Hardness of Motion Planning with Obstacle Uncertainty in 2 Dimensions

The International Journal of Robotics Research
XX(X):1–16

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Luke Shimanuki[1] and Brian Axelrod[2]

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 in configuration space, then efficiently searching the graph to find a collision-free path. We show that such an exact algorithm is unlikely to be practical in the domain with uncertain obstacles. In particular, we show that safe 2D motion planning among PGDF obstacles 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 MAXQHORNSAT and uses the risk of collision with an obstacle to encode variable assignments and literal satisfactions. This implies that, unlike in the known case, planning under uncertainty is hard, even when given a graph containing the solution. We further show by reduction from 3-SAT that both safe 3D motion planning among PGDF obstacles and the related minimum constraint removal problem remain NP-hard even when restricted to cases where each obstacle overlaps with at most a constant number of other obstacles.

Keywords

Completeness and Complexity, Motion and Path Planning, Obstalce Uncertainty

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 in a known environment with 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 decision process (POMDP) (Cassandra, Kaelbling and Kurien 1996), it has proven difficult to solve POMDPs for complicated real world problems despite large advances in POMDP solvers (Kurniawati, Hsu and Lee 2008; Somani, Ye, Hsu and Sun Lee 2013). 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 (Papadimitriou and Tsitsiklis 1987).

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, Kaelbling and Lozano-Pérez (2017, 2018) proposed solving an approximation of the navigation under uncertainty problem. 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 PGDF 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, Kaelbling and Lozano-Pérez (2018) 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? The cost function derived from the shadow approximation is only influenced by the portion of the trajectory close to the obstacle and has 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 (LaValle 2006). 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

[1]Massachusetts Institute of Technology, MA, USA
[2]Stanford University, CA, USA

Corresponding author:
Brian Axelrod, Stanford University
Stanford, CA 94305, USA
Email: baxelrod@cs.stanford.edu

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 search 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.

This paper answers the question posed by Axelrod, Kaelbling and Lozano-Pérez (2018) in the negative when an exact solution of FPTAS is required.

Theorem 1. *Safe path planning in the presence of uncertain obstacles in 2 dimensions is NP-hard.*

A more formal statement of this result is presented in Section 3. We prove this by reducing from MAXQHORNSAT using a construction based on and very similar to that used by Erickson and LaValle (2013) for the minimum constraint removal problem (MCR). We also show, via reduction from 3-SAT, that both safe path planning and MCR remain hard in three dimensions even when each obstacle overlaps with only a constant number of other obstacles, answering the question posed by Hauser (2014). Our first contribution is modifying the reduction to 2D MCR by Erickson and LaValle (2013) to instead reduce to the 2D safe path planning problem. Our second contribution is presenting a new reduction from 3-SAT to a restricted version of safe path planning and MCR in 3D.

While the proofs in this paper use PGDF obstacles, we do not use any property that is unique to PGDF obstacles. The results essentially depend on only several properties of the model. This includes tail behavior (probability of collision being small far away from the obstacle), monotonicity (larger shadows have a larger probabilities), and the correlations between the events that nearby locations are in collision. Since these properties are rather natural, we believe the same results would apply to many other reasonable classes of realistic models of estimated obstacles.

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, Hou and Srinivasa (2017). The absence of a Markov-like property for the risk over the path has important ramifications for the complexity of the problem. In particular, the collision risk at different points along a trajectory can be highly correlated.

2 Background

Motion planning for robotics has been extensively studied in many different settings. One important high-level distinction between settings is whether the environment and state are known exactly or estimated.

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 $\mathbb{R}^3$ (Reif 1979) and $\mathbb{R}^2$ (E. Hopcroft, Joseph and Whitesides 1984; Hopcroft, Joseph and Whitesides 1982) with respect to the number of degrees of freedom of a robot (and thus dimension of its configuration space). However, while Canny's (1988) work on singly-exponential time (with respect to number of degrees of freedom) roadmaps leads to a polynomial-time algorithm when the number of degrees of freedom is fixed, 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) (LaValle and Kuffner 1999; LaValle 2006) and Probabilistic Roadmaps (PRMs) (Kavraki, Svestka, Latombe and Overmars 1996) are both practically efficient and *probabilistically complete* under some regularity conditions (Kleinbort, Solovey, Littlefield, Bekris and Halperin 2018). Given effective heuristics, graph-based planners have also proved efficient and provide *resolution completeness* (LaValle 2006).

Searching for optimal plans, as opposed to simply feasible plans, further increases the difficulty. In a classic result, Canny and Reif (1987) show that the 3-d Shortest-Path Problem is NP-hard for a simple robot in terms of the number of obstacles. This ruled out results of the form of Canny's (1991) 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 (Karaman and Frazzoli 2011) and graph-based planning algorithms are able to provide bounds on the suboptimality of their solutions (Aine, Swaminathan, Narayanan, Hwang and Likhachev 2016).

Another motion-planning problem that lacks a Markov property is the minimum constraint removal problem (MCR), where the objective is to find a path that collides with the fewest obstacles. This problem was shown to be NP-hard in Cartesian spaces of dimension 3 (Hauser 2012, 2014), and shortly later, in dimension 2 (Erickson and LaValle 2013). Eiben, Gemmell, Kanj and Youngdahl (2018) improve on these results by showing that MCR remains hard when obstacles are restricted to line segments or axis-aligned rectangles. Hauser (2014) observes that MCR is in P when obstacles are connected and non-overlapping, and he suggests that the hardness seen in MCR is caused when an obstacle intersects with $O(n)$ other obstacles. Erickson and LaValle (2013), Hauser (2014), and Eiben, Gemmell, Kanj and Youngdahl (2018) further pose the open question of whether MCR remains hard when each obstacle overlaps with only a constant number of other obstacles. We then ask an analogous question: Is safe path planning in the presence of uncertain obstacles tractable when obstacles intersect only a constant number of other obstacles? One form of this question in 3D is answered in the negative in a recent WAFR paper (Shimanuki and Axelrod 2018). These questions are stated more formally and answered in the negative in Section 3. This

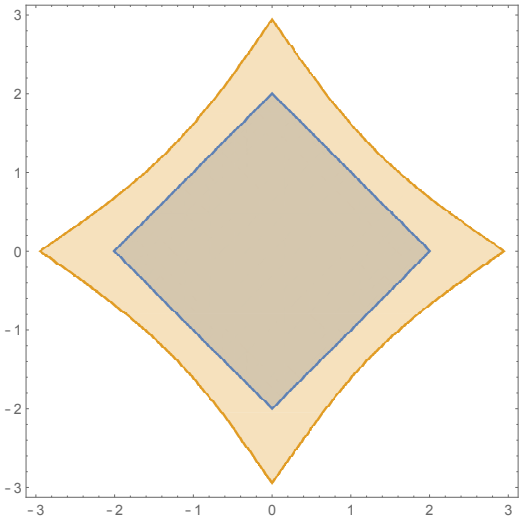

Figure 1. The orange set is a shadow of the obstacle. The blue set is the obstacle represented by the mean parameters.

paper is completely standalone and repeats all the necessary definitions. The results in two dimensions and the results with bounded obstacle overlap are new to this paper.

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 (Kaelbling and Lozano-Pérez 2013; Lee, Duan, Patil, Schulman, McCarthy, van den Berg, Goldberg and Abbeel 2013).

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 (Prentice and Roy 2007; Bry and Roy 2011) or control systems (Platt, Tedrake, Kaelbling and Lozano-Perez 2010).

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 (Ding, Pinto and Surana 2013; Feyzabadi and Carpin 2016). Other authors have used techniques from Signal Temporal Logic combined with an explicitly modeled uncertainty to generate plans that are heuristically safe (Sadigh and Kapoor 2016).

Recent work by Hauser (2014, 2012) applies an approximate minimum constraint removal algorithm to motion planning under obstacle uncertainty by randomly sampling many draws for each obstacle and finding the path that intersects with the fewest samples. With this approach, he demonstrates low runtime and error on average although with poor worst case performance.

In previous work, Axelrod, Kaelbling and Lozano-Pérez (2017, 2018) 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.

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 Preliminaries

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 $\mathbb{R}^n$. We will use e_i to denote the ith standard basis vector.

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

It is important that the obstacle distribution captures the fact that collision probabilities in different locations can be correlated. Consider the following toy example where a range sensor reports that an obstacle is 10 meters in front of the robot. At time $t = 1$ the robot drives forward 10 meters and then at $t = 2$ drives backwards 2 meters. If the robot did not

crash at time $t = 1$, it is unlikely to collide at time $t = 2$! Using a more realistic model that captures correlations allows systems to be both safer and less conservative.

