Noname manuscript No.
(will be inserted by the editor)

Active Sensing for Motion Planning in Uncertain Environments via Mutual Information Policies

Ryan A. MacDonald · Stephen L. Smith

the date of receipt and acceptance should be inserted later

Abstract This paper addresses path planning with real-time reaction to environmental uncertainty. The environment is represented as a robotic roadmap, or 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, modelling the likelihood of certain obstacles in the environment. The robot can locally sense the environment, and at a vertex, can determine the presence or absence of some subset of edges. Within this model, the Reactive Planning Problem 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 are guaranteed to navigate the robot to the goal, or determine no such path exists). We prove that the problem 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. We test the performance of the policy and the lower bound against that of the optimal policy and explore the effects of errors in the robot's prior information on performance. Simulations are run on a flexible factory scenario to demonstrate the scalability of the proposed approach. Finally, we present a method to extend this solution to robots with faulty sensors.

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 (Binney & Sukhatme 2012, Dames et al. 2012, Hollinger et al. 2012, Javdani et al. 2014, Kaelbling & Lozano-Pérez 2013, Yu et al. 2014). 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

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

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.

1.1 Related Work

In robotics, there are several effective methods for dealing with uncertainty. Point-to-point motion is addressed by Bhattacharya et al. (2015) using persistent paths, which maximize the probability of success. However, if the computed path is obstructed, the robot ends without finishing the task (i.e., failure). To avoid failure, Partially Observable Markov Decision Processes (POMDPs) can be used to compute reactive motion policies (Bai et al. 2014, Chen et al. 2016, Kaelbling & Lozano-Pérez 2013, Van Den Berg et al. 2012). A POMDP selects actions based on partially observed states, but the computation of policies is in general a PSPACE-Complete problem (Papadimitriou & Tsitsiklis 1987). 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 (LaValle 2006). To avoid the computational complexity, algorithms like lifelong planning A* and D* lite allow the robot to replan during execution (Koenig & Likhachev 2005, Koenig et al. 2004) for the case when the robot's location is fully observable. Replanning is also used by Kaelbling & Lozano-Pérez (2013) in their more general problem to form a compact policy, which is followed until the robot transitions to a state outside of the policy and triggers a replan. These replanning phases often provide much needed space complexity savings. In contrast, this work targets complete policies in which all reachable robot states are contained in a compact policy, and thus, the robot does not need to replan during execution.

The Informative Path Planning (IPP) problem is studied in several works (Javdani et al. 2014, Lim et al. 2015, Yu et al. 2014), 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 (Hollinger et al. 2012) to maximizing information from start to goal (Binney & Sukhatme 2012). Similarly, active sensing (Wang et al. 2016) and active perception (Best et al. 2016) allow autonomous robot(s) to intelligently collect data based on prior observations. These works plan policies or paths prior to deployment of the robot and react to new information collected by the robot, where the robot's 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 by Andreatta & Romeo (1988) provides possible obstacle locations, but assumes obstacles locations are such that there always exists a path to goal. The CTP, in which no prior information on obstacles is given, is a PSPACE-Complete problem (Papadimitriou & Yannakakis 1991). Remote sensing is added to the CTP by Bnaya et al. (2009) where the agent pays a sensing cost dependent on its location to determine the absence/presence of a particular obstacle. The authors look to minimize the sum of sensing and traversal costs and present a value of information model to decide when to use remote sensing. This decision is made for single obstacles where as in our work, the prior information available to the robot allows decisions with correlation between obstacles. To make the CTP more tractable, Polychronopoulos & Tsitsiklis (1996) introduce the problem R-SSPPR, in which the robot is operating in one of a finite number of different graphs (realizations), each defined over the same set of vertices and edges, but with different edge costs. The robot observes outgoing edge costs at each vertex and the goal

is to minimize the expected cost from start and goal. They present an optimal dynamic program as well as a feedback heuristic. In contrast, our work considers different graph topologies for each realization, and a generalized sensing model along with sensing costs. In transportation research, Issac & Campbell (2015) presents an integer linear program to solve a blocked route problem, in which they select a primary path and then switch to a secondary path when the primary fails. In contrast, we compute policies that minimize the expected cost for a robot to reach 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 (Charrow et al. 2014, Dames et al. 2016, 2012, Lim et al. 2015). Julian et al. (2014) shows that by maximizing mutual information within a mapping task, the robot is eventually attracted to unexplored regions. Hollinger & Sukhatme (2014) uses mutual information to generate a cost constrained path for an information collection task. Dames et al. (2012) presents a mutual information gradient controller, where multiple robots search for targets and avoid hazards. We use mutual information to quantify the robot's value of a sensing action and combine it with the cost of attaining this information in order to select the next action.

1.2 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 policy for 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.

A preliminary version of this work appeared in WAFR 2016 (MacDonald & Smith 2016). Some key contributions in comparison to this initial work are that we now provide a general sensor model, we present a method to compute the optimal policy, and we provide an extension to a class of faulty robot sensors. We also compare our policy to the optimal in simulation, and we explore the effects of inaccurate prior data on performance.

1.3 Organization

The organization of this paper is as follows. 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. An extension to robots with faulty sensor models is presented in Section 6. Finally, simulation results are provided in Section 7.

2 Background

In this section we briefly review graph terminology and the informative path planning problem by Lim et al. (2015).

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 $\mathcal{V}$ is the set of vertices that are connected by the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, 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 $e = (v, u) \in \mathcal{E}$ 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

Lim et al. (2015) defines the Informative Path Planning (IPP) problem under noiseless observation 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 Lim et al. (2015), IPP is NP-hard as it contains the optimal decision tree problem (Chakaravarthy et al. 2007) 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 (i.e., a robotic roadmap). The robot and environment models are defined using a weighted directed graph $G = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V}$ is a set of locations in the robot configuration space and $\mathcal{E}$ is the set of paths between configurations. The function c captures the costs of motion, and for each $e \in \mathcal{E}$ the value $c(e) \in \mathbb{R}_{\geq 0}$ defines the robot's cost for traversing the corresponding path. 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 an edge is free to traverse, it senses the edge, incurs a sensing cost and traverses it only if the outcome is unblocked. This is formalized in Section 3.2.

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 of edges $E_i \subseteq \mathcal{E}$ for $i \in \mathbb{N}_m$. The edge subset E_i captures the obstacle-free robot transitions in environment i. The robot is given the set of possible edge subsets $\mathcal{S} = \{E_1, \ldots, E_m\}$ along with a probability mass function (pmf) 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* $G_x = (V_x, E_x, c_x)$ where $c_x(e) = c(e)$ for all $e \in E_x$; 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 does not imply the absence or presence of any 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.

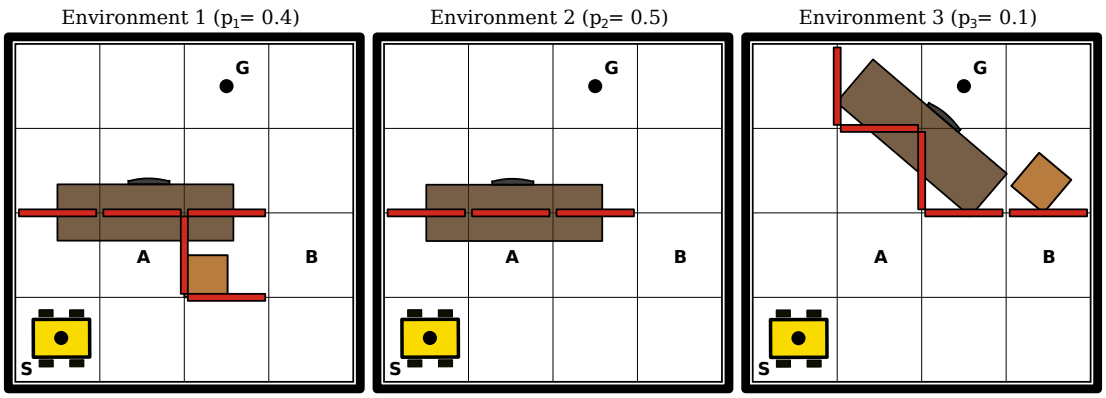

Fig. 1 Simple example environment where diagonal cells are not connected. Bars between cells represent an obstacle which implies the robot cannot traverse between these cells directly. Cells A and B are labelled for future examples.

