Bayesian Active Edge Evaluation on Expensive Graphs

Sanjiban Choudhury[1], **Siddhartha Srinivasa**[1], **Sebastian Scherer**[2]
[1] School of Computer Science and Engineering, University of Washington, USA
[2] The Robotics Institute, Carnegie Mellon University, USA
sanjibac@cs.uw.edu, siddh@cs.uw.edu, basti@cs.cmu.edu

Abstract

We consider the problem of real-time motion planning that requires evaluating a minimal number of edges on a graph to quickly discover collision-free paths. Evaluating edges is expensive, both for robots with complex geometries like robot arms, and for robots sensing the world online like UAVs. Until now, this challenge has been addressed via *laziness* i.e. deferring edge evaluation until absolutely necessary, with the hope that edges turn out to be valid. However, all edges are not alike in value - some have a lot of potentially good paths flowing through them, and some others encode the likelihood of neighbouring edges being valid. This leads to our key insight - instead of passive laziness, we can *actively* choose edges that reduce the uncertainty about the validity of paths. We show that this is equivalent to the Bayesian active learning paradigm of *decision region determination (DRD)*. However, the DRD problem is not only combinatorially hard, but also requires explicit enumeration of all possible worlds. We propose a novel framework that combines two DRD algorithms, DIRECT and BISECT, to overcome both issues. We show that our approach outperforms several state-of-the-art algorithms on a spectrum of planning problems for mobile robots, manipulators and autonomous helicopters.

1 Introduction

A widely-used approach for solving robot motion-planning problems is the construction of graphs, where vertices represent robot configurations and edges represent potentially valid movements of the robot between these configurations. We assume the following setting where we have an *explicit graph* (also called a roadmap [Kavraki *et al.*, 1996]) that is fixed across planning cycles, but the underlying validity of each edge depends on the configuration of obstacles. Consider the robot arm planning scenario in Fig. 1(a) where the roadmap remains the same, but the location of objects and the table may vary. Also consider receding horizon planning for a helicopter on a state lattice as shown in Fig. 1(b) where the lattice defined in the robot frame of refence remains the same, but obstacle configurations in the world vary.

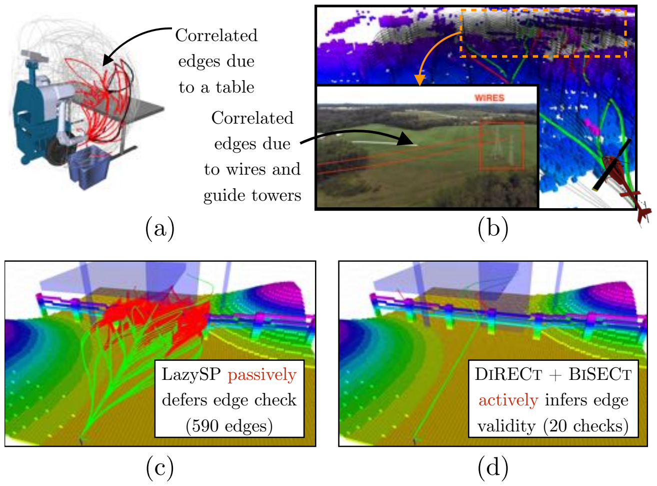
Correlated edges due to a table
Correlated edges due to wires and guide towers
WIRES
LazySP passively defers edge check (590 edges)
DIRECT + BISECT actively infers edge validity (20 checks)
(a)
(b)
(c)
(d)

Figure 1: Real world planning problems where edges are correlated such as (a) manipulating objects on a table (courtesy [Dellin and Srinivasa, 2016]) or (b) helicopter avoiding power-lines. In the helicopter application, real-time has a budget of 100 ms (based on speed and sensing range). Each edge requires ≈ 1.45 ms of evaluation time. (c) LazySP [Dellin and Srinivasa, 2016] finds the optimal path, but requires 590 checks (856ms) (d) Our approach finds a good enough *feasible* path in 20 checks (29ms)

The main computational bottleneck is the process of *edge-evaluation* which serves as the bottleneck to real-time performance. For example, in robot arm planning [Dellin *et al.*, 2016], evaluation requires expensive geometric intersection computations. In autonomous helicopter planning [Choudhury *et al.*, 2014], evaluation requires expensive reachability volume verification of the closed loop system. State-of-the-art planning algorithms [Dellin and Srinivasa, 2016] deal with expensive evaluation by resorting to *laziness* - they first compute a set of unevaluated paths quickly, and then evaluate them sequentially to find a valid optimal path.

We ask the question - Can we do better than laziness if the goal is to simply find a feasible path in the set as quickly as possible? We exploit a fundamental characteristic of the planning problem - *edges in a graph are implicitly correlated* as shown in Fig. 1. Evaluating certain edges provides valuable

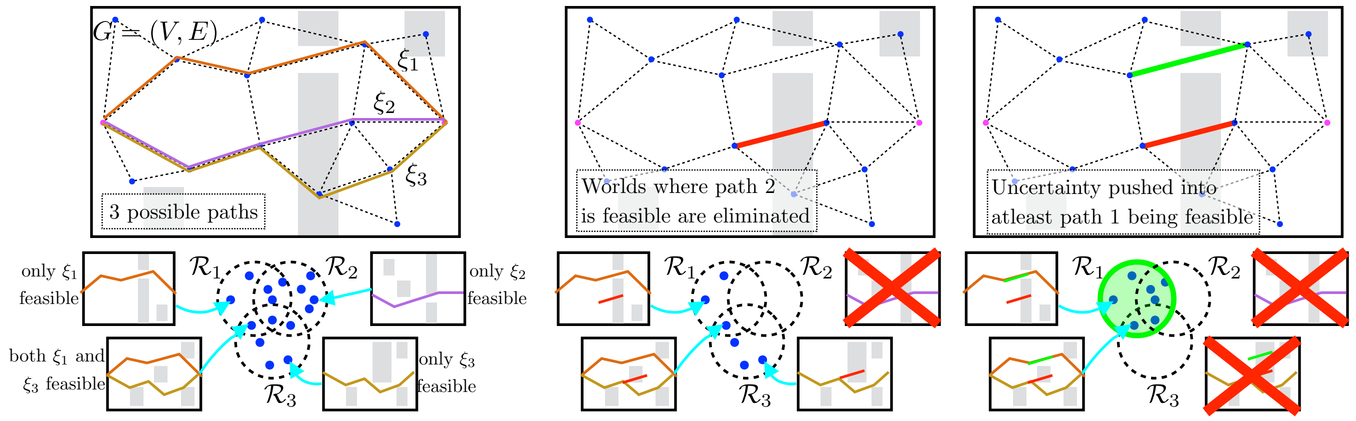

Figure 2: Equivalence between the feasible path identification problem and the decision region determination problem. A plausible world is equivalent to hypothesis (as shown by the blue dots in the lower row). A path ξ_i is equivalent to a region $\mathcal{R}_i$ over valid hypotheses where the path is feasible. A collision check is equivalent to a test whose outcome is valid (green) or invalid (red). Tests eliminate hypotheses and the algorithm terminates when uncertainty is pushed into a region ($\mathcal{R}_1$) and the corresponding path (ξ_1) is determined to be valid.

