Adaptive Informative Path Planning in Metric Spaces

Zhan Wei Lim David Hsu Wee Sun Lee

Department of Computer Science
National University of Singapore
Singapore, 117417, Singapore

Abstract

In contrast to classic geometric motion planning, *informative path planning* (IPP) seeks a path for a robot to sense the world and gain information. In *adaptive* IPP, the robot chooses the next sensing location conditioned on all information acquired so far, and the robot's goal is to minimize the travel cost required for identifying a true hypothesis. Adaptive IPP is NP-hard, because the robot must trade off information gain and travel cost optimally. This paper presents *Recursive Adaptive Identification* (RAId), a new polynomial-time approximation algorithm for adaptive IPP. We prove a polylogarithmic approximation bound when the robot travels in a metric space. Furthermore, our experiments suggest that RAId is practical and provides good approximate solutions for two distinct robot planning tasks. Although RAId is designed primarily for noiseless observations, a simple extension allows it to handle some tasks with noisy observations.

1 Introduction

Path planning typically seeks a collision-free path for a robot to reach a physical location. In contrast, *informative path planning* (IPP) seeks a path for the robot to sense the world and gain *information*:

- An unmanned aerial vehicle (UAV) searches a disaster region to pinpoint the location of survivors.
- A mobile manipulator moves around and senses an object with laser range finders (Platt Jr et al., 2011) or tactile sensors (Javdani et al., 2013) in order to estimate the object pose for grasping.
- An autonomous underwater vehicle inspects the submerged part of a ship hull (Hollinger et al., 2013).

In all these tasks, the robot has a set of hypotheses on the underlying state of the world—the location of survivors, the pose of an object, etc.—and must move to different locations in order to sense and eventually identify the true hypothesis. Each sensing operation provides new information, which enables the robot to act more effectively in the future. To acquire this information, the robot, however, must move around and incur movement cost, in addition to sensing cost. A key issue in designing efficient IPP algorithms is the *trade-off* between information gain and robot movement cost. This paper presents a new algorithm, *recursive adaptive identification* (RAId), which computes a near-optimal path for the robot to identify the true hypothesis.

IPP contains, as a special case, the well-studied optimal decision tree (ODT) problem (Chakaravarthy et al., 2007), in which we want to build

a decision tree that minimizes the expected number of tests required to identify a hypothesis. ODT is basically IPP with a single location containing all sensing operations. Thus the costs of all sensing operations are the same. Unfortunately, ODT, even with noiseless sensing, is not only NP-hard, but also NP-hard to approximate within a factor of $\Omega(\log n)$, where n is the total number of hypotheses (Chakaravarthy et al., 2007).

There are two general classes of algorithms for IPP, nonadaptive and adaptive. In *nonadaptive* planning, we compute a sequence of sensing operations in advance. A robot executes these operation in order, regardless of the outcomes of earlier operations. In *adaptive* planning, we choose, in each step, new sensing operations conditioned on the outcomes of earlier sensing operations. Consider the simple problem of searching for an element in an sorted array of n numbers. Linear search is nonadaptive. It compares the search key with each element of the array in order. The outcome of a comparison does not affect the next element chosen. In contrast, binary search is adaptive. Each comparison splits the array into two halves, and the outcome of the comparison determines which half is processed further. While linear search requires $\mathcal{O}(n)$ comparisons, binary search requires only $\mathcal{O}(\log n)$ comparisons. Clearly adaptive planning is more powerful in general.

RAId performs adaptive planning, just as binary search. Each recursive step of binary search chooses a single most discriminating comparison that prunes half of all hypotheses. RAId shares this basic idea, but is more complex. There are two difficulties. In binary search, each comparison has the same cost. In IPP, the costs of traveling to different sensing locations vary, and we must address the key trade-off between information gain and movement cost. Further, we cannot choose sensing locations independently one at a time, because different locations provide different sensing information and moving to a location affects future choices. The main idea of RAId is to construct a near-optimal adaptive plan in each recursive step by solving a group Steiner problem (Calinescu and Zelikovsky, 2005). Under the plan, the robot traverses a subset of sensing locations and terminates the traversal when it encounters an "informative" observation, which guarantees to eliminate a significant fraction of existing hypotheses.

In the following, Section 2 provides the background of this work. Section 3 defines informative path planning and presents RAId. Section 4 analyzes the performance of the algorithm. Section 5 presents experimental comparisons of RAId with several alternative algorithms. Although our algorithm is designed primarily for noiseless observations, Section 6 presents an extension of RAId to handle some tasks with noisy observations. Finally, Section 7 discusses limitations of this work and directions for future research.

2 Background

2.1 Related Work

IPP is important to robotics and various related fields. The importance and the difficulty of computing optimal solutions for IPP have attracted significant interest in recent years. One idea is to choose a set of "informative" sensing locations and then construct a minimum-cost tour to traverse them (Hollinger et al., 2013). The heuristic algorithm may work in practice, but it does not provide any theoretical performance guarantee. Another idea is to search for a plan over a finite horizon (Hollinger et al., 2011). The guarantee, if any, is limited by the search horizon. Finite-horizon search search can also be combined with sampling-based motion planning to achieve asymptotic optimality (Hollinger and Sukhatme, 2013). The NAIVE algorithm replans in each step, using a nonadaptive IPP algorithm, in order to achieve adaptivity (Singh et al., 2009b). It guarantees near-optimal performance when the *adaptivity gap* is small, in other words, when adaptive planning does not have significant advantage over nonadaptive planning. Unfortunately the adaptive gap can be exponentially large even for very simple problems (Hollinger et al., 2013). This is unsurpris-

ing in light of the well-known benefit of acting adaptively (Dean et al., 2004; Golovin and Krause, 2011). Furthermore, to achieve nontrivial performance bound, NAIVE requires explicit construction of a submodular function with the *locality* property (Singh et al., 2009b). This is not always easy or possible. One strength of NAIVE is its ability to handle noisy observations. Our current work makes the assumption of noiseless observations, though we are extending the algorithm to handle noisy observations (Section 6).

IPP is closely related to the adaptive traveling salesman (ATSP) problem (Gupta et al., 2010). In contrast to the standard TSP, the traveling salesman here services only a subset of locations with *requests*, but does not know this subset initially. When the salesman arrives at a location, he finds out whether there is a request there. The goal is to find an adaptive strategy for the salesman to service all requests and minimize the expected cost of traveling. IPP contains ATSP as a special case. Each hypothesis represents a subset of locations with requests. Each "sensing" operation is binary and answers the query whether the current location has a service request or not. RAId has its root in the isolation algorithm for ATSP (Gupta et al., 2010). To provide the theoretical performance bound, the isolation algorithm uses linear programming in the inner loop to solve the group Steiner problem (see Section 2.2). This is impractical. RAId solves the more general IPP problem, which allows arbitrary hypothesis space and removes the restriction of binary sensing. To solve the group Steiner problem, it uses a combinatorial approximation algorithm (Calinescu and Zelikovsky, 2005) that is far more effective in practice.

Our IPP algorithm contains three main ingredients: information gathering, robot movement cost, and adaptivity. It touches on several important research topics, which contain one or two, but not all three ingredients. If we focus on information gathering only and ignore location-dependent robot movement cost, IPP becomes sensor placement, view planning, or ODT, which admits efficient solutions through, e.g., submodular optimization, in both non-adaptive (Krause and Guestrin, 2009) and adaptive settings (Golovin and Krause, 2011; Javdani et al., 2013, 2014). Our work does not rely on adaptive submodularity in either the algorithm or the proofs. If we account for movement cost, there are several nonadaptive algorithms with performance guarantee, e.g., (Hollinger et al., 2009; Singh et al., 2009a).

Although active localization (Fox et al., 1998) and simultaneous localization and mapping (SLAM) (Feder et al., 1999) bear some similarity to IPP, they are in fact different, because IPP assumes that the robot location is fully observable. Reducing active localization or SLAM to IPP incurs significant representational and computational cost.

IPP, as well as other information-gathering tasks mentioned above, can all be modeled as partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998), which provide a general framework for planning under uncertainty. However, solving large-scale POMDP models near-optimally remains a challenge, despite the dramatic progress in recent years (Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008). The underlying structure of IPP allows simpler and more efficient solutions.

