Hardness of 3D Motion Planning Under Obstacle Uncertainty

Luke Shimanuki[1] and Brian Axelrod[2]✉

[1] Massachusetts Institute of Technology, Cambridge MA 02139, USA,
`lukeshim@csail.mit.edu`
[2] Stanford University, Stanford CA 94305, USA,
`baxelrod@cs.stanford.edu`

Abstract. We consider the problem of motion planning in the presence of uncertain obstacles, modeled as polytopes with Gaussian-distributed faces (PGDF). A number of practical algorithms exist for motion planning in the presence of known obstacles by constructing a graph using randomly sampled vertices, then efficiently searching the graph to find a collision-free path. We show that such a class of algorithms is unlikely to be efficient in the domain with uncertain obstacles. In particular, we show that safe 3D motion planning among PGDF obstacle is $NP-$hard with respect to the number of obstacles, and remains $NP-$hard after being restricted to a graph. Our reduction is based on a path encoding of 3−SAT and uses the risk of collision with an obstacle to encode the variable assignment. This implies that, unlike in the known case, planning under uncertainty is hard, even when given a graph containing the solution.

Keywords: Collision Avoidance, Completeness and Complexity, Motion and Path Planning

1 Introduction

Navigation under uncertainty is one of the most basic problems in robotics. While there are many methods to plan a trajectory between two points with a known environment and strong theoretical guarantees, few of them generalize to obstacles with locations estimated by noisy sensors. It has proven much harder to provide strong completeness, runtime, and optimality guarantees in this setting.

While some of the original work addressing planning under uncertainty was able to capture the additional richness of this problem by modeling it as a partially observable markov process (POMDP) [8], it has proven difficult to solve POMDPs for complicated real world problems despite large advances in POMDP solvers [16, 28]. In fact, solving POMDPs is PSPACE-complete in the finite horizon and undecidable otherwise, suggesting that it likely not possible to find a general, efficient algorithm for solving POMDPs [20].

Luckily, navigating among uncertain obstacles is a significantly more restricted problem class than POMDPs, giving us hope that we might find an algorithm that is efficient in practice and gives strong theoretical guarantees such as completeness and safety.

Axelrod et al. proposed solving an approximation of the navigation under uncertainty problem [4, 3]. Instead of trying to compute a path that minimizes the true probability of collision under any distribution of obstacles, they propose solving a restricted problem where the obstacles are limited to a structured class of distributions and the collision probability is approximated using a shadow (the geometric equivalent of a confidence interval). While shadow bounds are inherently loose (they overestimate the probability of collision when the obstacle is likely to be far away from the estimated location) they greatly decrease the computational complexity of bounding the probability of collision, since only space visited by the robot close to the obstacle affects the probability bound.

Axelrod et al. proposed the following question: Is there an efficient algorithm that, given a graph embedded in $\mathbb{R}^n$ and a set of obstacles, can find the path with minimal risk as computed via a shadow bound [3]? Unlike the original problem without the shadow approximation, the cost function was only influenced by the portion of the trajectory close to the obstacle and had submodular structure with respect to the graph. The fact that similar approximations have worked well for motion planning, and the existence of efficient algorithms for certain classes of submodular minimization problems gave the hope that it might be possible to find an efficient algorithm for this problem as well.

While motion planning is hard in general, practical and efficient algorithms have proven very successful under some assumptions [17]. One such body of work are the sampling-based motion-planning methods. These algorithms often have the assumption that the problem can be split into two pieces: First use a practically (though often not worst-case) efficient method to generate a small graph that contains a solution; then use a standard, efficient graph algorithm to find the solution in this graph. Algorithms based on this scheme have been successful even for high dimensional planning problems for robots with many degrees of freedom.

There are several other classes of practically efficient algorithms (including grid based and optimization-based planners) that rely on the assumption that part of the problem may be solved much more efficiently in the average case than in the worst case. We discuss this further in the background section.

1.1 Results

This paper answers the question posed by Axelrod et al. in the negative [3].

Theorem 1. *Safe path planning in the presence of uncertain obstacles in 3 dimension is NP-hard.*

A more formal statement of this result is presented in Section 3.

The proofs presented in this paper illuminate what makes this problem more difficult than the standard motion-planning problem with known obstacles. Searching for the minimum-risk path does not have a Markov-like structure. Unlike in the shortest-path problem on a graph, the risk of the second half of a trajectory is very much affected by the first half. This means that the problem is lacking the Bellman property, as identified by Salzman et al. [25].

The absence of a Markov-like property for the risk over the path seems almost necessary to capture important parts of the original problem. The probabilities that one collides with an obstacle are most certainly correlated across the trajectory since collision is inherently a local property. We further discuss sufficient properties of the random obstacle model to guarantee hardness in the conclusion.

2 Background

Motion planning for robotics has been extensively studied in many different settings. One high-level distinction is between motion planning in a known environment and planning in an environment that is not known.

2.1 Complexity in Motion Planning

The story of motion-planning algorithms in robotics has been one of walking the fine boundaries of complexity classes. On one hand, motion planning is PSPACE-hard in $\mathbb{R}^3$ [23] and $\mathbb{R}^2$ [10, 12] with respect to the number of arms of a robot (and thus dimension of its configuration space). However, while Canny's work on singly-exponential time (with respect to number of arms) roadmaps leads to a polynomial-time algorithm when the number of arms is fixed [6], a different set of algorithms is used in practice. The robotics community has been able to find practically efficient methods that provide meaningful theoretical guarantees weaker than completeness (finding a solution if one exists). Sampling-based planners such as Rapidly-Exploring Random Trees (RRTs) [18, 17] and Probabilistic Roadmaps (PRMs) [15] are both practically efficient and *probabilistically complete* under some regularity conditions. Given effective heuristics, graph-based planners have also proved efficient and provide *resolution completeness* [17].

