APPEARED IN
Proc. International Workshop on the Algorithmic Foundations of Robotics (WAFR), 2014

Adaptive Informative Path Planning in Metric Spaces

Zhan Wei Lim, David Hsu, and Wee Sun Lee

National University of Singapore, Singapore 117417, Singapore

Abstract. In contrast to classic robot 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 using all information acquired so far. The goal is to minimize the robot's travel cost required to identify a true hypothesis. Adaptive IPP is NP-hard. 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 efficient 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.

1 Introduction

Path planning usually 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 [18] or tactile sensors [13] in order to estimate the object pose for grasping.
- An autonomous underwater vehicle inspects a ship hull for the presence of explosive devices [10].

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. This paper presents a practical algorithm, *recursive adaptive identification* (RAId), which computes a near-optimal path for the robot to identify the true hypothesis with minimum movement cost.

IPP contains, as a special case, the well-studied optimal decision tree (ODT) problem, which basically has a single location with all sensing operations. 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 [2].

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 operations executed earlier. In *adaptive* planning, we choose, in each step, new sensing operations conditioned on the outcomes of sensing operations executed earlier. This is clearly more powerful. RAId belongs to the second class.

RAId takes a divide-and-conquer approach, somewhat similar to binary search. Each recursive step of binary search chooses a single most discriminating query that prunes half of all hypotheses. RAId shares the basic idea, but is more complex. There are two main difficulties. First, we cannot choose sensing locations one at a time in isolation, because different locations provide different sensing information and moving to a location affects future choices. Second, when choosing multiple sensing locations together, we must consider not only information gain, but also movement cost. Each recursive step of RAId constructs a near-optimal adaptive plan that traverses a subset of sensing locations, by solving a group Steiner problem [1]. The traversal terminates when the robot encounters an "informative" observation, which guarantees to eliminate a significant fraction of existing hypotheses.

In the following, Section 2 briefly surveys related work. Section 3 defines informative path planning and presents RAId. Section 4 analyzes the performance of the algorithm. Section 5 compares RAId with two widely used greedy 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 Related Work

IPP is important to robotics and various related fields. The importance and the difficulty in efficiently 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 [11]. Another idea is to search for a plan over a finite horizon [10]. Although these heuristic algorithms may work well in practice, they do not provide any theoretical performance guarantee. The NAIVE algorithm replans in each step, using a nonadaptive IPP algorithm, in order to achieve adaptivity [20]. 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 [10]. This is unsurprising in light of the well-known benefit of acting adaptively [4,7]. Furthermore, to achieve nontrivial performance bound, NAIVE requires explicit construction of a submodular function with the *locality* property [20]. 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 [9]. 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 [9]. To provide the theoretical performance bound, the isolation algorithm uses linear programming in the inner loop to solve the group Steiner problem. This is impractical. RAId solves the more general IPP problem, which allows arbitrary hypothesis space and non-binary sensing. To solve the group Steiner problem, it uses a combinatorial approximation algorithm [1] that is far more effective in practice.

Our IPP algorithm contains three key elements: information gathering, robot movement cost, and adaptivity. It touches on several important research topics, which contain one or two, but not all three elements. If we focus on information gathering only and ignore robot movement cost, IPP becomes sensor placement, view planning, or ODT, which admits efficient solutions through, e.g., submodular optimization, in both nonadaptive [15] and adaptive settings [7,13]. If we account for movement cost, there are several nonadaptive algorithms with performance guarantee (e.g., [12,19]).

Although active localization [6] and simultaneous localization and mapping (SLAM) [5] 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) [14], 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 [17,21,16]. The underlying structure of IPP allows simpler and more efficient solutions.

3 Informative Path Planning

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 $x, x' \in X$. Next, H is a finite set of hypotheses, and $\rho(h)$ specifies the prior probability of hypothesis $h \in H$ occurring. 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 for each location 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. We say that an observation o and a hypothesis h are *consistent*, if $Z_x(h, o) = 1$. In this work, we focus mainly on the noiseless case. Finally, r is the robot's start location. To simplify the presentation, we assume $r \notin X$, because either r provides no useful sensing information or the robot has already visited r and acquired the information.