Example 1 To illustrate the problem, consider Fig. 1 as a simplified model of a small room. A robot is tasked with delivering a package to cell G, and it starts in cell S. There is a large desk blocking 3 edges in both environment 1 and 2 but in environment 3 the desk now blocks off cell G. Environment 1 and 3 also has a small box next to the desk further obstructing traversal.

3.2 Robot Model

When the robot is located at vertex v, it has a finite set of *observations* $\Theta_v = \{O_1, O_2, \ldots\}$ where each $O \in \Theta_v$ is a subset of $\mathcal{E}$ (i.e., $O \subseteq \mathcal{E}$). Given an *observation* $O \in \Theta_v$, the function μ, with value $\mu(O) \in \mathbb{R}_{\geq 0}$, captures the cost of sensing which edges (i.e., paths between configurations) of O are free to traverse. Both Θ_v, for all $v \in V$, and μ are provided a priori, and we restrict Θ_v to satisfy $I_v \subseteq \cup_{O \in \Theta_v} O$. This ensures the robot may check if an outgoing edge from v is traversable. Some example *observation* models include limiting each *observation* to one edge (i.e., $\Theta_v = I_v$ for all $v \in \mathcal{V}$) or creating an omnidirectional sensor with one edge range (i.e., $\Theta_v = \{I_v\}$ for all $v \in \mathcal{V}$), which we call the *single hop* or *all neighbors* model. Fig. 2 shows three possible sensor models within Environment 1 from Example 1.

If the robot wishes to make an *observation* $O \in \Theta_v$, it pays $\mu(O)$, and it is returned an *outcome* as the subset of edges in O that are free to traverse in E_x for environment realization G_x.

Definition 1 (Observation-Outcome) Given a graph $(\mathcal{V}, \mathcal{E})$, a vertex $v \in \mathcal{V}$ and observation $O \in \Theta_v$ with $O \subseteq \mathcal{E}$, an observation is mapped to an outcome by $O \mapsto O \cap E_x$. The robot must occupy v to attain $O \cap E_x$.

Observations with respective *outcomes* allow the robot to rule out *environmental states*. If an *observation* contains edge e but its *outcome* does not, the robot knows all edge subsets containing e cannot be correct. Similarly, if the *outcome* contains e, all edge subsets missing e cannot be correct.

Definition 2 (Consistent) Given a set of observation-outcome pairs $\mathcal{O}$, an edge subset E is consistent with $\mathcal{O}$ if and only if $O \cap E = O \cap E_x$ for each $(O, O \cap E_x) \in \mathcal{O}$.

We define $Y \subseteq \mathbb{N}_m$ to be the set of *environmental states consistent* with *observation-outcome* pairs $\mathcal{O}$ collected by the robot. To avoid collisions with an obstacle in the environment, we impose the restriction that an edge e can be traversed only when the probability it is unblocked equals one, namely

$$\mathbb{P}(e|Y) = \frac{\sum_{i \in Y} \mathbb{P}(X = i \cap e \in E_i)}{\sum_{j \in Y} \mathbb{P}(X = j)} = 1 \,. \tag{1}$$

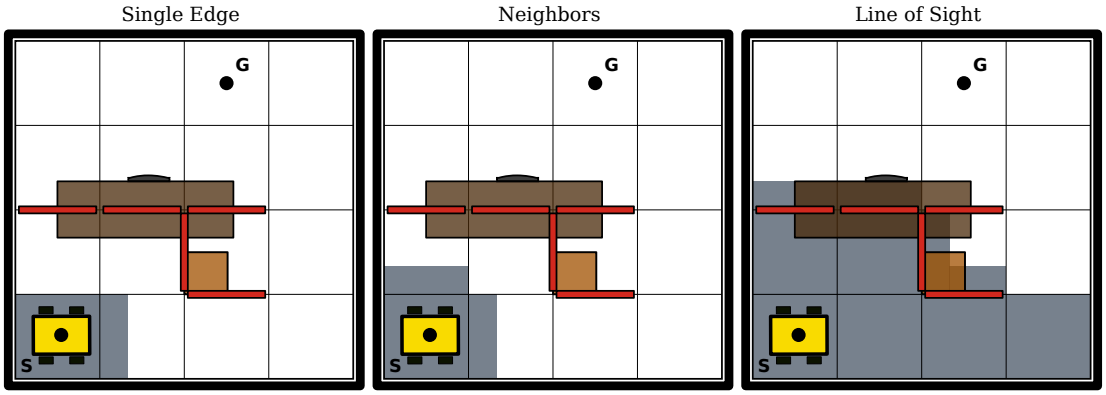

Fig. 2 Example *observation* models: single outgoing edges (left), all neighbors/single hop (middle) or line of sight (right). Edges between shaded cells are sensed.

If $e \in \mathcal{E}$ may be blocked (i.e., $\mathbb{P}(e|Y) \in (0,1)$), the robot can take an *observation* O with $e \in O$, incur *observation* cost $\mu(O)$ and proceed across the edge when e is in the *outcome*. Given a belief Y and *observation* O, the set of all possible *outcomes* is defined by $\Gamma_{(Y,O)} \equiv \{E \subseteq \mathcal{E} | E = O \cap E_i \text{ for } i \in Y\}$.

3.3 Policy Space

The robot state is characterized by the set of *environmental states* $Y \subseteq \mathbb{N}_m$ that are *consistent* with its *observation-outcome* pairs 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 from $(2^{\mathcal{E}}, \{\text{observe, move, terminate}\})$. The *observations* available to the robot are $(O, \text{observe})$ for each $O \in \Theta_v$. The move actions available to the robot are (e, move) where $e \in I_v$ and e is obstacle-free (i.e., $\mathbb{P}(e|Y) = 1$). Finally, the robot can terminate using the action $(\emptyset, \text{terminate})$. Thus, a policy π maps the robot state space to the set of actions, $\pi : (2^{\mathbb{N}_m}, \mathcal{V}) \to (2^{\mathcal{E}}, \{\text{observe, move, terminate}\})$.

Given a start and goal $s, g \in \mathcal{V}$, the *environmental state space* $\mathbb{N}_m$ is partitioned into $Y_{\text{goal}} = \{i \in \mathbb{N}_m \mid c(s,g) \text{ calculated on } G_i \text{ is finite}\}$ 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 finite 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 (Y, v) with $Y \subseteq Y_{\text{no goal}}$).

Note: There are environments for which no *complete* policy exists. Consider Fig. 3 with a single edge *observation* model, $\Theta_v = I_v$. The robot must move to A or B in order to identify the *realization* in which it resides. If the robot arrives at A and the edge to g is obstructed, then it must terminate, yet there still exists a path to goal (namely s to B to g). The same issue occurs if the robot instead initially travels to B. The following is a sufficient condition for a complete policy to exist. Given G_i for any $i \in \mathbb{N}_m$, the connected component containing the start must be strongly connected. In other words, the robot can always opt to retreat to the start.

A policy π defines a state transition function $f : (2^{\mathbb{N}_m}, \mathcal{V}) \times (2^{\mathcal{E}}, \{\text{observe, move, terminate}\}) \to (2^{\mathbb{N}_m}, \mathcal{V})$ where f updates Y after the observe command and updates v after the move command. Given a *realization* $X = x$ where x is drawn from $\mathbb{N}_m$ according to the pmf, a policy π emits a sequence of states and actions

$$(\mathbb{N}_m, s), a_1, (Y_2, v_2), a_2, \ldots, (Y_T, v_T), (\emptyset, \text{terminate})$$

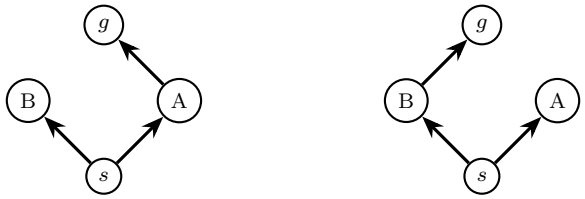

Fig. 3 Subgraph G_1 left with $\mathbb{P}(X = 1) = 0.5$ and subgraph G_2 right with $\mathbb{P}(X = 2) = 0.5$.

for some finite number T. The cost of an action a is

$$\text{cost}(a) = \begin{cases} c(e) & \text{if } a = (e, \text{move}) \\ \mu(O) & \text{if } a = (O, \text{sense}) \\ 0 & otherwise \end{cases} .$$

Given random draw x, the total cost incurred using π is given by,

$$\text{cost}(\pi | X = x) = \sum_{i=1}^{T} \text{cost}(a_i) . \tag{2}$$

Remark 1 (Policy Domain) Note that the domain of the policy has $n2^m$ states. In Section 4 we 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*,

$$\mathbb{E}_X(\pi) = \sum_{x \in \mathbb{N}_m} \text{cost}(\pi | X = x) \mathbb{P}(X = x) . \tag{3}$$

Problem 1 (Reactive Planning Problem, RPP) Given a graph G, start and goal vertices $s, g \in \mathcal{V}$, a set of edge subsets $\mathcal{S}$ with corresponding random variable X that has a known probability mass function and observations Θ_v for all $v \in \mathcal{V}$, 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 Reactive Planning Problem.