Searching for optimal plans, as opposed to simply feasible plans, further increases the difficulty. In a classic result, Canny shows that the 3-d Shortest-Path Problem is PSPACE-complete for a simple robot in terms of the number of obstacles [7]. This ruled out results of the form of Canny's roadmap algorithm that showed fixed parameter tractability in the feasible motion planning case.

However, the community has been able to find practically efficient algorithms regardless of these worst-case results. A modified version of the original sampling-based algorithms allows them to return nearly optimal solutions in the limit [14] and graph-based planning algorithms are able to provide bounds on the suboptimality of their solutions [1].

On the other hand, complexity results for motion planning under uncertainty have been quite rare, especially in the continuous setting. In contrast to the work on in the discrete setting [25, 8], we hope to introduce lower bounds in the continuous setting.

2.2 Planning under Uncertainty

While planning under uncertainty has been broadly studied in robotics, few methods have formal guarantees on solution quality and efficient runtime. We survey some of the related work below.

Many works assume some sort of uncertainty about the environment, but do not propose a model in which to rigorously quantify the uncertainty in the environment and provide guarantees about the success probability of the trajectory. Instead they often rely on heuristics that seem to provide the desired behavior.

One line of work focuses on uncertainty in the robot's position. Here the model of the robot itself is "inflated" before the collision checking, ensuring that any slight inaccuracy in the position estimate or tracking of the trajectory does not result in a collision.

Work that focuses on uncertainty in the environment sometimes does the exact opposite. They often inflate the occupied volume of the obstacle with a "shadow" and ensure that any planned trajectory avoids the shadow [13, 19].

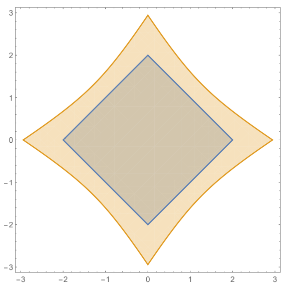


Fig. 1. The orange set is a shadow of the obstacle. The blue set is the obstacle represented by the mean parameters.

A more general approach that handles either or both of localization and obstacle uncertainty is belief-space planning. Belief space is the set of all possible beliefs about or probability distributions over the current state. Belief-space planning converts the uncertain domain in state space to belief space, then plans in belief space using trees [22, 5] or control systems [21].

Another line of work uses synthesis techniques to construct a trajectory intended to be safe by construction. If the system is modeled as a Markov Decision Process with discrete states, a safe plan can be found using techniques from formal verification [9, 11]. Other authors have used techniques from Signal Temporal Logic combined with an explicitly modeled uncertainty to generate plans that are heuristically safe [24].

Shadows In previous work, Axelrod et al. formalized the notion of a shadow in a way that allowed the construction of an efficient algorithm to bound the probability that a trajectory will collide with the estimated obstacles [4, 3].

We can now define a shadow rigorously:

Definition 1 (ϵ-shadow). *A set $S \subseteq \mathbb{R}^d$ is an ϵ-shadow of a random obstacle $O \subseteq \mathbb{R}^d$ if $Pr[O \subseteq S] \geq 1 - \epsilon$.*

Shadows are important because they allow for an efficient method to upper-bound the probability of collision. If there exists an $\epsilon-$shadow of an obstacle that does not intersect a given trajectory's swept volume, then the probability of the obstacle intersecting with the trajectory is at most ϵ. An example of a shadow for an obstacle is shown in figure 1.

3 Problem Formulation

3.1 Notation

In this section we will cover definitions and notation conventions that will be used in this paper.

- A vector will be marked in bold as in $\mathbf{u}$, in contrast to a scalar u.
- $\wedge$, $\vee$, and $\neg$ are the logical AND, OR, and NOT operators, respectively.
- The power set (set of all subsets) of S is denoted by $\mathcal{P}(S)$. A function mapping into the power set of $\mathbb{R}^n$ outputs subsets of $\mathbb{R}^n$.

Definition 2 (Standard Basis Vector). *A Standard Basis Vector in d dimensions is a vector $e_i \in \mathbb{R}^d$ that is 1 in the ith dimension and 0 in the remaining dimensions.*

3.2 Random Obstacle Model

In order to attempt to provide formal non-collision guarantees one must first model the uncertainty in the environment. At a high level we assume that each episode of the robot's interaction happens in the following sequence:

1. A set of obstacles is drawn from a known distribution (the conditional distribution for the obstacles given the sensor observations). These obstacles now remain static for the duration of the episode.
2. The robot computes, commits to and executes a trajectory.
3. The probability of collision in question is exactly the probability that this trajectory collides with at least one of the obstacles.

In this work we restrict ourselves to polytopes with Gaussian-distributed faces (PGDFs). A PGDF is the intersection of halfspaces with parameters drawn from a multivariate normal distribution. More formally a PGDF $O \subset \mathbb{R}^n$ is $O = \bigcap^{i} \alpha_i^T x \leq 0$, where $\alpha_i \sim \mathcal{N}(\mu_i, \Sigma_i)$. We can use homogenous coordinates to create obstacles not centered about the origin.