In this work we restrict ourselves to polytopes with Gaussian-distributed faces (PGDFs)—a model that is able to capture such correlations (Axelrod, Kaelbling and Lozano-Pérez 2017). Under the PGDF assumption, obstacles are the intersections of halfspaces with parameters drawn from multivariate normal distributions. More formally a PGDF $O \subset \mathbb{R}^n$ is $O = \bigcap^i {\alpha_\mathbf{i}}^T \mathbf{x} \leq 0$, where $\alpha_\mathbf{i} \sim \mathcal{N}(\mu_\mathbf{i}, \mathbf{\Sigma_i})$. We can use homogeneous 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 (Axelrod, Kaelbling and Lozano-Pérez 2017).

We note that this formulation differs from the notion of "risk-zones" evaluated by Salzman and Srinivasa (2016) and Salzman, Hou and Srinivasa (2017), where the cost of a trajectory is proportional to the amount of time spent within a risk-zone. These problems share the lack of an optimal substructure—the subpaths of an optimal path are not necessarily optimal. Salzman, Hou and Srinivasa (2017) provide a generalization of Dijkstra's algorithm that finds minimum-risk plans in their domain efficiently, 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 and obstacles exist in $\mathbb{R}^d$ (d is usually 2 or 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 2. 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 3. *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 4. *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 5. *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 6. 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 by sampling-based planners such as PRM 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 (LaValle and Kuffner 1999; Karaman and Frazzoli 2011).

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

Definition 7. δ-interior (Karaman and Frazzoli 2011). *A state $\mathbf{x} \in X_{free}$ is in the δ-interior of X_{free} if the closed ball of radius δ around $\mathbf{x}$ lies entirely in X_{free}. The a set is in the δ-interior of X_{free} if every point in the set is in the δ-interior of 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 8. *For $\delta > 0$, the δ-inflation of the set $\mathbf{X}$ is the set $Y = \bigcap_{\mathbf{x} \in X} \{\mathbf{y} \mid d(\mathbf{x}, \mathbf{y}) \leq \delta\}$.*

For the proofs in this paper, the particular choice of metric d does not matter.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 9. 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 10. Probabilistically Complete $(1+\alpha)$-approximate Safe Planning Algorithm. *A planning algorithm takes a set of PGDF obstacles O, 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.*

We also consider a special case where each obstacle overlaps with only a constant number of other obstacles.

Definition 11. *Two obstacles O_i, O_j overlap if there exists any point in space that both obstacles have a significant probability of intersecting. That is, there exists $\mathbf{x} \in \mathbb{R}^d$ such that $Pr[\mathbf{x} \in O_i] \geq \epsilon$ and $Pr[\mathbf{x} \in O_j] \geq \epsilon$ for $\epsilon = \frac{\epsilon_*}{|O|}$.*

Definition 12. Probabilistically Complete κ-overlap $(1+\alpha)$-approximate Safe Planning Algorithm. *A probabilistically complete κ-overlap $(1+\alpha)$-approximate safe planning algorithm is a planning algorithm that is probabilistically complete and $(1+\alpha)$-approximate safe for cases where the number of other obstacles that each obstacle overlaps with is at most κ.*

Axelrod, Kaelbling and Lozano-Pérez (2017) provide an extension of the RRT algorithm to the probabilistic domain using the shadow approximation. The uniqueness of paths between any two vertices in a tree makes finding the optimal (restricted to the tree) path trivial. However, while the paths it generates are indeed safe, the algorithm is not probabilistically complete.

However, the following extension of the RRG algorithm is probabilistically complete (Axelrod 2017).

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 ϵ_* is the minimum ϵ for which an ϵ-safe trajectory exists.

1: G = CONSTRUCT_RRG(**s**, **t**, n)
2: **return** GRAPH_SEARCH$(G, O, \mathbf{s}, \mathbf{t})$

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 13. *A $(1+\alpha)$-approximate safe graph-search algorithm is a procedure $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is a graph, O is a set of PGDF obstacles, and $\mathbf{s}$ and $\mathbf{t}$ 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 (Axelrod 2017). *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 (Axelrod 2017). This means that, while the probability of success increases with more samples, the worst-case running time is exponential. Sampling-based motion-planning algorithms work in practice in the known environment because efficient graph-search algorithms can quickly find collision-free paths within a graph. In order for the $SAFE\_RRG$ class of 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 due to 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.

4 Results

Unfortunately, as will be shown in the remainder of this paper, Problem 1 and Problem 2 are NP-HARD with respect to $n = \Theta(|G| + |O|)$, the size of the input, even with a point robot in two dimensions and given a graph containing the solution. We note that while our results preclude an FPTAS, it does not preclude all approximation algorithms.

Theorem 2. *Unless $P = NP$, there is no $(1+\alpha)$-approximate ϵ-safe graph-search algorithm that runs in $POLY(n)$, time when restricted to graphs that embed in $\mathbb{R}^d$, $d = O(k)$ where k is the number of obstacles and $\alpha = \Theta(\frac{1}{n})$.*

We can strengthen this result to show that the minimum-risk planning problem is hard in general, that is, even when not restricted to a graph.

Theorem 3. *The $(1+\alpha)$-approximate minimum-risk planning problem is NP-hard. That is, unless $P = NP$, there is no $(1+\alpha)$-approximate Safe Planning Algorithm for $\mathbb{R}^2$ that runs in $POLY(n)$ when $\alpha = \Theta\left(\frac{1}{n^2}\right)$, even when provided a graph containing the solution.*

We show Theorem 2 and Theorem 3 by constructing a minimum risk planning problem in 2 dimensions which solves

MAXQHORNSAT (an NP-complete problem). The proof follows the outline of the MCR hardness proof presented by Erickson and LaValle (2013). The main contribution of the work in this theorem is the connection to the minimum risk planning problem and the construction of the uncertain obstacles. However, the construction is not natural in the sense that certain constructed obstacles are used to correlate collision probabilities in disparate parts of the space. Some obstacles will be split by others and there is a high degree of overlap. We also show that the problem remains hard in 3D even when each obstacle overlaps with only a constant number of other obstacles.

Theorem 4. *The κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem is NP-hard for $\kappa = O(1)$ in 3 dimensions. That is, unless $P = NP$, there is no κ-overlap $(1+\alpha)$-approximate Safe Planning Algorithm for $\mathbb{R}^3$ that runs in $POLY(n)$ when $\alpha = \Theta(\frac{1}{n^2})$, even when provided a graph containing the solution and when restricted to cases where each obstacle overlaps with at most κ other obstacles.*

Furthermore, the proof can be extended to apply to MCR as well.

Theorem 5. *The κ-overlap minimum constraint removal problem is NP-hard for $\kappa = O(1)$ in 3 dimensions.*

We show Theorems 4 and 5 by constructing a minimum risk planning problem and related MCR problem in 3 dimensions which solves 3-SAT. We believe the construction presented here is of particular value because it is very natural – the construction is simple and does not require that any obstacle be immediately adjacent to more than a constant number of other obstacles.

5 Hardness Results in $\mathbb{R}^2$

5.1 Maximum Quadratic Horn Clause Satisfiability

Maximum Quadratic Horn Clause Satisfiability (MAXQHORNSAT) is an NP-complete problem whose input is a Boolean formula given in conjunctive normal form. It consists of the intersection of many clauses, each consisting of at most two literals (i.e. is quadratic), and each clause contains at most one positive literal (i.e. is a Horn clause) (Jaumard and Simeone 1987). In other words, it is of the form $((x_0 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2) \wedge ...)$. While QHORNSAT, the decision problem of determining the satisfiability of the input formula, is in P (F. Dowling and Gallier 1984), MAXQHORNSAT, the problem of determining the maximum number of clauses that can be satisfied, is NP-hard (Jaumard and Simeone 1987). MAXQHORNSAT was used by Erickson and LaValle (2013) to show that minimum constraint removal (MCR), a similar problem, is NP-hard. Our reduction is based on that used by Erickson and LaValle and will use a similar construction modified to apply to the safe planning problem. We will consider a formula with n_v variables, n_n clauses with two negative literals, n_p clauses with one positive literal and one negative literal, and n_s clauses with only a single literal. We also define the total number of clauses $n_c = n_s + n_p + n_n$ and the total size of the problem $n = n_v + n_c$.

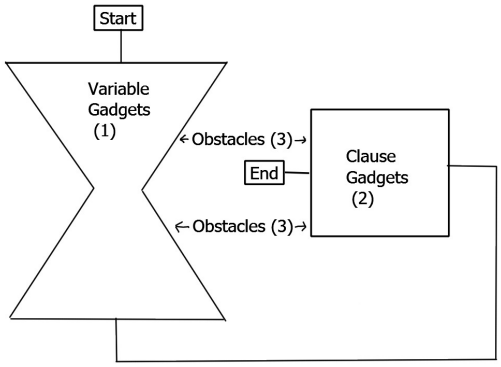

Figure 2. The spatial relationship between the different gadgets.