4.1 Action Properties

As the robot moves along a path P in G_x, it takes a set of *observations* $\mathcal{O}_v \subseteq \Theta_v \cup \emptyset$ at each vertex $v \in P$, where $\emptyset$ is used to denote that no *observation* is taken at v. We define this sequence of these sets of *observations* to be an *observed path*.

Definition 4 (Observed Path) Given a path $P = v_1, \ldots, v_k$ with observations $\mathcal{O}_v$ for all $v \in P$, the observed path is the sequence $\mathcal{O}_P = \mathcal{O}_{v_1}, \ldots, \mathcal{O}_{v_k}$.

The cost of an *observed path* can be found as the sum of travel costs and *observation* costs along the path:

$$\text{cost}(\mathcal{O}_P) = c(P) + \sum_{i=1}^{k} \sum_{O \in \mathcal{O}_{v_i}} \mu(O).$$

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 $\overline{G} = (\overline{V}, \overline{E}, c)$ induced by $\overline{E} = \{e \mid \mathbb{P}(e|Y) = 1\}$ is the known subgraph.

Definition 6 (Consistent Subgraph) Given a set of environmental states Y, the graph $\underline{G} = (\underline{V}, \underline{E}, c)$ induced by $\underline{E} = \{e \mid \mathbb{P}(e|Y) > 0\}$ 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. These graphs are updated as the robot collects *constructive observation-outcome* pairs of the environment. We say an *observation* is *constructive* if there are at least two different, possible *outcomes*.

Definition 7 (Constructive Observation) Given environmental states Y, an observation O is constructive if there exists $i, j \in Y$ such that $O \cap E_i \neq O \cap E_j$.

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 $\mathcal{O}_P$, a leg is a subpath of P, namely $v_i, v_{i+1}, \ldots, v_j$ where $\mathcal{O}_{v_i}$ and $\mathcal{O}_{v_j}$ are constructive observations, and each $\mathcal{O}_{v_{i+1}}, \ldots, \mathcal{O}_{v_{j-1}}$ is an empty set.

A *leg* can be thought of as a meta-edge between *constructive observations*. Since the robot can move only on edges that contain no obstacles, a *leg* is composed only of edges which are understood to be unobstructed after the *leg's* first *observation* set $\mathcal{O}_{v_i}$. Therefore, a *leg* is a sequence of move actions that join *constructive observation* actions.

The order in which *observations* can be visited depends on *observation-outcome* pairs 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 $O \in \Theta_u$ is reachable from v if there exists a path from v to u in $\overline{G}$.

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

Lemma 1 *Consider two consecutive constructive observations O_1 and O_2 on a path P. Let (Y, v) be the robot state after action $(O_1, \textit{observe})$ collects outcome $O_1 \cap E_x$. Then, in the known subgraph $\overline{G}$ defined by Y, observation O_2 is reachable from v.*

Proof After $O_1 \cap E_x$ the understanding of the environment, namely Y, is fixed until the robot gains new information at O_2. With no loss of generality, let $O_2 \in \Theta_u$ for *observation* location $u \in \mathcal{V}$. The robot can only select move actions for edges that cannot be blocked given Y. $\overline{G}$, 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 to u in $\overline{G}$. □

4.2 Control Policy Properties

We now show how a *complete* policy can be efficiently represented by a tree. The nodes of the tree are tuples (Y, O) where Y corresponds to the *consistent environmental states* prior to *constructive observation* O. The edges of the tree are defined by *legs* between *constructive observations*. Every non-leaf node (Y, O) must have one *leg incident-in* and $|\Gamma_{(Y,O)}|$ *legs incident-out*. The *incident-out legs* connect (Y, O) to (Y_E, O') for $E \in \Gamma_{(Y,O)}$ where $Y_E = \{i \in Y | E = O \cap E_i\}$. The robot knows which *leg* to traverse given *outcome* $O \cap E_x$ matches $E \in \Gamma_{(Y,O)}$. Informally, the tree stitches together the *observed paths*, starting from vertex s, for each $i \in \mathbb{N}_m$ until *observation-outcome* pairs disagree at which point the tree branches. This allows real-time reaction in every possible *environmental state* by Lemma 1. We now discuss the space complexity of this encoding.

Lemma 2 *Any two nodes $n_1 \neq n_2$ where n_1 is not an ancestor or descendent of n_2 satisfy $Y_1 \cap Y_2 = \emptyset$ (i.e., the realization at n_1 and at n_2 must be different).*

Proof Let (Y, O) be the youngest ancestor of both node n_1 and node n_2. Formally, $Y_1 \cup Y_2 \subseteq Y$ s.t. $Y \subset Y'$ for all other ancestors (Y', O'). Consider two *outcomes* $E^1, E^2 \in \Gamma_{(Y,O)}$ such that $E_i \cap O = E^1$ for all $i \in Y_1$ and $E_j \cap O = E^2$ for all $j \in Y_2$. If $E^1 = E^2$, there exists a node (Y', O') such that $Y' \subset Y$ and $Y_1 \cup Y_2 \subseteq Y'$, but this contradicts the definition of (Y, O). Therefore, $E^1 \neq E^2$ implies $i \neq j$ for any $i \in Y_1$ and any $j \in Y_2$ (i.e., $Y_1 \cap Y_2 = \emptyset$). □

Lemma 3 *A complete policy can be represented as a tree with $m-1$ nodes using $\mathbb{O}(nm + m^2)$ space where n is the number of vertices in G and m is the number of edge subsets.*

Proof The worst case encoding requires the robot to always learn random draw x before terminating. Given Lemma 2, we can bound the number of *constructive observations* the robot makes by $m-1$. Each *observation* is a node requiring $\mathbb{O}(m)$ space to encode Y and O. We know the lowest cost *leg* connecting these *observations* will visit at most n vertices because non-negative traversal cost allows a path without cycles to always be minimum cost. Therefore, the policy can be stored as a lookup table of size $\mathbb{O}(nm + m^2)$. □

Remark 2 (Policy Encodings) The policy size scales 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's *observations* are $\Theta_v = \{I_v\}$ for all $v \in \mathcal{V}$ with zero cost. Let moving between cells cost 1. Consider the two polices presented in Fig. 4. The robot moves to A or B and collects I_A or I_B respectively. Using Eq. 3, the left and right policies in Fig. 4 render expected costs of 6.7 and 7.3 respectively. These policies satisfies both the *reachability* condition in Lemma 1 and the *constructive observation* property, and note the left policy allows the robot to reach G without fully knowing x.

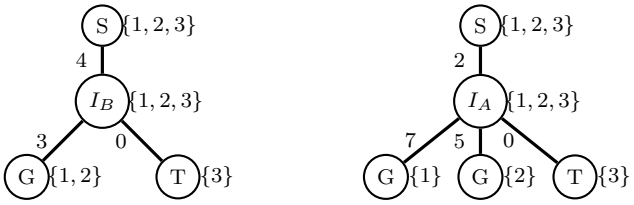

Fig. 4 Edge labels are the *leg* costs and node labels encode Y. T indicates no goal terminal state.

4.3 Computational Complexity

To establish the complexity of the reactive planning problem, we begin by considering the following variant.

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 and observations Θ_v for all $v \in \mathcal{V}$, find a policy π that identifies the induced subgraph G_x for a random draw x from X and minimizes $\mathbb{E}_X(\pi)$.

The objective in Probable World Problem is for the robot to map enough of the environment to determine which of the m environments it is operating in. On the other hand, the objective of the Reactive Planning Problem is to reach the goal vertex or determine that the goal cannot be reached.