2.2 Preliminaries on Group Steiner Trees

RAId makes critical use of the seemingly unrelated *group Steiner* problem to trade off information gain and robot movement cost. A group Steiner problem is defined by two elements. One is an edge-weighted graph $G = (V, E, W_E)$, where V is the set of vertices, E is the set of edges, and W_E contains the edge weights. The other is a collection of *groups* $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$ with corresponding group-weights $W_{\mathcal{V}} = \{\nu_1, \nu_2, \ldots, \nu_m\}$. Each group V_i contains a subset of vertices in V. A subgraph of G *covers* a group $V_i \subseteq V$ if the subgraph contains at least one vertex in V_i. In the standard group Steiner problem, the goal is to find a *minimum-edge-weight tree* that covers a subcollection of groups with total group-weight at least ν, for some given constant ν.

The group Steiner algorithm used in RAId constructs a tree T in a greedy manner (Calinescu and Zelikovsky, 2005). Define the *density* of a tree as the ratio of its total edge-weight over the total group-weight of the groups covered by the tree. Each step of the greedy algorithm constructs a low-density subtree T' and adds it to a partial solution T being constructed. The greedy step repeats until the total weight of groups covered by T exceeds the target ν. Each subtree T' is constructed by recursively applying the greedy algorithm on its children nodes with a series of different target values ν and then picking the lowest density one among all the subtrees found. It can be shown that a low-density subtree always exists and can be constructed efficiently. Furthermore, the union of low-density trees remains a low-density tree, which provides an approximately optimal solution to the group Steiner problem. The algorithm uses a series of technical ideas to limit the number of recursive calls so that it runs in polynomial time.

Theorem 1 (Calinescu and Zelikovsky 2005)**.** *Assume that the group-weights of a group Steiner problem are represented as non-negative integers. For any constant $\epsilon > 0$, there is a polynomial-time algorithm that computes a near-optimal group Steiner tree within a factor $\mathcal{O}(\log |V|^{2+\epsilon} \log \nu)$ of the optimal one.*

The constant ϵ is a parameter that can be tuned to improve the approximation ratio, but at a greater computational cost.

3 Informative Path Planning

3.1 Problem Definition

Formally an IPP problem is specified as a tuple $\mathcal{I} = (X, d, H, \rho, O, \mathcal{Z}, r)$. First, X is a finite set of sensing locations, with associated distance metric $d(x, x')$ for any two locations $x, x' \in X$. Next, H is a finite set of hypotheses, and $\rho(h)$ specifies the prior probability of hypothesis $h \in H$. We also have a finite set of observations O and a set of observation functions $\mathcal{Z} = \{Z_x \mid x \in X\}$, with one observation function Z_x associated with each location x. For generality, we define the observation functions probabilistically: $Z_x(h, o) = p(o|x, h)$. For noiseless observations, $Z_x(h, o)$ is either 1 or 0. In this work, we focus mainly on the noiseless case. Finally, r is the robot's start location. To simplify the presentation, we assume $r \notin X$. Either r provides no useful sensing information or the robot has already visited r and acquired the information.

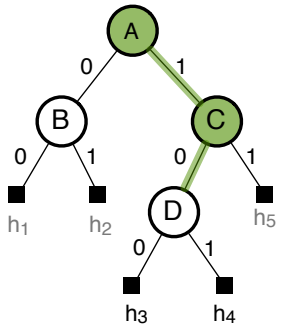


Figure 1: A policy tree with sensing locations $\{A, B, C, D\}$, observations $\{0, 1\}$, hypotheses $\{h_1, h_2, \ldots, h_5\}$. With noiseless observations, every path in a policy tree from the root to a leaf uniquely identifies a hypothesis. Suppose that a robot follows the shaded path σ. Then a hypothesis h is consistent with all observations received along σ if and only if h belongs to the subtree rooted at the node D, i.e., $\{h_3, h_4\}$.

We say that a hypothesis h is *consistent* with an observation o at a sensing location x if $Z_x(h, o) = 1$. Otherwise, it is *inconsistent*. If a hypothesis is inconsistent with a received observation, it clearly is not the true hypothesis and can be eliminated from further consideration.

In adaptive planning, the solution is a *policy* π, which can be represented as a tree. Each node of the policy tree is labeled with a sensing location $x \in X$, and each edge is labeled with an observation $o \in O$ (Fig. 1). To execute such a policy, the robot starts by moving to the location at the root of the policy tree and receives an observation o. It then follows the edge labeled with o and moves to the next location at the child node. The process con-

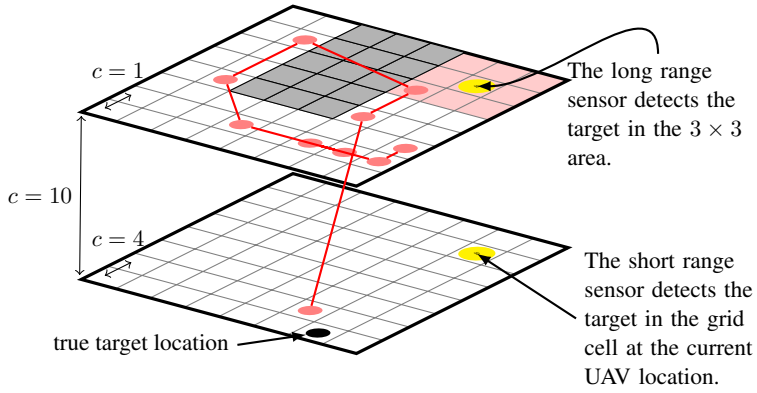


Figure 2: Search for a stationary target in an 8×8 grid. At the high altitude, the long-range sensor provides no information in the area shaded in gray, due to occlusion. The red curve indicates a sample path generated by RAId.

tinues until the robot identifies the true hypothesis. Thus every path in the policy tree of π uniquely identifies a hypothesis $h \in H$. Let $C(\pi, h)$ denote the total cost of traversing this path. Our goal is to find a policy that identifies the true hypothesis by taking observations at the chosen locations and minimizes the expected cost of travel.

We now state the problem formally:

Problem 1. *Given an IPP problem* $\mathcal{I} = (X, d, H, \rho, O, \mathcal{Z}, r)$*, compute an adaptive policy* π *that minimizes the expected cost to uniquely identify a hypothesis*

$$C(\pi) = \mathrm{E}_H C(\pi, h) = \sum_{h \in H} C(\pi, h)\rho(h). \quad (1)$$

We assume without loss of generality that in the worst case, the true hypothesis can be identified by visiting all locations in X.

Example. We now illustrate the definition above with a concrete example. A UAV searches for a stationary target in an area modeled as an 8×8 grid and must identify the grid cell that contains the target (Fig. 2). Initially the target may lie in any of the cells with equal probabilities.

The UAV can operate at two different altitudes. At the high altitude, it uses a long-range sensor that determines whether the 3×3 grid around its current location contains the target. At the low altitude, the UAV uses a more accurate short-range sensor that determines whether the current grid cell contains the target. Some grid cells are not visible from the high altitude because of occlusion, and the UAV must descend to the low altitude in order to search these cells.

The UAV starts at the low altitude. We use the Manhattan distance between two grid cells as the basis of calculating the movement cost. The cost of flying between two adjacent cells at the high altitude is 1. The corresponding cost at the low altitude is 4. The cost to move between high and low altitudes is 10.

Compared with the high altitude, the low altitude offers more accurate information over a smaller area and incurs a higher cost. The challenge is then to manage this trade-off.