5.2 Proof Outline

We prove Theorem 2 using a reduction from MAXQHORNSAT. Given a MAXQHORNSAT instance, we construct an $\mathbb{R}^2$ $(1+\alpha)$-approximate minimum-risk planning problem and graph containing the solution. Our construction will have two kinds of obstacles. High-risk obstacles will induce a sufficiently high risk that any "reasonable" solution will go through the minimal number of these obstacles. Low-risk obstacles will affect the collision probability much less and will be used to count how many clauses are satisfied. The sum of the potential risk of all low-risk obstacles will be less than that of a single high-risk obstacle. This means the optimal solution will always choose to avoid a high-risk obstacle whenever possible, regardless of how many low-risk obstacles it must pass in order to do so. This creates a measurable gap between the optimal solution and the next best one. The construction can be split into three pieces. Figure 2 describes the spatial relationship between the pieces described in the following caption.

1. Construct a portion of the graph for the algorithm to assign every variable by taking either the *left* or *right* branch, corresponding to setting the value of each variable to *true* or *false*, respectively. A high-risk obstacle corresponding to each branch will ensure that the optimal path only goes down one of the branches.
2. Construct a portion of the graph for the algorithm to select a literal from each clause to try to satisfy by taking either the *left* or *right* branch, corresponding to selecting the first or second literal, respectively. Choosing a branch which corresponds to a different assignment than in the first part would result in passing by an extra high-risk obstacle.
3. Construct low-risk obstacles such that there will be additional collision risk each time the selected literal is not satisfied by the chosen variable assignment.

The solution to this planning problem can then be transformed into a solution to the original MAXQHORNSAT instance in polynomial time (via observing which nodes were visited), demonstrating that $(1+\alpha)$-approximate safe graph search is at least as hard as MAXQHORNSAT.

5.3 Obstacle Templates

Throughout this reduction we will construct a number of obstacles using a few common templates, so for simplicity of notation we will define a few parameterized types of obstacles. These obstacle templates will fall into two categories based on how much we want them to affect the cost of a trajectory: low-risk obstacles and high-risk obstacles. In figures, high-risk obstacles will be denoted in blue and low-risk obstacles will be denoted in green.

The first kind of obstacle, shown in Figure 3, is a low-risk obstacle parameterized by a line segment $(\mathbf{u}, \mathbf{v})$. It is a long, thin obstacle that runs parallel to $(\mathbf{u}, \mathbf{v})$ and has one edge with uncertain position such that there is a risk of collision with points along $(\mathbf{u}, \mathbf{v})$.

$$\widehat{C}(\mathbf{u}, \mathbf{v}, \alpha) = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ \frac{1}{|\mathbf{v}-\mathbf{u}|}(\mathbf{v}-\mathbf{u})^T(\mathbf{x}-\mathbf{u}) \geq -\epsilon_C \\ \frac{1}{|\mathbf{u}-\mathbf{v}|}(\mathbf{u}-\mathbf{v})^T(\mathbf{x}-\mathbf{v}) \geq -\epsilon_C \\ \alpha \leq \frac{1}{|\mathbf{v}-\mathbf{u}|}(R_{\frac{\pi}{2}}(\mathbf{v}-\mathbf{u}))^T(\mathbf{x}-\mathbf{u}) \leq \epsilon_C \end{array} \right\}$$

for small constant ϵ_C, and where R_θ is the 2D rotation matrix for a clockwise rotation with angle θ. Note that $\widehat{C}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can define a distribution over such obstacles

$$C(\mathbf{u}, \mathbf{v}) = \widehat{C}(\mathbf{u}, \mathbf{v}, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_C, \sigma_C^2)$$

for small constants μ_C and σ_C. It guarantees that any point within distance ϵ_C of $(\mathbf{u}, \mathbf{v})$ has a risk of collision of at most $r_c = \Phi\left(-\frac{1}{\sigma_C}(\mu_C - \epsilon_C)\right)$ and at least $r'_c = \Phi\left(-\frac{1}{\sigma_C}(\mu_C + \epsilon_C)\right)$, and any point with distance further than $z_C\epsilon_C$ from $(\mathbf{u}, \mathbf{v})$ for some constant $z_C > 1$ has a risk of collision of at most $r_f = \Phi\left(-\frac{1}{\sigma_C}(\mu_C + \frac{1}{\sqrt{2}}z_C\epsilon_C)\right)$ (lower bounded by $r'_f = 0$ because risk becomes arbitrarily small as distance from the obstacle increases), where Φ is the cumulative distribution function of the standard normal distribution. Note that given some value of ϵ_C we can set μ_C, σ_C, and z_C to achieve any desired values of r_c, r'_c, and r_f. In particular, we can make r'_c/r_c arbitrarily close to 1 by decreasing ϵ_B, and make r_f/r_c arbitrarily close to 0 by increasing z_B.

The next kind of obstacle is identical to the line-segment obstacle defined above and shown in Figure 3 except it is a high-risk obstacle, so it has a higher probability of intersecting with points near the line segment (so it can be thought of as having a higher weight in terms of affecting the risk of a path).

$$\widehat{V}(\mathbf{u}, \mathbf{v}) = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} x \in \mathbb{R}^2 \\ \frac{1}{|\mathbf{v}-\mathbf{u}|}(\mathbf{v}-\mathbf{u})^T(\mathbf{x}-\mathbf{u}) \geq -\epsilon_V \\ \frac{1}{|\mathbf{u}-\mathbf{v}|}(\mathbf{u}-\mathbf{v})^T(\mathbf{x}-\mathbf{v}) \geq -\epsilon_V \\ \alpha \leq \frac{1}{|\mathbf{v}-\mathbf{u}|}(R_{\frac{\pi}{2}}(\mathbf{v}-\mathbf{u}))^T(\mathbf{x}-\mathbf{u}) \leq \epsilon_V \\ \alpha \sim \mathcal{N}(\mu_V, \sigma_V^2) \end{array} \right\}$$

for small constant ϵ_V. Note that $\widehat{V}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can define a distribution over such obstacles

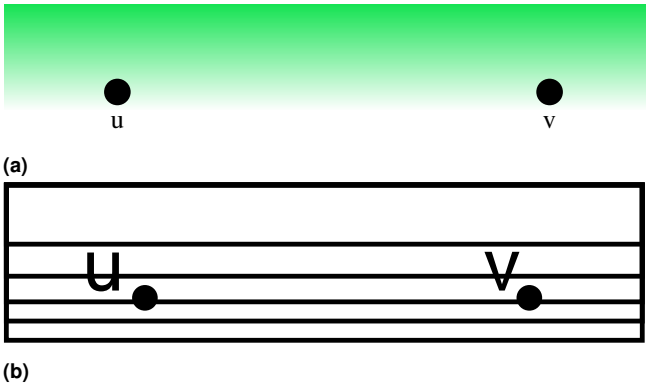

Figure 3. The illustrations of the obstacle template $C(u, v)$. Note that it is a PGDF obstacle with the height of the obstacle being the only part that is random. Figure 3a illustrates a probability density function of collision. The distance between u and the top line or between v and the top line is ϵ_C. Figure 3b illustrates several obstacles drawn from the obstacle template. Note that if u is in collision so is v and vice versa.

$$V(\mathbf{u}, \mathbf{v}) = \widehat{V}(\mathbf{u}, \mathbf{v}, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_V, \sigma_V^2)$$

for small constants μ_V and σ_V. As before, it guarantees that any point within distance ϵ_V of $(\mathbf{u}, \mathbf{v})$ has a risk of collision of at most $r_{Vc} = \Phi(-\frac{1}{\sigma_V}(\mu_V - \epsilon_V))$ and at least $r'_{Vc} = \Phi(-\frac{1}{\sigma_V}(\mu_V + \epsilon_V))$, and any point with distance further than $z_V\epsilon_V$ from $(\mathbf{u}, \mathbf{v})$ for some constant $z_V > 1$ has a risk of collision of at most $r_{Vf} = \Phi(-\frac{1}{\sigma_V}(\mu_V + \frac{1}{\sqrt{2}}z_V\epsilon_V))$ (lower bounded by $r'_{Vf} = 0$). Again, given some value of ϵ_V we can set μ_V, σ_V, and z_V to achieve any desired values of r_{Vc}, r'_{Vc}, and r_{Vf}, and so let us set the constants such that

$$\begin{aligned} r_{Vc} &= 5n_c r_c \\ r'_{Vc} &= 5n_c r'_c \\ r_{Vf} &= 5n_c r_f. \end{aligned}$$

The final type of obstacle, shown in Figure 4, is another low-risk obstacle parameterized by a single point $\mathbf{q}$ and a horizontal direction h (either 1 or -1, corresponding to right and left, respectively). It is a small obstacle that sits to the side of $\mathbf{q}$ in the direction specified by h and has one edge with uncertain position such that there is risk of collision with $\mathbf{q}$ and points nearby q.