Proposition 1 *The Probable World Problem is NP-Hard, even when all observations have zero cost.*

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 $b \in B$ for subset E_h only if $Z(a, h, b) = 1$. Set *observation* model $\Theta_v = \{e \in \mathcal{E} | e \in I_v\}$. 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 hypothesis h. Given IPP (perfect sensing) is NP-Hard (Lim et al. 2015), PWP must be NP-Hard. □

Theorem 1 *The Reactive Planning Problem is NP-Hard, even when all observations have zero cost.*

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 3. 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 $(q_i, g) \cap E_x = \emptyset$. 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 $\alpha(mn + m^2)$ is an upper bound on an optimal policy which is a contradiction. This shows RPP solves PWP. Given Proposition 1, RPP is NP-Hard. □

Remark 3 (NP-Complete) In the decision version of RPP we are given a budget and asked to find a *complete* policy with expected cost less than or equal the budget. From Lemma 3, it is straightforward to see that the decision version is in NP, and thus is NP-Complete. Polychronopoulos & Tsitsiklis (1996) provides a similar result for R-SSPPR.

5 Policy Generation

The Reactive Planning Problem seeks information to reach the goal. In this section we begin by proposing an efficient heuristic for generating a complete policy. We then show how this heuristic provides a lower

bound on the cost of the optimal policy. Finally, we provide a exponential-time dynamic program for computing the optimal policy.

5.1 Mutual Information Policies

In this section we propose an efficient algorithm for computing a complete, but suboptimal policy. The idea behind the algorithm is to incrementally construct the policy tree. Each node of the tree corresponds to a robot state (Y, v). We then evaluate the set of *constructive observations* that are reachable from that state with respect to two metrics:

1. mutual information, which quantifies the reduction in uncertainty that the observation has on the robot's belief Y, and
2. cost-to-go, which quantifies the expected cost for the robot to travel to that observation and then to the goal.

The first metric is used to encourage exploration to gain more information, while the second encourages exploitation of existing information. This is then coupled with a pruning condition that quantifies when a *constructive observation* is no longer cost effective, independent of how much the uncertainty is reduced.

Given a state (Y, v) we begin by defining the set of all *reachable constructive observations*. We denote this set as R_v, which contains *observation*-vertex pairs (O, u), and is defined as

$$R_v = \{(O, u) \mid O \in \Theta_u \text{ is } \textit{constructive} \text{ and is } \textit{reachable} \text{ from } v\}.$$

Recall the definitions of *constructive* and *reachable observations* are given in Definitions 7 and 9, respectively.

Exploration: Consider the RPP. By Lemma 1, information can only be collected at the set of *reachable observations*. To select which *constructive observation* is beneficial, we maximize mutual information extended from (Charrow et al. 2014, Dames et al. 2016, Lim et al. 2015).

Let X_Y encode the probability distribution over environments given a set of *consistent environmental states* Y. Its probability mass function is given by

$$\mathbb{P}(X_Y = i) = \begin{cases} \frac{\mathbb{P}(X=i)}{\sum_{j \in Y} \mathbb{P}(X=j)} & \text{for } i \in Y \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

Mutual information is the difference between entropy of X_Y and conditional entropy of X_Y, given a reachable constructive observation $(O, u) \in R_v$. Formally,

$$MI(X_Y, (O, E_O)) = H(X_Y) - H(X_Y | (O, E_O)) \,. \tag{5}$$

The entropy of X_Y, $H(X_Y)$, does not depend on the *observation outcome* pair (O, E_O); therefore, this problem can be reduced to minimization of conditional entropy,

$$H(X_Y | (O, E_O)) = - \sum_{E \in \Gamma_{(Y,O)}} \mathbb{P}(E_O = E) \sum_{i \in Y} \mathbb{P}(X_Y = i | E_O = E) \log(\mathbb{P}(X_Y = i | E_O = E)) \,. \tag{6}$$

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,

$$c_G(v,g) \leq c_G(v,u) + \mu(O) + c_G(u,g) \quad \text{for each } (O,u) \in R_v, \tag{7}$$

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,

$$\mathbb{C}_Y(u,g) = \sum_{i \in Y} c_{G_i}(u,g)\mathbb{P}(X_Y = i)\,, \tag{8}$$

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. 7 is augmented to include the robot's current environmental understanding and the expected cost-to-go. Given a $(O,u) \in R_v$, the pruning inequality can be written as

$$c_{\overline{G}}(v,g) \leq c_{\overline{G}}(v,u) + \mu(O) + \mathbb{C}_Y(u,g). \tag{9}$$

If this inequality is satisfied, then it is less expensive for the robot to traverse straight to the goal than for it to make observation O at vertex u, and thus this information should not be collected.

Lemma 4 *Given a robot state r with constructive observations R_v, if all $(O,u) \in R_v$ satisfy Eq. 9, then 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. 9, 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 $(O,u) \in R_v$ satisfy Eq. 9, 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 5 *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. □

Combining Exploration and Exploitation: To combine information gain and motion to goal, we can use any function of the exploration metric $H(X_Y|(O,E_O))$ and the exploitation metric $c_{\overline{G}}(v,u)+\mu(O)+\mathbb{C}_Y(u,g)$.

Weighted conditional entropy (Suhov et al. 2015) is a well-studied method for combining entropy with a second metric, and within this method, when at state (Y,v), we select *observations* satisfying

$$O_{\min}=\underset{(O,u)\in R_v}{\operatorname{argmin}}\ (c_{\overline{G}}(v,u)+\mu(O)+\mathbb{C}_Y(u,g))H\left(X_Y|(O,E_O)\right)\ . \tag{10}$$

We also test a weighted sum of the terms in order to select *observations*,

$$O_{\min}=\underset{(O,u)\in R_v}{\operatorname{argmin}}\ (c_{\overline{G}}(v,u)+\mu(O)+\mathbb{C}_Y(u,g))+\rho H\left(X_Y|(O,E_O)\right)\ , \tag{11}$$

where ρ is tuned based on the importance of information. In Section 7, we show the strength of each case.

Detailed Algorithm: Algorithm 1 selects *observations* that satisfy either Eq. 10 or Eq. 11. It requires the function $Reachable(G,\mathcal{S},(Y,v))$ 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 *constructive observations* which render finite path cost from v, where D_v is an array recording the distance $d[u]$ to each vertex u.

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_{move}

1 Compute $c_{G_i}(v,g)$ 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 **if** $c_{G_i}(v,g)=\infty$ *for all* $i\in Y$ **then**
6 Mark π, at v for Y, no goal terminal state;
7 **else**
8 Compute $(R_v,D_v)=Reachable(G,\mathcal{S},(Y,v))$;
9 Remove elements of R_v that satisfy Eq. 9;
10 **if** $|R_v|=0$ **then**
11 Add *leg* from v to g, marked goal terminal state, to π;
12 **else**
13 Let $(O,u)\in R_v$ be the minimum of Eq. 10 (or Eq. 11);
14 Add *leg* from v to u and node (Y,O) to π;
15 Add (Y_E,u) to Q for each $E\in\Gamma_{(Y,O)}$;
16 Let $l_{\text{move}}=\mathbb{C}_{\mathbb{N}_m}(s,g)$;
17 Return π and l_{move};

Remark 4 (Runtime) The runtime is dominated by the m calls to Dijkstra's Algorithm, which gives complexity $\mathbb{O}(m(|\mathcal{V}|+|\mathcal{E}|)\log|\mathcal{V}|)$ (priority queue implemented as a binary heap).

The biased cost when selecting an *observation* in Line 13 and the pruning condition in Line 9 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 moves to the goal.

Remark 5 (Parallel Computation) Given Lemma 2, the policy π, returned by Algorithm 1, is independent of the order in which states are removed from Q in Line 4 (potential for parallel computation).

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 (Y, v) for $v \neq g$ and there exists $y \in Y$ such that $y \in Y_{\text{goal}}$ (false negative).

False positive: Algorithm 1 must have directed the robot to travel over an obstructed edge because $Y \subseteq Y_{\text{no goal}}$. Since each *realization* G_i for $i \in Y$ does not have a path to goal, the cost $c_{G_i}(v, g)$ will be infinite for all $i \in Y$. Line 5 is satisfied and Line 6 sets this state to terminal no goal.

False negative: Algorithm 1 would not be able to find a path in each *realization* G_i for $i \in Y$, but since the *environmental state* y is still possible, $c_{G_y}(v, g)$ will have finite cost. This will not satisfy Line 5, and thus, this state cannot be marked no goal terminal state by Line 6. □

Remark 6 (Online Policy Generation) Algorithm 1 can be used in an online manner as follows. For a state (Y, v), find the *reachable observations* and select the minimizer of Eq. 10 or Eq. 11, namely $(O, u) \in R_v$ with $O \in \Theta_u$. Execute shortest path in $\overline{G}$ from v to u and take O. Update (Y, v) given $(O, O \cap E_x)$ and repeat.

