UNIVERSITY OF CALIFORNIA SAN DIEGO

Motion Planning Algorithms for Safety and Quantum Computing Efficiency

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Engineering Sciences (Aerospace Engineering)

by

Paul Lathrop

Committee in charge:

Sonia Martínez, Chair
Beth Boardman
Jorge Cortés
Sylvia Herbert
Michael Yip

2023

The Dissertation of Paul Lathrop is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude for Professor Sonia Martínez and Dr. Beth Boardman, for their support as research mentors, for technical guidance, and for the freedom to pursue an unconventional topic. I would also like to thank my committee members Professor Jorge Cortés, Professor Sylvia Herbert, and Professor Michael Yip for their support and feedback. Additionally, I would like to thank Troy Harden for his support as a research mentor during the beginning of my degree.

I am grateful for the members, past and present, of the UCSD Multi-Agent Robotics Lab and other colleagues of the Contextual Robotics Institute for lunchtime discussions, coffee walks, and office company, especially those who put up with me both in lab and in sport. I would like to acknowledge the E-3 group at LANL for support over the summers, and in London, and I acknowledge Los Alamos National Laboratory for the financial support during my degree and the opportunity to take part in three lab internships.

Lastly, I would like to express my deepest appreciation to my family: Dr. Lathrop, Dr. Lathrop, Dr. Lathrop, and John for their unwavering support and technical assistance during my Ph.D, to my friends: the UCSD and UMD triathlon teams, TE, and ZAPA for ensuring that I stay active in the great outdoors, and to my very good friend and colleague Dr. Alves and my chief of staff Calypso (feline) for their encouragement and strength.

Chapter 3, in full, is a reprint of the material "Distributionally Safe Path Planning: Wasserstein Safe RRT" as it appears in *IEEE Robotics and Automation Letters*. Lathrop, Paul, Beth Boardman, and Sonia Martínez. *IEEE Robotics and Automation Letters* 7.1, pp. 430-437, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 4, or portion thereof, is currently being prepared for publication of the material. Lathrop, Paul, Beth Boardman, and Sonia Martínez. "Mobile Robot Behavioral Classification with Uncertainty for Adaptive Safety." In preparation, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of material "Quantum Search Approaches to Sampling-

Based Motion Planning" as it appears in *IEEE Access*. Lathrop, Paul, Beth Boardman, and Sonia Martínez. *IEEE Access*, vol. 11, pp. 89506-89519, 2023, doi: 10.1109/ACCESS.2023.3307316. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, has been submitted for publication as "Parallel Quantum Rapidly-Exploring Random Trees" as it may appear in *IEEE Access*. Lathrop, Paul, Beth Boardman, and Sonia Martínez. The dissertation author was the primary investigator and author of this paper.

This dissertation is approved for public release under LA-UR-23-33571.

VITA

2019	B.S. in Aerospace Engineering, University of Maryland College Park
2021	M.S. in Engineering Sciences (Aerospace Engineering), University of California San Diego
2023	Ph.D. in Engineering Sciences (Aerospace Engineering), University of California San Diego

PUBLICATIONS

P. Lathrop, B. Boardman, and S. Martínez. "Distributionally Safe Path Planning: Wasserstein Safe RRT." *IEEE Robotics and Automation Letters* 7.1, pp. 430-437, 2021.

P. Lathrop. "Distributionally Safe Path Planning: Wasserstein Safe RRT." in 2023 *IEEE International Conference on Robotics and Automation*, London, UK.

P. Lathrop, B. Boardman, and S. Martínez. "Mobile Robot Behavioral Classification with Uncertainty for Adaptive Safety." In preparation, 2023.

P. Lathrop, B. Boardman and S. Martínez, "Quantum Search Approaches to Sampling-Based Motion Planning." *IEEE Access*, vol. 11, pp. 89506-89519, 2023.

P. Lathrop, B. Boardman, and S. Martínez, "Parallel Quantum Rapidly-Exploring Random Trees." *IEEE Access*, under review, 2023.

ABSTRACT OF THE DISSERTATION

Motion Planning Algorithms for Safety and Quantum Computing Efficiency

by

Paul Lathrop

Doctor of Philosophy in Engineering Sciences (Aerospace Engineering)

University of California San Diego, 2023

Sonia Martínez, Chair

Motion planning remains a fundamental problem in robotics. Sampling-based algorithms use randomization to allow efficient solutions to this complex problem. As mobile robots and autonomous vehicles become more prevalent in everyday life, motion planning must be applied to increasingly challenging scenarios. Safety has become a paramount concern in motion planning for ensuring robotic applications enrich human lives. To date, many motion planning techniques to increase safety in the face of uncertain and dynamic environments have been developed. This dissertation first addresses distributional safety of Rapidly-Exploring Random Trees (RRT) through our algorithm W-Safe RRT. To acknowledge distributional uncertainty and poor modeling, W-Safe RRT uses the Wasserstein metric to provide a probabilistic bound on the

distributional distance between a robot and obstacles. Human-interpretable environmental agent classification allows online safety margin adaptation. We propose and analyze an integrating-region method for online classification that increases actor labeling accuracy based on behavioral feature values when compared to state of the art methods. The method performs class assignments based on *local* maximum likelihood in a created behavioral feature-space, allowing a notion of classification uncertainty.

Model-based methods with safety guarantees can quickly become computationally intractable, especially with multiple agents, higher dimensions, and plentiful unknowns. Sampling-based algorithms have been parallelized for computation with multi-core computers and GPUs. We consider the use of quantum algorithms and computers for sampling-based motion planning for the first time. Quantum computing performs operations on superpositions of states and can solve certain problems much more efficiently than classical computers, but introduces previously unseen challenges. With Quantum-RRT, we recast the motion planning problem into a database-search structure and use Quantum Amplitude Amplification to find reachable states in the database with a quadratic performance increase over classical methods. We address two error sources with this method: quantum measurement and quantum oracle errors. We then extend this method to Parallel Quantum-RRT, which uses a manager-worker architecture with multiple parallel quantum workers to increase database search efficiency. We compare algorithm architectures and characterize probabilities of multiple workers finding solutions. Lastly, we test in simulation the quantum algorithms against classical versions in a wide variety of scenarios, concluding that a similar parallelization improvement is to be found in the quantum case as was found in the parallelization of classical RRT.

Chapter 1

Introduction

1.1 Introduction

MOTION planning at its heart concerns two things: motion, the act of changing position, and planning, the creation of a proposal to achieve desired motion. Motion planning remains one of the fundamental problems in robotics. Planning in online situations requires quickly combining a basic planning strategy with information about the robot, its dynamics, and the environment, in order to create a path from a starting location to a goal location. Several different broad strategies were created to address the challenges of robotic motion planning.

Exact roadmap methods use the connectivity of the planning environment to find paths from a start to a goal location. Graph search methods rely on discretizing the environment and then searching the resultant graph for a connected solution. The computationally intensive nature of graph searching in higher dimensions led to the development of the main subject of this dissertation, sampling-based planning algorithms, which use randomization to provide faster solutions. Sampling-based planners are ideal for non-convex high-dimensional environments, where they can quickly connect random state samples to provide a collision-free path from an initial to a goal configuration.

Several challenges facing such planners that have inspired the research in this dissertation are the need for collision free paths in dynamic environments, the need for paths generated in real-time scenarios, and the need for motion plans that reduce metrics such as distance, environmental

cost, or computational resources. Improving the safety, efficiency, and optimality of sampling-based planners has been at the forefront of efforts within the field of robotic motion planning. The research within this dissertation addresses the use of probabilistic uncertainty for safe motion planning, strategies for quickly identifying and responding to adversarial environmental behavior, and finally the use of quantum computers to more efficiently solve these problems.

1.2 Literature Review

In the field of robotic motion planning, many early efforts in deterministic path planning focused either on exact roadmap methods or cell decomposition and graph-search methods. Exact roadmap methods, such as visibility graphs [2] and Voronoi diagrams [3] use the connectivity of the configuration space for planning. Cell decomposition and graph-search methods, such as Dijkstra's Algorithm [4] or A* [5], first subdivide the configuration space into cells and subsequently perform a graph search. Discretization and graph-search methods suffer in high dimensional configuration spaces and with moving obstacles, which led to the development of sampling-based planners. An extended overview of the general field of motion planning can be found at the textbook [6], and an overview of motion planning for robotics and automated vehicles can be found at [7].