$$\widehat{B}(\mathbf{q}, h, \alpha) = \left\{ x \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ \mathbf{e_2}^T\mathbf{q} - \epsilon_B \leq \mathbf{e_2}^T\mathbf{x} \leq \mathbf{e_2}^T\mathbf{q} + \epsilon_B \\ (\mathbf{e_1}^T\mathbf{q} + \alpha)h \leq \mathbf{e_1}^T\mathbf{x}h \leq (\mathbf{e_1}^T\mathbf{q} + \epsilon_B)h \end{array} \right\}$$

for small constant ϵ_B (also recall that $\mathbf{e_i}$ refers to the ith standard basis vector). Note that $\widehat{B}$ defines a rectangular obstacle, where the position of one edge is parameterized by α. Then we can define a distribution over such obstacles

$$B(\mathbf{q}, h) = \widehat{B}(\mathbf{q}, h, \alpha) \text{ where } \alpha \sim \mathcal{N}(\mu_B, \sigma_B^2)$$

for small constants μ_B and σ_B. It guarantees that any point within a ball of radius ϵ_B around $\mathbf{q}$ has a risk of collision of at most $r_c = \Phi\left(-\frac{1}{\sigma_B}(\mu_B - \epsilon_B)\right)$ and at least $r'_c = \Phi\left(-\frac{1}{\sigma_B}(\mu_B + \epsilon_B)\right)$, and any point outside a ball of

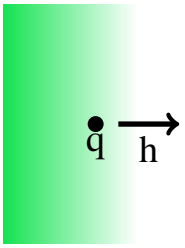

Figure 4. An example of the $B(q, h, \alpha)$ obstacle template. The distance between q and the left edge of the obstacle is ϵ_B

radius $z_B\epsilon_B$ around $\mathbf{q}$ for some constant $z_B > 1$ has a risk of collision of at most $r_f = \Phi\left(-\frac{1}{\sigma_B}(\mu_B + \frac{1}{\sqrt{2}}z_B\epsilon_B)\right)$. As before, note that given some value of ϵ_B we can set μ_B, σ_B, and z_B to achieve any desired values of r_c, r'_c, and r_f. Since these will also be low-risk obstacles, let us say that they will take on the same risk values as with the obstacles defined by C above.

5.4 Variable Gadgets

First, for each variable i in the MAXQHORNSAT problem, we construct a section of the graph where the choice of path corresponds to choosing either a true or false value of variable i. We illustrate an example graph in Figure 5 and an example trajectory through the graph in Figure 6 and formalize this below. For variable i we construct vertices

$$\begin{aligned}\mathbf{u_i^v} &= (0, 3i)\\ \mathbf{a_i^v} &= (-(n_v - i) - 1, 3i + 1)\\ \mathbf{b_i^v} &= (n_v - i + 1, 3i + 1)\\ \mathbf{v_i^v} &= (0, 3i + 2)\end{aligned}$$

and edges

$$\begin{aligned}&(\mathbf{u_i^v}, \mathbf{a_i^v})\\ &(\mathbf{u_i^v}, \mathbf{b_i^v})\\ &(\mathbf{a_i^v}, \mathbf{v_i^v})\\ &(\mathbf{b_i^v}, \mathbf{v_i^v}).\end{aligned}$$

Each pair of consecutive loops is connected by an additional edge $(\mathbf{v_i^v}, \mathbf{u_{i+1}^v})$ for all i..

This entire set of variable gadgets will be mirrored at the bottom, with the positive-negative clause gadgets (see Section 5.5.2 between them. The bottom set of variable gadgets will be needed for the negative-negative clause gadgets (see Section 5.5.3). Then for each variable i we construct a mirrored loop with vertices

$$\begin{aligned}\mathbf{u_i^{v'}} &= (0, 7n_v + 3n_p - 3i)\\ \mathbf{a_i^{v'}} &= (-(n_v - i) - 1, 7n_v + 3n_p - 3i + 1)\\ \mathbf{b_i^{v'}} &= (n_v - i + 1, 7n_v + 3n_p - 3i + 1)\\ \mathbf{v_i^{v'}} &= (0, 7n_v + 3n_p - 3i + 2)\end{aligned}$$

and edges

$$\begin{aligned}&(\mathbf{u_i^{v'}}, \mathbf{a_i^{v'}})\\ &(\mathbf{u_i^{v'}}, \mathbf{b_i^{v'}})\\ &(\mathbf{a_i^{v'}}, \mathbf{v_i^{v'}})\\ &(\mathbf{b_i^{v'}}, \mathbf{v_i^{v'}}).\end{aligned}$$

Likewise, consecutive loops are connected by an additional edge $(\mathbf{v_i^{v'}}, \mathbf{u_{i+1}^{v'}})$ for all i.

In order to ensure that the resulting path selects the same variable assignment in the top set and the mirrored set, for each variable i, we construct an obstacle that has risk of colliding with the *true* path in both versions, and another obstacle that has a risk of colliding with the *false* path in both versions. These obstacles are given by

$$\begin{aligned}&V(\mathbf{a_i^v}, \mathbf{a_i^{v'}})\\ &V(\mathbf{b_i^{v'}}, \mathbf{b_i^v}).\end{aligned}$$

Because the collision risks are correlated, selecting the same value in the bottom gadget as in the top gadget will incur no additional risk of colliding with the corresponding obstacle, but selecting a different value will incur the additional risk of colliding with the other obstacle.

5.5 Clause Gadgets

There are three types of clause gadgets, each of which will need to be handled separately: single literals, each with only a single positive or negative literal, positive-negative clauses, each with one positive literal and one negative literal, and negative-negative clauses, each with two negative literals.

5.5.1 Single Literal This first case is the simplest, as there is no choice to make about which literal to satisfy, so we do not need to add any additional components to the graph. For each single-literal clause j, let c_j^s denote the variable specified by the literal. Then we construct obstacles such that setting variable c_j^s to the opposite value in the variable assignment gadgets will incur additional risk. For a positive literal, the obstacles are constructed as

$$\begin{aligned}&C(\mathbf{a^v_{c_j^s}}, \mathbf{u^v_{c_j^s+1}})\\ &B(\mathbf{b^v_{c_j^s}}, 1)\end{aligned}$$

and for a negative literal, the obstacles are constructed as

$$\begin{aligned}&C(\mathbf{u^v_{c_j^s+1}}, \mathbf{b^v_{c_j^s}})\\ &B(\mathbf{a^v_{c_j^s}}, -1).\end{aligned}$$

Notice that each path through the variable gadget loop will incur risk of colliding with one of these obstacles, but at the end of the loop it will pass near the same obstacle that is near the branch corresponding to a satisfying assignment. Therefore selecting a variable assignment that satisfies this clause will risk collision with only one of these obstacles, whereas selecting a variable assignment that does not satisfy this clause will risk collision with both obstacles.

5.5.2 Positive and Negative Literal For each clause j with one positive literal and one negative literal, we construct a loop below the top set of variable gadgets, with vertices

$$\begin{aligned}\mathbf{u_j^p} &= (0, 4n_v + 3j)\\ \mathbf{a_j^p} &= (-1, 4n_v + 3j + 1)\\ \mathbf{b_j^p} &= (1, 4n_v + 3j + 1)\\ \mathbf{v_j^p} &= (0, 4n_v + 3j + 2)\end{aligned}$$

and edges

$$\begin{aligned}&(\mathbf{u_i^n}, \mathbf{a_i^n})\\ &(\mathbf{u_i^n}, \mathbf{b_i^n})\\ &(\mathbf{a_i^n}, \mathbf{v_i^n})\\ &(\mathbf{b_i^n}, \mathbf{v_i^n}).\end{aligned}$$

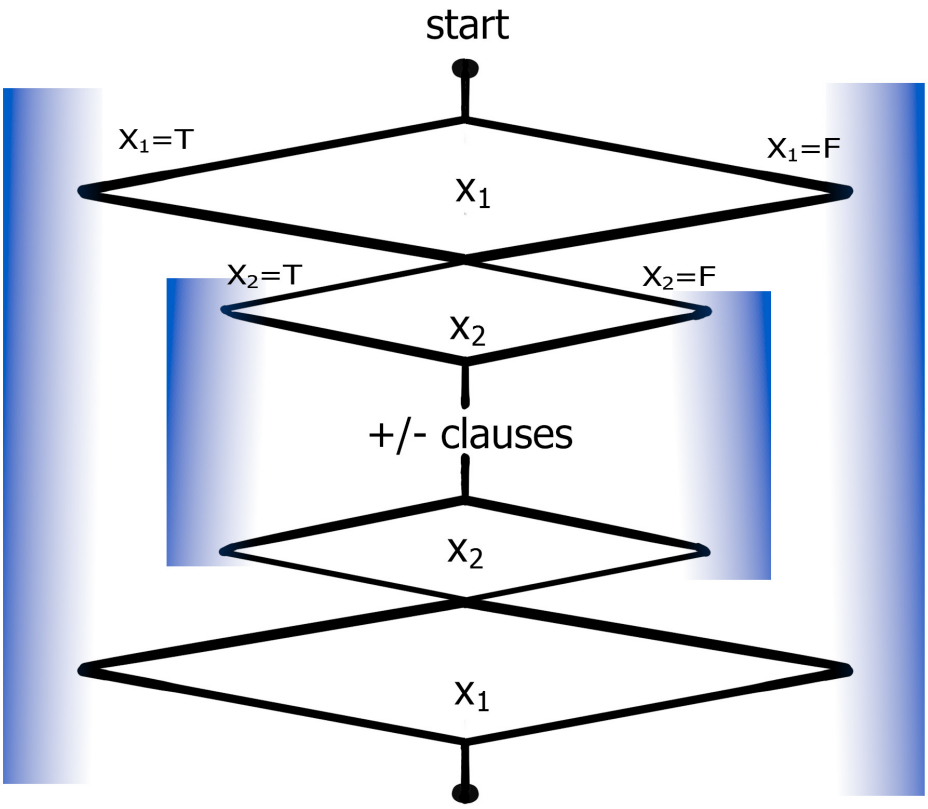

Figure 5. The variable gadget loops (solid lines) and obstacles (gradient-shaded rectangles). A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop to traverse. It must also select the same variable assignment in the mirrored gadgets at the bottom in order to avoid additional collision risk. Notice that only high-risk obstacles are used in these gadgets, as they are constructed with the V template. There are two mirrored copies of each loop, with the positive-negative clause (or $+/-$ clause) gadgets in the center. The positive-negative clause gadgets will be constructed in Section 5.5.2. Also notice how loops closer to the center have smaller width, so a straight line can be drawn from any loop to the center without intersecting any other loops.