In this example, the hypotheses are the grid cells that may contain the target: $H = \{(i, j) \mid i, j \in \{1, 2, \ldots, 8\}\}$. The prior probability ρ over the hypotheses is the uniform distribution. The sensing locations are the UAV locations at the two altitudes: $X = \{(i, j, k) \mid i, j \in \{1, 2, \ldots, 8\}, k \in \{1, 2\}\}$. The metric d is the shortest distance to move between two grid cells. There are two observation: $O = \{1, 0\}$, indicating whether the target is detected or not. The observation function $Z_x(h, o)$ specifies whether the hypothesis h is consistent

with the observation o received at location x. For example, if the UAV receives observation $o = 0$ at a low-altitude location $x = (2, 2, 1)$, a single hypothesis $h = (2, 2)$ is inconsistent and can be eliminated. In comparision, if the UAV receives $o = 0$ at the corresponding high-altitude location $x = (2, 2, 2)$, nine hypotheses corresponding to grid cells adjacent to $(2, 2)$ are inconsistent and can all be eliminated. □

3.2 Informative Observations

Let $H_{x,o} \subseteq H$ be the subset of hypotheses consistent with observation o at x:

$$H_{x,o} = \{h \in H \mid Z_x(h, o) = 1\}.$$

Let $p(H_{x,o})$ be the sum of probabilities of hypotheses in the subset. We consider an observation o at location x *informative* if $p(H_{x,o}) \leq 0.5$ and define the *informative observation set* at $x \in X$:

$$\Omega_x = \{o \in O \mid p(H_{x,o}) \leq 0.5\}.$$

If an observation o is informative, then by definition, $H_{x,o}$ has small probability (less than 0.5), and $H \backslash H_{x,o}$, the set of hypotheses inconsistent with o, has large probability (greater than 0.5). The observation o is informative, because it narrows down the consistent hypotheses to a small set measured in probability.

Let o^*_x be the most likely observation at x: $o^*_x = \arg\max_{o \in O} p(H_{x,o})$. It is interesting to observe that there are only two possibilities for Ω_x:

$$\Omega_x = \begin{cases} O & \text{if } p(H_{x,o}) \leq 0.5 \text{ for all } o \in O, \\ O \setminus \{o^*_x\} & \text{otherwise.} \end{cases}$$

Consider the UAV search example again. Initially, the observation $o = 1$ at every low-altitude location x is informative, as $p(H_{x,1}) = 1/64 \leq 0.5$. The notion of being informative is intuitively correct here, because the observation $o = 1$ at a low-altitude location identifies the target location exactly. In contrast, the observation $o = 0$ is not informative at any low-altitude location x, as $p(H_{x,0}) = 63/64 > 0.5$. It eliminates a single inconsistent hypothesis with probablity $1/64$ and does not help narrow down consistent hypotheses significantly.

3.3 Algorithm

RAId is a recursive divide-and-conquer algorithm. Each recursive step constructs a near-optimal adaptive plan to traverse a subset of sensing locations in X and eliminates inconsistent hypotheses using the observations received. The traversal terminates when it reduces the probability of the current hypothesis set H by at least a half. RAId then recurses on the remaining hypotheses until only one hypothesis remains. A sketch of the algorithm is shown in Algorithm 1.

The key step in RAId is to construct a traversal that significantly reduces the current hypothesis set at a low cost. Informative observation helps in eliminating inconsistent hypotheses. If a traversal encounters an informative observation o at location x, we can eliminate all hypotheses in $H \backslash H_{x,o}$, which has probability great than 0.5 by definition, and end the traversal. However, what happens if a traversal does not encounter any informative observations? To guarantee that each traversal reduces the probability of the current hypothesis set H by at least a half, RAId constructs and solves a group Steiner problem.

The underlying graph for the group Steiner problem is the complete graph over X, and the edge-weight between two vertices x and x' is $d(x, x')$.

Next, we define one group for every hypothesis $h \in H$:

$$X_h = \{x \in X \mid Z_x(h, o) = 1 \text{ for some } o \in \Omega_x\}, \tag{2}$$

which consists of all locations with *informative observations* consistent with h. The group-weight for X_h is simply $\rho(h)$. Our definition of a group implies that an uninformative observation $o \notin \Omega_x$ must be inconsistent with h at a location $x \in X_h$, because observations are noiseless and there is only one observation consistent with a given hypothesis. Thus, if a traversal encounters an uninformative observation at a location $x \in X_h$, we can eliminate h.

Finally, we set the target $\nu = \min\big(0.5, 1 - \max_{h \in H} \rho(h)\big)$. It would be desirable, but is not possible to simply set $\nu = 0.5$. If the true hypothesis has high probability, RAId may not be able to

Algorithm 1 RAId

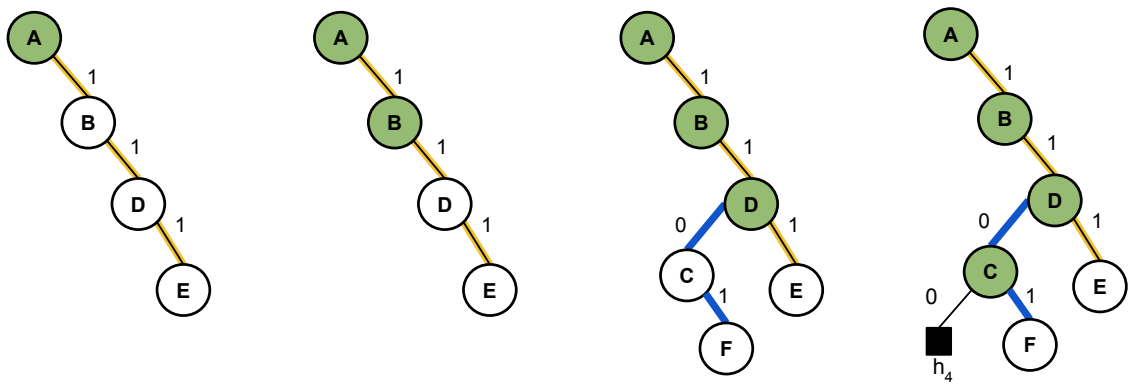
1: **procedure** RAId($X, d, H, \rho, O, \mathcal{Z}, r$)
2: **if** $|H| = 1$ **then**
3: **return** H.
4: **else**
5: $\nu \leftarrow \min\big(0.5, 1 - \max_{h \in H} \rho(h)\big)$.
6: $\tau \leftarrow$ GROUPSTEINERTOUR(X, $X \times X$, d, $\{X_h\}_{h \in H}$, ρ, ν), where $\tau = (x_0, x_1, \ldots, x_t)$ and $x_0 = x_t = r$.
7: $(H, r) \leftarrow$ EXECUTEPLAN(τ, H, r).
8: Renormalize the probability $\rho(h)$ for all $h \in H$ so that $\sum_{h \in H} \rho(h) = 1$.
9: RAId($X, d, H, \rho, O, \mathcal{Z}, r$)
10: **procedure** EXECUTEPLAN(τ, H, r)
11: $i \leftarrow 1$.
12: **repeat**
13: $r \leftarrow x_i$.
14: Visit location r and receive observation o.
15: Remove from H all hypotheses inconsistent with o.
16: $i \leftarrow i + 1$.
17: **until** $o \in \Omega_r$ or $i = t$.
18: $r \leftarrow x_t$.
19: Move to location r.
20: **return** (H, r).

achieve substantial pruning, as the remaining hypotheses have small total probability.

RAId guarantees that a traversal constructed from the group Steiner problem prunes inconsistent hypotheses that have total probability at least ν. If the robot encounters an informative observation o at a location x during the traversal, the inconsistent hypotheses $H \backslash H_{x,o}$ have probability greater than 0.5 by definition. Now suppose that the robot encounters only uninformative observations during the traversal. At each location $x \in X_h$ along the way, the robot eliminates the hypothesis h. Each hypothesis has an associated group in the group Steiner problem. The target value ν ensures that the total weight of groups covered by the traversal is greater than ν. So is the probability of eliminated hypotheses. The formal proof is given in Lemma 1.

In Algorithm 1, the procedure GROUPSTEINERTOUR($V, E, W_E, \mathcal{V}, W_{\mathcal{V}}, \nu$) solves the group Steiner problem defined in Section 2.2. However, it computes a group Steiner *tour*, i.e., a cycle in a graph-theoretic sense, instead of a tree. GROUPSTEINERTOUR consists of two steps. First, it solves for a group Steiner tree T using a greedy approximation algorithm (Calinescu and Zelikovsky, 2005). Next, it applies Christofides' metric TSP approximation algorithm (Christofides, 1976) to the vertex set of T and generates a tour. Both approximation algorithms rely critically on the metric property of the edge weight d.

