing from the graph. Regions 1 and 2 each contain one forklift obstruction (which corresponds to removing the two adjacent vertices it occupies). When regions 5 and 6 are obstructed, all other vertices exist.

We cast this as a Reactive Planning Problem by enumerating all combinations of the obstructed vertices and removing their *incident* edges. This generates $m = 34561$ edge subsets each with a corresponding probability. We compute policies from S to A, from S to B, from A to S, and from B to S. The robot can move faster when not loaded, so the movement costs of A to S and B to S are decreased by a factor of 2.

We implemented a mixed integer linear program for RPP, but thus far have not been able to scale to problems of this size (in our formulation, the number of variables scales with m^3). Instead, we will allow re-planning during the online phase for comparison purposes (note these results do not have constant action lookup time). We compare against A* and maximum probability of success ($\mathbb{P}_s$). Both approaches generate a path, which we follow until it is obstructed. Then we take the edges of the path as *observations* and use this new information to re-plan. This is completed for every realization $x \in X$. The cost of the corrected paths from s to a terminal state and X's PMF are used to calculate the expected cost found in Table 2. Note that the proposed algorithm provides significantly lower expected cost than the two comparison policies, and in three of the four cases, it also provides a lower variance. In addition, for each task the expected cost of Algorithm 1 is within 30% of the lower bound (L) on the optimal cost.

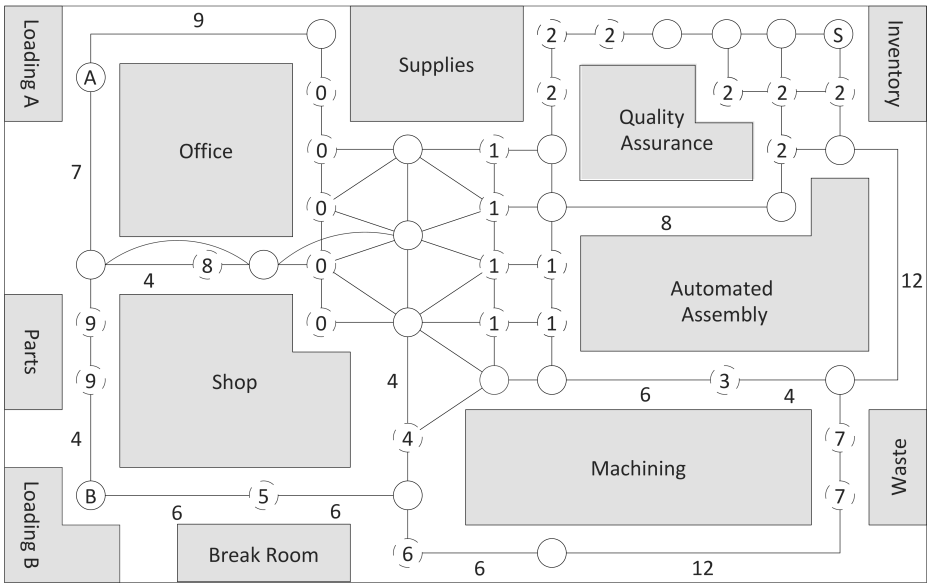

Fig. 3. Flexible Factory model: Dashed vertices may be obstructed. Edges with cost of 2 or 3 are not labelled for simplicity. The curved edges cost 2 more for waiting.

Table 1. Flexible Factory model parameters used in simulations.

Regions	Vertex Obstructions	Obstruction Probability	*Observation* Cost
0	≤ 2 independent	uniform over combinations	0.5
1,2	2 adjacent vertices	uniform over combinations	0.25
3,4	1	0.3	0.25
5,6	both or none	0.02	0.5
7,9	≤ 1 independent	0.1	0.5
8	1	0.4	0.5

Table 2. Flexible Factory simulation results. $|N|$ gives the number of nodes in the binary tree policy, $\mathbb{V}_X$ denotes the variance, Time indicates duration to compute the policy, and L denotes the lower bound.

	Algorithm 1					A*		Max $\mathbb{P}_s$	
Task	$\lvert N\rvert$	L	$\mathbb{E}_X(\pi)$	$\mathbb{V}_X(\pi)$	Time (s)	$\mathbb{E}_X$	$\mathbb{V}_X$	$\mathbb{E}_X$	$\mathbb{V}_X$
S→A	114	39.0	**42.5**	**12.6**	461	47.7	74.5	85.3	659
S→B	6869	41.4	**50.0**	**164.2**	739	50.1	282.0	61.7	206
A→S	84	19.5	**23.8**	**6.6**	510	30.2	81.9	43.6	135
B→S	175	20.7	**26.1**	51.3	497	28.3	**30.3**	33.9	135

6.2 Testing via Random Environment Generation

Since A* significantly outperformed the policy that maximizes the probability of success, we test Algorithm 1 against A* for a sequence of random problems. For simplicity the test graph is a grid where vertices have edges to move left, right, up and down (unless on the boundary). Travel costs are 1 and *observation* costs are drawn from the uniform random variable W on $[0, 1]$. Then edge subsets are generated by removing edges randomly. To do this, we incrementally relax the removal of edges on an edge subset until a path to goal exists for 950 cases and vice versa for no path to goal for another 50 cases ($m = 1000$). The robot starts at cell $(2, 2)$ with the goal located in cell $(\text{width} - 2, \text{depth} - 1)$. Finally, X's PMF is formed by normalizing m random draws of W.