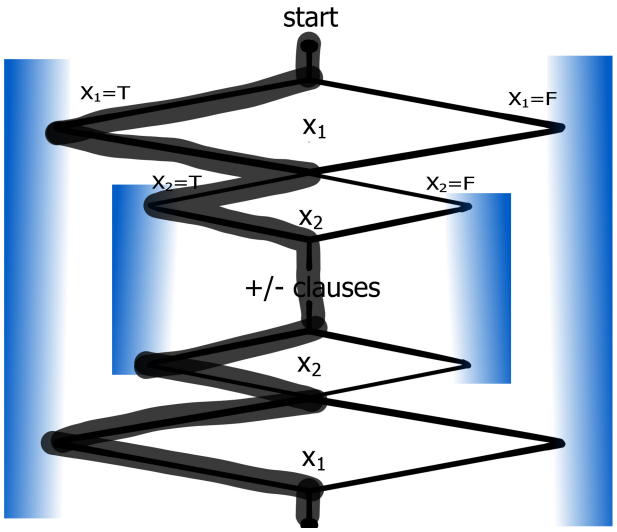

Figure 6. An example of a trajectory corresponding to $X_1 = T, X_2 = T$. Note that the top and bottom clauses match, otherwise excessive risk would be incurred.

As before, we also construct edges connecting consecutive loops $(\mathbf{v_i^n}, \mathbf{u_{i+1}^n})$ for all j, an edge from the end of the last variable gadget in the top set to the first positive-negative clause loop $(\mathbf{v_{n_v}^v}, \mathbf{u_0^n})$, and an edge from the last positive-negative clause loop to the first variable gadget in the mirrored set $(\mathbf{v_{n_n}^n}, \mathbf{u_{n_v}^{v\prime}})$. An example positive-negative clause gadget is illustrated in Figure 7.

Each loop gives a planning algorithm two paths corresponding to two literals to try to satisfy. Arbitrarily, for each such positive-negative clause j, let the left path correspond to the positive literal c_{jp}^p and the right path correspond to the negative literal c_{jn}^p. For each one, we construct two obstacles, each near one of the two paths of the corresponding variable gadget loop. However, for the obstacle near the path corresponding to assigning a value to the variable that satisfies the literal, we extend it to also be near the path for this literal in this clause gadget loop.

$$C\left(\mathbf{a_j^P}, \mathbf{a_{c_{jp}^P}^v}\right)$$
$$C\left(\mathbf{b_{c_{jn}^P}^v}, \mathbf{b_j^P}\right)$$

$$B\left(\mathbf{b_{c_{jp}^P}^v}, 1\right)$$
$$B\left(\mathbf{a_{c_{jn}^P}^v}, -1\right)$$

Therefore, taking a path corresponding to a satisfied literal risks collision with just that one obstacle, whereas a path

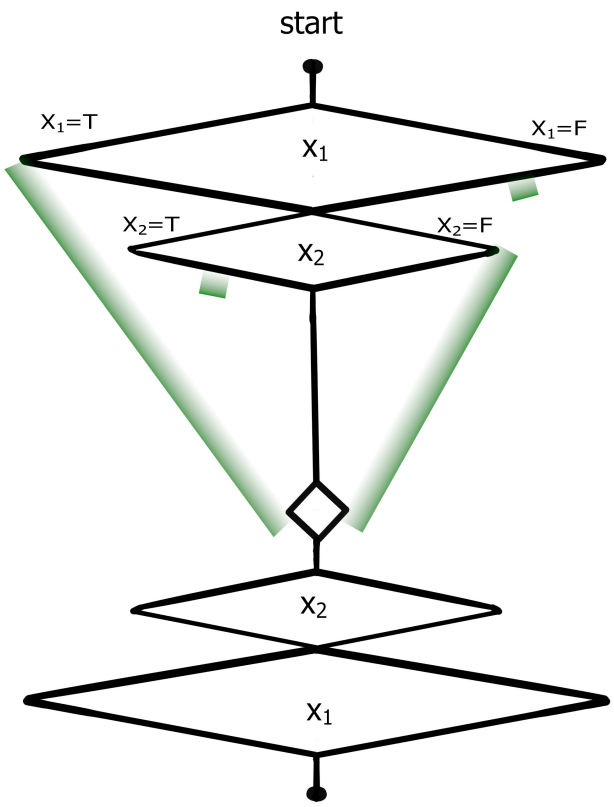

Figure 7. An example positive-negative clause gadget, for $X_1 \vee \neg X_2$, illustrating the variable gadget loops (obstacles not shown) and positive-negative clause gadget loops and obstacles. A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop, thereby risking collision with an obstacle. Then, there is no additional risk for taking the branch through the positive-negative clause gadget loop that corresponds to a satisfied literal. Notice that only low-risk obstacles are used in this gadget. For clarity, we have drawn the B-template obstacles along the edge of the branch rather than at the corner of the branch, as it is constructed in the template.

corresponding to a non-satisfied literal risks collision with both obstacles.

5.5.3 Two Negative Literals For each clause j with two negative literals, we construct a loop to the side of the other gadgets, with vertices

$$\begin{aligned}\mathbf{u_j^n} &= (2n_v + 3n_s - 3j, 4n_v)\\ \mathbf{a_j^n} &= (2n_v + 3n_s - 3j - 1, 4n_v + 1)\\ \mathbf{b_j^n} &= (2n_v + 3n_s - 3j - 1, 4n_v - 1)\\ \mathbf{v_j^n} &= (2n_v + 3n_s - 3j - 2, 4n_v)\end{aligned}$$

and edges

$$\begin{aligned}&(\mathbf{u_i^p}, \mathbf{a_i^p})\\ &(\mathbf{u_i^p}, \mathbf{b_i^p})\\ &(\mathbf{a_i^p}, \mathbf{v_i^p})\\ &(\mathbf{b_i^p}, \mathbf{v_i^p}).\end{aligned}$$

We construct edges connecting consecutive loops $(\mathbf{v_i^p}, \mathbf{u_{i+1}^p})$ for all j. We also construct an intermediate vertex $\mathbf{d} = (2n_v + 3n_s, 7n_v + 3n_p + 2)$ to connect the last variable gadget in the mirrored set to the first negative-negative clause loop with two edges $(\mathbf{v_0^v}, \mathbf{d})$ and $(\mathbf{d}, \mathbf{u_0^p})$. An example negative-negative clause gadget is illustrated in Figure 8.

Each loop gives a planning algorithm two paths corresponding to two literals to try to satisfy. For each such negative-negative clause j, let c_{j1}^n denote the variable specified by the first literal and c_{j2}^n denote the variable specified by the second literal. For each literal, we construct two obstacles, each near one of the two paths of the corresponding variable gadget loop (for the literal corresponding to the top path of the negative-negative clause loop, we will use the top set of variable gadgets, and for the literal corresponding to the bottom path of the negative-negative clause loop, we will use the mirrored set of variable gadgets). However, for the obstacle near the path corresponding to a negative assignment assigning (which satisfies the literal), we extend it to also be near the path for this literal in this clause gadget loop.

$$\begin{aligned}&C(\mathbf{b_{c_{j1}^n}^v}, \mathbf{a_j^n})\\ &C(\mathbf{b_j^n}, \mathbf{a_{c_{j2}^n}^{v\prime}})\end{aligned}$$

$$\begin{aligned}&B(\mathbf{a_{c_{j1}^n}^v}, -1)\\ &B(\mathbf{a_{c_{j2}^n}^{v\prime}}, -1)\end{aligned}$$

Therefore, taking a path corresponding to a satisfied literal risks collision with just that one obstacle, whereas a path corresponding to a non-satisfied literal risks collision with both obstacles.