RAId is an online algorithm, which interleaves planning and plan execution. In the planning phase, RAId computes a tour (Algorithm 1, line 6), which is a partial plan. The robot executes the plan by traversing the locations on the tour (Algorithm 1, line 7). At each location, the robot prunes all hypotheses inconsistent with the received observation. If the robot receives an uninformative observation, it moves to the next location on the tour. If the robot receives an informative observation or exhausts the tour, it ends the traversal and returns to



(*a*) (*b*) (*c*) (*d*)

Figure 3: A example run of RAId with two recursive calls. Shaded edges in color indicate the two tours planned in the recursive calls. Shaded nodes indicate the locations that the robot traverses while executing the plans. We assume that observation 0 is informative and observation 1 is uninformative. (*a*) RAId's first recursive call generates the tour (A, B, D, E). The robot moves to the first location A on the tour and receives observation 1. (*b*) Since observation 1 is uninformative, the robot next moves to B and receives observation 1 again. (*c*) Upon receiving the first informative observation 0 at D, the robot ends the traversal. RAId replans in the second recursive call and generates a new tour (C, F). (*d*) The robot moves to C. It receives 0 at C and identifies the hypothesis h_4.

the start location. RAId then replans a new tour, and the whole process repeats.

Returning to the start location simplifies the analysis in Section 4. However, it is not required in practice. In our experiments in Section 5, the robot starts the new traversal from its current location without returning. Fig. 3.3 shows an example run of RAId.

4 Analysis

The analysis of RAId focuses two main issues:

- the total probability of hypotheses eliminated in each traversal, and
- the associated travel cost.

We proceed in two main steps. In the first step, we analyze a variant of IPP, called *rooted IPP*, in which the robot must return to the start location r in the end. Our main idea is to show that each group Steiner tour computed enables the robot to either prune inconsistent hypotheses with probability at least 0.5 or identify the true hypothesis (Lemma 1). Furthermore, the robot traversing such a tour incurs a cost not more than twice the expected cost of an optimal policy (Lemmas 2 and 3). By bounding the number of recursive calls to RAId, we then obtain a result on its performance for rooted IPP (Theorem 2). In the second step, we exploit this result to bound the performance of RAId for IPP itself (Theorem 3).

We consider only rooted IPP for Lemma 1–4 and Theorem 2.

Lemma 1. *Let $H' \subset H$ be the set of remaining hypotheses after a single recursive call to RAId. Then, either $p(H') \leq 0.5$ or $|H'| = 1$.*

Proof. In each recursive call to RAId, the robot follows a group Steiner tour τ. If it receives an observation $o \in \Omega_x$ at some location x on τ, then the robot returns to r immediately (Algorithm 1, line 19) and $p(H') = p(H_{x,o}) \leq 0.5$ by definition of Ω_x. Otherwise, the robot visits every location x on τ and receives at every x an observation $o^*_x \notin \Omega_x$. Consider $x \in X_h$ for some x on τ and $h \in H$. If the robot receives the observation $o^*_x \notin \Omega_x$ at x, then h is inconsistent with

o^*_x by the definition of X_h and is pruned. Since the target of our group Steiner problem is ν, the pruned hypotheses has probability at least ν, and the remaining hypothesis set H' has probability at most $1-\nu$. If there is a single hypothesis h^* with $p(h^*) \geq 0.5$, then h^* must be the only remaining hypothesis. Otherwise, $p(H') \leq 1-\nu \leq 0.5$. □

Next, we bound the edge-weight of an optimal group Steiner tour.

Lemma 2. *Let π^* be an optimal policy for a rooted IPP problem $\mathcal{I}$. Let W^* be the total edge-weight of an optimal group Steiner tour for $\mathcal{I}$. Then $W^* \leq 2C(\pi^*)$.*

Proof. First, we extract a path σ from an optimal policy tree π^* and use σ to construct a feasible, but not necessarily optimal solution σ_{r} to the group Steiner problem for $\mathcal{I}$. Next, we show that the optimal policy traverses σ with probability at least 0.5. This allows us to bound the total edge-weight of σ_{r} and thus that of an optimal group Steiner tour by the cost of the optimal policy.

Let $(r, x_1, x_2, \ldots, r)$ be a path in the optimal policy tree π^* such that every edge following a node x_i in the path is labeled with the most likely observation $o^*_{x_i} = \arg\max_{o \in O} p(H_{x_i,o})$. For any subpath ϕ, $H_\phi = \{h \in H \mid Z_{x_i}(h, o^*_{x_i}) = 1 \text{ for all } x_i \text{ in } \phi\}$ is the set of hypotheses consistent with the observations received at all locations in ϕ. Let $\sigma = (r, x_1, x_2, \ldots, x_s)$ be the shortest subpath of $(r, x_1, x_2, \ldots, r)$ such that $p(H_\sigma) \leq 1-\nu$, where the length of σ is measured in the number of nodes in the path.

We now show that the tour $\sigma_{\mathrm{r}} = (r, x_1, x_2 \ldots, x_s, r)$ is a feasible solution to the group Steiner tour problem. The key issue is to determine the total group-weight of $\mathcal{X}$, the collection of groups covered by $x_1, x_2, \ldots, x_s$. At each location x_i on σ, the robot receives an observation $o^*_{x_i}$. If a hypothesis $h \in H$ is inconsistent with $o^*_{x_i}$, then h must be consistent with some $o \neq o^*_{x_i}$, i.e., $Z_{x_i}(h, o) = 1$ for $o \in \Omega_{x_i}$. Then $x_i \in X_h$ by definition. In other words, x_i covers X_h if h is inconsistent with $o^*_{x_i}$ at x_i, and $\mathcal{X} = \{X_h \mid Z_{x_i}(h, o^*_{x_i}) = 0 \text{ for some } x_i \text{ in } \sigma\}$. Since $p(H_\sigma) \leq 1-\nu$, the total group-weight of $\mathcal{X}$ must be least ν. This proves that σ_{r} is a feasible group Steiner tour.

Now consider the subpath $\sigma' = (r, x_1, x_2, \ldots, x_{s-1})$. We have $p(H_{\sigma'}) > 1-\nu$, as σ is the *shortest* path with $p(H_\sigma) \leq 1-\nu$. To bound the expected cost of the optimal policy π^*,

$$C(\pi^*) = \sum_{h \in H} \rho(h) C(\pi^*, h) \geq \sum_{h \in H_{\sigma'}} \rho(h) C(\pi^*, h).$$

$H_{\sigma'}$ can be interpreted as the set of hypotheses that visit $x_1, \ldots, x_s$ but not necessarily receive $o^*_{x_s}$ at x_s. Hence for any $h \in H_{\sigma'}$, the path that leads to h in the optimal policy tree π^* must contain σ as a subpath. Thus,

$$C(\pi^*) \geq \sum_{h \in H_{\sigma'}} \rho(h) w(\sigma_{\mathrm{r}}) \geq (1-\nu) w(\sigma_{\mathrm{r}}) \geq (1-\nu) W^*,$$

where $w(\sigma_{\mathrm{r}})$ is the total edge-weight of the tour σ_{r}. Rearranging the inequality above, we get

$$W^* \leq \frac{1}{1-\nu} \cdot C(\pi^*) \leq 2C(\pi^*).$$

□

Lemma 3. *If RAId computes an optimal group Steiner tour, then the robot travels a path with cost at most $2C(\pi^*)$ in each recursive step of RAId.*

Proof. In each recursive step of RAId, the robot travels a path whose cost is bounded by the total edge-weight of the group Steiner tour computed. The conclusion then follows directly from Lemma 2. □

Before moving to our first theorem, we need to connect a rooted IPP problem to its subproblems, as RAId is recursive.

Lemma 4. *Suppose that π^* is an optimal policy for a rooted IPP problem $\mathcal{I}$ with hypothesis set H and prior probability distribution ρ. Let $\{H_1, H_2, \ldots, H_n\}$ be a partition of H, and let π^*_i be an optimal policy for the subproblem $\mathcal{I}_i$ with hypothesis set H_i and prior probability distribution*

ρ_i, where $\rho_i(h) = \rho(h)/\rho(H_i)$ for each $h \in H_i$. Then we have

$$\sum_{i=1}^{n} \rho(H_i) C(\pi_i^*) \leq C(\pi^*).$$

Proof. For each subproblem $\mathcal{I}_i$, we can construct a feasible policy π_i for $\mathcal{I}_i$ from the optimal policy π^* for $\mathcal{I}$. Consider the policy tree π^*. Every path from the root of π^* to a leaf uniquely identifies a hypothesis $h \in H$. So we choose the policy tree π_i as the subtree of π^* that consists of all the paths leading to hypotheses in H_i. Clearly π_i is feasible, as it identifies all the relevant hypotheses. Then,

$$\begin{aligned}\sum_{i=1}^{n} \rho(H_i) C(\pi_i^*) &\leq \sum_{i=1}^{n} \rho(H_i) C(\pi_i) \\ &\leq \sum_{i=1}^{n} \rho(H_i) \sum_{h \in H_i} \frac{\rho(h)}{\rho(H_i)} \cdot C(\pi_i, h) \\ &= \sum_{h \in H} \rho(h) C(\pi^*, h) = C(\pi^*).\end{aligned}$$

□

We are now ready to bound the performance of RAId for rooted IPP, under an assumption which we relax later.