The purpose of this section is to introduce, at a high level, sampling-based motion planning and the themes, questions, and algorithms which are studied, extended, used throughout this dissertation. Sampling-based motion planning is the branch of motion planning which concerns algorithms that generally employ random samples in the quest to create robotic paths. This approach allows algorithms to find faster solutions to more difficult problems, such as high dimensional spaces or environments with complex dynamics. However, the approach suffers from the point of view of completeness. Completeness is the notion that if there is a solution, then an algorithm will find it. Sampling-based planners are probabilistically complete, meaning if there is a solution, as runtime goes to infinity, then the algorithm is guaranteed to find the

solution.

Two sampling-based planning algorithms, the Probabilistic Roadmap (PRM) [8] algorithm and the Rapidly-Exploring Random Tree (RRT) [9] algorithm form the core algorithms for using random sampling to create roadmaps and trees, respectively, through an environment. A major difference between the two is that PRMs are typically used for multi-shot path planning, where roadmaps are reused (and reconnected to) to find paths through environments, and RRTs are typically used for single-shot path planning, where a tree from one state is branched through the environment to find a goal state, then discarded. Pseudocode for PRM and RRT are included in Section 2.2.

At a glance, the RRT algorithm consists of a few relatively simple, looped steps. After initializing the tree at the start node, the algorithm produces a random state and then attempts to connect that state to the existing tree. This sample-connect process is repeated until the goal is found. The basic structure remains consistent across all sampling-based planning algorithms, and the exact details of the steps are changed, or additional steps added, to produce desired results.

Sampling-based planning algorithms have been modified and extended to address shortfalls, encourage performance, and optimize a host of different heuristics and metrics. Paths produced with RRT have a tendency to appear jagged due to the random sampling, and without post-processing the returned path can vary significantly from the solution that optimizes metrics such as path length. This effect led to several important modifications to RRT: guided sampling, and near-vertices and rewiring. Guided sampling approaches change what portion of the planning space is selected for random sampling. Methods such as Dynamic-Domain RRT [10], which proposes an adaptive sampling framework based on visibility regions of tree nodes, medial axis sampling [11], and Gaussian obstacle sampling [12] can encourage the expansion of trees in parts of the configuration space which can lead to more optimal path and search behavior [13].

The RRT* [14] (pronounced RRT 'Star') algorithm introduced two processes, called near-vertices and rewiring, which can guarantee the asymptotic optimality of the algorithm. Asymptotic optimality is the quality that as the runtime of the algorithm goes to infinity, the

returned path will approach some idealized path which optimizes some pre-defined metric or cost. A large number of works have set about defining this metric in different ways [15, 16], what it can mean in different contexts, under what conditions asymptotic optimality is met [17], and how quickly optimality can be achieved [18]. Rewiring consists of re-testing connections of a selected node to a batch of the nearest existing vertices, and re-making the connection as the connection which minimizes the running cost to the selected node.

Of special interest in sampling-based planning are the concepts of safety and uncertainty. Motion planning with safety can loosely be thought of as planning with constraint satisfaction, and sampling-based planners such as RRT are inherently able to handle and check constraints in the 'connect' portion of the sample-connect process loop [19]. Early efforts to create safety-enforcing constraints involved obstacle collision constraints, but more elaborate and abstract notions have arisen in response to questions and applications regarding moving obstacles [20], uncertain obstacles [21], disturbances, and errors in modeling [22]. Uncertainty representations acknowledge lack of state certainty, and some works, such as Chance Constrained RRT (CC-RRT) [23] and CC-RRT* [24], use the notion of chance constraints to reason probabilistically about safety constraint satisfaction. The field of safety in robotic motion planning is an active open research area, especially with respect to uncertain environments and dynamics, system control guarantees, and incorporating sensing and perception. Work in this dissertation addresses safety through the additional acknowledgment of *model* uncertainty (on top of state uncertainty) for chance constraint creation, followed by how class uncertainty can be incorporated in the scenario of obstacle classification.

Constraints can also take the form of kinematic constraints, such as rigid body constraints for robotic arms, and kinodynamic constraints, which include velocity and acceleration limits, and trajectory limits that satisfy known dynamic models for a robot or vehicle. Kinodynamic RRT* [25]) introduced a branch of RRTs which use the connection procedure to check dynamic reachability. This ensures that new nodes in the tree are within the set of states reachable by a robot, within a certain time and set of controls, when subjected to the dynamics of the

robot. Ensuring notions such as reachability in the connect phase of the planner is referred to as local planning [26], and often forms the most computationally challenging portion of RRT variants [27, 28], which has led to efforts to optimize runtime characteristics of sampling-based planners. Local planning under dynamic, safety, and uncertainty constraints can present severe runtime issues, especially in higher dimensional spaces. In this dissertation we explore an entirely new direction for optimizing sampling-based motion planning runtime: the use of quantum computing. In a process called quantum parallelism [29], quantum computers are able to perform simultaneous calculations on superpositions of states, and we exploit this feature to parallelize the local planning step of motion planners. A deeper introduction into quantum computing as it applies to this dissertation can be found in Section 2.3.

One such non-quantum method to increase tree creation speed and efficiency comes in the form of writing parallel sampling-based architectures for computation on GPUs and multi-core computers [28]. PRMs have been described as 'embarrassingly parallel' [30], and even simple strategies such as OR-Parallel RRT [31], which involves multiple concurrent and independent trees trying to find the solution, show promise in increasing the speed with which solutions are found. More nuanced strategies such as scheduler-processor schemes [32] allow a central processor to assign specific work to processors. This led to such strategies as collaboratively creating trees with spatial decomposition [33] and functional decomposition, such as a manager-worker paradigm where the manager accesses the tree and workers perform assigned work [34,35]. In contrast, in Distributed RRT local copies of the tree are maintained by each worker and tree expansions are communicated via a messaging scheme [36]. In this dissertation we apply the manager-worker paradigm (due to results expressed in [37]) to quantum computers performing RRT tree creation in an effort to further increase efficiency (combined with quantum computing).

An extended overview of sampling-based motion planning can be found at [27], and readers are encouraged to consult the textbook [6] for a thorough discussion of motion planning. Introductions to quantum computing from a circuits perspective can be found at [38] and [39]. For a comprehensive review on the state of the art of quantum computing as it applies to robotics,

especially with respect to open research questions, can be found at [40]. Specific literature reviews for each chapter can be found within each chapter.

1.3 Contributions and Organization

In Chapter 2, we introduce the notation used throughout this dissertation, general algorithm definitions for PRM and RRT for use as an introduction as they are extended in further chapters, and a brief introduction to quantum computing as it applies to the final two chapters.

In Chapter 3, we propose the algorithm Wasserstein-Safe Rapidly Exploring Random Trees (W-Safe RRT) which uses probabilistic representations of uncertainty to create motion plans with a guarantee that a robot's probability distribution and obstacle probability distributions are a certain Wasserstein distance apart, given some environmental sampling strategy. This algorithm enables safer path planning by explicitly acknowledging that environmental uncertainty distributions can be incorrectly modeled, maintained, and sampled.

In Chapter 4, we consider a scenario where a robot is planning in an environment with several classes of actors with varying behaviors, and we propose a strategy for allowing uncertainty in the observation-based classification of environmental actors. We propose an integral-calculating uncertainty method, which assigns the local maximum likelihood class to actors during online planning. We analyze actor data in a feature space, which condenses actor behavior from raw observed trajectories to features, or high level behavioral qualities. This strategy enables behavioral classifications that acknowledge the uncertain nature of environmental observations and allows safer path planning through more likely classification labels.

In Chapter 5, we provide an algorithmic basis to solving sampling-based motion planning problems using quantum computers, with the algorithm Quantum Rapidly Exploring Random Trees (q-RRT). We recast the problem into a database search structure by considering databases of possible states to connect dynamically to the path planning tree. We then search this database for admittable solutions using Quantum Amplitude Amplification, which leverages quantum

mechanical properties to perform fast and efficient parallel computations. We show simulated examples of the quadratic run-time advantage of this strategy compared to classical RRT.