5.6 Path Risk Encoding MAXQHORNSAT

We define the $(1+\alpha)$-approximate safe graph search problem as $\phi(G, O, \mathbf{s}, \mathbf{t})$, where G is the set of vertices and edges constructed in the variable and clause gadgets above, O is the set of obstacles constructed in the variable and clause gadgets above, and

$$\begin{aligned}\mathbf{s} &= \mathbf{u_0^v}\\ \mathbf{t} &= \mathbf{v_{n_s}^n}.\end{aligned}$$

G and O were constructed such that there will exist a gap between the risk of a path corresponding to an optimal assignment and a path corresponding to a suboptimal assignment. Each variable gadget loop in the top set passes near a single high-risk obstacle, and there will exist a path through the mirrored set that does not pass near any additional high-risk obstacles, and passing near an additional high-risk obstacle incurs more risk than passing near every low-risk obstacle, so a minimum-risk path will pass near exactly n_v high-risk obstacles. Any path must also pass near at least one obstacle for each clause gadget, and it will pass near an additional obstacle for each unsatisfied clause, so a minimum-risk path will pass near $(n_s + n_p + n_n + \delta)$ low-risk obstacles, where δ is the minimum number of unsatisfied clauses. Recall that there exists a gap between the induced risk close to an obstacle r_c and far away from the obstacle r_f (or r_{Vc} and r_{Vf} in the case of high-risk obstacles). Then a path corresponding to an optimal solution to the MAXQHORNSAT problem will incur risk at most

$$n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta) r_c + (3n_c - \delta) r_f$$

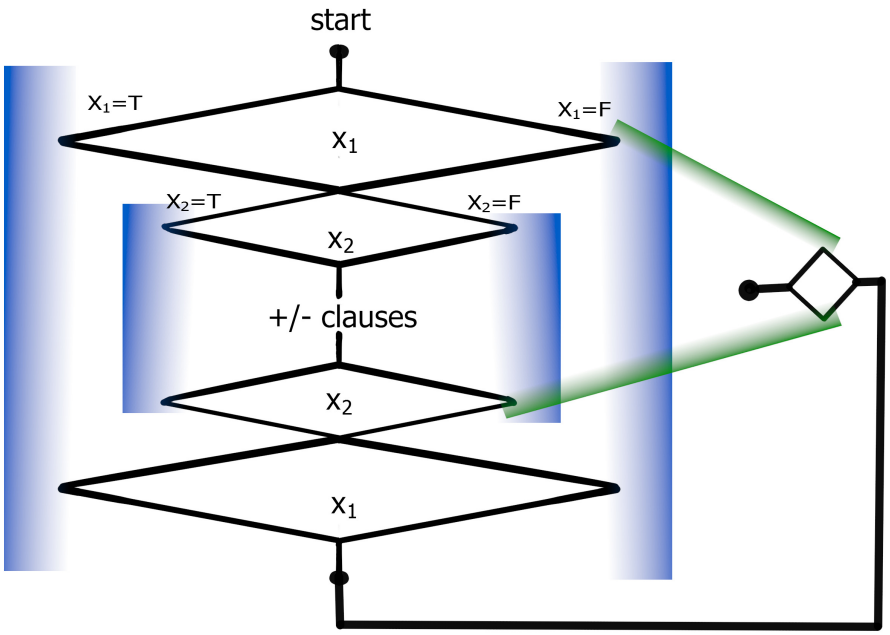

Figure 8. An example negative-negative clause gadget, for $\neg X_1 \vee \neg X_2$, illustrating the variable gadget loops and obstacles (blue) and negative-negative clause gadget loops and obstacles (green). A path assigns a *true* or *false* value to each variable by selecting a branch through the variable gadget loop. It must also select the same variable assignment in the mirrored gadgets at the bottom in order to avoid additional collision risk. Then, there is no additional risk for taking the branch through the negative-negative clause gadget loop that corresponds to a satisfied literal.

and a path corresponding to a suboptimal solution to the MAXQHORNSAT problem will incur risk at least

$$n_v r'_{Vc} + n_v r'_{Vf} + (n_c + \delta + 1)r'_c + (3n_c - \delta - 1)r'_f.$$

This allows us to compute a lower bound on the ratio of the risks between a suboptimal solution and an optimal solution as

$$\begin{aligned}
&\frac{n_v r'_{Vc} + n_v r'_{Vf} + (n_c + \delta + 1)r'_c + (3n_c - \delta - 1)r'_f}{n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
\geq &\frac{n_v r'_{Vc} + (n_c + \delta + 1)r'_c}{n_v r_{Vc} + n_v r_{Vf} + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
= &\frac{5n_c n_v r'_c + (n_c + \delta + 1)r'_c}{5n_c n_v r_c + 5n_c n_v r_f + (n_c + \delta)r_c + (3n_c - \delta)r_f} \\
\geq &\left(\frac{r'_c}{r_c \beta} + \frac{r'_c}{20 n_c n_v r_c}\right) + \frac{r'_c}{20 n_c n_v r_c} \\
\geq &1 + \theta\left(\frac{1}{n^2}\right)
\end{aligned}$$

if we set the obstacle constants such that $20n_c n_v + \beta \geq 20n_c n_v \beta$, where $\beta = 1 + 3\frac{r_f}{r_c}$

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 $1 + \Theta(\frac{1}{n^2})$ can also solve MAXQHORNSAT with polynomial overhead. □

5.7 Hardness of Continuous Planning Problem

We prove Theorem 3 by extending the above reduction to still apply even without the graph restriction (thereby reducing to $(1+\alpha)$-approximate minimum-risk planning). Given the graph G and set of obstacles O constructed above, we surround G with additional obstacles such that a path cannot deviate from G by more than $2\epsilon_P$, for some small constant ϵ_P. We divide the space of $\mathbb{R}^2$ into a grid with cells of size $\epsilon_P \times \epsilon_P$ and construct a square obstacle in every cell that does not intersect with the graph and is within a window containing all of the gadgets.

$$C_{ij} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^2 \\ i\epsilon_P \leq \mathbf{e_1}^T \mathbf{x} \leq (i+1)\epsilon_P \\ j\epsilon_P \leq \mathbf{e_2}^T \mathbf{x} \leq (j+1)\epsilon_P \end{array} \right\}$$

for all $i, j \in \mathbb{Z}$

$$O' = O \cup \left\{ C_{ij} \;\middle|\; \begin{array}{l} i, j \in \mathbb{Z} \\ -\frac{1}{\epsilon_P} 2n_v \leq i, j \leq \frac{1}{\epsilon_P}(8n_v + 4n_p) \\ C_{ij} \cap G = \emptyset \end{array} \right\}$$

Note that there are a polynomial number of such obstacles, so if ϵ_P is sufficiently small, the solution to the resulting $(1+\alpha)$-approximate minimum-risk planning problem or lack thereof is approximately equivalent to that for the original $(1+\alpha)$-approximate safe graph search problem, and so $(1+\alpha)$-approximate minimum-risk planning is also NP-hard. □

6 Hardness with Constraints on Overlapping Obstacles

Hauser (2014) observed that his 3D MCR reduction as well as the 2D MCR reduction presented by Erickson and LaValle (2013) required that each obstacle be allowed to overlap with

$O(n)$ other obstacles. Similarly, we note that the reduction we present above for 2D motion planning under obstacle uncertainty also requires that each obstacle overlap with $O(n)$ other obstacles. This is a relatively unnatural problem instance, as most real-world problems will not have this degree of overlap. In this section, we show that the problem remains hard in 3D even when each obstacle only overlaps with a constant number of other obstacles.

6.1 3SAT

3SAT is an NP-complete problem that is commonly used to prove the hardness of other problems (Sipser 1996). 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 $(x_{j_u} \vee \neg x_{j_v} \vee x_{j_w})$.

6.2 Proof Outline

We prove Theorem 4 using a reduction from 3SAT. Given a 3SAT instance, we construct a κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem as follows.

1. Construct a set of variable assignment layers where each branch corresponds to a variable assignment.
2. Construct a set of clause layers where each branch corresponds to selecting a literal to satisfy.
3. For each variable assignment layer, construct a pair of obstacles for each variable that will encode whether the variable is set to *true* or *false*. There will be additional collision risk for a path that selects different values in each variable assignment layer, as well as for a path that selects a literal that is not satisfied by the value selected in the preceding variable assignment layer.

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

6.3 Proof

6.3.1 Variable Gadgets

First, we will construct a variable assignment layer for each clause j. A variable assignment layer consists of two PGDF obstacles for each variable i in the 3SAT problem.