Theorem 2. *Let π denote the policy that RAId computes for a rooted IPP problem. If RAId computes an* optimal *group Steiner tour in each step, then*

$$C(\pi) \leq 2\left(\log\left(1/\delta\right) + 1\right) C(\pi^*),$$

where $C(\pi)$ is the expected cost of RAId and $\delta = \min_{h \in H} \rho(h)$.

Proof. By Lemma 1, if a recursive step of RAId does not terminate, it reduces the probability of consistent hypotheses by a factor of $1/2$. For any $h \in H$, the number of recursive steps required is then at most $\log(1/\delta) + 1$.

We now complete the proof by induction on the number of recursive calls to RAId. For the base case of $k = 1$ call, $C(\pi) \leq 2C(\pi^*)$ by Lemma 3. Assume that $C(\pi) \leq 2(k-1)C(\pi^*)$ when there are at most $k-1$ recursive calls. Now consider the induction step of k calls. The first recursive call partitions the hypothesis set H into a collection of mutually exclusive subsets, $H_1, H_2, \ldots, H_n$. Let $\mathcal{I}_i$ be the subproblem with hypothesis set H_i and optimal policy π_i^*, for $i = 1, 2, \ldots, n$. After the first recursive call, it takes at most $k-1$ additional calls for each $\mathcal{I}_i$. In the first call, the robot incurs a cost at most $2C(\pi^*)$ by Lemma 3. For each $\mathcal{I}_i$, the robot incurs a cost at most $2(k-1)C(\pi_i^*)$ in the remaining $k-1$ calls, by the induction hypothesis. Putting together this with Lemma 4, we conclude that the robot incurs a total cost of at most $2kC(\pi^*)$ when there are k calls. □

Finally, we use Theorem 2 to analyze the performance of RAId on IPP rather than rooted IPP. To start, we argue that a rooted IPP solution provides a good approximate solution for IPP.

Lemma 5. *An α-approximation algorithm for rooted IPP is a 2α-approximation algorithm for IPP.*

Proof. Let C^* and C_{r}^* be the expected cost of an optimal policy for an IPP problem $\mathcal{I}$ and for a corresponding rooted IPP problem $\mathcal{I}_{\mathrm{r}}$, respectively. Since any policy for $\mathcal{I}$ can be turned into a policy for $\mathcal{I}_{\mathrm{r}}$ by retracing the solution path back to the start location, we have $C_{\mathrm{r}}^* \leq 2C^*$. An α-approximation algorithm for rooted IPP computes a policy π for $\mathcal{I}_{\mathrm{r}}$ with expected cost $C_{\mathrm{r}}(\pi) \leq \alpha C_{\mathrm{r}}^*$. It then follows that $C_{\mathrm{r}}(\pi) \leq \alpha C_{\mathrm{r}}^* \leq 2\alpha C^*$ and this algorithm provides a 2α-approximation to the optimal solution of $\mathcal{I}$. □

To obtain our main result, we need to address two remaining issues. First, Theorem 2 assumes that RAId computes an optimal group Steiner tour. This is, however, not achievable in polynomial time under standard assumptions. RAId uses a polynomial-time greedy algorithm (Calinescu and Zelikovsky, 2005) that computes a group Steiner tree T with a guaranteed approximation factor. It then applies Christofides' metric TSP algorithm (Christofides, 1976) to the vertex set of T and generates a tour, instead of traversing T directly, because Christofides algorithm provides a guaranteed $3/2$-approximation to the optimal TSP

tour. Second, the greedy group Steiner approximation algorithm assumes integer group-weights. To apply this algorithm and obtain the approximation bound, we assume that the prior probabilities are coded in non-negative integers. We remove the renormalization step (Algorithm 1, line 8) and make other minor changes accordingly. Normalization of probabilities is not necessary for RAId. It only simplifies presentation.

Theorem 3. *Let $\mathcal{I} = (X, d, H, \rho, O, \mathcal{Z}, r)$ be an IPP problem. Assume that the prior probability distribution ρ is represented as non-negative integers with $\sum_{h\in H} \rho(h) = P$. Let $\delta = \min_{h\in H} \rho(h)/P$. For any constant $\epsilon > 0$, RAId computes a policy π for $\mathcal{I}$ in polynomial time such that $C(\pi) \in O((\log|X|)^{2+\epsilon} \log P \log(1/\delta) C(\pi^*))$.*

Proof. In the group Steiner problem for $\mathcal{I}$, the vertex set is X. From Theorem 1, the greedy approximation in RAId computes an α-approximation T to the optimal group Steiner tree T^*, with $\alpha \in O((\log|X|)^{2+\epsilon} \log P)$. The total edge-weight of an optimal group Steiner tree, $w(T^*)$, must be less than that of an optimal group Steiner tour, W^*, as we can remove any edge from a tour and turn it into a tree. Thus, $w(T) \le \alpha\, w(T^*) \le \alpha\, W^*$. Applying Christofides' metric TSP to the vertices of T produces a tour τ, which has weight $w(\tau) \le 2w(T)$, using an argument similar to that in (Christofides, 1976). It then follows that $w(\tau) \le 2\alpha W^*$. In other words, RAId obtains a 2α-approximation to the optimal group Steiner tour. Putting this together with Theorem 2 and Lemma 5, we get the desired approximation bound. The algorithm clearly runs in polynomial time. □

IPP is an NP-hard optimization problem. RAId provides a polylogarithmic approximation algorithm that runs in polynomial time. The computational bottleneck of RAId lies in the recursive calls to GROUPSTEINERTOUR, which computes an approximate solution to the group Steiner problem. The running time of GROUPSTEINERTOUR is roughly linear in the number of hypotheses and the number of locations.

5 Experiments in Simulation

5.1 Setup

For comparison, we implemented three types of algorithms: greedy algorithms, finite-horizon lookahead search, and submodular optimization. The experiments focus on two main differentiating characteristics of these algorithms: planning horizon and adaptivity. See Table 1 for a summary and the subsections below for detailed explanation.

Greedy Algorithms

We first describe two greedy algorithms, which are simple and widely used in practice: *information gain* (IG) and *information gain with cost* (IGC). Let Q denote the random variable representing the true hypothesis. Suppose that the robot is currently located at x. If it receives observation o at the next location x', the information gain is $\mathbb{H}(Q) - \mathbb{H}(Q|x', o)$, where $\mathbb{H}$ denotes the Shannon entropy. Entropy measures the uncertainty in a random variable. Reducing entropy is the same as gaining information. IG always chooses the next location x' to maximize the *expected* information gain

$$f_{\mathrm{IG}}(x') = \sum_{h\in H} \sum_{o\in O} \Big(\mathbb{H}(Q) - \mathbb{H}(Q \mid x', o)\Big) p(o|x', h) p(h).$$

in a greedy manner. When there are only two observations, IG is equivalent to *generalized binary search* (Nowak, 2008), as shown by Zheng et al. (2005).

To account for robot movement cost, IGC maximizes information gain per unit movement cost

$$f_{\mathrm{IGC}}(x') = \sum_{h\in H} \sum_{o\in O} \frac{\mathbb{H}(Q) - \mathbb{H}(Q \mid x', o)}{d(x, x')}\, p(o|x', h) p(h),$$

again in a greedy manner.

Greedy algorithms are *myopic*: they do not reason over the long term. They achieve limited adaptivity by replanning in each step.

Table 1: The main characteristics of algorithms under comparison.