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 continues 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

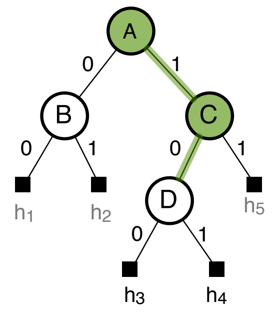

Fig. 1. A policy tree with sensing locations $\{A, B, C, D\}$, observations $\{0, 1\}$, hypotheses $\{h_1, h_2, \ldots, h_5\}$. With noiseless observations, every path in a policy tree from the root to a leaf uniquely identifies a hypothesis. Suppose that a robot follows the shaded path σ. Then a hypothesis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

hypothesis by taking observations at the chosen locations and minimizes the expected cost of traveling.

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

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

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

RAId is a recursive divide-and-conquer algorithm. In each recursive step, it 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 the robot receives an "informative" observation that reduces the probability of the current hypothesis set H by a half. RAId then recurses on the remaining hypotheses, until identifying the true hypothesis. A sketch of the algorithm is shown in Algorithm 1.

To generate such a traversal, RAId solves a group Steiner problem. A group Steiner problem is defined by two elements. One is an edge-weighted graph $G = (V, E, W_E)$. 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. The usual goal of a group Steiner problem is to find a minimum-edge-weight tree that covers a sub-collection of groups with total group-weight at least ν, for some given constant ν. In Algorithm 1, the procedure GROUPSTEINERTOUR$(V, E, W_E, \mathcal{V}, W_{\mathcal{V}}, \nu)$ computes a group Steiner *tour*, i.e., a cycle in a graph-theoretic sense, instead of a tree.