One reason that PGDF obstacles are important is that we have methods of computing shadows for PGDF obstacles efficiently [4].

We note that this formulation differs from the notion of "risk-zones" evaluated by Salzman et al., where the cost of a trajectory is proportional to the amount of time spent within a risk-zone [26, 25]. These problems share the lack of an optimal substructure—the subpaths of an optimal path are not necessarily optimal. Salzman et al. provide a generalization of Dijkstra's algorithm that finds minimum-risk plans in their domain efficiently [25], but as we will show, there are no such techniques for our problem.

3.3 Algorithmic Question

Now that we have defined shadows and PGDF obstacles, we can define what it means for a path to be safe. Suppose the robot operates, and obstacles are, in $\mathbb{R}^d$ (d is usually 3). This is commonly referred to as the task space. Furthermore, suppose the configuration space of the robot is parametrized in $\mathbb{R}^k$ (usually corresponding to the k degrees of freedom of the robot).

Since planning usually happens in the robot's configuration space, but the obstacles are in task space, we need to be able to convert between the two.

Definition 3 (Embedding Map). *A function $f : \mathbb{R}^k \to \mathcal{P}(\mathbb{R}^d)$ is an embedding map if it maps robot configurations into the subset of $\mathbb{R}^d$ that is occupied by the robot at that configuration.*

The embedding map can usually be constructed by combining the forward kinematics and robot model.

Definition 4 (Configuration Space Trajectory). *A configuration space trajectory $\tau : [0, 1] \to \mathbb{R}^k$ is a map from a "time" index into the trajectory to the robot configuration at that point in the trajectory.*

Definition 5 (Task-space Trajectory). *A task-space trajectory $\tau' : [0, 1] \to \mathcal{P}(\mathbb{R}^d)$ is defined as the map between an index into the trajectory and the space occupied by the robot at that point in the trajectory.*

Alternatively, if given a configuration space trajectory τ, $\tau'(t) = f(\tau(t))$ where f is the embedding map.

For the rest of the paper we will only concern ourselves with task-space trajectories, noting that it is easy to go from a configuration space trajectory to a task-space trajectory using the embedding map.

Definition 6 (Swept Volume). *The swept volume X of a task-space trajectory τ is the set of task-space points touched by the robot while executing trajectory τ.*

Said differently, $X = \bigcup_{t \in [0,1]} \tau(t)$.

This allows us to formally define what it means for a trajectory to be safe.

Definition 7 (ϵ-safe trajectory). *Given a joint distributions over random obstacles, a task-space trajectory is ϵ-safe if the corresponding swept volume has at* most *ϵ probability of intersecting at least one obstacle.*

This leads to the following algorithmic question, of finding safe plans for a known distribution of PGDF obstacles.

Problem 1 (ϵ-safe Planning Problem) *Given the parameters of PGDF distributions for each obstacle and initial and end points $\mathbf{s}, \mathbf{t}$ in configuration space, find an ϵ-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$.*

Note that there exists reductions between the safe planning problem and finding a path that minimizes the risk of collision. Since the probability ϵ is confined to $[0, 1]$, a binary search over ϵ yields an efficient algorithm that can approximately compute the minimum risk given an $\epsilon-$safe planner. For convenience, our proofs will consider the approximate minimum-risk planning problem, though the construction applies directly to $\epsilon-$safe planning as well.

Problem 2 ($(1+\alpha)$-approximate minimum-risk planning problem) *Given the parameters of PGDF distributions for each obstacle and initial and end points* $\mathbf{s}, \mathbf{t}$ *in configuration space, return a* $((1+\alpha)\epsilon_*)$*-safe trajectory from* $\mathbf{s}$ *to* $\mathbf{t}$*, where* ϵ_* *is the minimum* ϵ *for which an* ϵ*-safe trajectory exists.*

3.4 Graph Restriction

We start by considering the class of motion-planning algorithms that first construct a graph embedded in the robot's configuration space, and then run a graph-search algorithm to find a path within the graph. This class of algorithms has been shown to be practical in the known environment, with sampling-based planners such as RRT and RRG. Conditioned on there being a nonzero probability of sampling a solution, these algorithm are guaranteed to find a collision-free path with probability approaching 1 as the number of iterations approaches infinity. [18, 14].

More formally this condition can be articulated as the existence of a path in the δ-interior of the free space X_{free}.

Definition 8 (δ-interior [14]). *A state* $x \in X_{free}$ *is in the* δ*-interior of* X_{free} *if the closed ball of radius* δ *around* x *lies entirely in* X_{free}.

This condition is necessary because it guarantees that finding a plan does not require waiting for a zero probability event. However this formulation does not extend well to the domain with uncertain obstacles; there is no concept of "free space" because the locations of the obstacles are not known. Instead we will use the equivalent view of inflating the path instead of shrinking the free space.

Definition 9 (δ-inflation). *The* δ*-inflation of the set* X *is the set* $Y = \bigcap_{x \in X} \{y \mid d(x, y) \leq \delta\}$.

We note that in the deterministic setting, if a trajectory is in the δ-interior of X_{free}, then the δ-inflation of the trajectory is entirely in X_{free}. This allows us to consider problems with the following regularity condition: there exists a δ-inflated task-space trajectory that has a low risk of collision.

Definition 10 (ϵ-safe δ-inflated task-space trajectory). *A task-space trajectory is an* ϵ*-safe* δ*-inflated trajectory if its* δ*-inflation intersects an obstacle with probability at most* ϵ.