5.2 Lower Bound on Observation Costs

The lower bound l_{move} does not account for any *observation* costs. We formulate a lower bound on the expected *observation* cost such that its sum with l_{move} remains a lower bound on an optimal policy. To accomplish this, a new *observation* is created for a state (Y, v) from the union of all *constructive observations* in the reachable set R_v, namely $\theta = \cup_{(O,u)\in R_v} O$. This *observation* is available at v for the cost $\mu(\theta) = \min_{(O,u)\in R_v} \mu(O)$. We make this new *observation* if

$$c_{\overline{G}|Y}(v, g) \geq \sum_{E \in \Gamma_{(Y,\theta)}} \mathbb{P}(X_Y \in Y_E) c_{\overline{G}|Y_E}(v, g) + \mu(\theta)\,, \tag{12}$$

where subscript $\overline{G}|Y$ indicates costs are found for $\overline{G}$ defined by Y.

Algorithm 2: RPP Observation Lower Bound

Data: Graph G, edge subsets $\mathcal{S}$, vertices s & g, states $\mathbb{N}_m$, probabilities p
Result: Lower bound on expected observation cost l_{obs}

```
1  Compute c_{G_i}(v, g) for all v ∈ V and i ∈ N_m;
2  Add N_m to Q and set l_obs = 0;
3  while Q not empty do
4      Remove Y from Q;
5      if c_{G_i}(s, g) ≠ ∞ for any i ∈ Y then
6          Compute R_s = Reachable(G, S, (Y, s));
7          if Eq. 12 then
8              l_obs = l_obs + P(X ∈ Y) μ(θ);
9              Add Y_E to Q for each E ∈ Γ_{(Y,θ)};
10 Return l_obs;
```

Lemma 6 *Algorithm 2 finds a lower bound on the expected observation cost.*

Proof First, we must show Algorithm 2 will not make more than the minimum number of *observations* required. If Line 5 is not met then all $i \in Y$ have no path to goal; thus, the robot may terminate because $Y \subseteq Y_{\text{no goal}}$. Consider Eq. 12. The left-hand-side is the lowest cost path from s to g without any more *observations*. The right-hand-side is the expected cost from s to g, given θ's *outcome*, for the cost $\mu(\theta)$. When this inequality holds, the added freedom to select a path with the *outcome's* information saves more than it costs to make the *observation*. Otherwise, the *observation* may not improve the known path for future states; thus, the robot will not take the *observation*.

If an *observation* is required, the optimal *observation* is in R_v by Lemma 1. This information is collected for the minimum cost of any *observation* in R_v in Line 8. This lower bounds the expected cost of the optimal *observation*. Therefore, no more than the minimum number of *observations* are made, each with minimum cost, showing that l_{obs} lower bounds the expected *observation* cost. □

5.3 A Dynamic Program for the Optimal Policy

The research by Polychronopoulos & Tsitsiklis (1996) presents an exponential-time dynamic program to solve the R-SSPPR problem. Their dynamic program runs in $\mathbb{O}(2^m(m|\mathcal{V}| + |\mathcal{V}|^2))$, which from a practical standpoint, makes problems with a large number of possible realizations m intractable. Building on their formulation, we can create a dynamic programming solution with the same runtime to the Reactive Planning Problem as follows. We define the subproblem $V(Y, v)$ to be the optimal expected cost to go from v in belief Y to the goal g. Any belief Y such that $Y \subseteq Y_{\text{no goal}}$, has an optimal cost to go of $V(Y, v) = 0$ for all $v \in \mathcal{V}$. Following Polychronopoulos & Tsitsiklis (1996), the dynamic programming recursion is

$$V(Y, v) = \min_{(O,u)\in R_v} \left[c_{\overline{G}}(v, u) + \mu(O) + \mathbb{E}_{X_{Y_E}}[V(Y_E, u)]\right], \tag{13}$$

where Y_E is the *consistent environmental state* after *observation* O gives the *outcome* $E \in \Gamma_{(Y,O)}$. The recursion looks over all *reachable constructive observations* and determines the cost to obtain that *observation*, followed by the optimal expected cost-to-go to the goal, versus the cost to go directly to the goal from the current state (Y, v). Note that in the worst case, to solve the subproblem $V(Y, v)$ we may require the solution to the subproblems $V(Y_E, u)$ for all $Y_E \subset Y$ and $u \in V$, which grows exponentially with m.

To implement this, we use memoization and propagate the solution forward from initial state $(\mathbb{N}_m, s)$. This complements the fact that the next *observation* must be in the *reachable* set and often limits the number of states examined. However, note that the number of subproblems $V(Y, v)$ can in general be exponential in the number of realizations m, since $Y \subseteq \mathbb{N}_m$. Our solution uses recursion on each *reachable observation* to identify the best *observation* to make, and we compare this to Algorithm 1's simplified selection method in the simulations.

6 Extension to Faulty Sensors

In some applications, the robot sensor may erroneously miss obstacles (false negatives) or detect obstacles that are not actually present (false positives). The reactive planning problem can be extended to certain faulty sensor models. We begin by discussing the difficulties of a general faulty sensor model, and then discuss a special case that can be handled directly.

In a general model, we have a probability $p_O \in [0, 1]$ for each *observation* $O \in \cup_{v\in\mathcal{V}}\Theta_v$, where the robot receives the correct *outcome* $O \cap E_x$ with probability p_O and an incorrect subset of O with probability $(1 - p_O)$, which can consist of both false negatives and false positives. In this case, an *observation-outcome*

pair (O, E) cannot eliminate any environmental state unless it makes an *observation* O for which $p_O = 1$. Instead of updating Y, *observation-outcome* pairs update the distribution of X_Y. A natural way to address this is to allow the robot selection of action (e, move) at v even when $\mathbb{P}(e|Y) < 1$. The robot then either successfully traverses the edge, or detects an obstruction using a proximity sensor, or by physical contact, returning to v and incurring cost $c(e)$. This model now bears close resemblance to the form of POMDP presented by Papadimitriou & Tsitsiklis (1987), and we conjecture that as with POMDPs, this general extension is PSPACE-hard. This however, would imply that the optimal policy structure no longer takes the compact form discussed in Lemma 3. Thus, we limit our attention to the following class of faulty sensors for which our policy structure still applies.

Structured Sensor Failures: To motivate the idea of structured failures, consider deploying a robot in a building that may contain obstacles with reflective surfaces. Certain sensors have difficulty detecting these surfaces and thus may return incorrect *outcomes*. We term this as a failure type and quantify it with a faulty edge subset $E \in \mathcal{F}$ where $\mathcal{F} = \{E_{m+1}, E_{m+2}, \ldots, E_{m+f}\}$ is the set of $f \in \mathbb{Z}^+$ possible failure types. The failure experienced by the robot is encoded as a random variable W taking values in $\{0, 1, \ldots, f\}$. The outcome $W = 0$ corresponds to no sensor failure (i.e., each *outcome* is $O \cap E_x$), and occurs with probability $p_{\text{nf}} \in [0, 1]$. The failure type $w \in \mathbb{N}_f$, corresponding to edge subset E_{m+w} occurs with probability $\mathbb{P}(W = w)$, where $\sum_{w=1}^{f} \mathbb{P}(W = w) = 1 - p_{\text{nf}}$. The robot knows the pmf of W, but not its random draw w. If the sensor experiences failure type $w \in \mathbb{N}_f$, then the *outcome* of each *observation* O will be $O \cap E_{m+w}$, which may not agree with $O \cap E_x$. Thus, this model captures correlated failures: if failure type w occurs when the robot is operating in *realization* x, then all edges in $E_{m+w} \setminus E_x$ will give false negative, and all edges in $E_x \setminus E_{m+w}$ will give false positives.

The robot state is expanded to capture both the possible *realizations* of G_x, namely $Y \subseteq \mathbb{N}_m$, and the indices of edge subsets *consistent* with the *observation-outcome* pairs, namely $Z \subseteq \mathbb{N}_{m+f}$. Formally, the robot state is $(2^{\mathbb{N}_m}, 2^{\mathbb{N}_{m+f}}, \mathcal{V})$. The set Y records all environmental *realizations* that could be *consistent* with the *observations* (given that they may be faulty). The set Z records the possible edge subsets in $(E_1, \ldots, E_{m+f})$ that are *consistent* with the measurements. Under this new sensor model, we provide the actions that update the robot's environmental information.