In Chapter 6, we extend Quantum RRT for parallel computation on multiple quantum devices with the algorithm Parallel Quantum Rapidly Exploring Random Trees (Pq-RRT). We create a parallel formulation where a classical computing manager creates a single database and assigns work to multiple quantum computing workers in order to produce multiple simultaneous solutions from a single database for addition to a planning tree. This allows more efficient database search that helps to bridge the problem of quantum measurement collapse, where information about multiple solutions is lost when a single quantum device measures a superposition to find a single solution. We compare Pq-RRT, q-RRT, a parallel RRT, and RRT for run-time, efficiency, and their abilities to expand quickly and solve specific problems.

Chapter 2

Preliminaries

2.1 Notation

We introduce here the general notation that are used throughout this dissertation. Let $d, p \in \mathbb{N}$, and let $\mathbb{N}_N$ be the set of natural numbers from one to N. Let $\mathbb{R}^d$ be the d-dimensional real space with $x \in \mathbb{R}^d$ denoting a vector in it. We denote the p-norm on $\mathbb{R}^d$ as $\|x\|_p = \sqrt[p]{\|{}^1x\|^p + \cdots + \|{}^dx\|^p}$, $x \in \mathbb{R}^d$, with the Euclidean norm as $\|\cdot\| \equiv \|\cdot\|_2$ and ${}^1x, \ldots, {}^dx$ as the components of x. The diameter of a set S in the p-norm is denoted as $\mathrm{diam}_p(S) := \sup_{x,y\in S} \|x-y\|_p$. Let $\bar{S} = \mathbb{R}^d \setminus S$ be the complement of set $S \subseteq \mathbb{R}^d$.

Let the Borel σ-algebra on $\mathbb{R}^d$ be denoted as $\mathcal{B}(\mathbb{R}^d)$, and the set of probability distributions on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ as $\mathcal{P}(\mathbb{R}^d) \equiv \mathcal{P}$. In what follows, we identify probability distributions $\mathbb{P} \in \mathcal{P}$ with the measures μ used to generate them. Throughout this dissertation, the empirical distribution built on a set of samples $\hat{\xi}_i$, $i \in \{1, \ldots, N\}$ is $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} \delta_{\hat{\xi}_i}$, where δ is the Dirac delta function. Let $X \in \mathbb{R}^d$ be a random vector variable. We denote the probability that $X \in S$, for $S \subseteq \mathbb{R}^d$, as $\mathbb{P}(X \in S) \in [0,1]$, for a given $\mathbb{P} \in \mathcal{P}$.

Let $\mathcal{N}(y, \Sigma)$ refer to the Normal distribution with mean $y \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$. Let $|z\rangle$ refer to the quantum state represented by the qubit z. Let $\mathbb{E}$ be the expectation operator. Let $\mathbb{U}(C)$ be the uniform distribution over C.

2.2 Algorithm Definitions

In this section we define the two standard sampling-based motion planning algorithms that we extend and analyze in this dissertation, Probabilistic Roadmaps and Rapidly-Exploring Random Trees. The Probabilistic Roadmap algorithm is used to create a roadmap through an environment. Pseudocode for PRM is included in Alg. 1. The PRM algorithm consists of two

Algorithm. 1 Probabilistic Road-maps (PRM)

Input: Obstacles
Output: Roadmap R

```
1:  while Roadmap R not initiated do
2:      x_rand = random point
3:      Check x_rand against obstacles
4:      if x_rand collision free then
5:          Init roadmap R with x_rand
6:      end if
7:  end while
8:  while Still building R do
9:      x_rand = random point
10:     L_adj = list of all nodes in R within distance d of x_rand
11:     for All x_adj in L_adj do
12:         Check path from x_rand to x_adj for collisions
13:         if path collision free then
14:             Add x_rand to R with edge to x_adj
15:         end if
16:     end for
17:     if R large enough then
18:         Stop building R
19:     end if
20: end while
21: Return R
```

loops, one to initiate the roadmap, and the other to build the roadmap. Between lines 1 and 7, the algorithm attempts to place a single obstacle free point to serve as the initial node. A random point x_{rand} is chosen, checked against known obstacles, and used to initiate the roadmap R if it is obstacle free. In the second loop, between lines 8 and 20, a more in depth path check is performed to admit additional nodes to R. After a random point is drawn, a list of candidate

existing nodes is found on line 10. In the above formulation, all nodes within a certain distance d of x_{rand} are chosen as candidate nodes for connecting x_{rand} to R. The ensuing loop between lines 11 and 16 checks each candidate node x_{adj} within L_{adj} as to whether there is a collision free path between x_{rand} and x_{adj}. If the path is collision free, then on line 14, x_{rand} is added to R by an edge to x_{adj}. For subsequent x_{adj} that also have a collision free path, just the edge is added to R. After R is large enough, it is returned.

After the roadmap is returned, when a new path planning query arises in the form of a starting node x_{start} and goal node x_{goal}, first the two nodes are connected to the roadmap. Then, the roadmap (with x_{start} and x_{goal}) can be queried with a graph search algorithm such as Dijkstra's Algorithm [4] or A* [5].

Algorithm. 2 Rapidly Exploring Random Trees (RRT)

Input: $x_{\text{start}}, x_{\text{goal}}$
Output: Path γ

```
1:  Init tree T with root at x_start and no parent
2:  while goal not found do
3:      x_rand = random point
4:      x_parent = closest parent of x_rand in T
5:      Redefine x_rand to be near enough to x_parent
6:      Check path from x_rand to x_parent for collisions
7:      if path collision free then
8:          Add x_rand to T with parent x_parent
9:      end if
10:     if x_rand and x_goal close enough then
11:         Mark goal as found
12:     end if
13: end while
14: Unwrap path γ from T
15: Return γ
```

The Rapidly-Exploring Random Tree (RRT) [9] algorithm takes as inputs a starting state x_{start} and a goal state x_{goal}, as it is generally a single-shot planner. It outputs a path γ from x_{start} to x_{goal}. The root node of the tree T is set at x_{start}, then the loop from lines 2 to 13 populates the tree until the goal is found. First, a random node x_{rand} is chosen and the nearest parent x_{parent} (to

x_{rand}) is found. If the path from x_{rand} to x_{parent} is collision free, x_{rand} is added to T with parent x_{parent}. In some formulations, multiple possible parents within a distance d of x_{rand} can be put in a list and iterated, similar to the PRM description in Alg. 1. If that is the case, the first (closest) parent is selected as the chosen parent, and the rest discarded so the tree structure is maintained (different from PRM). Typically, RRT can be terminated by adding a node within some region of x_{goal}, with the implicit assumption that the last x_{rand} and x_{goal} can be connected (with x_{rand} serving as the parent). After that, the path is unwrapped by tracing from x_{goal} to x_{start} via parents (then reversed). Lastly, the path γ can be returned. The RRT algorithm rapidly grows a tree toward free space, as the planner is more likely to randomly sample from larger Voronoi regions. Pseudocode for RRT is included in Alg. 2.

2.3 Quantum Computing Introduction

To aid in understanding of the final two chapters of this dissertation, in this section we introduce quantum computing basics, how quantum algorithms can be used to solve motion planning problems, and an explanation of Quantum Amplitude Amplification (QAA), the quantum algorithm we have selected for use. An extended introduction can be found at [38] and [39]. A summary of pertinent information from these sources is presented below.

Instead of encoding information classically in bits of either 0 or 1 states, quantum computers encode information in basic units called quantum bits or *qubits* [41]. A qubit is given as the superposition of two basis quantum states, $|0\rangle$ and $|1\rangle$. The latter two correspond to the two physical states 0 and 1, or the classical computing states. However, a qubit $|\Psi\rangle$ can exist in a superposition of $|0\rangle$ and $|1\rangle$, of the form $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$. We say that $\{|0\rangle, |1\rangle\}$ defines a basis of quantum states. In this way, a qubit can be given as a weighted superposition of the basis states, meaning it can be thought of as physically existing simultaneously in many states at once.

Quantum states in a superposition maintain probability amplitudes α and β, or the relative

likelihoods of measuring a particular state of the superposition. The measurement process of the quantum state involves the collapse of the quantum state $|\Psi\rangle$ to a base state $\{|0\rangle, |1\rangle\}$ according to the measurement probabilities α^2 and β^2 (also known as the Born rule [42]).

Qubits are placed in superpositions using the Walsh-Hadamard transform, a multidimensional Fourier operator which forms the quantum Hadamard gate [43]. This is a unitary operator mapping a quantum state to an equal superposition of all qubit states. Since the Hadamard gate creates the superposition, it is key to simultaneous computation.