information about the feasibility likelihood of other edges. We wish to compute such a policy that judiciously chooses edges to evaluate by exploiting such correlations.

We show that this problem is equivalent to the Bayesian active learning problem of *decision region determination (DRD)* [Javdani *et al.*, 2014; Chen *et al.*, 2015] - given a set of tests (edges), hypotheses (worlds), and regions (potential paths), the objective is to select a sequence of tests that drive uncertainty into a single decision region. The DRD problem has one key distinction from the general active learning problem [Dasgupta, 2004] - we only need to know enough about the world to ascertain if a path is feasible. To solve the DRD problem in our context, we need to address two issues:

(a) Enumeration of all possible worlds.

(b) Solving the DRD problem in general is NP-hard [Javdani *et al.*, 2014].

Fortunately, [Chen *et al.*, 2015] provide an algorithm, DIRECT, to address (b) by maximizing an objective function that satisfies *adaptive submodularity* [Golovin and Krause, 2011] - a natural diminishing returns property that endows greedy policies with near-optimality guarantees. However, DIRECT requires (a) to be solved. Explicitly enumerating all possible worlds is impractical even as an offline operation - since each world is a configuration of edges, a graph with E edges can induce $\mathcal{O}\left(2^E\right)$ possible worlds.

[Choudhury *et al.*, 2017a] addresses (a) by examining the DRD problem when edges are *independent*. They propose an efficient near-optimal algorithm BISECT which reduces the computation from $\mathcal{O}\left(2^E\right)$ to $\mathcal{O}\left(E\right)$. However, this independence assumption is too strong for certain environments (such as those in Fig 1) thus leading to excessive edge evaluations.

Our key idea is to combine the two approaches. We sample a finite database of worlds and apply DIRECT offline on this database to compute a decision tree of edges to evaluate. At test time we execute the tree. When we reach a leaf node, we have either solved the problem or the world lies outside of the database. In such cases we execute BISECT, which implicitly reasons about all possible worlds, and accept the performance loss due to the independence assumption. By controlling the bias term in BISECT, we can also choose to not over-fit to the training database. We make the following contributions:

1. We show an equivalence between the edge evaluation problem and the decision region determination problem.
2. We propose a framework to combine two DRD algorithms, DIRECT and BISECT, that near-optimally solves the decision region problem, overcomes issues pertaining to finite databases and can be executed efficiently online.
3. We evaluate our approach on a spectrum of planning problems including a manipulator and a helicopter.

2 Problem Formulation

We now describe the edge evaluation problem, showing the equivalence to the DRD problem along the way. Let $G = (V, E)$ be an explicit graph that consists of a set of vertices V and edges E. Given a pair of start and goal vertices, $(v_s, v_g) \in V$, a search algorithm computes a path $\xi \subseteq E$ - a connected sequence of valid edges. The search is performed on an underlying world ϕ which corresponds to a specific validity status of edges. Our belief about the robot's environment defines a distribution $P(\phi)$. We address applications where the cost of evaluating an edge is $c(e)$. We make a simplification to the problem - instead of searching G online for a path, we frame the problem as identifying a valid path from a library of 'good' candidate paths $\Xi = (\xi_1, \xi_2, \ldots, \xi_m)$ [1].

Let $\mathcal{H} = \{h_1, \ldots, h_n\}$ be a set of "hypotheses", each of which is analogous to a world. Hence, we have a prior distribution $P(h)$ on this set corresponding to $P(\phi)$. A "test" $t \in \mathcal{T}$ is performed by querying a corresponding edge $e \in E$ for evaluation, which returns a binary outcome $x \in \{0, 1\}$ denoting if an edge is valid or not. Thus each hypothesis can be

[1]For example a library of k-shortest paths

considered a function, $h : \mathcal{T} \to \{0, 1\}$, mapping tests to corresponding outcomes. The cost of performing a test is $c(t)$. A path $\xi_i \in \Xi$ corresponds to a set of worlds on which that path is valid. Hence each path $\xi_i \in \Xi$ corresponds to a "decision region" $\mathcal{R}_i \subseteq \mathcal{H}$ over the space of hypotheses. Let $\{\mathcal{R}_i\}_{i=1}^{m}$ be the set of "decision regions" corresponding to Ξ.

For a set of tests $\mathcal{A} \subseteq \mathcal{T}$ that are performed, let the observed outcome vector be denoted by $\mathbf{x}_{\mathcal{A}}$. Let the version space $\mathcal{H}(\mathbf{x}_{\mathcal{A}})$ be the set of hypotheses consistent with outcome vector $\mathbf{x}_{\mathcal{A}}$, i.e. $\mathcal{H}(\mathbf{x}_{\mathcal{A}}) = \{h \in \mathcal{H} \mid \forall t \in \mathcal{A}, h(t) = \mathbf{x}_{\mathcal{A}}(t)\}$.

We define a policy π as a mapping from the current outcome vector $\mathbf{x}_{\mathcal{A}}$ to the next test to select. A policy terminates when at least one region is valid, or all regions are invalid. Let h be the underlying world on which it is evaluated. Denote the outcome vector of a policy π as $\mathbf{x}_{\mathcal{A}}(\pi, h)$. The expected cost of a policy π is $c(\pi) = \mathbb{E}_h [c(\mathbf{x}_{\mathcal{A}}(\pi, h)]$ where $c(\mathbf{x}_{\mathcal{A}})$ is the cost of all tests $t \in \mathcal{A}$. The objective is to compute a policy π^* with minimum cost such that at least one region is valid,

$$\pi^* \in \arg\min_{\pi} c(\pi) \text{ s.t } \forall h, \exists \mathcal{R}_d \ : \ P(\mathcal{R}_d \mid \mathcal{H}(\mathbf{x}_{\mathcal{A}})) = 1 \tag{1}$$

We summarize the equivalence which is illustrated in Fig. 2. A world is equivalent to a hypothesis. We do not know the true hypothesis, but have a belief over them. Evaluating an edge is equivalent to performing a test. Based on the outcome, we can update this belief. Each path in the library is equivalent to a region. The set of worlds for which a path is valid correspond to the set of hypothesis belonging to the region. The goal is to drive uncertainty to at least one path, i.e. one region.