Definition 10 (Constructive Observation) Given state (Y, Z, v) with $O \in \Theta_v$, $(O, \text{observe})$ is constructive if there exist $i, j \in Z$ such that $O \cap E_i \neq O \cap E_j$.

Definition 11 (Reactive Move) Given state (Y, Z, v) with $e \in I_v$, (e, move) is reactive if there exist $i, j \in Y$ such that $e \in E_i$ and $e \notin E_j$. The robot remains at v if $e \notin E_x$ and incurs $c(e)$.

Given Z, a *constructive observation* O has a set, $\Gamma_{(Z,O)} \equiv \{E \subseteq \mathcal{E} | E = O \cap E_i, i \in Z\}$, of possible *outcomes* where *outcome* $E \in \Gamma_{(Z,O)}$ updates Z to $Z_E = \{i \in Z | E = O \cap E_i\}$. A *reactive move*, (e, move), partitions Y and Z into edge subsets that contain e and those that do not. If the robot state (Y, Z, v) satisfies $Z \subseteq \mathbb{N}_m$ (i.e., the robot knows no fault types have been encountered), the problem returns to the RPP under perfect sensing with start state (Z, v). Note the robot may reach a terminal state without satisfying this special case.

The tree policy can be extended by changing the nodes to (Y, Z, a) where action a is a *constructive observation* or *reactive move*. To calculate its expected cost, we extract each root to leaf *observed path*. Given *realization* $x \in \mathbb{N}_m$, there are $f + 1$ *observed paths* corresponding to sensing $x \in \mathbb{N}_m$ (*observed paths* $A_{x|x}$) and sensing $w \in \mathbb{N}_f$ (*observed paths* $A_{m+w|x}$). The expected cost of policy π is,

$$\mathbb{E}(\pi) = \sum_{x \in \mathbb{N}_m} \left(p_{\text{nf}} \mathbb{P}(X = x) \text{cost}(A_{x|x}) + \sum_{w \in \mathbb{N}_f} \mathbb{P}(W = w) \text{cost}(A_{m+w|x}) \right).$$

Lemma 7 *A complete policy tree has at most $2m + f - 2$ nodes where m is the number of realizations and f is the number of faulty edge subsets.*

Proof The largest policy tree requires the robot to learn if it is getting incorrect *outcomes* and then find the true *realization* x via *reactive moves*. Every *constructive observation* partitions Z and it requires at most $m + f - 1$ *constructive observations* to identify z or x. If it identifies z, only *reactive moves* can partition Y and it takes $m - 1$ *reactive moves* to learn every possible x. This can be encoded into $2m + f - 2$ nodes. □

Algorithm 1 can be altered to handle this faulty sensor model as follows. The queue now holds the new state, and Y is still used to calculate $\overline{G}$. The function $Reachable(G, \mathcal{S} \cup \mathcal{F}, (Y, Z, v))$ must also find the *reactive moves* that can be reached with finite cost. Finally, Eq. 6 must be altered to capture the conditional entropy of the new state given a *reactive move* or a *constructive observation*.

7 Simulation Results

In this section we provide simulation results on a large scale practical example and on randomly generated environments. We compare against online algorithms and the optimal solution presented in Section 5.3. Tests were run on a single Intel Core i7-6700 at 3.4GHz.

7.1 Flexible Factory

Flexible factories often spend considerable downtime between contracts due to changes in infrastructure and machinery. Consider Fig. 5 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 34561 edge subsets each with a corresponding probability and 48 vertices. We compute policies from S to A, from S to B, from A to S, and from B to S. The robot is faster when not loaded, so the movement costs of A to S and B to S are decreased by a factor of 2.

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

Due to the size of the environment, we will allow replanning 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 replan. This is completed

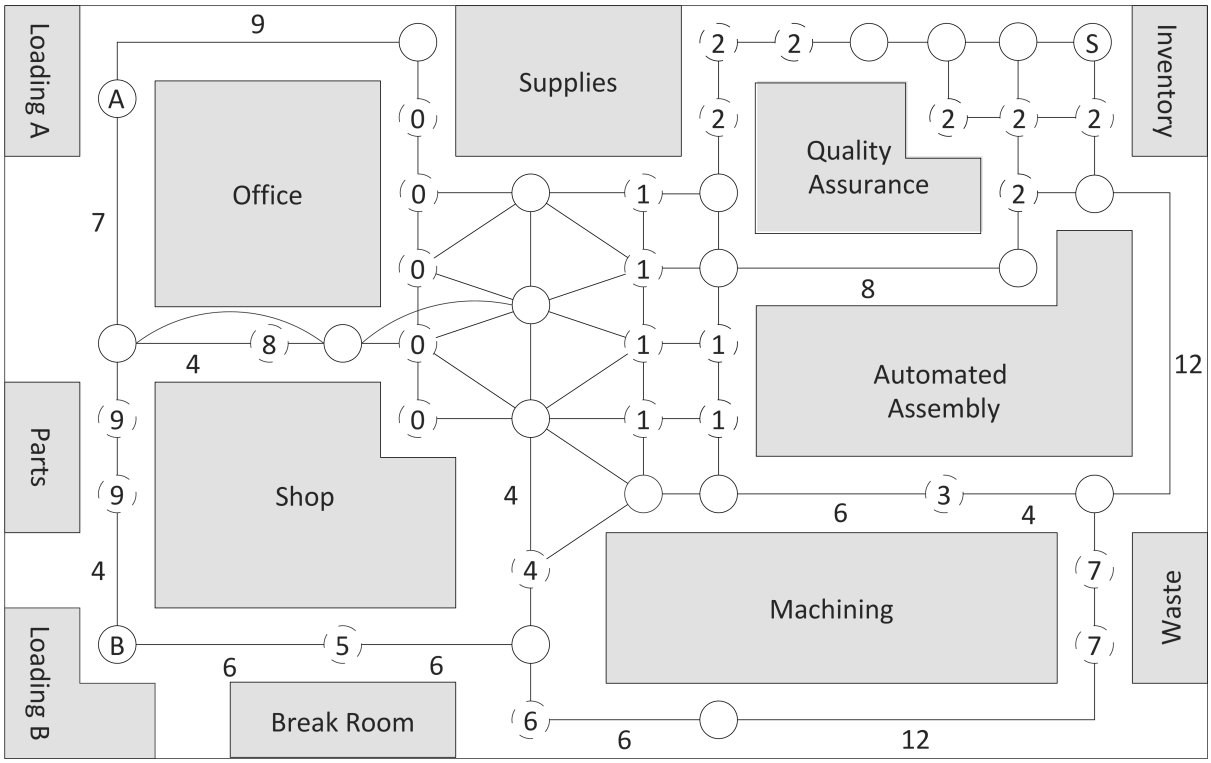

Fig. 5 Flexible Factory model: Dashed vertices may be obstructed. Edges with cost of 2 or 3 are not labelled for simplicity. The curved edges cost 2 more for waiting.

for every realization $x \in \mathbb{N}_m$. 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. We also compare against the feedback heuristic from Polychronopoulos & Tsitsiklis (1996). In contrast to the two prior algorithms, if any new information is collected the robot replans with this new information. Given the difference between the R-SSPPR and the RPP, the optimal expected cost path to goal may not be well-defined. To remedy this, we let the robot find the shortest path in $\underline{G}$ and make the lowest cost *constructive observations* required to use this path. If an *observation* is made, the robot replans with the acquired information applied to $\underline{G}$.

Table 2 Flexible Factory simulation results. The column obs. gives the number of *observations* in the policy. The Eq. 11 column shows expected cost for optimized ρ.

	Algorithm 1:		Eq. 10	Eq. 11	A*	Max $\mathbb{P}_s$	Feedback
Task	obs.	$l_{\text{move}} + l_{\text{obs}}$	$\mathbb{E}_X(\pi)$	$\mathbb{E}_X(\pi)$	$\mathbb{E}_X$	$\mathbb{E}_X$	$\mathbb{E}_X$
S→A	114	40.0	**42.5**	42.6	47.7	85.3	47.3
S→B	6869	42.3	**50.0**	50.0	50.1	61.7	53.7
A→S	84	20.6	**23.3**	23.3	30.2	43.6	29.9
B→S	175	22.0	25.7	**25.4**	28.3	33.9	27.6

The proposed algorithm provides lower expected cost than the three online solutions. In addition, for each task the expected cost of Algorithm 1 is within 20% of the lower bound ($l_{\text{move}} + l_{\text{obs}}$) on the optimal expected cost. In general the feedback heuristic should always perform at least as well as A*. Notice this is not the case for task S→B, but this occurs due to the different tie-breaking methods for shortest path in each solution. When comparing Eq. 10 and Eq. 11, the solution for weighted conditional entropy closely compares to the additive method for optimized ρ and is computed only once verses multiple runs for different values of ρ. We contribute this to the correlated environment structure, and in the next section we show, when the environments are randomized, this is not always the case.