Quantum algorithms use superposition as a tool to perform fast and efficient parallel computations on superpositions of states. A unitary transformation will act on all basis vectors of the quantum state and can simultaneously evaluate many values of a function $f(x)$ for many inputs x in a process known as quantum parallelism [29]. Although the probability amplitudes α and β of the system cannot be known explicitly [44], quantum algorithms use quantum parallelism to manipulate the amplitudes. Planning algorithms written for quantum methods can be thought of as fully parallelized. To accomplish motion planning, we intend to use quantum algorithms in the following general way:

1. Identify an oracle function (or quantum black box) to check for configuration feasibility or path reachability.
2. Construct a database of possible paths or points.
3. Encode the database as a qubit register (i.e. a system comprising multiple qubits).
4. Create a superposition across all database elements.
5. Repeatedly apply QAA to increase the probability amplitude of the correct database elements.
6. Measure the qubit to return a single element.
7. Check the measured answer and repeat the process.

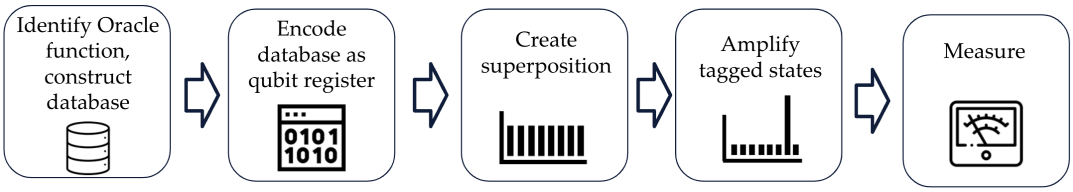

Figure 2.1. A visualization of the high level workflow, inspired by [1], which we will adapt to the problems of motion planning. Icons from Flaticon.

This succinct description on how to apply QAA to a specific problem is shown in Fig. 2.1 and is inspired by the work [1], which applies quantum algorithms to financial analysis. We will use a Boolean oracle function to evaluate the feasibility of a path and later, the reachability to a state. In the context of quantum computing, a Boolean oracle function, represents a black-box function that is handed inputs and produces a Boolean, or True/False, output [45]. They are widely used in quantum algorithms to study complexity and runtime comparisons [46]. We refer to feasibility as the connectivity of a pair of points, and provable reachability refers to whether, given a set of dynamics and a type of controller, we can steer the system from a state to another with a reachable obstacle-free path. Further discussion on applications and local planning is included in Chapters 5 and 6. The actual state and environmental parameters are not required to be explicitly known, but the Boolean output of this oracle is assumed to be available.

Quantum Amplitude Amplification uses a Boolean oracle function $\mathcal{X}$ to increase the probability of measuring a good state Ψ. Ψ is defined in terms of being a good state if and only if $\mathcal{X}(\Psi) = 1$. The oracle function can be described as a Phase Oracle, and it is a unitary operator which shifts all qubit inputs by a constant phase. The QAA operator Q then performs a pair of probability amplitude reflections based upon the output of the oracle. This results in the probability amplitude magnification of good states and decrease of bad states. The QAA precise definition and mechanism of action can be found at [39], page 56. In what follows, the QAA operator using oracle $\mathcal{X}$ is denoted as $Q(\mathcal{X})$.

We will take advantage of the fact that QAA can perform a quantum search on a size-N unordered database for an oracle-tagged item in $\mathcal{O}(N^{1/2})$ oracle calls, whereas classical search

algorithms require $\mathcal{O}(N)$ calls [39].

Quantum oracles are often used to study *relativized* complexities of algorithms in terms of oracle calls, or how many times an oracle must be queried to solve a problem. In Chapters 5 and 6 we use a quantum oracle that returns a Boolean value representing reachability (according to some dynamics) from inputs encoding robot state, environment, and dynamics. The action within the oracle is constructed from quantum circuits and gates, all of which need to be reversible, according to the reversibility postulate of quantum mechanics. Reversibility, when applied to quantum computing gates, refers to the ability to exactly construct the input given the output of any gate [47]. The action of the oracle seems to violate this postulate, as a Boolean decision cannot be used to construct the state, environment, and dynamic model input to the oracle. This problem is studied in quantum information theory, and a solution which allows quantum gate universality (construction of all classical computing algorithms and gates) involves the use of ancilla bits, uncomputation, and the Toffoli gate [48], which has been physically realized in superconducting circuits [49] and trapped ions [50]. In short, in this dissertation we do not directly create the action of the quantum oracle and instead of reasoning at the quantum circuit level, we reason at the algorithmic architecture level. However, it is generally possible to create quantum oracles which can solve arbitrary classical computing queries. A detailed discussion into quantum information theory can be found at [51] and [52].

Chapter 3

Wasserstein Safe RRT: Distributionally Safe Path Planning

3.1 Introduction

Safety is an essential requirement on the operation of autonomous systems in close proximity to humans, from self-driving cars to more complex human-robot teams. Our ability to guarantee safety is directly related to how we understand and manage uncertainty [53], from the epistemic kind —on environment, obstacle, and system modeling errors—to the aleatoric type —in the form of random noises.

In motion planning, the probabilistic modeling of uncertainty has enabled the integration of sensing, motion, and environmental uncertainty in a principled manner. This results in less conservative plans at the expense of higher computational and time complexity costs. However, distributional errors and lack of knowledge of the underlying probability distributions can nullify efforts to leverage this approach and guarantee safety. To address this, this work accounts for both state and obstacle distributional modeling errors through the Wasserstein metric and ambiguity sets. Using these, we approximate uncertain distributions with a quantification of the error for finite samples, and create a distributionally robust path planning algorithm for any finite sample set. The algorithm is applicable to vehicles navigating obstacles in a physical environment and extends to more abstract state spaces.

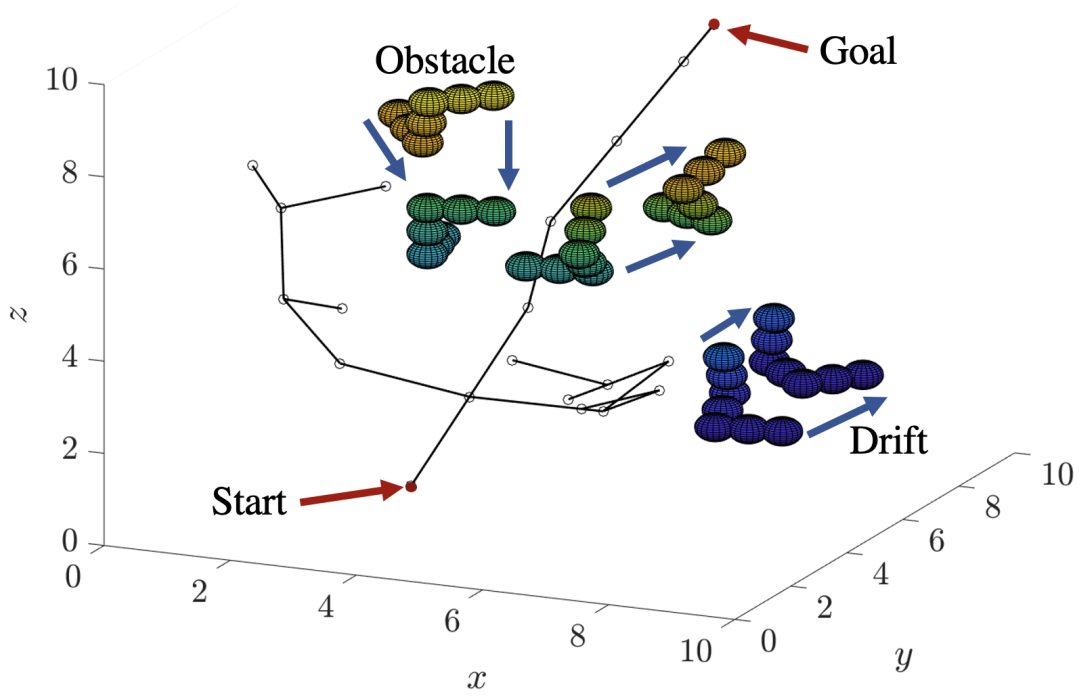

Figure 3.1. Search tree created by W-Safe RRT to path plan in a multi-nonconvex-obstacle 3D configuration space with drift and rotational uncertainty present.