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

$$o = \left\{ \mathbf{x} \;\middle|\; {\alpha_\mathbf{i}}^T \mathbf{x} \leq 1, \alpha_\mathbf{i} \sim \mathcal{N}(\mu_\mathbf{i}, \mathbf{\Sigma_i}) \right\}.$$

For obstacle i, the *true* obstacle will be defined as the intersection of

$$\begin{aligned} {\alpha_\mathbf{i}}^T \mathbf{x} &\leq 1 \\ i \leq {\mathbf{e_2}}^T \mathbf{x} &\leq i+1 \\ 2j - \frac{3}{2} < {\mathbf{e_3}}^T \mathbf{x} &\leq 2j + \frac{3}{2} \end{aligned}$$

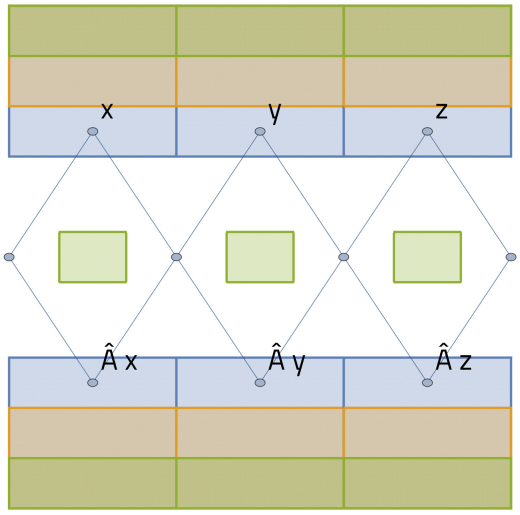

Figure 9. A path through this gadget must go near either the *true* or *false* obstacle for each variable, thereby selecting a variable assignment.

where $\alpha_\mathbf{i} \sim \mathcal{N}(2\mathbf{e_1}, \mathbf{e_1}{\mathbf{e_1}}^T)$. The "negative" obstacle will similarly be defined with

$$\begin{aligned} {\beta_\mathbf{i}}^T \mathbf{x} &\leq 1 \\ i \leq {\mathbf{e_2}}^T \mathbf{x} &\leq i+1 \\ 2j - \frac{3}{2} < {\mathbf{e_3}}^T \mathbf{x} &\leq 2j + \frac{3}{2} \end{aligned}$$

where $\beta_\mathbf{i} \sim \mathcal{N}(-2\mathbf{e_1}, \mathbf{e_1}{\mathbf{e_1}}^T)$.

Intuitively the covariance $\mathbf{e_1}{\mathbf{e_1}}^T$ means that $\alpha_\mathbf{i}$ 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. Also note that each obstacle overlaps with the corresponding obstacle in the layer below and the layer above, which we will later show to be important in ensuring that variable assignments are consistent across layers.

Then we will construct the variable assignment graph, as illustrated in figure 9. Said formally, indexing over the variable with index i, we embed nodes in locations

$$\begin{aligned} &(2j-1)\mathbf{e_3} + i\mathbf{e_2} \\ &(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1} \\ &(2j-1)\mathbf{e_3} + (i+1)\mathbf{e_2}. \end{aligned}$$

We then draw edges from $(2j-1)\mathbf{e_3} + i\mathbf{e_2}$ to both of $(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1}$, and from both of $(2j-1)\mathbf{e_3} + (i+\frac{1}{2})\mathbf{e_2} \pm \mathbf{e_1}$ to $(2j-1)\mathbf{e_3} + (i+1)\mathbf{e_2}$.

6.3.2 Clause Gadgets

For each clause j we will construct an additional graph "layer" in between consecutive pairs of variable layers that lets the algorithm choose which literal to satisfy, as illustrated in figure 10.

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

$$2j\mathbf{e_3}, 2j\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3}$$

drawing edges between consecutive nodes, and letting '$\pm$' represent '-' if x_{j_u} is given in negated form and '+' otherwise. Then construct nodes at

$$2j\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$
$$2j\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_u + \frac{1}{2}\right)\mathbf{e_2}$$

(the first and last were already constructed previously), drawing edges between consecutive nodes, and similarly setting '$\pm$' based on the negation of literal x_{j_v}. Then construct nodes at

$$2j\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$
$$2j\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{1}{3}\right)\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2} \pm \mathbf{e_1}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_w + \frac{1}{2}\right)\mathbf{e_2}$$
$$\left(2j + \frac{2}{3}\right)\mathbf{e_3} + \left(j_v + \frac{1}{2}\right)\mathbf{e_2}$$

(the first and last were already constructed previously), drawing edges between consecutive nodes, and similarly setting '$\pm$' 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.

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.

6.3.3 Full Reduction Now we combine the variable and clause gadgets, as seen in figure 11. As in Section 5.7, we construct a grid of deterministic obstacles to force the path to remain near the graph. Given the graph G and set of obstacles O constructed above, we surround G with additional obstacles such that a path cannot deviate from G by more than $2\epsilon_P$, for some small constant ϵ_P. We divide the space of $\mathbb{R}^{\not\vDash}$ into a grid with cells of size $\epsilon_P \times \epsilon_P \times \epsilon_P$ and construct a cubic obstacle in every cell that does not intersect with the graph and is within a window containing all of the gadgets.

$$C_{\mathbf{z}} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{R}^3 \\ z_i\epsilon_P \le x_i \le (z_i+1)\epsilon_P \;\forall\, i \in \{1,2,3\} \end{array} \right\}$$
for all $\mathbf{z} \in \mathbb{Z}^3$

$$O' = O \cup \left\{ C_{\mathbf{z}} \;\middle|\; \begin{array}{l} z \in \mathbb{Z}^3 \\ -\frac{4}{\epsilon_P} \le \frac{1}{\sqrt{3}}|\mathbf{z}| \le \frac{1}{\epsilon_P}(k+m+4) \\ C_{\mathbf{z}} \cap G = \emptyset \end{array} \right\}$$

6.3.4 Path Risk Encoding 3SAT This 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 the PGDF obstacle model as well as most reasonable alternative formulations, there exists a gap between the induced risk close to the obstacle and far away from the obstacle. In particular, there is some r_c that lower-bounds the risk computed from the shadow approximation for the closer points and r_f that upper-bounds the computed risk for the further point. A path through the variable assignment portion of the graph will go near km obstacles for the k variable assignments it makes, each repeated m times. Then it will be "close" to km obstacles and "far" from the other km obstacles. Therefore, it will incur risk $kmr_c + kmr_f$.

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 $(km+1)r_c + (km-1)r_f$. This allows us to compute a lower bound on ratio between the two risks:

$$\begin{aligned} &\frac{(km+1)r_c + (km-1)r_f}{kmr_c + kmr_f} \\ =&\frac{kmr_c + kmr_f + r_c - r_f}{kmr_c + kmr_f} \\ =&1 + \frac{r_c - r_f}{kmr_c + kmr_f} \\ =&1 + \Theta\left(\frac{1}{km}\right). \end{aligned}$$

We note that each obstacle only overlaps with at most a constant number of other obstacles. In particular, each obstacle will overlap with the two corresponding obstacles in the layer above and the layer below, as well as the constant number of deterministic obstacles forming dividers between layers.

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

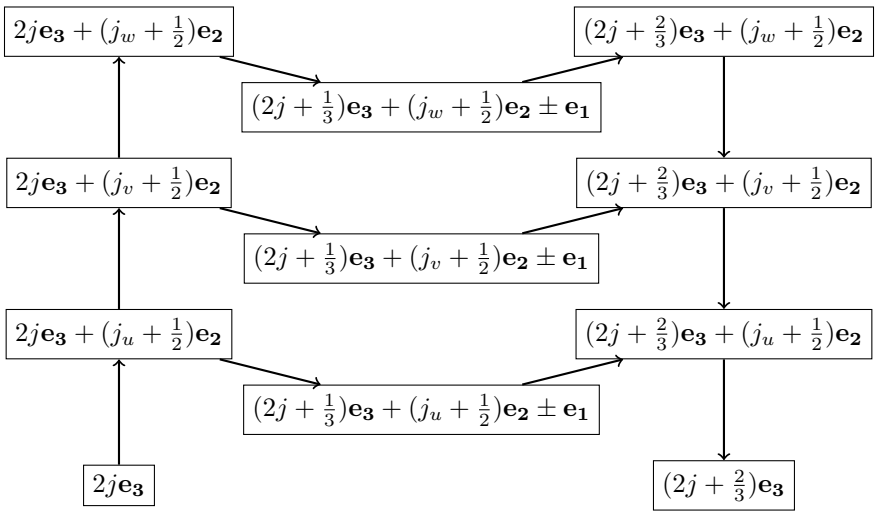

Figure 10. A path through this gadget must select one of three paths to go through, each going near the obstacle for the corresponding literal.