3 Related Work

We examine the problem class of expensive edge evaluation in motion planning which has inspired a variety of 'lazy' approaches. The LazyPRM algorithm [Bohlin and Kavraki, 2000] only evaluates edges on the shortest path while FuzzyPRM [Nielsen and Kavraki, 2000] evaluates paths that minimize probability of collision. The Lazy Weighted A* (LWA*) algorithm [Cohen *et al.*, 2015] delays edge evaluation in A* search and is reflected in similar techniques for randomized search [Gammell *et al.*, 2015; Choudhury *et al.*, 2016a; Hauser, 2015]. We base our work on the LazyShortestPath (LazySP) framework [Dellin and Srinivasa, 2016] which examines the problem of which edges to evaluate on the shortest path. The Anytime Edge Evaluation (AEE*) framework [Narayanan and Likhachev, 2017] also deals with a similar problem however it makes an independent edge assumption.

Efficient collision checking has its own history in the context of motion planning. [Bialkowski *et al.*, 2016] creates a data-structure to store the distances to obstacles from queries to speed-up future queries. However, this assumes access to an interpretable distance value, and benefits are only asymptotic in nature. Other approaches model belief over the configuration space to speed-up collision checking [Huh and Lee, 2016; Choudhury *et al.*, 2016b], sample vertices in promising regions [Bialkowski *et al.*, 2013] or grow the search

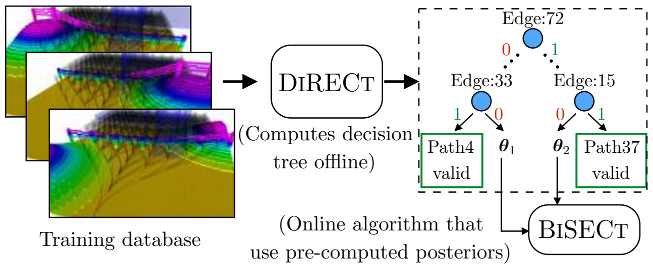

Figure 3: The overall approach framework. *Offline:* DiRECT is executed on a database to create a decision tree. *Online:* Tree is executed until leaf node is reached. If the problem is unsolved, BiSECT is invoked with bias term $\boldsymbol{\theta}_i$

tree to explore the configuration space [Hsu *et al.*, 1997; Burns and Brock, 2005; Lacevic *et al.*, 2016]. However, these approaches make geometric assumptions, requires creating a data-structure online which maybe expensive and do not exploit correlations from prior data. Our approach, in comparison, is completely general and applicable to any domain.

We draw a novel connection between motion planning and optimal test selection which has a wide-spread application in medical diagnosis [Kononenko, 2001] and experiment design [Chaloner and Verdinelli, 1995]. Optimizing the ideal metric, decision theoretic value of information [Howard, 1966], is known to be NP$^{\text{PP}}$ complete [Krause and Guestrin, 2009]. For hypothesis identification (known as the Optimal Decision Tree (ODT) problem), Generalized Binary Search (GBS) [Dasgupta, 2004] provides a near-optimal policy. For disjoint region identification (known as the Equivalence Class Determination (ECD) problem), EC^2 [Golovin *et al.*, 2010] provides a near-optimal policy. When regions overlap (known as the Decision Region Determination (DRD) problem), HEC [Javdani *et al.*, 2014] provides a near-optimal policy. The DiRECT algorithm [Chen *et al.*, 2015], a computationally more efficient alternative to HEC, forms the basis of our approach. We also employ the BiSECT algorithm [Choudhury *et al.*, 2017a], which solves the DRD problem under edge independence assumptions.

4 Approach

4.1 Overview

Fig. 3 shows an overview of our approach. We sample a finite database of worlds to create a training dataset. We employ a greedy yet near-optimal algorithm DiRECT [Chen *et al.*, 2015] to solve the DRD problem. DiRECT chooses decisions to prune inconsistent worlds from the database until it can ascertain if a path is valid. The decisions of DiRECT can be compactly stored in the form of a decision tree which is computed offline. At test time, the tree is executed until the leaf node is reached. At this point, either the problem is solved or the fraction of consistent worlds drops below a threshold η, i.e. it is likely that the test world is not in the database. In the latter case, we invoke another DRD algorithm, BiSECT. BiSECT implicitly reasons about the exhaustive set of $O(2^E)$ worlds and does this efficiently by assuming edges are independent. BiSECT is invoked with a bias vector of edge like-

Algorithm 1: DIRECT $(\mathcal{H}_{\text{act}}, \mathbf{R}, \mathbf{X}, \mathbf{c})$

```
1 $f_{\text{old}} = \texttt{DRD}(\mathcal{H}_{\text{act}}, \mathbf{R})$ ;              ▷ Compute old $f_{\text{DRD}}$
2 for $t \in \mathcal{T}$ do
3     $\Delta(t) \leftarrow 0$;
4     for $x_t \in \{0,1\}$ do
5         $\mathcal{H}_{\text{cond}} \leftarrow \{h \in \mathcal{H}_{\text{act}} \mid \mathbf{X}(h,t) = x_t\}$ ;   ▷ Prune
6         $p \leftarrow \frac{|\mathcal{H}_{\text{cond}}|}{|\mathcal{H}_{\text{act}}|}$ ;   ▷ Probability of outcome
7         $\Delta(t) \leftarrow \Delta(t) + p\ (\texttt{DRD}(\mathcal{H}_{\text{cond}}, \mathbf{R}) - f_{\text{old}})$;
8     $\Delta(t) \leftarrow \frac{\Delta(t)}{\mathbf{c}(t)}$;
9 return $\arg\max_{t \in \mathcal{T}} \Delta(t)$;
```

Algorithm 2: DRD $(\mathcal{H}', \mathbf{R})$

```
1 $v \leftarrow 1$;
2 for $i \in \{1, \ldots, m\}$ do
3     $v \leftarrow v \left(\frac{\texttt{WeightEC}(\mathcal{H}', \mathbf{R}, i)}{\texttt{WeightEC}(\mathcal{H}, \mathbf{R}, i)}\right)$ ; ▷ Weight of each ECD
4 return $1 - v$;
```

lihoods $\boldsymbol{\theta}$ which can be chosen to prevent over-fitting to the training database. The combined behaviour of the framework is as follows - the tree makes a set of evaluations to quickly collapse the posterior on to a set of candidate paths, while BISECT completes the episode being guided by the obtained posterior. We now describe each component.