3.2 Literature Review

Various existing path planners have been designed to manage uncertainty. Chance-constrained optimization formulations in sampling-based planning [54–56] have taken the limelight because they can limit the probability of collision in a straightforward way provided that models are accurate. Two such algorithms, Chance Constrained Rapidly Exploring Random Trees (CC-RRT) [23], and Particle Chance Constrained RRT (PCC-RRT) [57], use continuous and discrete distributions, respectively, to model the vehicle state and limit the probability of collision with known obstacles. For tractability, CC-RRT uses continuous Gaussian distributions to represent the state, while PCC-RRT uses sample approximations that allow non-Gaussian models. Both algorithms return a probabilistic guarantee on safety for a given uncertainty model of the state (approximated in [57]) and fail to consider distributional modeling errors or unmodeled locational obstacle uncertainty. The work [58] characterizes path planning probabilistic

completeness under uncertainty, but fails to consider distributional uncertainty and associated probabilistic guarantees.

Wasserstein-based ambiguity sets can handle these types of distributional errors [59], while retaining tractability in associated optimization problems. Thus, they have been applied in robust policy optimization against random disturbances [60], and distributed decision-making [61]. The work [62] characterizes the evolution of ambiguity sets under dynamic linear-systems transformations, which becomes useful in uncertainty quantification. Furthermore, chance constraints on moment-based ambiguity sets have been used in optimal motion planning [22], but require known, convex obstacles. Distributionally-robust chance constrained optimization for convex functions in the decision variables is studied in [63]. Here, for tractability, the distributional problem is inner-approximated by means of a conditional value-at-risk (CVaR)-like function. Returning trajectories with bounded-risk guarantees and probabilistic locations of obstacles is studied in [64], but these require an obstacle model and obstacle movement model. Furthermore, this work considers bounding risk contours through an obstacle environment and not dynamic and distributional modeling errors.

Beyond this, risk-aware motion planning algorithms have been developed [65,66] to limit the risk of collision as modeled by the CVaR metric. While [65] does not consider distributional uncertainty, [66] employs Wasserstein ambiguity sets around random obstacle drift vectors to constrain a distributionally robust model predictive control problem. However, the robot state is assumed to be known with certainty, and obstacles are modeled as non-rotating, convex polytopes, with random shifts that also belong to a known convex polytope. This results in large, poorly scaling optimization problems that are solved approximately via McCormick relaxations. Additionally, no results are shown employing the claimed sample guarantees.

In this chapter, we consider a motion planning problem where we account for uncertainty in not-necessarily-convex obstacles and in vehicle motion through a novel sampling-based planner. W-Safe RRT is based upon checking possible paths and states in pre-planning rather than solving directly constrained optimal control problems online. Because our algorithm is a pre-

planner, we create and maintain environmental and state models. Unlike previous Gaussian and particle-based planners, our algorithm makes use of the Wasserstein metric measured between individual vehicle-obstacle states to create a distributionally safe and probabilistically complete random path planner. By exploiting recent results on the number of samples required to obtain probabilistic guarantees over compact spaces, we can probabilistically guarantee with a precise bound that a resulting vehicle path is greater than a certain Wasserstein distance (W-distance) away from a moving obstacle model. We create a minimum encompassing ball algorithm inspired by PCC-RRT [57] that receives the same information with the same assumptions as W-Safe RRT for comparison purposes. We show that W-Safe RRT outperforms the comparison algorithm in simple convex obstacle and rotating non-convex multi-obstacle environments at the expense of computation time.

3.3 Contributions

We propose a Wasserstein metric-based random path planning algorithm. Wasserstein Safe RRT (W-Safe RRT) provides finite-sample probabilistic guarantees on the safety of a returned path in an uncertain obstacle environment. Vehicle and obstacle states are modeled as distributions based upon state and model observations. We define limits on distributional sampling error so the Wasserstein distance between a vehicle state distribution and obstacle distributions can be bounded. This enables the algorithm to return safe paths with a confidence bound through combining finite sampling error bounds with calculations of the Wasserstein distance between discrete distributions. W-Safe RRT is compared against a baseline minimum encompassing ball algorithm, which ensures balls that minimally encompass discrete state and obstacle distributions do not overlap. The improved performance is verified in a 3D environment using single, multi, and rotating non-convex obstacle cases, with and without forced obstacle error in adversarial directions, showing that W-Safe RRT can handle poorly modeled complex environments.

3.4 Problem Formulation

We consider a pre-planning mobile robot problem where we assume obstacle and state dynamics to allow dynamic path planning. Before the algorithm is run, in an observational period, we observe the environment and robot details to create $\mathcal{M}_r$ and $\mathcal{M}_O$, the assumed dynamic robot and obstacle models, and the initial states. To cope with incorrect modeling and unknown disturbances, we will make use of Wasserstein ambiguity sets, see Section 3.5. The robot has state $x \in \mathbb{R}^d$ which is constrained within a compact configuration space, $Q \subseteq \mathbb{R}^d$, and control $u \in \mathbb{R}^n$, with,

$$\mathcal{M}_r : \quad x(t+1) = Ax(t) + Bu(t) + w(t), \quad \forall t \in \mathbb{N},$$

where (A,B) is controllable. $\mathcal{M}_r$ is a linear time invariant dynamic model chosen to approximate unknown, possibly nonlinear dynamics. If bounds on the control $u(t)$ are known, they can be directly applied in the calculation of $x(t+1)$ above. The noise model is chosen as $w(t) \sim \mathcal{N}(0, P_w)$. The robot must avoid N_O rigidly rotating and translating obstacles, $\mathcal{O}_1, \ldots, \mathcal{O}_{N_O} \subset Q$, to stay safe. Obstacle movement is modeled as a rigid body transformation,

$$\mathcal{M}_O : \quad \mathcal{O}_k(t+1) = \hat{R}_k \mathcal{O}_k(t) + \hat{\gamma}_k, \text{ for } k \in [1, N_O],$$

where $\hat{R}_k$ is a time-invariant rotation matrix and $\hat{\gamma}_k \in \mathbb{R}^d$ is a time-invariant vector translation. $\hat{R}_k$ and $\hat{\gamma}_k$ are estimates of the unknown true dynamics R_k and γ_k based upon the observation period. The goal is for the robot to navigate a path within the free space $\bar{\mathcal{O}}(t) := Q \setminus (\mathcal{O}_1(t) \cup \cdots \cup \mathcal{O}_{N_O}(t))$, from the initial state $x_I \in \mathbb{R}^d$ to the goal state $x_G \in \mathbb{R}^d$. The path is denoted as an ordered set of states $Z : x_I, x_2, \ldots, x_G$. For the path to be considered safe, $x_i \in \bar{\mathcal{O}}(t),\ \forall t \geq 0$.

To account for state modeling errors, the state is represented by a distribution,

$$\mathbb{P}_1(t) := \mathcal{N}(x(t), P_x(t)),\ P_x(t+1) = AP_x(t)A^T + P_w. \tag{3.1}$$

To account for obstacle model errors, each obstacle location is represented by a distribution evolving according to,

$$\mathbb{P}_{2,k}(t) := \mathcal{N}(\mathcal{O}_k^C(t), P_k(t)),\ P_k(t+1) = P_k(t) + \kappa_k, \tag{3.2}$$

for $k \in [1, N_O]$, where $\mathcal{O}_k^C(t)$ represents the center of mass of $\mathcal{O}_k(t)$, and κ_k represents the distribution spread since the observation period.

3.5 Algorithm: Wasserstein-Safe RRT

In this section, we define a novel path planning algorithm for uncertain robotic states and environments based on Rapidly Exploring Random Trees [67]. In Proposition 1, we leverage the recent probabilistic guarantees on the discrete W-distance from [62], [68] to create a safety search criterion.

To determine whether states at time t are safe, Algorithm 3 uses empirical distributions sampled from $\mathbb{P}_1(t)$ and $\mathbb{P}_{2,1}(t), \ldots, \mathbb{P}_{2,N_O}(t)$, which are available for sampling for all $t \geq 0$ because the algorithm creates and maintains them. To sample $\mathbb{P}_1(t)$ and $\mathbb{P}_{2,1}(t), \ldots, \mathbb{P}_{2,N_O}(t)$ at time t, a temporarily truncated and finitely supported version of each distribution is created. Wasserstein truncation errors can be found and added to ε_1 and ε_2 at each time step to maintain probabilistic guarantees. Alternatively, any finitely supported distributions can be used.