We want to find an algorithm that satisfies the completeness and safety guarantees defined below.

Definition 11 (Probabilistically Complete $(1{+}\alpha)$-approximate Safe Planning Algorithm). *A planning algorithm takes a set of PGDF obstacles, a start state* $\mathbf{s}$*, and a goal state* $\mathbf{t}$ *as input and generates a path as output. A planning algorithm is probabilistically complete and* $(1+\alpha)$*-approximate safe if, with* n *samples, the probability that it finds a* $((1+\alpha)\epsilon_*)$*-safe trajectory approaches* 1 *as* n *approaches* ∞*, where* ϵ_* *is the minimum* ϵ *for which an* ϵ*-safe trajectory exists.*

Axelrod et al. provides an extension of the RRT algorithm to the probabilistic domain using the shadow approximation [4]. The uniqueness of paths between any two vertices in a tree, makes the finding the optimal (restricted to the tree) path trivial. However, while the paths it generates are indeed safe, the algorithm is not probabilistically complete.

The following extension of the RRG algorithm is probabilistically complete [2].

Algorithm 1 SAFE_RRG

```
Input: End points $\mathbf{s}, \mathbf{t} \in \mathbb{R}^d$, set of PGDF obstacles $O$, and number of samples $n$.
Output: A $((1+\alpha)\epsilon_*)$-safe trajectory from $\mathbf{s}$ to $\mathbf{t}$, where $\epsilon_*$ is the minimum $\epsilon$ for which
    an $\epsilon$-safe trajectory exists.
1: G = CONSTRUCT_RRG($\mathbf{s}$, $\mathbf{t}$, n)
2: return GRAPH_SEARCH($G, O, s, t$)
```

We note that as n increases, the probability that there is a sample near any given point x in the space approaches 1. Here, GRAPH_SEARCH is a $(1+\alpha)$-approximate safe graph-search algorithm as defined below.

Definition 12 ($(1{+}\alpha)$-approximate safe graph-search algorithm). *A* $(1+\alpha)$*-approximate safe graph-search algorithm is a procedure* $\phi(G, O, s, t)$*, where* G *is a graph,* O *is a set of PGDF obstacles, and* s *and* t *are the start and end nodes in* G*, respectively. It returns a* $((1+\alpha)\epsilon_*)$*-safe trajectory in* G*, where* ϵ_* *is the minimum* ϵ *for which an* ϵ*-safe trajectory exists.*

Theorem 2 ([2]). *SAFE_RRG is probabilistically complete and* $(1{+}\alpha)$*-approximate safe as long as GRAPH_SEARCH is complete and* $(1+\alpha)$*-approximate safe.*

However, no graph-search procedure, beyond the naïve, exponential-time search procedure, is provided [2]. Sampling-based motion-planning algorithms succeed in the known environment because efficient graph-search algorithms can quickly find collision-free paths within a graph. In order for this class of motion-planning algorithms to be practical, we would need a corresponding graph-search algorithm in the probabilistic domain. Because the cost of a path depends on what set of shadows it intersects, the state space of the graph search is not just the current node but also includes the accumulated risk incurred by each obstacle. This means that the typical approaches for searching graphs with known obstacles, which make use of dynamic programming, cannot be applied in the same manner to graphs with unknown obstacles.

Unfortunately, as will be shown in the remainder of this paper, this problem is NP-HARD with respect to $n = \Theta(|G| + |O|)$, the size of the input.

Theorem 3. *Unless $P = NP$, there is no $(1+\alpha)$-approximate safe graph-search algorithm that, given a graph with n nodes, runs in $POLY(n)$ time, when restricted to $O(n)$ obstacles in $\mathbb{R}^{O(n)}$, where $\alpha = \Theta(\frac{1}{n})$.*

And we can strengthen it to show that the minimum-risk planning problem in constant dimension is hard in general, that is, even when not restricted to a graph.

Theorem 4. *The $(1+\alpha)$-approximate minimum-risk planning problem is NP-hard. That is, unless $P = NP$, there is no Probabilistically Complete $(1+\alpha)$-approximate Safe Planning Algorithm for $\mathbb{R}^3$ that runs in $POLY(n)$ when $\alpha = \Theta(\frac{1}{n})$.*

4 Basic Hardness Result

4.1 3SAT

3SAT is an NP-complete problem that is commonly used to prove the hardness of other problems [27]. The problem input is a Boolean formula given in conjunctive normal form, where each clause consists of three literals, or in other words, it is of the form $((x_0 \vee \neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge \ldots)$. The algorithm must then decide whether there exists any variable assignment that satisfies the formula. We will consider a 3SAT problem with k variables $x_0, x_1, \ldots$ and m clauses, where each clause j is of the form $(x_{j_u} \vee \neg x_{j_v} \vee x_{j_w})$.

4.2 Proof Outline

We prove Theorem 3 using a reduction from 3SAT. Given a 3SAT instance, we construct a $(1+\alpha)$-approximate safe graph-search problem as follows.

1. Construct a pair of obstacles for each variable that will encode whether the variable is set to *true* or *false*.
2. Construct a portion of the graph to force the algorithm to assign every variable by going near either the *true* or *false* obstacle for each variable.
3. Construct the remainder of the graph to force the algorithm to satisfy every clause. There will be additional collision risk for a path that goes by both the *true* and *false* obstacles (i.e. uses both x and $\neg x$).