4.2 The Decision Region Edge Cutting Algorithm (DIRECT)

In order to solve the DRD problem in (1), we adopt the framework of *Decision Region Edge Cutting (*DIRECT*)* [Chen *et al.*, 2015]. The intuition behind the method is as follows - as tests are performed, hypotheses inconsistent with test outcomes are pruned away. Hence, tests should be incentivized to push the probability mass over hypotheses into a region as fast as possible. [Chen *et al.*, 2015] derive a surrogate objective function that not only provides such an incentive, but also exhibits the property of adaptive submodularity [Golovin and Krause, 2011] - greedily maximizing such an objective results in a near-optimal policy.

DIRECT uses a key result from the EC^2 algorithm [Golovin *et al.*, 2010] which solves the *Equivalence Class Determination* (ECD) problem - a special case of the DRD problem (1) when *regions are disjoint*. The EC^2 algorithm defines a graph $\mathcal{G}_{\text{EC}} = (\mathcal{V}_{\text{EC}}, \mathcal{E}_{\text{EC}})$ where the nodes are hypotheses and edges are between hypotheses in different decision regions $\mathcal{E}_{\text{EC}} = \cup_{i \neq j} \{\{h, h'\} \mid h \in \mathcal{R}_i, h' \in \mathcal{R}_j\}$. The weight of an edge is defined as $w(\{h, h'\}) = P(h)P(h')$. An edge is said to be 'cut' by an observation if either hypothesis is inconsistent with the observation. The aim is to cut all edges, i.e. to drive total weight to 0. EC^2 efficiently computes the total weight $w_{\text{EC}}(\{\mathcal{R}_i\}) = \frac{1}{2}\left((\sum_i P(\mathcal{R}_i))^2 - \sum_i P(\mathcal{R}_i)^2\right)$. It then defines an objective function $f_{\text{EC}}(\mathbf{x}_{\mathcal{A}})$ that measures the ratio of the weight orig-

Algorithm 3: WeightEC $(\mathcal{H}', \mathbf{R}, i)$

```
1 $a \leftarrow \sum_{h \in \mathcal{H}'} \mathbf{R}(h, i)$ ;      ▷ Number of hyp in region
2 $b \leftarrow |\mathcal{H}| - a$ ;                        ▷ Remaining hyp
3 return $w^i_{\text{EC}} = \frac{1}{2|\mathcal{H}|^2}\left((a+b)^2 - a^2 - b\right)$
```

inally to the weight of pruned regions $\mathcal{R}_i \cap \mathcal{H}(\mathbf{x}_{\mathcal{A}})$, i.e.

$$f_{\text{EC}}(\mathbf{x}_{\mathcal{A}}) = 1 - \frac{w_{\text{EC}}(\{\mathcal{R}_i\} \cap \mathcal{H}(\mathbf{x}_{\mathcal{A}}))}{w_{\text{EC}}(\{\mathcal{R}_i\})} \tag{2}$$

EC^2 uses the fact that $f_{\text{EC}}(\mathbf{x}_{\mathcal{A}})$ is *adaptive submodular* ([Golovin and Krause, 2011]) to define a greedy algorithm. Let the expected marginal gain of a test be $\Delta_{f_{\text{EC}}}(t \mid x) = \mathbb{E}_{x_t}\left[f_{\text{EC}}(\mathbf{x}_{\mathcal{A} \cup \{t\}}) - f_{\text{EC}}(\mathbf{x}_{\mathcal{A}}) \mid \mathbf{x}_{\mathcal{A}}\right]$. It greedily selects a test $t^* \in \arg\max_t \frac{\Delta_{f_{\text{EC}}}(t \mid \mathbf{x}_{\mathcal{A}})}{c(t)}$.

We now return to the general DRD problem where regions are not disjoint. DIRECT reduces the DRD problem with m regions to m instances of the ECD problem. Each ECD problem is a 'one region versus all'. ECD problem i is defined over the following disjoint regions: the first region is $\mathcal{R}_i$ and the remaining regions are singletons containing only one hypothesis $h \notin \mathcal{R}_i$. The EC^2 objective corresponding to this problem is $f^r_{\text{EC}}(\mathbf{x}_{\mathcal{A}})$. The key idea is that *solving any one ECD problem solves the DRD problem.* The DIRECT algorithm then combines them in a *Noisy-OR* formulation by defining the following combined objective

$$f_{\text{DRD}}(\mathbf{x}_{\mathcal{A}}) = 1 - \prod_{r=1}^{m}(1 - f^r_{\text{EC}}(\mathbf{x}_{\mathcal{A}})) \tag{3}$$

DIRECT uses the fact that $f_{\text{DRD}}(\mathbf{x}_{\mathcal{A}})$ is also *adaptive submodular* to greedily select a test $t^* \in \arg\max_t \frac{\Delta_{f_{\text{DRD}}}(t \mid \mathbf{x}_{\mathcal{A}})}{c(t)}$. For details on the theoretical guarantees and proofs, we refer the reader to [Chen *et al.*, 2015].

To aid in implementation, we provide a pseudo-code for DIRECT in Alg. 1, 2 and 3. The pseudo-code is derived by expanding and simplifying $\Delta_{f_{\text{DRD}}}(t \mid \mathbf{x}_{\mathcal{A}})$ which we omit for brevity. Alg. 1 describes the main subroutine of DIRECT algorithm that greedily selects a test to execute based on the current state. $\mathcal{H}_{\text{act}}$ is the set of active hypotheses which have remained consistent so far with test outcomes. $\mathbf{R} \in \mathbb{R}^{n \times m}$ is a binary membership matrix where $\mathbf{R}(h, r) = 1$ if $h \in \mathcal{R}_r$. $\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{T}|}$ is the test outcome matrix where $\mathbf{X}(h, t) = h(t)$. $\mathbf{c} \in \mathbb{R}^{|\mathcal{T}| \times 1}$ is a vector of test costs. Line 1 computes the current value of f_{DRD} by invoking the DRD function. Line 2 iterates over each test. Line 4 iterates over the two possible outcomes. Line 5 computes $\mathcal{H}_{\text{cond}}$, the set of consistent hypotheses conditioned on the test outcome. Line 6 computes the new f_{DRD} value by invoking the DRD function and computes the gain. Line 9 returns the test with the highest gain.