Proposition 1. *Let $\mathbb{P}_1$ and $\mathbb{P}_2$ be two probability distributions over the compact configuration space $Q \subseteq \mathbb{R}^d$. Let $\mathbb{P}_i$ be supported on $B_i \subseteq Q$ with $\rho_i = 1/2 \operatorname{diam}_\infty(B_i)$, $\rho_i < \infty$. Let $\hat{\mathbb{P}}_1^N$ and $\hat{\mathbb{P}}_2^M$ be the empirical distributions defined from taking N and M samples of $\mathbb{P}_1$ and $\mathbb{P}_2$, respectively, at an arbitrary time t. Given a confidence $1-\beta \in [0,1]$, for any $p \geq 1$, $N \geq 1$ when $p < d/2$, it holds that*

$$W_p(\mathbb{P}_1, \mathbb{P}_2) \geq W_p(\hat{\mathbb{P}}_1^N, \hat{\mathbb{P}}_2^M) - \varepsilon_1^N - \varepsilon_2^M, \tag{3.3}$$

with ε_i^N *given by,*

$$\varepsilon_i^N(\beta,\rho_i) = \left(\frac{\ln(C\beta^{-1})}{c}\right)^{1/d} \frac{\rho_i}{N^{1/d}},$$

constants C and c given by,

$$C = \frac{(C_*)^d}{2\sqrt{d}^d}, \text{ and } c = \frac{1}{2^d\sqrt{d}^d}, \tag{3.4}$$

and constant C_* *given by,*

$$C_* = \sqrt{d}\,2^{\frac{d-2}{2p}}\left(\frac{1}{1-2^{p-d/2}} + \frac{1}{1-2^{-p}}\right)^{\frac{1}{p}}. \tag{3.5}$$

Proof. Note first that by the triangular inequality,

$$W_p(\mathbb{P}_1,\mathbb{P}_2) \geq W_p(\hat{\mathbb{P}}_1^N,\hat{\mathbb{P}}_2^M) - W_p(\mathbb{P}_1,\hat{\mathbb{P}}_1^N) - W_p(\hat{\mathbb{P}}_2^M,\mathbb{P}_2). \tag{3.6}$$

Let the W-distance between a distribution and an empirical one built with samples of said distribution be constrained within a bound ε,

$$W_p(\mathbb{P}_1,\hat{\mathbb{P}}_1^N) \leq \varepsilon_1^N, \quad W_p(\mathbb{P}_2,\hat{\mathbb{P}}_2^M) \leq \varepsilon_2^M. \tag{3.7}$$

Proposition 1 Equation (3.3) follows via substitution of (3.7) into (3.6). According to [62, Proposition 6 and 20], the nominal bound ε_i between the continuous and empirical distributions with a confidence $1-\beta$ is given below. Consider a sequence $(X_i)_{i\in\mathbb{N}}$ of i.i.d $\mathbb{R}^d$-valued random variables supported compactly on distribution μ. Then, for $p < d/2$ and $N \geq 1$, we have $P(W_p(\mu^N,\mu) \leq \varepsilon^N(\beta,\rho)) \geq 1-\beta$, where,

$$\rho = \frac{1}{2}\text{diam}_\infty(\text{supp}(\mu)). \tag{3.8}$$

Under the assumptions of Proposition 1, we can select ε^N,

$$\varepsilon^N(\beta,\rho) = \rho(C_* N^{-\frac{1}{d}} + \sqrt{d}(2\ln\beta^{-1})^{\frac{1}{2p}} N^{-\frac{1}{2p}}), \tag{3.9}$$

with C_* as given in (3.5). Proposition 1 Equation (3.4) follows as shown in [62, Corollary 21]. □

We design a sampling-based path planner similar to RRT, but which compares $\mathbb{P}_1(t)$ and $\mathbb{P}_{2,k}(t)$ for each obstacle k instead of using a collision checker. In particular, the algorithm bounds the W-distance between the probability distribution of the vehicle state and the obstacle state by means of the associated empirical distributions and the bounds of Proposition 1. The Wasserstein metric is used over other approaches, such as Kullback-Leibler divergence, due to its consistency with state-space Euclidean distance [69]. The inputs to Algorithm 3 are the initial vehicle state $x_I \in \mathbb{R}^d$, the goal vehicle state $x_G \in \mathbb{R}^d$, the robot dynamic model $\mathcal{M}_r$, the obstacle model $\mathcal{M}_O$, the confidence β, and the W-distance threshold value θ. The output of the algorithm is the W-Safe path $Z = \{x_I, x_2, \ldots, x_G\}$.

To create a tree T, Algorithm 3 generates a random sample, finds the nearest node in the tree, and simulates control of the vehicle from the parent node to the sample with intermediate states. The state distribution and obstacle distributions are simulated via $\mathcal{M}_r$ and $\mathcal{M}_O$, and then intermediate states are checked for safety with confidence $1-\beta$. If the sample and intermediates are safe, as explained next, the sample is added to the tree. At intermediate state $x(j)$, we have the empirical state distribution $\hat{\mathbb{P}}_1^N(j)$ sampled from $\mathbb{P}_1(j)$, and the N_O empirical obstacle distributions $\hat{\mathbb{P}}_{2,k}^M(j)$ sampled from $\mathbb{P}_{2,k}(j)$ for $k \in [0, N_O]$. $W_p(\hat{\mathbb{P}}_1^N(j), \hat{\mathbb{P}}_{2,k}^M(j))$ for each obstacle is found through [70], based upon [71], then Proposition 1 is used in Method 1: WCheck to ensure,

$$W_p(\hat{\mathbb{P}}_1^N(j), \hat{\mathbb{P}}_{2,k}^M(j)) - \varepsilon_1^N - \varepsilon_2^M \geq \theta, \tag{3.10}$$

as shown in Fig 3.2, so that $W_p(\mathbb{P}_1(j), \mathbb{P}_{2,k}(j)) \geq \theta$ for each obstacle k. New samples are added

to T until $x_G \in T$.