7.2 Performance versus Optimal

In order to test the performance of the lower bound as well as the proposed algorithm, we compare against the optimal solver discussed in Section 5.3 as well as the feedback heuristic discussed in Section 7.1. The test environments are cell worlds where edges connect left, right, up and down cells. Traversal and *observation* costs are set uniformly at random between $[5, 6]$ and $[1, 2]$ respectively. The edge subsets are generated by adding obstacles randomly. To do this, we incrementally relax the addition of obstacles into a *realization* until a path to goal exists and vice versa for no path to goal. Finally, X's pmf is formed iteratively by selecting a realization uniformly at random from the remaining unselected *realizations* and assigning it one fourth the remaining probability (the final realization takes the mass left). Due to the exponential runtime of the optimal solver, we test on grids of fewer than 40 vertices and up to 100 possible *realizations*. To demonstrate the time scaling, we used our optimal solver for three 6x6 grids of 50, 65 and 75 *realizations* each with a single edge *observation* model. The computation time for 50 and 65 *realizations* were 48 and 54 seconds respectively, but the solver could not find a policy within 20 minutes for the 75 *realizations*.

The results in Fig. 6 highlight a shortcoming in the lower bound, $l_{\text{move}} + l_{\text{obs}}$. The lower bound does not perform well when the number of *realizations* that do not have a path to goal is large relative to the total number of *realizations*. This is expected because the lower bound l_{move} assumes the robot does not move when a realization does not contain a path to the goal. In practice, the robot must traverse and observe to identify if it can make its way to the goal.

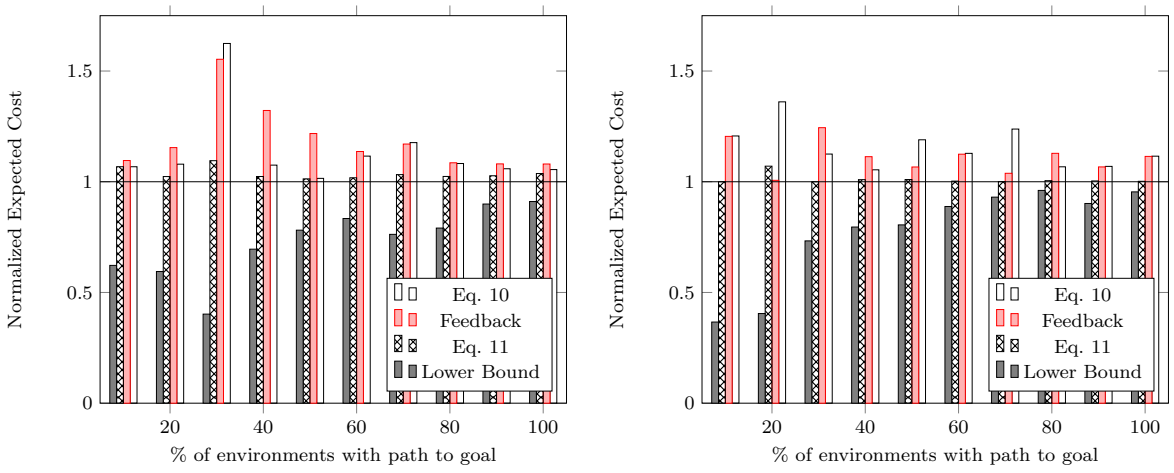

Fig. 6 Left: G is a 5x5 grid with single edge observations and 40 realizations. Right: G is a 6x6 grid with single hop observations and 100 realizations.

Fig. 6 also displays a strength in Eq. 11 over Eq. 10 and the feedback heuristic. First, Eq. 11 with an optimized ρ is always within 10% of the optimal while both Eq. 10 and the feedback heuristic are within 63% of the optimal. These environments are randomly generated and thus its structure does not foster a strong relationship between obstacles. When this is the case, the weighted additive method of Eq. 11 allows the robot to put less importance on information gathering, $H(X_Y, (O, E_O))$, and more importance on the cost of traversal, $(c_{\overline{G}}(v, u) + \mu(O) + \mathbb{C}_Y(u, g))$.

We also test the runtime of Algorithm 1 by fixing the number of vertices in the grid while varying the number of *realizations* and fixing the number of *realizations* while varying the number of vertices. Ten percent of the *realizations* have no path to goal. The timing results in Fig. 7 are generated based on a C-programming implementation. For low number of *realizations*, a single *observation* greatly impacts where the robot can traverse; in contrast, as more *realizations* are introduced, the robot requires more *observations* to traverse similar area (more while loop iterations) as seen in the left plot. The right plot shows computation time dominated by Dijkstra's algorithm as expected given the number of vertices are growing while the number of *realizations* remain the same.

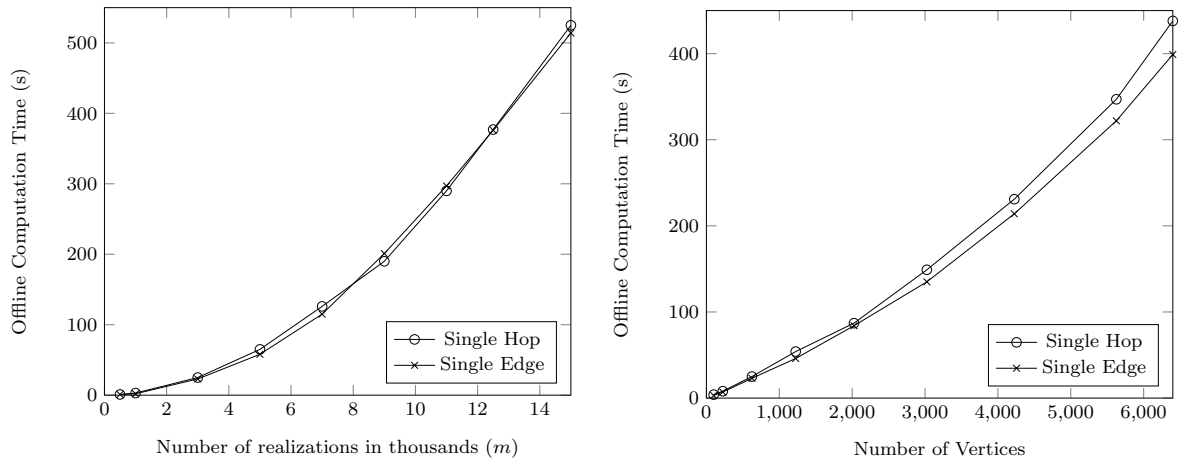

Fig. 7 Left: G has 625 vertices and 2600 edges. Right: G's vertices and edges are varied with 3000 realizations.

7.3 Performance with Inaccurate Prior Data

Often prior data fails in that it may not accurately represent the physical environment the robot functions in (i.e., the realized edge subset is not an element of $\mathcal{S}$). We test Algorithm 1's policy by randomly selecting k edges of the realized environment and complementing their obstruction (i.e., adding an obstacle if one does not exist and vice versa). The robot follows the policy until it reaches a terminal state or finds an inconsistency, namely state $(\emptyset, v)$. If $Y = \emptyset$, the robot switches to the A* algorithm presented in Section 7.1 where all uncertain edges are assumed to exist but still must be sensed. The A* algorithm always uses a single hop (i.e., all neighbors) *observation* model.

The environment is a 12x12 grid with 400 *realizations* where 10% have no path to goal. For a fixed percentage of inaccurate edges, we perform 20 independent tests using both single edge and single hop *observation* models. Prior tests show Eq. 10 strongly values information; thus, we test Algorithm 1, using Eq. 10, to show how corrupted information can adversely affect the expected cost.