The solution to the planning problem can then be transformed into a solution to the 3SAT instance in polynomial time, demonstrating that the $(1+\alpha)$-approximate safe graph-search problem is at least as hard as 3SAT.

The proof for $\mathbb{R}^3$ will use a similar technique with a more complicated construction that folds the graph into $\mathbb{R}^3$.

4.3 Proof

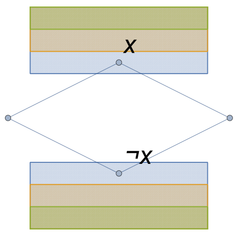


Fig. 2. A path through this gadget must select one of the obstacles to go near, either the positive assignment x (obs_{2i}) or the negative assignment $\neg x$ (obs_{2i+1}).

Our graph and obstacle set will be in space $\mathbb{R}^d$ where $d = k + 1$. Each of the first k dimensions will correspond to each variable, and dimension $t = k + 1$ can be thought of as "time" or some monotonically increasing value as we progress along a path.

Variable Assignment First, for each variable i in the 3SAT problem, we will construct two halfspace PGDF obstacles.

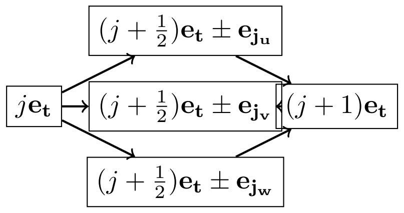
Note that we define a PGDF obstacle as the intersection of halfspaces of the form $\alpha^T x \leq 0$ for α normally distributed and x represented in homogeneous coordinates. Here we will work with just one face and standard coordinates for convenience.

That is, each obstacle i will be defined as $\alpha_i^T x \leq 1$ for $\alpha \sim (\boldsymbol{\mu}_\mathbf{i}, \Sigma_i)$.

For each variable i we define a *true* obstacle and a *false* obstacle with the the following parameters; *true*: $\alpha \sim \mathcal{N}(2e_i, e_i e_i^T)$, *false*: $\alpha \sim \mathcal{N}(-2e_i, e_i e_i^T)$.

Intuitively the covariance $e_i e_i^T$ means that α has variance 1 in the direction of the normal of the face. This is important because it means that there is no variance in the orientation of the face.

Then we will construct a graph that will force any path to select a *true* or *false* assignment for each variable as illustrated in figure 2. Said formally, indexing over the variable with index i, we embed nodes in locations $-(i+1)e_t, -(i+\frac{1}{2})e_t \pm e_i, -ie_t$. We then draw edges from $-(i+1)e_t$ to both $-(i+\frac{1}{2})e_t \pm e_i$ and from both $-(i+\frac{1}{2})e_t \pm e_i$ to $-ie_t$.



$$\begin{tikzcd}
 & \boxed{(j+\frac{1}{2})\mathbf{e_t} \pm \mathbf{e_{j_u}}} \arrow[dr] & \\
\boxed{j\mathbf{e_t}} \arrow[ur] \arrow[r] \arrow[dr] & \boxed{(j+\frac{1}{2})\mathbf{e_t} \pm \mathbf{e_{j_v}}} \arrow[r] & \boxed{(j+1)\mathbf{e_t}} \\
 & \boxed{(j+\frac{1}{2})\mathbf{e_t} \pm \mathbf{e_{j_w}}} \arrow[ur] &
\end{tikzcd}$$

Fig. 3. A path through this gadget must take one of three paths, each extending in different dimensions. Each path goes near an obstacle corresponding to the respective literal.

Clause Gadget For each clause j, we will construct a graph that lets the algorithm choose which variable with which to satisfy the clause as shown in figure 3.

Recall that each clause j is of the form $(x_{j_u} \vee \neg x_{j_v} \vee x_{j_w})$. First indexing over j, construct "via" nodes at je_t and $(j+1)e_t$, respectively. Then if x_{j_u} appears positively, construct a node connected to the "via" nodes at $(j+\frac{1}{2})e_t + e_{j_u}$ and at $(j+\frac{1}{2})e_t - e_{j_u}$ if the variable appears negatively. Repeat for x_{j_v} and x_{j_w}. This is illustrated in figure 3.

A path through this gadget must pick one of the literals in the clause to satisfy and pass near the obstacle that corresponds to that variable and the value the literal requires it to have. In doing so, it may incur risk of intersecting with the obstacle. If this variable was assigned to the value the literal specifies, then the path would have already gone near this obstacle so no further risk is incurred. However, if the literal contradicts the variable assignment, the path will incur additional risk for going near this obstacle.

Path Risk Encoding 3SAT The above graph was constructed such that there will exist a gap between the risk of a satisfying assignment and of a non-satisfying assignment.

First we note that for any reasonable shadow approximation (including the ones presented in [4]), there exists a gap between the induced risk close to the obstacle and far away from the obstacle. For shadow approximation schemes where this is the case, there is some r_{close} that lower-bounds the risk computed from the shadow approximation for the closer points and r_{far} that upper-bounds the computed risk for the further point. This holds true for all shadows derived from the methods in [3] but may not hold for shadows that are computed via Monte Carlo algorithms.

A path through the variable assignment portion of the graph will go near k obstacles for the k variable assignments it makes. Then it will be "close" to k obstacles and "far" from the other k obstacles. Therefore, it will incur risk $kr_{close} + kr_{far}$.