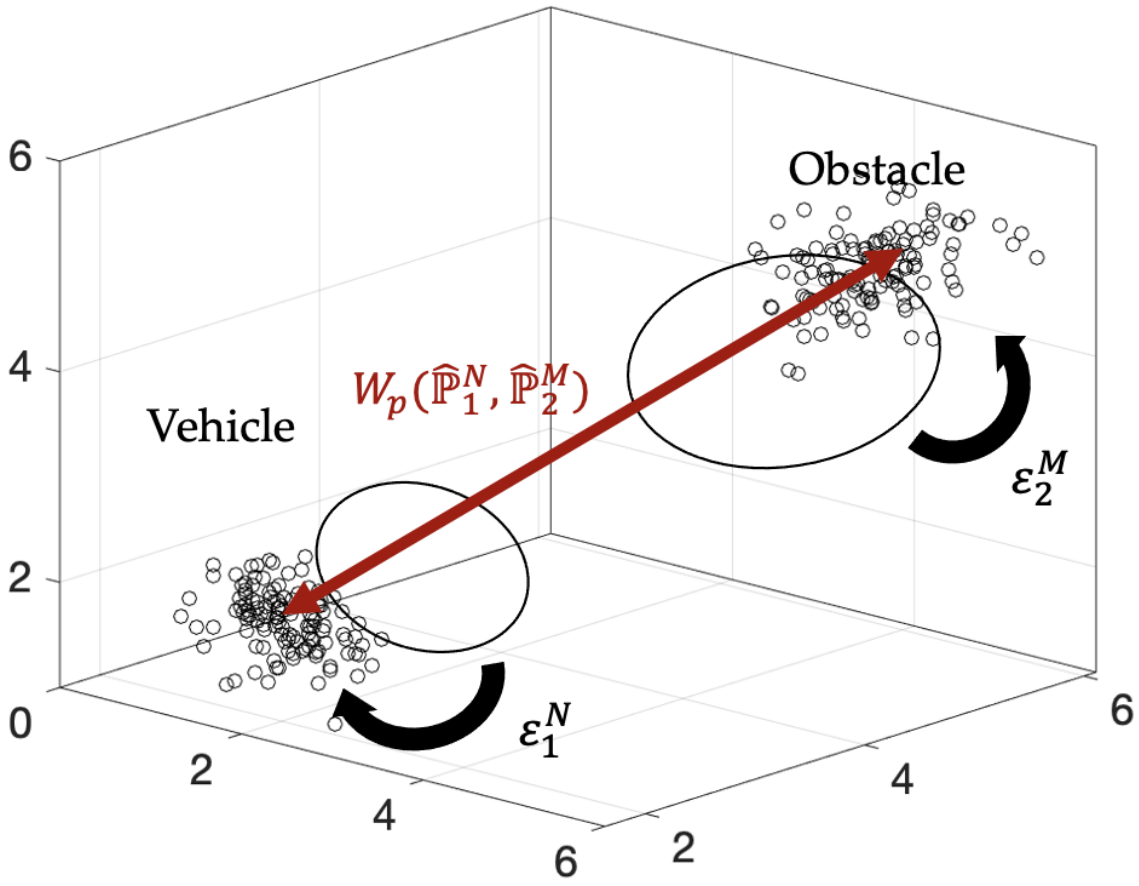

Figure 3.2. Discrete W-distance between robot and obstacle distributions $W_p(\hat{\mathbb{P}}_1^N, \hat{\mathbb{P}}_2^M)$ shown compared to maximally-poor distributional sampling errors ε_1^N and ε_2^M, with continuous distribution locations approximated by ovals.

In contrast to RRT, $N_O + 1$ distinct distribution trees, $T_{\mathbb{P}1}$ and $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$, keep track of $\mathbb{P}_1$ and $\mathbb{P}_{2,1}, \ldots, \mathbb{P}_{2,N_O}$ at each vertex in T. The vertices of $T_{\mathbb{P}1}$ and $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$ are distributions, and the edges are identical to those in T. Once a parent node $x_{\text{parent}} \in T$ is found for a new sample x_{sample}, the distributions $\mathbb{P}_1^{x_{\text{parent}}}$ and $\mathbb{P}_{2,k}^{x_{\text{parent}}}$, $\forall k \in \mathbb{N}_{N_O}$ corresponding to x_{parent} can be quickly found by accessing the same indices in $T_{\mathbb{P}1}$ and $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$. If x_{sample} is added to T, the distributions $\mathbb{P}_1^{x_{\text{sample}}}$ and $\mathbb{P}_{2,1}^{x_{\text{sample}}}, \ldots, \mathbb{P}_{2,N_O}^{x_{\text{sample}}}$ corresponding with that sample are added to $T_{\mathbb{P}1}$ and $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$.

Under certain conditions, W-Safe RRT is a probabilistically complete planner, and the

Algorithm. 3 Wasserstein-Safe RRT

Input: $x_I,\ x_G,\ \mathcal{M}_r,\ \mathcal{M}_O,\ \beta,\ \theta$
Output: $Z:\ x_I, x_2, \ldots, x_G$

Initialize Tree T with vertex x_I, and distribution trees $T_{\mathbb{P}1}$ and $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$ using $\mathcal{M}_O$, (3.1), and (3.2)
while $x_G \notin T$ **do**
  Randomly draw state x_{new}
  Find $x_{\text{parent}} \in T$, the nearest vertex to x_{new}
  Find $\mathbb{P}_1^{x_{\text{parent}}} \in T_{\mathbb{P}1}$ and $\mathbb{P}_{2,1}^{x_{\text{parent}}}, \ldots, \mathbb{P}_{2,N_O}^{x_{\text{parent}}} \in T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$ corresponding with x_{parent}
  Using $\mathcal{M}_r$, advance state from x_{parent} to x_{new} with intermediate states $x(1), \ldots, x(j)$
  Using $\mathcal{M}_O$, advance distr. from $\mathbb{P}_1^{x_{\text{parent}}}$, $\mathbb{P}_{2,1}^{x_{\text{parent}}}, \ldots, \mathbb{P}_{2,N_O}^{x_{\text{parent}}}$ to $\mathbb{P}_1^{x_{\text{new}}}$, $\mathbb{P}_{2,1}^{x_{\text{new}}}, \ldots, \mathbb{P}_{2,N_O}^{x_{\text{new}}}$ with intermediate distr. $\mathbb{P}_1(1), \ldots, \mathbb{P}_1(j)$, $\{\mathbb{P}_{2,1}(1), \ldots, \mathbb{P}_{2,1}(j)\}, \ldots, \{\mathbb{P}_{2,N_O}(1), \ldots, \mathbb{P}_{2,N_O}(j)\}$
  Unsafe = False
  for $i = 1 : j$ **do**
    W_p = WCheck($\mathbb{P}_1(i), \{\mathbb{P}_{2,1}(i), \ldots$
    $\ldots, \mathbb{P}_{2,N_O}(i)\}, \beta, N, M$)
    if $W_p < \theta$ **then**
      Continue
    else
      Unsafe = True, Break
    end if
  end for
  if Unsafe **then**
    Continue
  else
    Add x_{new} to T, add $\mathbb{P}_1^{x_{\text{new}}}$ to $T_{\mathbb{P}1}$
    Add $\mathbb{P}_{2,1}^{x_{\text{new}}}, \ldots, \mathbb{P}_{2,N_O}^{x_{\text{new}}}$ to $T_{\mathbb{P}2,1}, \ldots, T_{\mathbb{P}2,N_O}$
  end if
end while
Trace T from x_G to x_I to extract path
Return path $Z:\ x_I, x_2, \ldots, x_G$

Method 1. WCheck, from Algorithm 3 Line 9

Input: $\mathbb{P}_1(i), \{\mathbb{P}_{2,1}(i), \ldots, \mathbb{P}_{2,k}(i)\}, \beta, N, M$
Output: W_p

1: Truncate $\mathbb{P}_1(i), \{\mathbb{P}_{2,1}(i), \ldots, \mathbb{P}_{2,k}(i)\}$ to $\rho_1, \rho_{2,1}, \ldots, \rho_{2,k}$ as defined in (3.8)
2: Calculate C_* via (3.5)
3: Calculate ε_1^N and $\varepsilon_{2,1}^M, \ldots, \varepsilon_{2,k}^M$ via (3.9)
4: Sample $\hat{\mathbb{P}}_1^N = \{\hat{\xi}_1, \ldots \hat{\xi}_N\}$ from $\mathbb{P}_1(i)$
5: Sample $\hat{\mathbb{P}}_{2,l}^M = \{\hat{\zeta}_1^l, \ldots \hat{\zeta}_M^l\}$ from $\mathbb{P}_{2,l}(i)$ for each $l \in \mathbb{N}_k$
6: Find W-distance $W_p(\hat{\mathbb{P}}_1^N, \hat{\mathbb{P}}_{2,l}^M)$ between $\{\hat{\xi}_1, \ldots \hat{\xi}_N\}$ and $\{\hat{\zeta}_1^l, \ldots \hat{\zeta}_M^l\}$ for each $l \in \mathbb{N}_k$
7: Return $\inf_{l \in \mathbb{N}_k}(W_p(\hat{\mathbb{P}}_1^N, \hat{\mathbb{P}}_{2,l}^M) - \varepsilon_1^N - \varepsilon_{2,l}^M)$ as per (3.10)

statement of probabilistic completeness follows.