Alg. 2 describes the f_{DRD} computation for $\mathcal{H}'$. Line 2 iterates over every region (i.e. every ECD problem) and multiplies the weight ratio of individual ECD problem. Line 3 returns the value.

Alg. 3 calculates the weight of the i^{th} ECD problem. Line 1 computes the number of hypothesis in the region $\mathcal{R}_i$. Line 2 computes the number of hypothesis outside the region. Line 3 computes w^i_{EC}. The computational complexity of Alg. 1 is $\mathcal{O}(|\mathcal{T}|\, mn)$. Speedups can be obtained by lazy gain evaluation and graph coloring to reduce the number of ECD problems [Chen *et al.*, 2015].

Let's examine the the situation when all the uncertainty is pushed into a region $\mathcal{R}_i$, i.e. when the problem is solved. In Alg. 3, the value of $b = 0$. Hence the algorithm returns $w^i_{\text{EC}} = 0$. Alg.2 assigns $v = 0$ as result (irrespective of other weights) and returns a value of 1. Since 1 is the maximum value of f_{DRD}, this corresponds to the maximum possible gain $\Delta(t)$ in line 7 of Alg. 1.

4.3 Offline Decision Tree using DIRECT

We presented an algorithm DIRECT for selecting a test in Alg. 1. One approach is to run this algorithm at test time to make decisions about which edge to select for evaluation. However, the algorithm needs access to the entire training database $\mathbf{R}$ and $\mathbf{X}$ at runtime. This can be expensive for storage and computational reasons. We need an approach that has minimal edge selection time.

Algorithm 4: `CreateDecisionTree` $(\Gamma, \mathcal{H}_{\text{act}}, \mathbf{R}, \mathbf{X}, \mathbf{c})$

```
1 $t \leftarrow$ DIRECT $(\mathcal{H}_{\text{act}}, \mathbf{R}, \mathbf{X}, \mathbf{c})$ ;    ▷ Invoke algorithm
2 $\Gamma$.test $\leftarrow t$ ;    ▷ Assign test to tree node
3 $\mathcal{H}^1_{\text{act}} \leftarrow \{h \in \mathcal{H}_{\text{act}} \mid \mathbf{X}(h,t) = 0\}$ ; ▷ Prune false hyp
4 CreateDecisionTree $(\Gamma.\text{left}, \mathcal{H}^1_{\text{act}}, \mathbf{R}, \mathbf{X}, \mathbf{c})$;
5 $\mathcal{H}^2_{\text{act}} \leftarrow \{h \in \mathcal{H}_{\text{act}} \mid \mathbf{X}(h,t) = 0\}$ ;  ▷ Prune true hyp
6 CreateDecisionTree $(\Gamma.\text{right}, \mathcal{H}^2_{\text{act}}, \mathbf{R}, \mathbf{X}, \mathbf{c})$;
```

We circumvent this problem by computing a decision tree offline using DIRECT. The process is illustrated in Fig. 3. The nodes of the tree encode which edge to evaluate. The tree branches represent the outcome of the evaluation. The tree is created by recursively calling a subroutine described in Alg. 4. Line 1 invokes DIRECT to get a test. Line 2 stores this test in the current node of the tree. To evaluate the left branch, line 3 assumes the test outcome is 0 and prunes inconsistent hypothesis. Line 4 calls the subroutine with the left node. The right branch is also similarly evaluated in lines 5 and 6 assuming the test outcome is 1. Note that the number of nodes of the tree is bounded by $|\mathcal{H}|$ which is much smaller than $2^{|\mathcal{T}|}$.

4.4 Executing BISECT from the Leaf Node

If we reach the leaf node of the tree and the problem is still unsolved, we need to execute an online algorithm that can run to completion by reasoning over the exhaustive set of worlds. We use the *Bernoulli Subregion Edge Cutting (*BISECT*)* algorithm [Choudhury *et al.*, 2017a] as our online algorithm. BISECT addresses the DRD problem under the assumption that test outcomes are independent Bernoulli random variables. It leverages this assumption to reduce computational complexity from $\mathcal{O}(2^E)$ to $\mathcal{O}(E)$ and has a closed form expression which we omit here for brevity.

BISECT needs as input a bias vector which corresponds to the independent likelihood of an edge being free. Since DIRECT has made a set of decisions to collapse the posterior, albeit on a finite database, we wish to use this to inform BISECT. We do this by growing the DIRECT decision tree only until the version space $\mathcal{H}_\eta$ drops below a fraction η of consistent worlds, i.e. $|\mathcal{H}_\eta| \leq \eta\, |\mathcal{H}|$. This is then used to create a bias vector $\boldsymbol{\theta}$ with a mixture term to ensure non-zero support for all plausible worlds. The bias term for a test t is

$$\boldsymbol{\theta}(t) = \alpha\, \frac{1}{|\mathcal{H}_\eta|} \sum_{h \in \mathcal{H}_\eta} \mathbf{X}(h,t) + (1-\alpha)\, 0.5 \qquad (4)$$

We can control overfitting to training data by tuning α. Using BISECT with a bias of 0.5 is a principled approach when one does not have any prior knowledge about the world.

5 Experiments

5.1 Dataset Construction

We evaluated our approach on a collection of datasets from insightful 2D problems to more realistic high dimension problems as encountered by a helicopter or a robot arm. The robot dynamics information is used to create an explicit graph $G = (V, E)$. A dataset of n worlds is sampled from a designed generative model. Each edge is evaluated on each world to create a test outcome matrix $\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{T}|}$. A library of paths is created along with a binary membership matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ encoding the validity of a path on a world. 10% of the data is used for test, remainder for training. Typical values used are $n : 1000$, $m : 500$. Details about the dataset generation are described in [Choudhury *et al.*, 2017b]. Opensource code and details can be found here: https://github.com/sanjibac/matlab_learning_collision_checking

5.2 Baseline Algorithms

Our primary baseline is BISECT [Choudhury *et al.*, 2017a] which treats each edge as independent Bernoulli random variables. We additionally use information theoretic baselines from [Choudhury *et al.*, 2017a] which were competitive with BISECT, i.e the MAXPROBREG version of MAXTALLY, SETCOVER and MVOI.