If a path through the variable assignment portion encodes a satisfying assignment to the 3SAT problem, there will exist a path through the remainder of the graph that will not incur any additional cost. If there is no satisfying assignment, then any path through the remaining portion must go near an obstacle that it did not go near in the variable assignment portion, so for some variable i, the optimal path must go close to both the *true* and *false* obstacles, incurring cost at least $(k+1)r_{close} + (k-1)r_{far}$. This allows us to compute a lower bound on ratio between the two risks:

$$\begin{aligned} risk\_ratio &= \frac{(k+1)r_{close} + (k-1)r_{far}}{kr_{close} + kr_{far}} \\ &= \frac{kr_{close} + kr_{far} + r_{close} - r_{far}}{kr_{close} + kr_{far}} \\ &= 1 + \frac{r_{close} - r_{far}}{kr_{close} + kr_{far}} \\ &= 1 + \Theta\left(\frac{1}{k}\right). \end{aligned}$$

Each gadget can be constructed in polynomial time, and the number of gadgets is polynomial, so this reduction can be constructed in polynomial time. Thus any algorithm that can approximate the minimum-risk planning problem in a graph to a factor better than $1 + \Theta\left(\frac{1}{k}\right)$ can also solve 3SAT with polynomial overhead. □

5 Hardness Result in $\mathbb{R}^3$

We prove Theorem 4 by reducing from 3SAT using the same outline as for Theorem 3 but with more complicated gadgets. We also add deterministic obstacles to force any procedure that computes a safe plan to decide the 3SAT problem, showing that the problem is hard with and without the graph restriction.

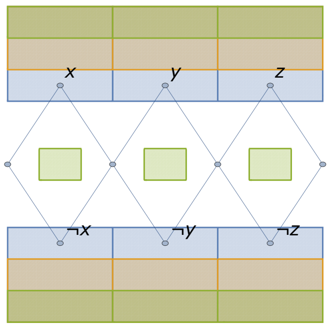


Fig. 4. A path through this gadget must go near either the *true* or *false* obstacle for each variable, thereby selecting a variable assignment.

5.1 Proof

Variable Gadgets As before, we will construct two PGDF obstacles for each variable, but this time they will each have three faces. We will continue to use standard coordinates for the remainder of this proof.

For obstacle i, the *true* obstacle will be defined as the intersection of $\alpha_i^T x \leq 1$ and $i \leq e_2^T x \leq i+1$, where $\alpha_i \sim \mathcal{N}(2e_1, e_1 e_1^T)$. The "negative" obstacle will similarly be defined with $\beta_i^T x \leq 1$ and $i \leq e_2^T x \leq i+1$, where $\beta_i \sim \mathcal{N}(-2e_1, e_1 e_1^T)$.

Then we will construct the variable assignment graph, as illustrated in figure 4. Said formally, indexing over the variable with index i, we embed nodes in locations $ie_2, (i+\frac{1}{2})e_2 \pm e_1, (i+1)e_2$. We then draw edges from ie_2 to both of $(i+\frac{1}{2})e_2 \pm e_1$, and from both of $(i+\frac{1}{2})e_2 \pm e_1$ to $(i+1)e_2$.

Because for this proof we allow any path, not just those restricted to the graph, we need an additional obstacle to block the path from crossing in between the variable obstacles. This obstacle will be deterministic and be defined as the intersection of $-\frac{1}{8} \leq e_1^T x \leq \frac{1}{8}$, $i+\frac{1}{8} \leq e_2^T x \leq i+\frac{7}{8}$, and $e_3^T x \leq \frac{1}{2}$. We also need an obstacle above the gadget to prevent a path from just going straight up without first passing through the gadget. This obstacle is defined as the intersection of $e_2^T x \leq k+\frac{1}{2}$ and $\frac{1}{4} \leq e_3^T x \leq \frac{3}{4}$.

Clause Gadgets For each clause j we will construct an additional graph "layer" as illustrated in figure 5.

Recall that each clause j is of the form $x_{j_u} \vee \neg x_v \vee x_w$. Without loss of generality, let $j_u < j_v < j_w$. Indexing over j, construct nodes at $(j+1)e_3, (j+1)e_3+(j_u+\frac{1}{2})e_2, (j+1\frac{1}{3})e_3+(j_u+\frac{1}{2})e_2 \pm e_1, (j+1\frac{2}{3})e_3+(j_u+\frac{1}{2})e_2, j+1\frac{2}{3})e_3$, drawing edges between consecutive nodes, letting '±' represent '-' if x_{j_u} is given in negated form and '+' otherwise. Then construct nodes at $(j+1)e_3+(j_u+\frac{1}{2})e_2, (j+1)e_3+(j_v+\frac{1}{2})e_2, (j+1\frac{1}{3})e_3+(j_v+\frac{1}{2})e_2 \pm e_1, (j+1\frac{2}{3})e_3+(j_v+\frac{1}{2})e_2, (j+1\frac{2}{3})e_3+(j_u+\frac{1}{2})e_2$ (the first and last were already constructed previously), drawing edges between consecutive nodes, similarly setting '±' based on the negation of literal x_{j_v}. Then construct nodes at $(j+1)e_3+(j_v+\frac{1}{2})e_2, (j+1)e_3+(j_w+\frac{1}{2})e_2, (j+1\frac{1}{3})e_3+(j_w+\frac{1}{2})e_2 \pm e_1, (j+1\frac{2}{3})e_3+(j_w+\frac{1}{2})e_2, (j+1\frac{2}{3})e_3+(j_v+\frac{1}{2})e_2$ (the first and last were already constructed previously), drawing edges between consecutive nodes, similarly setting '±' based on the negation of literal x_{j_w}. Intuitively, this creates three possible routes through the graph, each going near the obstacle corresponding to a particular value assigned to a variable.