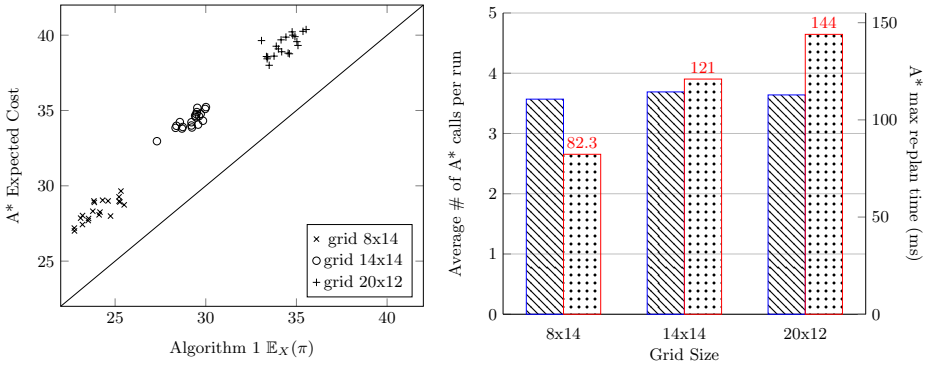

Fig. 4. Results of random environment generation.

The results in Fig. 4 show Algorithm 1 consistently outperforms A*. We also show the number of times A* re-plans and the maximum amount of time spent in a re-plan. The average runtimes, in seconds, of Algorithm 1 were 161, 302 and 569 for 8x14, 14x14 and 20x12 respectively. The average gaps between Algorithm 1's expected cost and L were 6.3, 5.9 and 6.0, meaning Algorithm 1 typically provides solutions within 25% of the optimal. We ran one 40x40 grid to test timing in a larger environment. The max re-plan time for A* was over 1(s) which is not viable for many real-time applications (Algorithm 1: 3349(s)).

7 Conclusion

A reactionary *complete* policy based on environmental *observations* was presented that finishes a task or identifies it is impossible to finish. For future work, we wish to remove the requirement of prior environmental knowledge so the robot may learn environmental trends during repetitive tasks. We are also interested in allowing multiple termination conditions (including multiple goals), faulty sensor models and extending Algorithm 1 to parallel computation.

References

1. Binney, J., Sukhatme, G.S.: Branch and bound for informative path planning. In: ICRA, Citeseer (2012) 2147–2154
2. Yu, J., Schwager, M., Rus, D.: Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In: International Conference on Intelligent Robots and Systems, IEEE (2014) 342–349
3. Hollinger, G.A., Englot, B., Hover, F.S., Mitra, U., Sukhatme, G.S.: Active planning for underwater inspection and the benefit of adaptivity. The International Journal of Robotics Research (2012) 3–18
4. Javdani, S., Chen, Y., Karbasi, A., Krause, A., Bagnell, D., Srinivasa, S.S.: Near optimal bayesian active learning for decision making. In: AISTATS. (2014) 430–438
5. Dames, P., Schwager, M., Kumar, V., Rus, D.: A decentralized control policy for adaptive information gathering in hazardous environments. In: 51st IEEE Conference on Decision and Control (CDC), IEEE (2012) 2807–2813
6. Bhattacharya, S., Ghrist, R., Kumar, V.: Persistent homology for path planning in uncertain environments. IEEE Transactions on Robotics **31**(3) (2015) 578–590
7. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of markov decision processes. Mathematics of operations research **12**(3) (1987) 441–450
8. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
9. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2) (1968) 100–107
10. Koenig, S., Likhachev, M.: Fast replanning for navigation in unknown terrain. IEEE Transactions on Robotics **21**(3) (2005) 354–363
11. Lim, Z.W., Hsu, D., Lee, W.S.: Adaptive informative path planning in metric spaces. The International Journal of Robotics Research **35**(5) (2015) 585–598
12. Andreatta, G., Romeo, L.: Stochastic shortest paths with recourse. Networks **18**(3) (1988) 193–204
13. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Computer Science **84**(1) (1991) 127–150
14. Polychronopoulos, G.H., Tsitsiklis, J.N.: Stochastic shortest path problems with recourse. Networks **27**(2) (1996) 133–143
15. Issac, P., Campbell, A.M.: Shortest path problem with arc failure scenarios. EURO Journal on Transportation and Logistics (2015) 1–25
16. Dames, P.M., Schwager, M., Rus, D., Kumar, V.: Active magnetic anomaly detection using multiple micro aerial vehicles. IEEE Robotics and Automation Letters **1**(1) (2016) 153–160
17. Charrow, B., Kumar, V., Michael, N.: Approximate representations for multi-robot control policies that maximize mutual information. Autonomous Robots **37**(4) (2014) 383–400
18. Chakaravarthy, V.T., Pandit, V., Roy, S., Awasthi, P., Mohania, M.: Decision trees for entity identification: approximation algorithms and hardness results. In: Proceedings ACM Symp. on Principles of Database Systems, ACM (2007) 53–62
19. Suhov, Y., Stuhl, I., Sekeh, S.Y., Kelbert, M.: Basic inequalities for weighted entropies. Aequationes mathematicae (2015) 1–32