To compare with a planning baseline, we use the LAZYSP algorithm [Dellin and Srinivasa, 2016] which operates on the original graph G. LAZYSPSET is provably optimal with respect to edge evaluations when the planning algorithm is given no prior information. We also introduce LAZYSPSET which is restricted to the library of paths Ξ.

5.3 Summary of Results

Table 1 shows the normalized evaluation cost of an algorithm, i.e $\frac{\text{cost(alg)}}{\text{cost(DIRECT +BISECT)}} - 1$. The two numbers are lower and upper 95% confidence intervals. The best performance on each dataset is highlighted. We present a set of observations to interpret these results.

O 1. DIRECT +BISECT *has a consistently competitive performance across all datasets.*

Table 1 shows on 16 datasets, DIRECT is at par with the best — on 8 of those it is exclusively the best.

	LAZYSP	LAZYSPSET	MAXTALLY	SETCOVER	MVOI	BISECT	DIRECT + BISECT
2D Geometric Planning: Variation across environments							
Forest	$(10.90, 18.48)$	$(1.84, 3.02)$	$(0.17, 0.40)$	$(0.14, 0.51)$	$(0.30, 0.55)$	$(0.014, 0.20)$	$(0.00, 0.00)$
OneWall	$(7.47, 16.01)$	$(0.30, 0.71)$	$(0.00, 0.30)$	$(0.08, 0.34)$	$(0.09, 0.36)$	$(-0.06, 0.22)$	$(0.00, 0.00)$
TwoWall	$(21.54, 26.68)$	$(0.00, 0.21)$	$(0.20, 0.92)$	$(0.12, 0.58)$	$(0.31, 0.56)$	$(0.00, 0.53)$	$(0.00, 0.00)$
MovingWall	$(1.33, 3.01)$	$(1.00, 1.54)$	$(0.43, 1.17)$	$(0.35, 0.91)$	$(-0.03, 0.57)$	$(0.11, 0.92)$	$(0.00, 0.00)$
Baffle	$(7.86, 11.26)$	$(2.30, 3.83)$	$(0.33, 1.06)$	$(0.36, 0.74)$	$(0.26, 0.89)$	$(0.11, 0.55)$	$(0.00, 0.00)$
Maze	$(14.39, 19.66)$	$(1.16, 1.81)$	$(0.12, 0.34)$	$(0.00, 0.17)$	$(0.41, 0.87)$	$(0.44, 0.76)$	$(0.00, 0.00)$
Bugtrap	$(7.40, 8.57)$	$(2.74, 3.53)$	$(0.51, 0.84)$	$(-0.12, 0.54)$	$(-0.12, 0.53)$	$(0.43, 0.91)$	$(0.00, 0.00)$
2D Geometric Planning: Heterogeneous datasets by mixing multiple environments							
Mixture	$(1.82, 2.55)$	$(0.44, 0.95)$	$(0.0, 0.49)$	$(0.04, 0.47)$	$(-0.11, 0.23)$	$(-0.07, 0.24)$	$(0.00, 0.00)$
SE(2) Nonholonomic Path Planning: Variation across environments							
OneWall	$(2.22, 4.18)$	$(0.15, 0.57)$	$(0.16, 0.48)$	$(-0.11, 0.07)$	$(0.00, 0.28)$	$(-0.07, 0.12)$	$(0.00, 0.00)$
MovingWall	$(-0.14, 0.23)$	$(-0.14, 0.15)$	$(0.24, 0.49)$	$(0.13, 0.41)$	$(0.00, 0.36)$	$(0.10, 0.54)$	$(0.00, 0.00)$
Baffle	$(7.74, 10.48)$	$(2.88, 4.81)$	$(1.86, 3.21)$	$(1.35, 2.32)$	$(0.70, 1.47)$	$(1.14, 1.70)$	$(0.00, 0.00)$
Bugtrap	$(3.75, 6.51)$	$(2.27, 4.69)$	$(0.22, 0.52)$	$(0.05, 0.43)$	$(0.26, 0.55)$	$(0.12, 0.44)$	$(0.00, 0.00)$
Autonomous Helicopter Path Planning: Variation across environments							
Wires	$(17.42, 75.85)$	$(1.15, 3.08)$	$(0.55, 0.96)$	$(0.00, 0.25)$	$(-0.08, 0.08)$	$(0.08, 0.23)$	$(0.00, 0.00)$
Canyon	$(0.73, 1.27)$	$(1.41, 2.00)$	$(0.15, 0.52)$	$(0.07, 0.40)$	$(0.43, 0.72)$	$(0.06, 0.47)$	$(0.00, 0.00)$
7D Arm Planning: Variation across environments							
Clutter	$(0.49, 1.08)$	$(0.09, 0.57)$	$(-0.04, 0.05)$	$(0.00, 0.13)$	$(0.10, 0.32)$	$(0.00, 0.10)$	$(0.00, 0.00)$
Table+Clutter	$(0.94, 1.84)$	$(-0.22, 0.17)$	$(0.06, 0.51)$	$(0.05, 0.27)$	$(0.11, 0.46)$	$(0.06, 0.36)$	$(0.00, 0.00)$

Table 1: Normalized cost (with respect to our approach) of different algorithms on different datasets (lower and upper bounds of 95% C.I.)

O 2. DIRECT *is effective on environments with spatial correlation.*

Fig. 4 shows a test point from in the Maze dataset where there are 5 hallways with one interconnecting passage. DIRECT is able to locate this passage with a few checks and has better performance than BISECT which assumes independence between edges. LAZYSPSET suffers as it goes through the shorter paths first. The myopic heuristic MVOI greedily evaluates the most probable path which fails for this environment where a large number of paths are equiprobable.

O 3. DIRECT +BISECT *improves in performance with more data.*

Fig. 5(a) shows that both mean and variance reduce as the size of the dataset is increased. This is not only due to DIRECT having better realizability, but also due to BISECT having a more accurate bias term.