$$\begin{tikzcd}
\boxed{(j+1)e_3+(j_w+\frac{1}{2})e_2} \arrow[dr] & & \boxed{(j+1\frac{2}{3})e_3+(j_w+\frac{1}{2})e_2} \arrow[dd] \\
& \boxed{(j+1\frac{1}{3})e_3+(j_w+\frac{1}{2})e_2 \pm e_1} \arrow[ur] & \\
\boxed{(j+1)e_3+(j_v+\frac{1}{2})e_2} \arrow[uu] \arrow[dr] & & \boxed{(j+1\frac{2}{3})e_3+(j_v+\frac{1}{2})e_2} \arrow[dd] \\
& \boxed{(j+1\frac{1}{3})e_3+(j_v+\frac{1}{2})e_2 \pm e_1} \arrow[ur] & \\
\boxed{(j+1)e_3+(j_u+\frac{1}{2})e_2} \arrow[uu] \arrow[dr] & & \boxed{(j+1\frac{2}{3})e_3+(j_u+\frac{1}{2})e_2} \arrow[dd] \\
& \boxed{(j+1\frac{1}{3})e_3+(j_u+\frac{1}{2})e_2 \pm e_1} \arrow[ur] & \\
\boxed{(j+1)e_3} \arrow[uu] & & \boxed{(j+1\frac{2}{3})e_3}
\end{tikzcd}$$

Fig. 5. A path through this gadget must select one of three paths to go through, each going near the obstacle for the corresponding literal.

As with the variable gadget, we need a few additional deterministic obstacles in each layer to force paths to follow the graph, in the sense that it must still go near the same obstacles as a path restricted to the graph would. First, we put boundaries below and above each layer to prevent paths from just going straight up without passing through the gadgets. These obstacles are defined as the intersection of $\frac{1}{2} \le e_2^T x$ and $j+\frac{9}{12} \le e_3^T x \le j+\frac{11}{12}$, and as the intersection of $\frac{1}{2} \le e_2^T x$ and $j+1\frac{9}{12} \le e_3^T x \le j+1\frac{11}{12}$. Then we construct an obstacle in the middle of the layer that a path must go around, defined as the intersection of $-\frac{1}{4} \le e_1^T x \le \frac{1}{4}$ and $j+1\frac{1}{6} \le e_3^T x \le j+1\frac{1}{2}$.

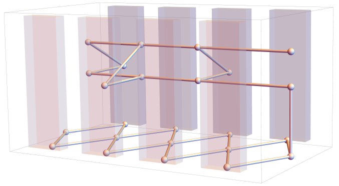


Fig. 6. The bottom layer is the variable assignment gadget. The top layer is a single clause gadget. There would usually be many more clause gadgets stacked on top.

We construct obstacles blocking a path from going around the middle obstacle except for via one of the paths in the graph. For each variable i and value $+/-$, if the literal $(\neg)x_i$ (where the '$\neg$' is dependent on whether the value is '$+$' or '$-$') appears in clause j, construct the obstacle defined as the intersection of $\pm e_1^T x \ge \frac{1}{8}$, $i \le e_2^T x \le i+1$, and $j+\frac{11}{12} \le e_3^T x \le j+1\frac{11}{12}$. Finally, we construct an obstacle after the last variable obstacle to prevent the path from just bypassing all the obstacles. This obstacle is defined as the intersection of $e_2^T x \le k+\frac{3}{4}$, and $j+\frac{11}{12} \le e_3^T x \le j+1\frac{11}{12}$.

Now we combine the variable and clause gadgets, as seen in figure 6. We see that as in the proof of theorem 3, if a path through the variable assignment portion encodes a satisfying assignment, there will exist a path through the remainder of the graph that will not incur any additional cost, where as a nonsatisfying assignment must incur some additional cost in the clause gadgets.

Therefore, we see the same gap between a satisfying assignment and a nonsatisfying assignment: $\frac{(k+1)r_{close}+(k-1)r_{far}}{kr_{close}+kr_{far}} = 1+\Theta(\frac{1}{k})$. Each gadget can be constructed in polynomial time, and the number of gadgets is polynomial, so this reduction can be constructed in polynomial time. Thus, any $(1+\alpha)$-approximate Safe Planning Algorithm can also solve 3SAT with polynomial overhead. □

6 Conclusions and Future work

We have shown that the minimum-risk planning problem on graphs is NP-hard, even in dimension 3. Furthermore, the fact that it remains hard after restriction to a small graph suggests that the design paradigm used for many sampling-based motion-planning algorithms is unlikely to yield practically efficient algorithms with correctness and optimality guarantees.

This suggests that the field should pursue other directions towards solving motion planning under uncertainty. Trajectory-optimization-based methods, for example, might lead to efficient practical solutions. Furthermore, barring stronger hardness-of-approximation results, it is possible that there is a practical approximation algorithm for solving the minimum-risk planning problem on graphs.