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($X,\ X \times X,\ d,\ \{X_h\}_{h \in H},\ \rho,\ \nu$),
           where $\tau = (x_0, x_1, \ldots, x_t)$ and $x_0 = x_t = r$.
7:         $(H, r) \leftarrow$ EXECUTEPLAN($\tau, H, r$).
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($\tau, H, r$)
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 $(H, r)$.
```

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

A key step in our construction is to define the groups. Let $H_{x,o} \subseteq H$ be the subset of hypotheses consistent with observation o at x. We define the *informative observation set* at x:

$$\Omega_x = \{\ o \mid p(H_{x,o}) \leq 0.5\}\,. \qquad (2)$$

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). As a result, upon receiving o, each recursive step of RAId prunes all inconsistent hypotheses $H \backslash H_{x,o}$ and reduces the probability of remaining consistent hypotheses by at least a half (see Lemma 1). In this sense, each observation $o \in \Omega_x$ is informative. 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

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

Now we define one group for each hypothesis $h \in H$:

$$X_h = \{x \in X \mid Z_x(h, o) = 1 \text{ for some } o \in \Omega_x\}, \qquad (3)$$

which contains all locations having informative observations consistent with h. The group-weight for X_h is simply $\rho(h)$.

Finally, we set the target $\nu = \min\big(0.5, 1 - \max_{h \in H} \rho(h)\big)$. RAId guarantees that by traversing such a group Steiner tour, the robot will prune inconsistent hypotheses with total probability at least ν. 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 achieve substantial pruning, as the remaining hypotheses have small total probability.

GROUPSTEINERTOUR first solves for a group Steiner tree T using a greedy approximation algorithm [1] and then applies Christofides' metric TSP approximation algorithm [3] to the vertex set of T in order to generate a tour. Both approximation algorithms rely critically on the the metric property of the edge weight d.

RAId is an online algorithm, which interleaves planning and plan execution. It plans a tour (Algorithm 1, line 6). The robot then traverses the locations on the tour (Algorithm 1, line 7). At each location, the robot prunes all hypotheses inconsistent with the received observation. It ends the traversal and returns to the start location, after receiving an observation in the informative observation space or exhausting the tour. RAId guarantees that the traversal either reduces the probability of consistent hypotheses by a half or identifies the true hypothesis (see Lemma 1).

4 Analysis

Our analysis consists of 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 1). In the second step, we exploit this result to bound the performance of RAId for IPP itself (Theorem 2).

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

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^*,

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

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.

Theorem 1. *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 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 1 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 1 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 [1] that computes a group Steiner tree T with a guaranteed approximation factor. It then applies Christofides' metric TSP algorithm [3] 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 2. *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. The greedy approximation in RAId computes an α-approximation T to the optimal group Steiner tree T^* [1], 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, $w(T) \leq \alpha\, w(T^*) \leq \alpha\, W^*$. Applying Christofides' metric TSP to the vertices of T produces a tour τ, which has weight $w(\tau) \leq 2w(T)$, using an argument similar to that in [3]. It then follows that $w(\tau) \leq 2\alpha W^*$. In other words, RAId obtains a 2α-approximation to the optimal group Steiner tour. Putting this together with Theorem 1 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. We further show in the next section that RAId works well in practice.

Table 1. Performance comparison. "Cost" is the average cost of a computed policy over all hypotheses. "Time" is the average total planning time, excluding the time for plan execution.

	$\|X\|$	$\|H\|$	$\|O\|$	**Cost**			**Time** (second)		
				IG	IG-Cost	RAId	IG	IG-Cost	RAId
2-Star (d=10, n=5)	37	32	2	25.3	32.9	19.0	0.03	0.03	0.13
2-Star (d=10, n=6)	70	64	2	27.9	22.3	21.0	0.10	0.12	0.34
2-Star (d=53, n=6)	70	64	2	102.1	62.0	64.0	0.13	0.75	1.07