O 4. BISECT *is essential as a post-processing step.*

We defined an algorithm, DIRECTONLY that runs DIRECT to completion and randomly returns a path from the consistent set of paths, i.e. the a path DIRECT believes should be feasible. Fig. 5(b) shows the failure rate of DIRECTONLY with training size, i.e. the returned path being infeasible. The plot shows the failure does not go to zero. BISECT is essential to reason about the remaining paths and in which order to check edges to ascertain which path is free.

O 5. *Our approach is robust to both heterogeneous datasets as well as changes in train/test distribution*

We applied DIRECT +BISECT on different heterogeneous datasets by combining Forest, TwoWall and Maze. We see that even on this dataset, our approach outperforms some of the baselines. This is due to the fact that DIRECT is still able to disambiguate between the type of worlds and subsequently exploit correlation. However, we note that the variance of the results are also higher - hence we conclude no one method dominates significantly in this case.

To check the robustness of our approach to changes in the test dataset, we trained DIRECT +BISECT on the TwoWall distribution (which is highly structured). We then created a test dataset by blending in this distribution with an unbiased bernoulli distibution where every edge can be 0/1 with probability 0.5. The mixing fraction is ρ. We compare against LAZYSPSET which does not use the prior and is not affected by this data corruption. When $\rho = 0$, we see that our approach not only outperforms LAZYSPSET, the variance is significantly lower. As ρ increases, our approach becomes comparable to LAZYSPSET- both having a large variance. Note that LAZYSPSET does not use any prior knowledge. The fact that our approach is robust to this change in test environment can be attributed to the ability of BISECT to reason about all worlds.

We take a closer look at a simple illustrative example from the Baffle dataset for SE(2) path planning as shown in Fig. 6. The combination of the narrow gap between two walls and the curvature constraint of the robot makes this a challenging problem for BISECT. We see that the prior over edge validity is not informative enough for BISECT to find the gap. However, as DIRECT proceeds to collision check edges, it is quickly able to localize the gap between the two walls. Interestingly, it is *relatively uncertain about the actual vertical location of the wall* - this is reflective of DIRECT judiciously reducing uncertainty only enough to make a region valid (i.e to know if a candidate path would be feasible). The posterior is much more informative for BISECT which is able to easily find a feasible path.

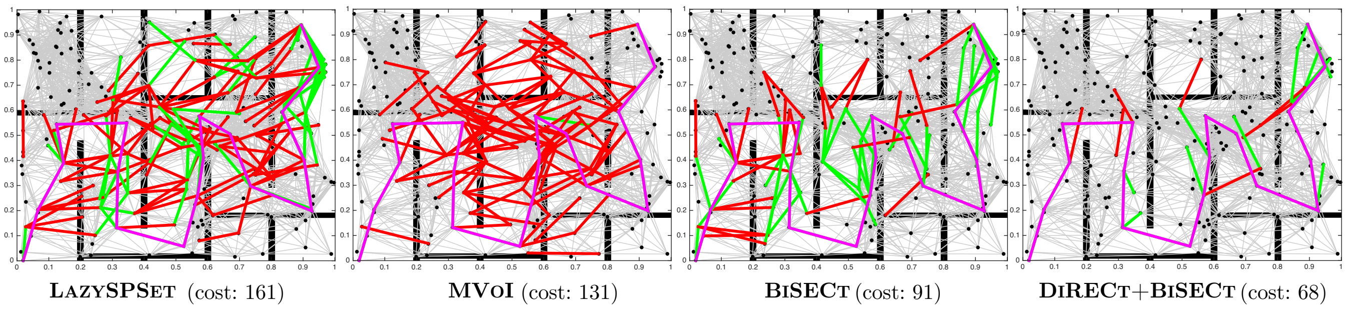

Figure 4: Comparison of algorithms on the Maze dataset. DIRECT +BISECT exploits the rectilinear structure of the maze.

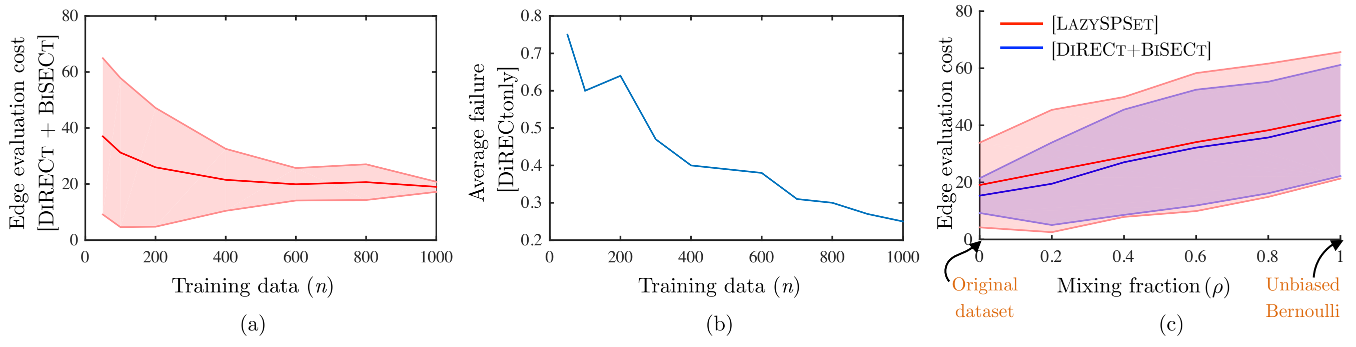

Figure 5: (a) Mean and variance of edge evaluation cost of DIRECT +BISECT with increasing training size. (b) The average failure (to indentify a feasible path) rate when only using DIRECT (without BISECT). (c) Mean and variance of evaluation cost of our approach and LAZYSPSET when test data is corrupted by mixing it with a unbiased bernoulli distribution.