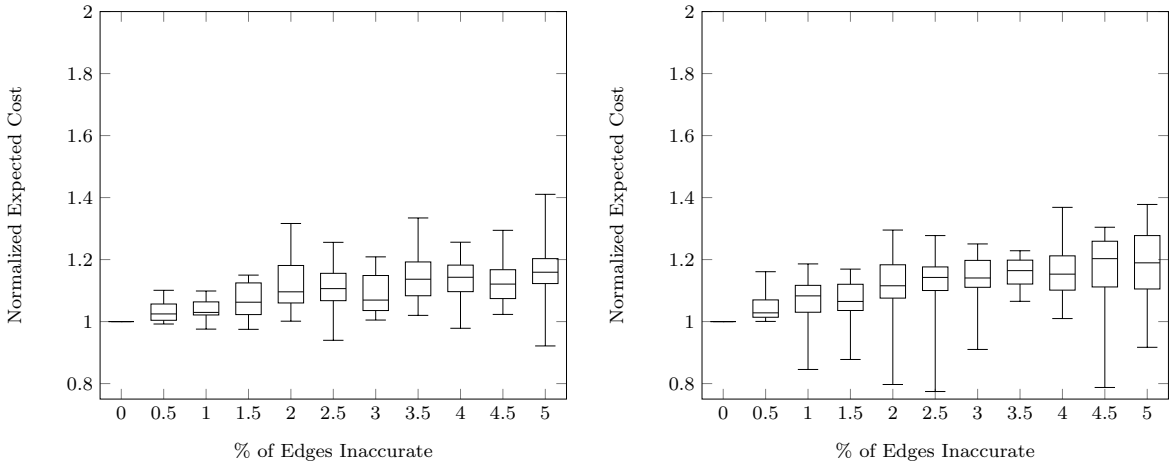

Fig. 8 Single edge observation model (left) and single hop observation model (right). Each realization has 20 independent error tests. We normalize the expected cost by Algorithm 1's policy found assuming no errors.

The results in Fig. 8 show the single hop *observation* model has higher expected cost than the single edge *observation* model. This occurs because the single hop model is more likely to identify errors in the environment given it often collects more information; thus as error grows, this model abandons its prior knowledge sooner causing a tendency to increase the expected cost (seen in the right plot). Also, the single edge *observation* model only senses edges that may be blocked so an error will only be encountered on a *leg*. The robot belief may not satisfy $x \in Y$, yet it often continues travelling toward the goal before abandoning the prior knowledge.

Notice values less than one in Fig. 8 correspond to particular errors in the environment. Edges may become unblocked causing a lower cost to goal. The robot may become stuck before reaching a terminal state as the strongly connect component containing s may be broken (see Fig. 3 for an example). Also, a terminal no goal state may be reached when originally there was a path to goal due to added obstructions.

8 Conclusion

A reactionary *complete* policy based on environmental *observations* was presented that directs a robot to complete a task or to identify it is impossible complete. A lower bound on the expected cost of the optimal policy was provided and experimentally shown to perform well when the number of *realizations* with no path to goal is small relative to the total number *realizations*. Multiple *observation* models were presented as well as an extension to robots with faulty sensors.

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 goal locations. Given Lemma 2, we look to formulate an extension of Algorithm 1 to parallel computation in order to provide increased runtime savings.

References

Andreatta, G. & Romeo, L. (1988), 'Stochastic shortest paths with recourse', *Networks* **18**(3), 193–204.

Bai, H., Hsu, D. & Lee, W. S. (2014), 'Integrated perception and planning in the continuous space: A POMDP approach', *The International Journal of Robotics Research* **33**(9), 1288–1302.

Best, G., Faigl, J. & Fitch, R. (2016), Multi-robot path planning for budgeted active perception with self-organising maps, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3164–3171.

Bhattacharya, S., Ghrist, R. & Kumar, V. (2015), 'Persistent homology for path planning in uncertain environments', *IEEE Transactions on Robotics* **31**(3), 578–590.

Binney, J. & Sukhatme, G. S. (2012), Branch and bound for informative path planning., *in* 'IEEE International Conference on Robotics and Automation', pp. 2147–2154.

Bnaya, Z., Felner, A. & Shimony, S. E. (2009), Canadian traveler problem with remote sensing, *in* 'International Joint Conference on Artificial Intelligence', pp. 437–442.

Chakaravarthy, V. T., Pandit, V., Roy, S., Awasthi, P. & Mohania, M. (2007), Decision trees for entity identification: approximation algorithms and hardness results, *in* 'Proceedings ACM Symp. on Principles of Database Systems', ACM, pp. 53–62.

Charrow, B., Kumar, V. & Michael, N. (2014), 'Approximate representations for multi-robot control policies that maximize mutual information', *Autonomous Robots* **37**(4), 383–400.

Chen, M., Frazzoli, E., Hsu, D. & Lee, W. S. (2016), POMDP-lite for robust robot planning under uncertainty, *in* 'IEEE International Conference on Robotics and Automation', pp. 5427–5433.

Dames, P. M., Schwager, M., Rus, D. & Kumar, V. (2016), 'Active magnetic anomaly detection using multiple micro aerial vehicles', *IEEE Robotics and Automation Letters* **1**(1), 153–160.

Dames, P., Schwager, M., Kumar, V. & Rus, D. (2012), A decentralized control policy for adaptive information gathering in hazardous environments, *in* 'IEEE Conference on Decision and Control', pp. 2807–2813.

Hollinger, G. A., Englot, B., Hover, F. S., Mitra, U. & Sukhatme, G. S. (2012), 'Active planning for underwater inspection and the benefit of adaptivity', *The International Journal of Robotics Research* pp. 3–18.

Hollinger, G. A. & Sukhatme, G. S. (2014), 'Sampling-based robotic information gathering algorithms', *The International Journal of Robotics Research* **33**(9), 1271–1287.

Issac, P. & Campbell, A. M. (2015), 'Shortest path problem with arc failure scenarios', *EURO Journal on Transportation and Logistics* pp. 1–25.

Javdani, S., Chen, Y., Karbasi, A., Krause, A., Bagnell, D. & Srinivasa, S. S. (2014), Near optimal bayesian active learning for decision making., *in* 'International Conference on Artificial Intelligence and Statistics', pp. 430–438.

Julian, B. J., Karaman, S. & Rus, D. (2014), 'On mutual information-based control of range sensing robots for mapping applications', *The International Journal of Robotics Research* **33**(10), 1375–1392.

Kaelbling, L. P. & Lozano-Pérez, T. (2013), 'Integrated task and motion planning in belief space', *The International Journal of Robotics Research* **32**(9-10), 1194–1227.

Koenig, S. & Likhachev, M. (2005), 'Fast replanning for navigation in unknown terrain', *IEEE Transactions on Robotics* **21**(3), 354–363.

Koenig, S., Likhachev, M. & Furcy, D. (2004), 'Lifelong planning A*', *Artificial Intelligence* **155**(1-2), 93–146.

LaValle, S. M. (2006), *Planning algorithms*, Cambridge university press.

Lim, Z. W., Hsu, D. & Lee, W. S. (2015), 'Adaptive informative path planning in metric spaces', *The International Journal of Robotics Research* **35**(5), 585–598.

MacDonald, R. A. & Smith, S. L. (2016), Reactive motion planning in uncertain environments via mutual information policies, *in* 'Workshop on the Algorithmic Foundations of Robotics'.

Papadimitriou, C. H. & Tsitsiklis, J. N. (1987), 'The complexity of markov decision processes', *Mathematics of operations research* **12**(3), 441–450.

Papadimitriou, C. H. & Yannakakis, M. (1991), 'Shortest paths without a map', *Theoretical Computer Science* **84**(1), 127–150.

Polychronopoulos, G. H. & Tsitsiklis, J. N. (1996), 'Stochastic shortest path problems with recourse', *Networks* **27**(2), 133–143.

Suhov, Y., Stuhl, I., Sekeh, S. Y. & Kelbert, M. (2015), 'Basic inequalities for weighted entropies', *Aequationes mathematicae* pp. 1–32.

Van Den Berg, J., Patil, S. & Alterovitz, R. (2012), 'Motion planning under uncertainty using iterative local optimization in belief space', *The International Journal of Robotics Research* **31**(11), 1263–1278.

Wang, R., Veloso, M. & Seshan, S. (2016), Active sensing data collection with autonomous mobile robots, *in* 'IEEE International Conference on Robotics and Automation', pp. 2583–2588.

Yu, J., Schwager, M. & Rus, D. (2014), Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks, *in* 'IEEE International Conference on Intelligent Robots and Systems', pp. 342–349.