	RAId	IG, IGC	IGC-2	NAId-Replan
Nonmyopic	yes	no	finite horizon	yes
Adaptive	yes	replanning	replanning	replanning

Finite-Horizon Lookahead Search

To alleviate the weakness in greedy algorithms, one idea is to search over a finite horizon k for a depth-k policy tree (Fig. 1) with the best expected heuristic value (Hollinger et al., 2009). We use IGC as the heuristic and call the resulting algorithm IGC-k. The original greedy IGC algorithm corresponds to IGC-1. IGC-k replans in each step. It performs a lookahead search for the best policy tree, and the robot executes the first step of the chosen policy. The process then repeats. Since each policy tree node chooses among $|X|$ sensing locations and branches on $|O|$ observations, there are $\mathcal{O}(|X|^k|O|^{k-1})$ policy trees of depth k. Clearly, with large $|X|$ and $|O|$, k must be kept small for the finite-horizon search to be practical. Some of the tasks in our experiments can have up to 170 sensing locations and 22 observations at each location. We had to set $k = 2$ to keep the total running time reasonable.

The planning horizon IGC-k is longer than that of its greedy counterpart, but is bounded by the finite constant k a priori. IGC-k achieves limited adaptivity through replanning, just as the greedy algorithms.

Submodular Optimization

Submodular optimization is another interesting idea for IPP, e.g., the NAIVE algorithm (Section 2.1). NAIVE requires a submodular function with the locality property for guaranteed performance. It is unclear how to construct such functions for the tasks in our experiments. Instead, we use an expected version space reduction function to search for a near-optimal path σ:

$$f_{\mathrm{VSR}}(\sigma) = 1 - \sum_{h \in H} \big(p(H_{\sigma,h}) - p(h)\big)p(h),$$

where $H_{\sigma,h}$ denotes the set of hypotheses with the same observation as h at every location on σ. Intuitively, maximizing f_{VSR} results in a path that maximally reduces the set of confounding hypotheses. If a path σ always eliminates all confounding hypotheses, then $p(H_{\sigma,h}) = p(h)$ for all $h \in H$, and $f_{\mathrm{VSR}}(\sigma) = 1$. The function f_{VSR} is submodular, but may not satisfy the locality property required by NAIVE.

Finding a minimum-cost path σ such that $f_{\mathrm{vsr}}(\sigma) = 1$ is a minimum-cost submodular coverage problem. To solve it, we use the greedy polymatroid Steiner algorithm (Calinescu and Zelikovsky, 2005). Although both submodular optimization and RAId make use of the polymatroid Steiner algorithm (group Steiner algorithm is a special case), they differ in their objectives. Submodular optimization searches for an open-loop plan, i.e., a path that maximizes f_{VSR}. It does not consider future observations during planning and is nonadaptive.

There are two ways to execute the computed path. One is to have the robot traverse every location on the path until the end. Alternatively, NAIVE replans in every step. It plans a path, but the robot visits only the first location on the path. The process then repeats. We follow NAIVE's approach: it is more adaptive, but has a higher computational cost. We call the resulting algorithm *nonadaptive hypothesis identification with replanning* (NAId-Replan).

An alternative way of solving the minimum-cost submodular coverage problem is the recursive greedy algorithm (Chekuri and Pal, 2005) used in

(Singh et al., 2009b). We implemented this algorithm, but found it too slow to be practical for our tasks.

In summary, NAId-Replan is nonmyopic. It shares the same basic idea as RAId, but performs nonadaptive planning. It achieves limited adaptivity through replanning. NAId-Replan is also related to NAIVE. It performs submodular optimization, but the submodular function used does not possess the locality property required by NAIVE for theoretical performance guarantee.

We implemented all algorithms in the Clojure language and compared their performance on a set of tasks in simulation. For each task, we ran the algorithms on every hypothesis in H and calculated the average policy cost weighted by the prior probabilities. The running times were obtained on a computer server with an Intel Xeon 2.4GHz processor.

5.2 Results

Overall, RAId obtains the best or nearly the best policies in all tasks in our experiments, according to their average policy costs (Table 2). The other algorithms may perform well in some tasks, but very poorly in others. While RAId has performance guarantees, it will be not surprising for greedy algorithm to outperform RAId on some problems due to the approximation factors in the performance bound. In general, it is difficult to tell the effectiveness of an algorithm in advance. As IGC is easy to implement, one could try it as a first approach for the problem of interest.

While the average policy cost is our main performance measure, we also report the total *planning* time for completeness (Table 3). RAId is slower that the greedy algorithms. This is expected, as greedy algorithms perform only short-term planning. RAId are much faster than IGC-2 and NAId-Replan, which both perform longer-term planning.

Although our implementation is not optimized as a result of the implementation language, the running times, which are on the order of seconds for these moderate-scale tasks, are useful for a range of online robot planning tasks.

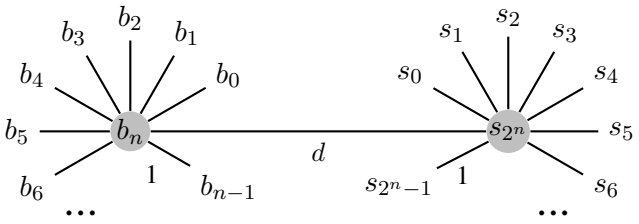


Figure 4: The 2-star graph.

2-Star Graph

We start with a simple example to gain some understanding of the key issues. There are a total of 2^n possible hypotheses $H = \{0, 1, 2, \ldots, 2^n - 1\}$, with equal probability of occurring. Each hypothesis $h \in H$ is coded in its binary representation.

To identify the true hypothesis, the robot visits the nodes in a graph consisting of two connected stars (Fig. 4). One star has center b_n and n peripheral nodes $b_0, b_2, \ldots, b_{n-1}$. The other star has center s_{2^n} and 2^n peripheral nodes $s_0, s_1, \ldots, s_{2^n-1}$. There is an edge connecting the two centers nodes, with edge-weight d. The weight of an edge between a center and a connected peripheral node is 1. The set X contains only the peripheral nodes and not the two centers, b_n and s_{2^n}, which serve only the purpose of connecting the peripheral nodes. The robot is initially located at s_{2^n}.

At each node b_i in X, the robot receives observation 1 if the ith bit of the true hypothesis h is 1, and receives 0 otherwise. At each node s_i in X, the robot receives observation 1 if $h = i$, and receives 0 otherwise. Clearly the b-nodes provide much more informative observations than the s-nodes. Visiting b-nodes is similar to binary search, while visiting s-nodes is similar to linear search. Since the robot starts at s_{2^n}, the main issue is to decide whether to pay the high cost of traversing the inter-star edge in order to benefit from the more informative observations at the b-nodes. Unfortunately, even in this very simple example, the issue cannot be resolved locally in a greedy manner.

In this experiment, the two nonmyopic algorithms, RAId and NAId-Replan, consistently ob-

Table 2: Average cost of a computed policy over all hypotheses.

	Cost				
	RAId	IG	IGC	IGC-2	NAId-Replan
2-Star (d=10, n=5)	19.0	25.3	32.9	33.9	19.0
2-Star (d=10, n=6)	21.0	27.9	22.3	52.5	21.0
2-Star (d=53, n=6)	65.0	102.1	62.0	68.9	78.8
2-Star (d=53, n=7)	66.0	102.4	127.4	118.5	66.0
2-Star (d=53, n=8)	68.0	100.9	257.7	258.7	68.0
Adaptive 2-Star	73.0	84.3	127.8	132.2	136.2
Grasping	562.8	2822.9	839.9	775.1	597.3
UAV Search	83.6	97.2	142.7	133.6	151.4

Table 3: Average total planning time, excluding the time for plan execution.