There is also the related direction of investigating models of uncertainty over obstacles. We focus on the PGDF model in this work because it captures certain desirable characteristics and has been used in prior work. However, the PGDF model has certain surprising characteristics, particularly near the tails of the distribution [4]. Perhaps there is a model that is a better fit for obstacle estimates in practice, that also permits efficient algorithms. In exploring this direction, it is important to note that we do not strongly invoke the structure of PGDF obstacles. Interesting directions for future work also include finding a good minimal condition on the obstacle distribution to make the problem NP-hard.

Another direction of future work is finding upper bounds on the safe motion-planning problem. While, when there is some "slack" in the shadows for the optimal solution there is a trivial algorithm for finding an approximate solution by exhaustively iterating through an $\epsilon-$net of shadow configurations (each one reduces to a motion planning instance that can be solved by Canny's roadmap algorithm), no exact algorithm is known.

7 Funding

We gratefully acknowledge support from the Thomas and Stacey Siebel Foundation, NSF Fellowship grant DGE-1656518 and NSF grants CCF-1763299, 1420316, 1523767 and 1723381. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors

References

[1] Sandip Aine et al. “Multi-Heuristic A*”. In: *The International Journal of Robotics Research* 35.1-3 (2016), pp. 224–243.

[2] Brian Axelrod. “Algorithms for Safe Robot Navigation”. MA thesis. Massachusetts Institute of Technology, 2017.

[3] Brian Axelrod, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. “Provably safe robot navigation with obstacle uncertainty”. In: *The International Journal of Robotics Research* (2018).

[4] Brian Axelrod, Leslie Kaelbling, and Tomas Lozano-Perez. “Provably Safe Robot Navigation with Obstacle Uncertainty”. In: *Robotics Science and Systems* 13 (2017).

[5] Adam Bry and Nicholas Roy. “Rapidly-exploring random belief trees for motion planning under uncertainty”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 723–730.

[6] John Canny. “Some algebraic and geometric computations in PSPACE”. In: *Proceedings of the Annual ACM Symposium on Theory of Computing*. Jan. 1988, pp. 460–467.

[7] John Canny and John Reif. “New lower bound techniques for robot motion planning problems”. In: *Foundations of Computer Science*. Nov. 1987, pp. 49–60. ISBN: 0-8186-0807-2.

[8] A.R. Cassandra, L.P. Kaelbling, and James Kurien. “Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation”. In: *International Conference on Intelligent Robots and Systems*. Vol. 12. Dec. 1996, 963–972 vol.2. ISBN: 0-7803-3213-X.

[9] Xu Chu Ding, Alessandro Pinto, and Amit Surana. “Strategic planning under uncertainties via constrained Markov decision processes”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 4568–4575.

[10] John E. Hopcroft, Deborah Joseph, and Sue Whitesides. “Movement Problems for 2-Dimensional Linkages”. In: *SIAM Journal on Computing*. Vol. 13. Aug. 1984, pp. 610–629.

[11] Seyedshams Feyzabadi and Stefano Carpin. “Multi-objective planning with multiple high level task specifications”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 5483–5490.

[12] John Hopcroft, Deborah Joseph, and Sue Whitesides. “On the Movement of Robot Arms in 2-Dimensional Bounded Regions”. In: *SIAM Journal on Computing*. Vol. 14. Dec. 1982, pp. 280–289.

[13] Leslie Pack Kaelbling and Tomás Lozano-Pérez. “Integrated task and motion planning in belief space”. In: *The International Journal of Robotics Research* (2013).

[14] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[15] Lydia E Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[16] Hanna Kurniawati, David Hsu, and Wee Sun Lee. “SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces”. In: *Robotics: Science and Systems*. 2008.

[17] S. M. LaValle. *Planning Algorithms*. Available at http://planning.cs.uiuc.edu/. Cambridge, U.K.: Cambridge University Press, 2006.

[18] Steven M. LaValle and James J. Kuffner. “Randomized Kinodynamic Planning”. In: *ICRA*. 1999.

[19] Alex Lee et al. “Sigma hulls for Gaussian belief space planning for imprecise articulated robots amid obstacles”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 5660–5667.

[20] Christos H Papadimitriou and John N Tsitsiklis. “The complexity of Markov decision processes”. In: *Mathematics of operations research* 12.3 (1987), pp. 441–450.

[21] R. Platt et al. “Belief space planning assuming maximum likelihood observations”. In: *Proceedings of Robotics: Science and Systems*. Zaragoza, Spain, June 2010.

[22] Sam Prentice and Nicholas Roy. “The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance”. In: *Proceedings of the 13th International Symposium of Robotics Research (ISRR)*. Hiroshima, Japan, Nov. 2007.

[23] John Reif. “Complexity of the Mover’s problem and generalizations”. In: *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*. Nov. 1979, pp. 421–427.

[24] Dorsa Sadigh and Ashish Kapoor. “Safe Control under Uncertainty with Probabilistic Signal Temporal Logic”. In: *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan, June 2016.

[25] Oren Salzman, Brian Hou, and Siddhartha Srinivasa. “Efficient motion planning for problems lacking optimal substructure”. In: *arXiv preprint arXiv:1703.02582* (2017).

[26] Oren Salzman and Siddhartha Srinivasa. “Open problem on risk-aware planning in the plane”. In: *arXiv preprint arXiv:1612.05101* (2016).

[27] Michael Sipser. *Introduction to the Theory of Computation*. 1st. International Thomson Publishing, 1996. ISBN: 053494728X.

[28] Adhiraj Somani et al. “DESPOT: Online POMDP planning with regularization”. In: *Advances in Neural Information Processing Systems*. Vol. 58. Jan. 2013.