2-Star (d=53, n=7)	135	128	2	102.4	127.4	69.4	0.40	5.58	1.22
2-Star (d=53, n=8)	264	256	2	100.9	257.7	68.0	1.39	3.35	4.96
Grasping	170	144	154	2822.9	839.9	690.1	2.39	4.14	6.43
UAV Search	128	64	2	97.2	142.7	74.7	0.45	2.54	7.23

5 Implementation and Experiments

It is probably unsurprising that the robot actually does not need to return to the start position, line 18–19) in each recursive step (Algorithm 1). This is mainly to simplify the analysis. For the experiments, we implemented a RAId variant without these two lines.

For comparison, we also implemented two greedy algorithms. The first one, *Information gain* (IG), is widely used in practice. Let Q denote that random variable that represents 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 to maximize the *expected* information gain in a greedy manner:

$$\max_{x' \in X} \sum_{h \in H} \sum_{o \in O} \Big(\mathbb{H}(Q) - \mathbb{H}(Q \mid x', o) \Big) p(o|x', h) p(h).$$

To account for robot movement cost, one simple way is to maximize information gain per unit movement cost (IG-Cost), again in a greedy manner:

$$\max_{x' \in X} \sum_{h \in H} \sum_{o \in O} \frac{\mathbb{H}(Q) - \mathbb{H}(Q \mid x', o)}{d(x, x')} p(o|x', h) p(h).$$

We implemented all three algorithms in the Clojure language and compared their performance in simulation (Table 1). For each test case, we ran the algorithms on every hypothesis in H and calculated the average policy cost weighted by the prior probabilities. Although cost is our main performance measure, we also recorded total *planning* time for completeness (Table 1). The running times were obtained on a computer server with an Intel Xeon 2.4GHz processor. Overall, RAId takes longer computation time than the two greedy algorithms, but produces much better policies. 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 test problems, are adequate for a range of online robot planning tasks.

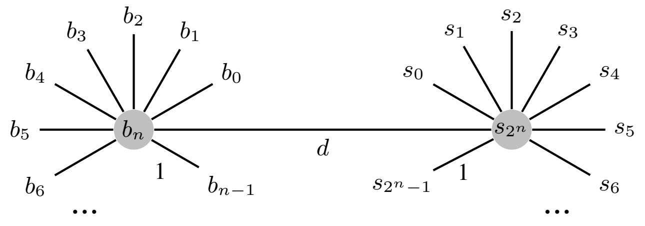

Fig. 2. The 2-star graph.

5.1 2-Star Graph

We start with a simple example to gain some understanding of the main 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. 2). 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 for 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 only serve the purpose of connecting the peripheral nodes. The robot is initially located at s_0.

At each node b_i in X, the robot receives observation 1 if the ith bit of a 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. The observations at b-nodes behave like binary search, while the observations at s-nodes behave like sequential search. Since the robot starts at s_0, 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.

RAId has the best or close to the lowest cost in all instances (Table 1). IG-Cost reasons about cost, but it is unable to decide optimally whether to jump to b-nodes or stay on s-nodes. When $d = 10$, IG-Cost transits to the b-nodes because it is not deterred by distance. However, it turns out to be profitable to jump to b-nodes only when $n = 6$. Hence, IG-Cost performs worse in $n = 5$. When we increase distance d to 53, IG-Cost is misled by the greedy local analysis and decides to stay at the s-nodes simply because it is cheaper to reach them. Its performance degrades quickly as the number of hypotheses increases. In fact, IG-Cost's regret, measured against the optimal solution, increases exponentially, as n increases. Interestingly, IG sometimes performs better than IG-Cost. This is, however, coincidence. By completely ignoring the movement cost, IG naturally moves the b-nodes, which provide more informative observations.

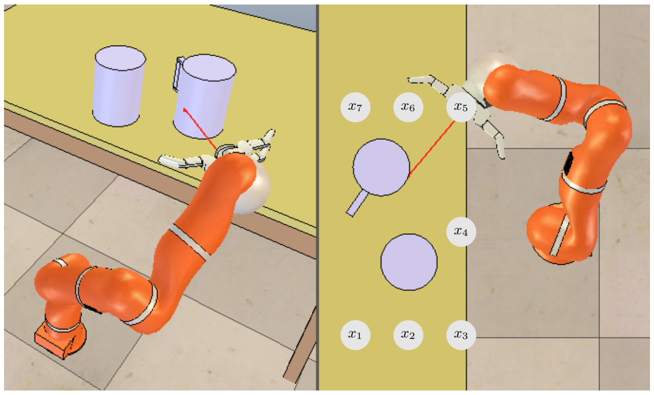

Fig. 3. Grasp the cup with a handle. The figure shows the side view and the top view of the same robot configuration with the robot hand on the right side of the table.

5.2 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. 3). 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 must face away from the 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. 3). 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 again has the lowest cost. 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-Cost does not perform as well here. 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. It turns out that 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.

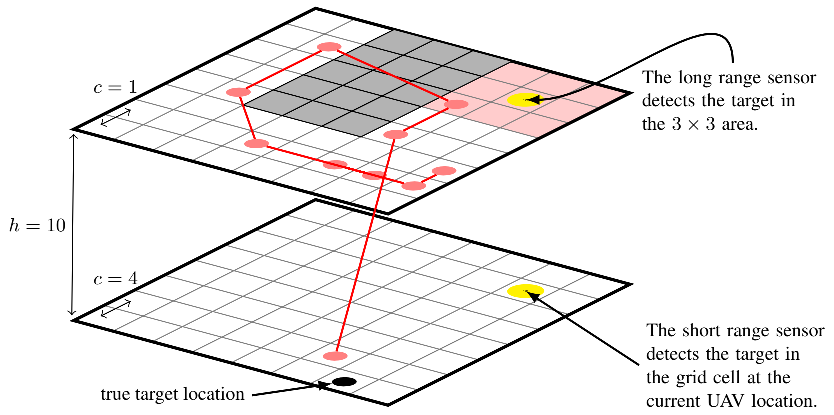
$c = 1$
$h = 10$
$c = 4$
The long range sensor detects the target in the 3×3 area.
The short range sensor detects the target in the grid cell at the current UAV location.
true target location

Fig. 4. 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 one sample path generated by RAId.

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.