	Time (seconds)				
	RAId	IG	IGC	IGC-2	NAId-Replan
2-Star (d=10, n=5)	0.5	0.0	0.0	1.6	7.8
2-Star (d=10, n=6)	1.4	0.1	0.1	15.3	68.6
2-Star (d=53, n=6)	1.3	0.1	0.8	16.9	1045.4
2-Star (d=53, n=7)	5.2	0.4	5.6	3.3	684.2
2-Star (d=53, n=8)	22.6	1.4	3.4	4305.5	8415.7
Adaptive 2-Star	3.3	0.2	3.4	417.5	10290.6
Grasping	22.9	2.4	4.1	88.7	4523.5
UAV Search	25.5	0.5	2.5	157.4	16753.5

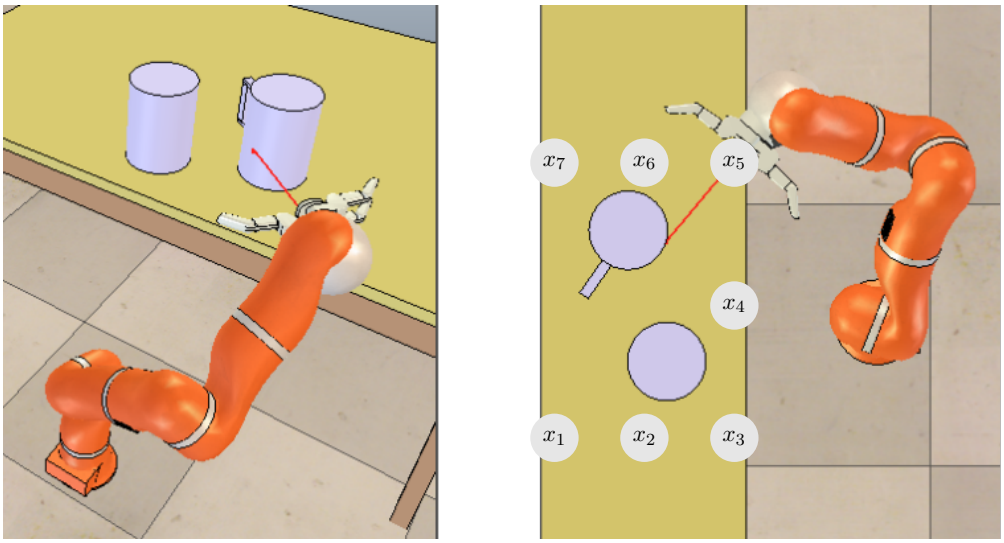


Figure 5: Grasp the cup with a handle. The figure shows the side view (left) and the top view (right) of the same robot configuration with the robot hand on the right side of the table.

tain good policies (Table 2).

The greedy algorithms do not perform as well. Curiously IG sometimes outperforms IGC. This is, however, coincidence. By completely ignoring the movement cost, IG naturally moves to the more informative b-nodes. IGC reasons about cost, but it is unable to decide optimally whether to jump to b-nodes or stay on s-nodes. In the two instances with $d = 10$, the optimal policy stays with the s-nodes when $n = 5$; it jumps to the b-nodes when $n = 6$. IGC always moves to the b-nodes, simply because the movement cost is low. Hence, IGC underperforms when $n = 5$. In the instances with $d = 53$, IGC is again misled by the greedy local analysis and decides to stay at the s-nodes, because it is cheaper to reach them. This is optimal when $n = 6$, but the performance degrades quickly when $n = 7$ or 8. In fact, IGC's regret, measured against the optimal policy, increases exponentially, as n grows.

Compared with the greedy algorithms, IGC-2 has longer planning horizon. Although it takes more computational time, IGC-2 fails to obtain better policies. It seems that a horizon of 2 is still insufficient for the tasks here.

It is somewhat surprising that the optimal policies for our 2-star graph instances are in fact nonadaptive. Intuitively the optimal policy would either (i) always stay on the s-nodes or (ii) jump to the b-nodes and stay there, depending on the d and n values, until the true hypothesis is identified. The traversal does not depend on the observations received, and adaptivity is not required. This is confirmed by examining the results computed by RAId. The nonadaptive optimal policies explain why RAId and NAId-Replan achieve comparable performance.

Adaptive 2-Star Graph

To better understand the issue of adaptivity, let us now modify the 2-star graph so that the optimal policy is adaptive. For $i = 0, 1, \ldots, n-1$, replace each peripheral node b_i in the 2-star graph by m copies, $b_{i,0}, b_{i,1}, \ldots b_{i,m-1}$, each connected to the center b_n by an edge of weight 1. For each i, only one of the m copies is *informative*. A function $g(h, i)$ specifies the index of the informative node for every $h \in H$ and $i \in [0, n-1]$. At an informative node $b_{i,j}$, the observation provides two values: the binary value of the ith bit of h and the index of the informative node for the next bit, $g(h, i+1)$. At an uninformative node, the observation provides no information. With this modification, an optimal policy must locate the informative b-nodes based on the observation received.

With suitable d and n values, an optimal policy visits $b_{0,0}, b_{0,1}, b_{0,2}, \ldots$ until reaching the first informative node. It then uses the information from the received observation to move to the next informative node and so on. This is clearly an adaptive policy. A nonadaptive policy cannot change its behavior based on the observation received and is suboptimal.

This example is constructed, but not necessarily artificial. The basic idea is that each informative node contains a "map" that points to the next location of interest.

In the experiment, $d = 53$, $n = 7$, and $m = 5$. The function g is randomly generated, but remains fixed for all runs. RAId significantly outperforms all of IGC, IGC-2, and NAId-Replan. Although NAId-Replan achieves some level of adaptivity through replanning, it is inadequate.

Grasping a Cup

There are two cups on the table, one with a handle and one without. A robot arm needs to lift the cup with a handle by grasping on the handle (Fig. 5). Using an external camera placed on the left side of the table, the robot can accurately sense the positions of the two cups. However, due to occlusion, it is uncertain which cup has a handle and where the handle is.

Each hypothesis (κ, θ) has two parameters: κ is a binary value that indicates which cup has a handle, and θ is the cup's orientation, which determines the handle location. The handle faces away from the external camera. So those hypotheses have higher prior probabilities.

The robot arm has a single-beam laser range

finder mounted at its the wrist. The range finder reports the (discretized) distance to the nearest object in the direction that the range finder is facing.

We sample seven wrist positions $x_1, x_2, \ldots, x_7$ around the cups (Fig. 5). At each position, the robot can pan the range finder in the plane parallel to the tabletop. Panning by a fixed amount incurs a cost of 4. Moving the wrist from one position to another incurs a higher cost: the distance between the current position and the target position, scaled up by a factor of 15. The robot arm starts at wrist position x_1 on the left side of the table.

RAId achieves the lowest cost in this experiments. Under RAId, the robot moves progressively from x_1 to x_7 and pans the range finder at each position to take observations. This is a good strategy, because it avoids excessive robot arm movement, which incurs high cost.

IG performs very poorly, because it completely ignores the difference in action costs and moves the robot arm excessively between the various wrist positions in order to seek sometimes minor additional information gain. IGC does not perform well either. Under IGC, the robot moves to x_6 in the first step, because it expects to see the handle from there with high probability according to the prior. However, with small probability, the cup is oriented so that the handle is not visible from x_6. In this case, the robot must pay a high cost to travel back to the other positions. On the average, the aggressive move to x_6 does not pay off. This example clearly shows the weakness of greedy strategies, which do not plan *multiple steps* ahead.

IGC-2 achieves lower cost than IGC, because of its slightly longer planning horizon, but it is substantially worse than RAId.

NAId-Replan achieves comparable, but slightly worse result than RAId. NAId-Replan is nonmyopic. It is also adaptive, to a limited extent. We suspect that similar to the 2-star graph, adaptivity has limited benefit for this task, but there is no easy way to verify this.

UAV Search

This is the example described in Section 3. One may think that the optimal strategy is for the UAV to rise to the high altitude, search and locate the target in a 3×3 area, and finally descend to the low altitude in order to localize the target precisely. RAId, however, does not always do this, because the cost of descending is high. Fig. 2 shows a sample run of RAId. After identifying the 3×3 area, the UAV stays at the high altitude. It moves around in the neighborhood and fuses the observations received to localize the target precisely without descending.

IGC does not perform well, again because it does not plan multiple steps ahead. It fails to recognize that although the cost of climbing to the high altitude seems high in one step, the cost can be amortized over many future high-altitude observations, which are more informative. Under IGC, the UAV always stays on the low altitude and does not climb up. The result does not improve much even with 2-step lookahead in IGC-2. Under IGC-2, the UAV climbs up only occasionally in some instances. NAId-Replan does not perform well, either. Replanning does not provide sufficent adaptivity for this task.