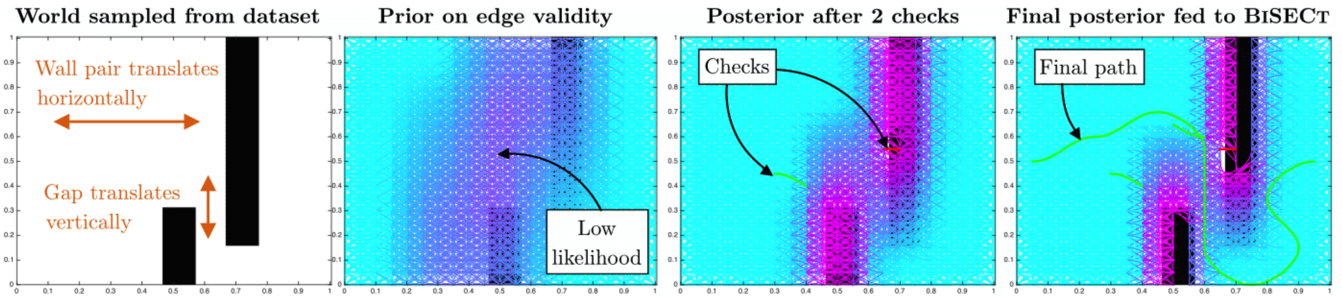
World sampled from dataset
Wall pair translates horizontally
Gap translates vertically
Prior on edge validity
Low likelihood
Posterior after 2 checks
Checks
Final posterior fed to BISECT
Final path

Figure 6: Illustration of belief propagation when applying DIRECT on a simple example from the baffle dataset(SE(2)). Two walls force the path to maneuver through the gap. DIRECT is able to collapse the uncertainty enough to locate the gap. BISECT finishes off the problem.

6 Conclusion

In this paper, we addressed the problem of identification of a feasible path from a library while minimizing the expected cost of edge evaluation given priors on the likelihood of edge validity. We showed that this problem is equivalent to a DRD problem where the goal is to select tests (edges) that drive uncertainty into a single decision region (a valid path). We proposed an approach that combines two DRD algorithms, DIRECT and BISECT, to efficiently solve the problem. We validated our approach on a spectrum of problems against state of the art heuristics and showed that it has a consistent performance across datasets. These results demonstrate the utility of prior data to achieve real-time robot planning.

We have only taken a first step towards forming a bridge between motion planning and active learning. There are several open questions and research directions:

1. *Generalization to all paths:* How can we reason about set of all paths without explicit enumeration?
2. *Incorporate solution quality:* What is an optimal policy for identifying the shortest path in a library?
3. *Dealing with data starvation:* Can we leverage learning oracles that continuously predict plausible worlds?

Acknowledgements

Sebastian Scherer acknowledges support of ONR contract N00014-12-C-0671. The authors thank Shushman Choudhury and Shervin Javdani.

References

[Bialkowski *et al.*, 2013] Joshua Bialkowski, Michael Otte, and Emilio Frazzoli. Free-configuration biased sampling for motion planning. In *IROS*, 2013.

[Bialkowski *et al.*, 2016] Joshua Bialkowski, Michael Otte, Sertac Karaman, and Emilio Frazzoli. Efficient collision checking in sampling-based motion planning via safety certificates. *The International Journal of Robotics Research*, 35(7):767–796, 2016.

[Bohlin and Kavraki, 2000] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *ICRA*, 2000.

[Burns and Brock, 2005] Brendan Burns and Oliver Brock. Sampling-based motion planning using predictive models. In *ICRA*, 2005.

[Chaloner and Verdinelli, 1995] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[Chen *et al.*, 2015] Yuxin Chen, Shervin Javdani, Amin Karbasi, Drew Bagnell, Siddhartha Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *AAAI*, 2015.

[Choudhury *et al.*, 2014] Sanjiban Choudhury, Sankalp Arora, and Sebastian Scherer. The planner ensemble and trajectory executive: A high performance motion planning system with guaranteed safety. In *AHS 70th Annual Forum*, 2014.

[Choudhury *et al.*, 2016a] Sanjiban Choudhury, Jonathan D. Gammell, Timothy D. Barfoot, Siddhartha Srinivasa, and Sebastian Scherer. Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In *ICRA*, 2016.

[Choudhury *et al.*, 2016b] Shushman Choudhury, Christopher M Dellin, and Siddhartha S Srinivasa. Pareto-optimal search over configuration space beliefs for anytime motion planning. In *IROS*, 2016.

[Choudhury *et al.*, 2017a] Sanjiban Choudhury, Shervin JAvdani, Siddhartha Srinivasa, and Sebastian Scherer. Near-optimal edge evaluation in explicit generalized binomial graphs. In *NIPS*, 2017.

[Choudhury *et al.*, 2017b] Sanjiban Choudhury, Shervin JAvdani, Siddhartha Srinivasa, and Sebastian Scherer. Near-optimal edge evaluation in explicit generalized binomial graphs. *Arxiv*, 2017.

[Cohen *et al.*, 2015] Benjamin Cohen, Mike Phillips, and Maxim Likhachev. Planning single-arm manipulations with n-arm robots. In *Eigth Annual Symposium on Combinatorial Search*, 2015.

[Dasgupta, 2004] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, 2004.

[Dellin and Srinivasa, 2016] Christopher M Dellin and Siddhartha S Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *ICAPS*, 2016.

[Dellin *et al.*, 2016] Christopher M Dellin, Kyle Strabala, G Clark Haynes, David Stager, and Siddhartha S Srinivasa. Guided manipulation planning at the darpa robotics challenge trials. In *Experimental Robotics*, 2016.

[Gammell *et al.*, 2015] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch Informed Trees: Sampling-based optimal planning via heuristically guided search of random geometric graphs. In *ICRA*, 2015.

[Golovin and Krause, 2011] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 2011.

[Golovin *et al.*, 2010] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, 2010.

[Hauser, 2015] Kris Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *ICRA*, 2015.

[Howard, 1966] Ronald A Howard. Information value theory. *IEEE Tran. Systems Science Cybernetics*, 1966.

[Hsu *et al.*, 1997] David Hsu, J-C Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *ICRA*, 1997.

[Huh and Lee, 2016] Jinwook Huh and Daniel D Lee. Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees. In *ICRA*, 2016.

[Javdani *et al.*, 2014] Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *AISTATS*, 2014.

[Kavraki *et al.*, 1996] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[Kononenko, 2001] Igor Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence in Medicine*, 2001.

[Krause and Guestrin, 2009] Andreas Krause and Carlos Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research*, 35:557–591, 2009.

[Lacevic *et al.*, 2016] Bakir Lacevic, Dinko Osmankovic, and Adnan Ademovic. Burs of free c-space: a novel structure for path planning. In *ICRA*. IEEE, 2016.

[Narayanan and Likhachev, 2017] Venkatraman Narayanan and Maxim Likhachev. Heuristic search on graphs with existence priors for expensive-to-evaluate edges. In *ICAPS*, 2017.

[Nielsen and Kavraki, 2000] Christian L Nielsen and Lydia E Kavraki. A 2 level fuzzy prm for manipulation planning. In *IROS*, 2000.