5.3 UAV Search

A UAV searches for a stationary target in an area modeled as an 8×8 grid (Fig. 4) and must identify the grid cell that contains the target. Initially the target lies 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.

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. 4 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.

IG-Cost 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 IG-Cost, the UAV always stays on the low altitude and does not climb up.

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 5.3) with two different noise levels for the high-altitude sensor. The termination condition γ was set to 0.99. We evaluated multiple settings with different numbers

Table 2. The performance of Noisy RAId on the UAV Search task 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	110.1	104.6	106.1
0.05	131.9	135.5	131.3

of samples. For each setting, we run one trial for every hypothesis $h \in H$ and averaged performance statistics. The results, reported in Table 2, 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.

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 efficient in practice and provide 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, theoretical guarantees for RAId no longer hold when there are noisy observations. Our simple extension to RAId may benefit from borrowing ideas from algorithms for noisy Bayesian active learning such as [8].

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. 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.

References

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

2. Chakaravarthy, V., Pandit, V., Roy, S., Awasthi, P., Mohania, M.: Decision trees for entity identification: approximation algorithms and hardness results. In: Proc. ACM Symp. on Principles of Database Systems (2007)
3. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Tech. Rep. 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)
4. Dean, B., Goemans, M., Vondrdk, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. In: Proc. IEEE Symp. on Foundations of Computer Science. pp. 208–217 (2004)
5. Feder, H., Leonard, J., Smith, C.: Adaptive mobile robot navigation and mapping. Int. J. Robotics Research 18(7), 650–668 (1999)
6. Fox, D., Burgard, W., Thrun, S.: Active Markov localization for mobile robots. Robotics & Autonomous Systems 25(3), 195–207 (1998)
7. Golovin, D., Krause, A.: Adaptive submodularity: Theory and applications in active learning and stochastic optimization. J. Artificial Intelligence Research 42(1), 427–486 (2011)
8. Golovin, D., Krause, A., Ray, D.: Near-optimal bayesian active learning with noisy observations. In: NIPS. vol. 10, pp. 766–774 (2010)
9. Gupta, A., Nagarajan, V., Ravi, R.: Approximation algorithms for optimal decision trees and adaptive TSP problems. In: Proc. Int. Conf. on Automata, Languages & Programming, LNCS, vol. 6198, pp. 690–701. Springer (2010)
10. Hollinger, G., Mitra, U., Sukhatme, G.: Active classification: Theory and application to underwater inspection. In: Proc. Int. Symp. on Robotics Research. Springer (2011)
11. Hollinger, G., Englot, B., Hover, F.S., Mitra, U., Sukhatme, G.S.: Active planning for underwater inspection and the benefit of adaptivity. Int. J. Robotics Research 32(1), 3–18 (2013)
12. Hollinger, G., Singh, S., Djugash, J., Kehagias, A.: Efficient multi-robot search for a moving target. Int. J. Robotics Research 28(2), 201–219 (2009)
13. Javdani, S., Klingensmith, M., Bagnell, J., Pollard, N., Srinivasa, S.: Efficient touch based localization through submodularity. In: Proc. IEEE Int. Conf. on Robotics & Automation (2013)
14. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1–2), 99–134 (1998)
15. Krause, A., Guestrin, C.: Optimal value of information in graphical models. J. Artificial Intelligence Research 35, 557–591 (2009)
16. Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science and Systems (2008)
17. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proc. Int. Jnt. Conf. on Artificial Intelligence. pp. 477–484 (2003)
18. Platt Jr, R., Kaelbling, L., Lozano-Perez, T., Tedrake, R.: Simultaneous localization and grasping as a belief space control problem. In: Proc. Int. Symp. on Robotics Research (2011)
19. Singh, A., Krause, A., Guestrin, C., Kaiser, W.: Efficient informative sensing using multiple robots. J. Artificial Intelligence Research 34(2), 707–755 (2009)
20. Singh, A., Krause, A., Kaiser, W.: Nonmyopic adaptive informative path planning for multiple robots. In: Proc. Int. Jnt. Conf. on Artificial Intelligence (2009)
21. Smith, T., Simmons, R.: Point-based POMDP algorithms: Improved analysis and implementation. In: Proc. Uncertainty in Artificial Intelligence (2005)