6 Noisy Observations

Although RAId is designed for noiseless observations, we now describe a simple extension, *Noisy RAId*, to handle noisy observations. Our strategy is first to create a noiseless IPP problem $\mathcal{I}' = (X, d, H', \rho', O, \mathcal{Z}', r)$ from the original noisy one $\mathcal{I} = (X, d, H, \rho, O, \mathcal{Z}, r)$, by associating a hypothesis with observations. For noiseless observations, each hypothesis h has a unique observation vector $(o_1, o_2, \ldots, o_{|X|})$, where $Z_{x_i}(h, o_{x_i}) = 1$ for each location $x_i \in X$. This one-to-one relationship allows us to represent a hypothesis by its associated observation vector. The hypothesis space H is then simply a set of points in $O^{|X|}$. For noisy observations, the one-to-one relationship no longer holds, but the intuition of associating hypotheses

with their observation vectors remains valid.

Formally we set $H' = O^{|X|}$. For a hypothesis $h' = (o_1, o_2, \ldots, o_{|X|})$ in H', the prior probability of h' is the probability of observing h' if the robot visits all locations in X: $\rho'(h') = \sum_{h \in H} \rho(h) \prod_{i=1}^{|X|} Z_{x_i}(h, o_i)$. Finally, the observation function $Z'_{x_i}(h', o) = 1$ if $o = o_i$.

Noisy RAId applies RAId to $\mathcal{I}'$ with three changes:

- For computational efficiency, we sample a set of n hypotheses from H' in each recursive step of RAId and use it an approximate representation of H'.
- Although $\mathcal{I}$ is transformed into $\mathcal{I}'$, our goal is still to acquire information on the original hypothesis space H. We maintain a probability distribution over H. Initially, $b = \rho$. Because of noise, we cannot use an observation to *eliminate* a hypothesis $h \in H$, but we can update their probabilities using the Bayes rule. Suppose that the robot receives a new observation o at location x. We replace Algorithm 1, line 15 with

  $$b(h) \leftarrow \eta \, Z_x(h, o) b(h) \text{ for every } h \in H,$$

  where η is a normalization constant.
- Finally, we terminate RAId if the most likely hypothesis $h^* = \arg\max_{h \in H} b(h)$ has probability greater than or equal to a given constant $\gamma \in (0, 1]$).

Under the assumption of noiseless observations, Noisy RAId reverts back RAId. To see this, note that in the first change, we may exhaustively sample every hypothesis in H and make $H' = H$. In the second change, $Z_x(h, o)$ is either 1 or 0. Bayesian update is then equivalent to hypothesis elimination. In the third change, we set $\gamma = 1$.

We performed preliminary experiments to evaluate this idea on the UAV Search task (Section 5.2) with two different noise levels for the high-altitude sensor. The termination condition γ was set to 0.99.

Table 4: The performance of Noisy RAId on UAV Search with noisy observations. Noise level σ means that the high-altitude sensor reports a false observation with probability σ, and n is the number of samples.

Noise	**Cost**		
	$n = 128$	$n = 192$	$n = 320$
0.01	88.5	94.3	112.3
0.05	133.8	133.2	143.7

Table 5: The average total planning time of Noisy RAId on UAV Search with noisy observations.

Noise	**Time (seconds)**		
	$n = 128$	$n = 192$	$n = 320$
0.01	20.8	28.8	40.7
0.05	44.1	52.1	55.7

We evaluated multiple settings with different numbers of samples. For each setting, we run 200 trials and averaged performance statistics. The results, reported in Tables 4 and 5, are promising. Although the size of H' is 2^{128}, the algorithm identifies the true hypothesis correctly for every trial with only a few hundred samples in all settings. In other words, it always identifies the correct hypothesis according to the ground truth. In general, the robot's travel cost increases with noisy observations, as expected. With more samples, we expect the algorithm to compute a better policy with lower cost. However, the trend in the data is not definitive. Either a small number of samples is sufficient in this case to produce a near-optimal policy or a much larger number of samples is needed for significant improvement. Further investigation is required. The average total planning time scales roughly linearly with the number samples taken.

7 Conclusion

RAId is a new algorithm for the NP-hard informative path planning problem. We show that it computes a polylogarithmic approximation to the optimal solution in polynomial time, when the robot

travels in a metric space. Furthermore, our experiments demonstrate that RAId is effective in practice and provides good approximate solutions for several distinct robot planning tasks. Although RAId is designed primarily for noiseless observations, a simple extension allows it to handle some tasks with noisy observations. However, the theoretical guarantee for RAId no longer holds when the observations are noisy.

To expand the use of RAId, there are two main challenges. One is to develop a principled and practical treatment of noisy observations with performance guarantee, possibly by borrowing ideas from Bayesian active learning (Golovin et al., 2010). The other is scalability. Currently, RAId uses a "flat" representation, which explicitly enumerates every possible hypothesis. Hierarchical or factored representations will be needed in order to scale up to very large hypothesis spaces.

Acknowledgments. This work is supported in part by A*STAR grant R-252-506-001-305, MoE AcRF grant 2010-T2-2-071, National Research Foundation Singapore through the SMART IRG research program (Subaward Agreement No. 41), and the US Air Force Research Laboratory under agreement number FA2386-12-1-4031.

We thank the anonymous reviewers whose comments helped considerably in improving the presentation of this work.

References

G. Calinescu and A. Zelikovsky. The polymatroid Steiner problems. *J. Combinatorial Optimization*, 9 (3):281–294, 2005.

V.T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania. Decision trees for entity identification: approximation algorithms and hardness results. In *Proc. ACM Symp. on Principles of Database Systems*, pages 53–62, 2007.

C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 245–253, 2005.

N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

B.C. Dean, M.X. Goemans, and J. Vondrdk. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 208–217, 2004.

H.J.S. Feder, J.J Leonard, and C.M. Smith. Adaptive mobile robot navigation and mapping. *Int. J. Robotics Research*, 18(7):650–668, 1999.

D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics & Autonomous Systems*, 25(3):195–207, 1998.

D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artificial Intelligence Research*, 42(1):427–486, 2011.

D. Golovin, A. Krause, and D. Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 766–774. 2010.

A. Gupta, V. Nagarajan, and R Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. In *Proc. Int. Conf. on Automata, Languages & Programming*, volume 6198 of *LNCS*, pages 690–701. Springer, 2010.

G. Hollinger and G. Sukhatme. Sampling-based motion planning for robotic information gathering. In *Proc. Robotics: Science and Systems*, 2013.

G. Hollinger, S. Singh, J. Djugash, and A. Kehagias. Efficient multi-robot search for a moving target. *Int. J. Robotics Research*, 28(2):201–219, 2009.

G. Hollinger, U. Mitra, and G. Sukhatme. Active classification: Theory and application to underwater inspection. In *Proc. Int. Symp. on Robotics Research*. Springer, 2011.

G. Hollinger, B. Englot, F.S. Hover, U. Mitra, and G.S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *Int. J. Robotics Research*, 32(1):3–18, 2013.

S. Javdani, M. Klingensmith, J.A. Bagnell, N. Pollard, and S.S. Srinivasa. Efficient touch based localization through submodularity. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2013.

S. Javdani, Y. Chen, A. Karbasi, A. Krause, J.A. Bagnell, and S. Srinivasa. Near Optimal Bayesian Active Learning for Decision Making. In *Proc. of Int. Conf. on Artificial Intelligence & Statistics*, 2014.

L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

A. Krause and C. Guestrin. Optimal value of information in graphical models. *J. Artificial Intelligence Research*, 35(1):557–591, 2009.

H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

R. Nowak. Generalized binary search. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 568–574, 2008.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, pages 477–484, 2003.

R. Platt Jr, L.P. Kaelbling, T. Lozano-Perez, and R. Tedrake. Simultaneous localization and grasping as a belief space control problem. In *Proc. Int. Symp. on Robotics Research*, 2011.

A. Singh, A. Krause, C. Guestrin, and W.J. Kaiser. Efficient informative sensing using multiple robots. *J. Artificial Intelligence Research*, 34(2):707–755, 2009a.

A. Singh, A. Krause, and W.J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, 2009b.

T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Uncertainty in Artificial Intelligence*, 2005.

A. X. Zheng, I. Rish, and A. Beygelzimer. Efficient Test Selection in Active Diagnosis via Entropy Approximation. *Proc. Uncertainty in Artificial Intelligence*, 2005.