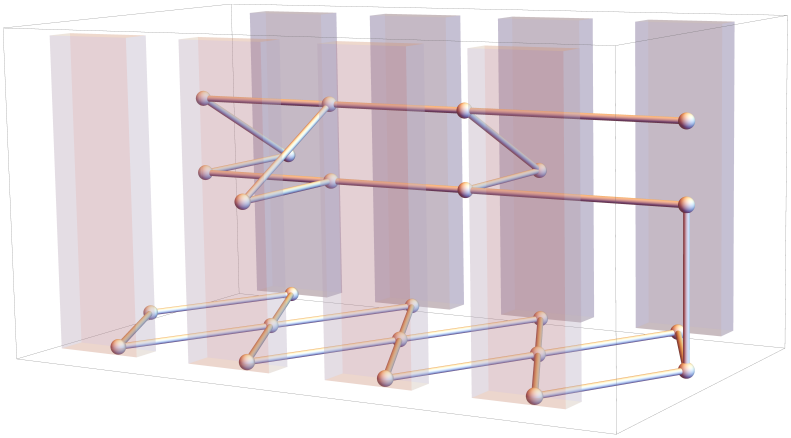

Figure 11. The bottom layer is the first variable assignment layer. The top layer is the first clause gadget. There would usually be many more clause gadgets stacked on top with additional variable gadget layers in between.

that can approximate the κ-overlap $(1+\alpha)$-approximate minimum-risk planning problem (regardless of whether a graph containing the solution is provided) to a factor better than $1 + \Theta\left(\frac{1}{km}\right)$ can also solve 3SAT with polynomial overhead. □

6.3.5 Extension for Minimum Constraint Removal This reduction can also be extended to apply to the Minimum Constraint Removal (MCR) Problem, proving Theorem 5 and answering the question posed by Hauser (2014). We replace each uncertain obstacle in the ϵ-safe planning problem with an MCR obstacle covering the r_c-shadow of the obstacle. As before, a path corresponding to a satisfying assignment will collide with km obstacles, whereas a path corresponding to a nonsatisfying assignment must collide with at least $km + 1$ obstacles. Then the ratio between the costs of a nonsatisfying path and a satisfying path is lower bounded by

$$\frac{km+1}{km} = 1 + \Theta\left(\frac{1}{km}\right).$$

Therefore, MCR remains NP-hard even when each obstacle is connected and intersects no more than κ obstacles (κ held constant).

7 Conclusions and future work

We have shown that the minimum-risk planning problem on graphs is NP-hard, even in two dimensions. Furthermore, the fact that it remains hard after restriction to a small graph indicates that exact algorithms reducing to a graph search are likely to be impractical in the uncertain domain. However, 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 potential for algorithms that demonstrate fixed parameter tractability.

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 (Axelrod, Kaelbling and Lozano-Pérez 2017). 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 (1991) roadmap algorithm), no exact algorithm is known. Finally, we hope that this work is not taken as evidence that the planning problem with uncertain obstacles is impossible. After all, the robotics community has a long history of finding heuristic methods that work well on problems NP-hard in the worst case. We hope this work highlights why worst case is difficult and leads to the identification of structure that makes real world problems solvable in practice, perhaps even with provable guarantees.

Acknowledgements

We would like to thank Tomas Lozano Perez and Leslie Pack Kaelbling for their advice and guidance throughout this project. We are also grateful for our discussions with Gustavo Goretkin about the connections between the safe planning problem and MCR.

Declaration of conflicting interests

The Authors declare that there is no conflict of interest.

Funding

We gratefully acknowledge support from the Thomas and Stacey Siebel Foundation; from NSF Fellowship grant DGE-1656518; from NSF grants CCF-1763299, CCF-1763311, 1420316, 1523767, and 1723381; from AFOSR grant FA9550-17-1-0165; from ONR grant N00014-18-1-2847; from Honda Research; and from Draper Laboratory. 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

Aine S, Swaminathan S, Narayanan V, Hwang V and Likhachev M (2016) Multi-heuristic a*. *The International Journal of Robotics Research* 35(1-3): 224–243.

Axelrod B (2017) *Algorithms for Safe Robot Navigation*. Master's Thesis, Massachusetts Institute of Technology.

Axelrod B, Kaelbling L and Lozano-Pérez T (2017) Provably safe robot navigation with obstacle uncertainty. *Robotics Science and Systems* 13. URL http://lis.csail.mit.edu/pubs/axelrod-rss-17.pdf.

Axelrod B, Kaelbling LP and Lozano-Pérez T (2018) Provably safe robot navigation with obstacle uncertainty. *The International Journal of Robotics Research* .

Bry A and Roy N (2011) Rapidly-exploring random belief trees for motion planning under uncertainty. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 723–730. URL http://ieeexplore.ieee.org/abstract/document/5980508/.

Canny J (1988) Some algebraic and geometric computations in pspace. In: *Proceedings of the Annual ACM Symposium on Theory of Computing*. pp. 460–467.

Canny J and Reif J (1987) New lower bound techniques for robot motion planning problems. In: *Foundations of Computer Science*. ISBN 0-8186-0807-2, pp. 49 – 60.

Canny JF (1991) Computing roadmaps of general semi-algebraic sets. In: Mattson HF, Mora T and Rao TRN (eds.) *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-38436-6, pp. 94–107.

Cassandra A, Kaelbling L and Kurien J (1996) Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In: *International Conference on Intelligent Robots and Systems*, volume 12. ISBN 0-7803-3213-X, pp. 963 – 972 vol.2.

Ding XC, Pinto A and Surana A (2013) Strategic planning under uncertainties via constrained markov decision processes. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, pp. 4568–4575. URL http://ieeexplore.ieee.org/document/6631226/.

E Hopcroft J, Joseph D and Whitesides S (1984) Movement problems for 2-dimensional linkages. In: *SIAM Journal on Computing*, volume 13. pp. 610–629.

Eiben E, Gemmell J, Kanj I and Youngdahl A (2018) Improved results for minimum constraint removal. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16615.

Erickson L and LaValle S (2013) A simple, but np-hard, motion planning problem. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

F Dowling W and Gallier J (1984) Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming* 1: 267–284. DOI:10.1016/0743-1066(84)90014-1.

Feyzabadi S and Carpin S (2016) Multi-objective planning with multiple high level task specifications. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 5483–5490. URL http://ieeexplore.ieee.org/document/7487762/.

Hauser K (2012) The minimum constraint removal problem with three robotics applications. In: *Workshop on the Algorithmic Foundations of Robotics*.

Hauser K (2014) The minimum constraint removal problem with three robotics applications. In: *The International Journal of Robotics Research*, volume 33.

Hopcroft J, Joseph D and Whitesides S (1982) On the movement of robot arms in 2-dimensional bounded regions. In: *SIAM Journal on Computing*, volume 14. pp. 280 – 289.

Jaumard B and Simeone B (1987) On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters* 26: 1–4. DOI:10.1016/0020-0190(87)90028-7.

Kaelbling LP and Lozano-Pérez T (2013) Integrated task and motion planning in belief space. *The International Journal of Robotics Research* URL http://journals.sagepub.com/doi/abs/10.1177/0278364913484072.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7): 846–894.

Kavraki LE, Svestka P, Latombe JC and Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and*

Automation 12(4): 566–580.

Kleinbort M, Solovey K, Littlefield Z, Bekris KE and Halperin D (2018) Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters* 4(2): x–xvi.

Kurniawati H, Hsu D and Lee WS (2008) Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In: *Robotics: Science and Systems*.

LaValle SM (2006) *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

LaValle SM and Kuffner JJ (1999) Randomized kinodynamic planning. In: *ICRA*.

Lee A, Duan Y, Patil S, Schulman J, McCarthy Z, van den Berg J, Goldberg K and Abbeel P (2013) 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, pp. 5660–5667. URL http://ieeexplore.ieee.org/document/6697176/.

Papadimitriou CH and Tsitsiklis JN (1987) The complexity of markov decision processes. *Mathematics of operations research* 12(3): 441–450.

Platt R, Tedrake R, Kaelbling L and Lozano-Perez T (2010) Belief space planning assuming maximum likelihood observations. In: *Proceedings of Robotics: Science and Systems*. Zaragoza, Spain. DOI:10.15607/RSS.2010.VI.037. URL http://www.roboticsproceedings.org/rss06/p37.html.

Prentice S and Roy N (2007) 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.

Reif J (1979) Complexity of the mover's problem and generalizations. In: *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*. pp. 421–427.

Sadigh D and Kapoor A (2016) Safe control under uncertainty with probabilistic signal temporal logic. In: *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan. DOI:10.15607/RSS.2016.XII.017. URL http://www.roboticsproceedings.org/rss12/p17.html.

Salzman O, Hou B and Srinivasa S (2017) Efficient motion planning for problems lacking optimal substructure. *arXiv preprint arXiv:1703.02582* .

Salzman O and Srinivasa S (2016) Open problem on risk-aware planning in the plane. *arXiv preprint arXiv:1612.05101* .

Shimanuki L and Axelrod B (2018) Hardness of 3d motion planning under obstacle uncertainty. In: *International Workshop on the Algorithmic Foundations of Robotics*. Springer, pp. 852–867.

Sipser M (1996) *Introduction to the Theory of Computation*. 1st edition. International Thomson Publishing. ISBN 053494728X.

Somani A, Ye N, Hsu D and Sun Lee W (2013) Despot: Online pomdp planning with regularization. In: *Advances in Neural Information Processing Systems*, volume 58.