Theorem 2. *Let $\mathbb{P}_{\text{obs},k}(\ell)$, be the distributions of obstacles k at time ℓ, for $k \in \mathbb{N}_{N_O}$ and $\ell \in [0, \infty)$, $\mathbb{P}_{\text{rob}}^0$ be initial robot configuration, and B_{goal} be a ball around the goal configuration in a compact configuration space Q. Suppose that $\forall\, k$ the support of each $\mathbb{P}_{\text{obs},k}(\ell)$ remains in a ball $B_{\text{obs},\,k}(\ell)$. Let $S \subset Q$ be $\bigcup_{k=1}^{N_O} S_k$, with S_k such that $\forall\, \ell, B_{\text{obs},k}(\ell) \subset S_k$. Assume that for each ℓ there exist controls such that the robot distribution is transformed into $\mathbb{P}_{\text{rob}}(\ell)$, and assume the support of this distribution remains in a compact ball $B_{\text{rob}}(\ell)$. Assume there is a path $\mathbb{P}_{\text{rob}}(\ell)$ from $\mathbb{P}_{\text{rob}}^0$ to B_{goal} such that the final robot distribution mean $\mu_{\text{rob}} \in B_{\text{goal}}$ and $\forall\, \ell$, the set distance $\text{dist}_p(B_{\text{rob}}(\ell), S) > \delta$,[1] for some $\delta > 0$. Then, if we take N, M samples such that Eq. (3.3) from Proposition 1 holds, W-Safe RRT returns a path from $\mathbb{P}_{\text{rob}}^0$ to B_{goal} with probability 1 that satisfies $W_p(\mathbb{P}_{\text{rob}}(\ell), \mathbb{P}_{\text{obs},k}(\ell)) > \delta - \varepsilon_1^N - \varepsilon_2^M$, for all $k \in \mathbb{N}_{N_O}$ and $\ell \in [0, \infty)$ with confidence $1 - \beta$.*

Proof. The proof follows from the probabilistic completeness of RRT [67] and Proposition 1. These results can be combined because the sampling processes w.r.t. configurations in Q and that of obtaining samples of each distribution are independent. From the probabilistic completeness of RRT, the algorithm will produce a path from $\mathbb{P}_{\text{rob}}^0$ to B_{goal} such that $\forall\, \ell$, $\text{dist}_p(B_{\text{rob}}(\ell), S) > \delta$, for some $\delta > 0$ with probability 1. Note that $\forall\, k$ and $\forall\, \ell$,

$$W_p(\hat{\mathbb{P}}_{\text{rob}}(\ell), \hat{\mathbb{P}}_{\text{obs,k}}(\ell)) \geq \text{dist}_p(B_{\text{rob}}(\ell), B_{\text{obs,k}}(\ell)) > \delta,$$

[1] Here, $\text{dist}_p(A, B) = \inf\{\|x - y\|^p \,|\, x \in A, y \in B\}$.

implying via Proposition 1,

$$W_p(\mathbb{P}_{\text{rob}}(\ell), \mathbb{P}_{\text{obs,k}}(\ell)) \geq \delta - \varepsilon_1^N - \varepsilon_2^M,$$

with confidence $1-\beta$. □

3.6 Results and Discussion

In this section, we show simulations and results that are performed in a three dimensional environment run with MATLAB ver. R2020b on a machine with an Intel i5-4690K CPU, 32GB RAM, and an AMD Radeon R9 290X GPU. Algorithm 3 is compared against a baseline minimum encompassing sphere method that extends PCC-RRT to handle the same assumptions, modeling, and scenarios as W-Safe RRT. The goal is to compare W-Safe RRT to this baseline, with and without forced distributional obstacle errors, in a variety of environments.

The chosen configuration space $Q \subseteq \mathbb{R}^3$ is $Q = \{(x,y,z) | x,y,z \in [0,10]\}$. The following vehicle model dynamics and control policy,

$$x(t+1) = A(x(t) - x_{\text{new}}) + Bu(t) + x_{\text{new}}\,,\, x(0) = x_{\text{parent}},$$

$$A = \begin{bmatrix} .9 & -.05 & .1 \\ .05 & .9 & -.1 \\ 0 & .08 & .85 \end{bmatrix}, B = \begin{bmatrix} .85 & .2 & 0 \\ -.15 & .85 & .1 \\ -.1 & .1 & .9 \end{bmatrix},$$

$$u(t) = 0.4\frac{-K(x(t) - x_{\text{new}})}{\|u(t)\|},$$

$$K = \begin{bmatrix} 2.36 & -0.51 & 0.18 \\ 0.45 & 1.93 & -0.27 \\ 0.21 & -0.18 & 1.55 \end{bmatrix},$$

are implemented from x_{parent} to x_{new} each time a random state is drawn. The control policy

performs reference tracking for controlling to x_{new}. The constant matrix K can be any gain matrix such that the closed loop system is stable.

A single, spherical obstacle is used in the initial simulation results. For each path planning problem, the true obstacle location $\mathcal{O}(t)$ is chosen to be a ball of radius 0.5 centered randomly near the center of Q and drifting in a random direction. The obstacle uncertainty model is,

$$\mathbb{P}_{2,1}(t) = \mathcal{N}\left(\mathcal{O}^C(t), \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.15 \end{bmatrix}\right),$$

where $\mathcal{O}^C(t) \in \mathbb{R}^3$ is the geometric center of $\mathcal{O}(t)$. The state uncertainty model, $\mathbb{P}_1(t)$, centered at state (x, y, z), is,

$$\mathbb{P}_1(t) = \mathcal{N}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.15 \end{bmatrix}\right).$$

For sampling, the support of the distribution in each dimension is an interval centered at $x, y,$ or z, with length 1 in all dimensions. The start state, x_I, and the goal state, x_G, are chosen uniformly randomly to be on opposite sides of the obstacle, forcing the path to interact with the obstacle, as shown in Fig. 3.3. For all simulations except the drift error percentage study, each algorithm assumed a drift vector $\hat{\gamma}$ with 10 percent error from the true drift γ. In simulation, the maximum distance between x_{new} and the closest parent state $x_{\text{parent}} \in T$ is capped, and the random state x_{new} draw is chosen to be toward the goal state x_G with 0.3 probability. The W_p-distance is calculated with $p = 1$ and $d = 3$, so that $p < d/2$. To use higher p values within the W_p-distance, the state can be lifted into higher dimensions to achieve $d > 3$.

3.6.1 Algorithm Performance

For the baseline comparison algorithm, the Method 1 WCheck is swapped out for the comparison Method 2 CompCheck. In Method 2, both the vehicle and obstacle distributions, $\mathbb{P}_1(j)$ and $\mathbb{P}_{2,1}(j)$, respectively, are sampled to create two circumspheres, which are then checked for overlap. The circumspheres are found through Welzl's Algorithm [72], which is chosen for simplicity. If there is overlap, the state is not safe with respect to the obstacle. This algorithm, which is an extension of PCC-RRT, includes similar simulated dynamics, particle sampling and re-sampling from distributions, and a probabilistic check to admit robot states to the RRT-style tree. It is modified to accept obstacle uncertainty representations instead of a known convex obstacle polytope by changing the comparison between two distributions to be via Welzl's Algorithm.

Method 2. CompCheck

Input: $\mathbb{P}_1(j), \mathbb{P}_2(j), N, M$
Output: s
1: Sample $\{\hat{\xi}_1, \ldots \hat{\xi}_N\}, \{\hat{\zeta}_1, \ldots \hat{\zeta}_M\}$ from $\mathbb{P}_1(j), \mathbb{P}_{2,1}(j)$
2: Find minimum encompassing spheres characterized by c_ξ, r_ξ and c_ζ, r_ζ, using Welzl's Algorithm
3: **if** $\|c_\xi - c_\zeta\| < r_\xi + r_\zeta$ **then**
4: $\quad s$ = false
5: $\quad$ **else** s = true
6: **end if**

The algorithms are evaluated by comparing returned paths with reality in two ways, nominally and adversarially. The nominal evaluation analyzes the returned path ensuring it avoids the 0.5 radius spherical obstacle. The adversarial evaluation assumes that at any state, $x_i \in Z$, each obstacle will have shifted one unit from $\mathcal{O}(t)$ toward x_i to try to block the path. The post processing evaluations assess the safety of returned paths and therefore algorithm performance. Paths are not discarded by the algorithms based on the evaluations.

Table 3.1 records the number of correctly returned paths out of 1000 path planning problems, the average time to solve each problem, the average path length of each solution, and

Table 3.1. W-Safe RRT performance comparison in the three dimensional single obstacle case.

	Ball Method	W-Safe RRT
Average Time	5.345 s	12.786 s
Ave. Path Length	14.89 u	15.23 u
Ave. No. Nodes	40.38	43.76
Nominal Check	997/1000	1000/1000
Adversarial Check	681/1000	998/1000

the average number of nodes per solution. Algorithm 3 using Method 1 took an average of 12.786 seconds to solve each problem and returned 998 safe paths when checked adversarially. Method 2 took an average of 5.345 seconds and returned 681 safe paths when checked adversarially. This time penalty is largely due to calculating the discrete W-distance, $W_p(\hat{\mathbb{P}}_1^N, \hat{\mathbb{P}}_{2,1}^M)$ in Algorithm 1, which requires solving an optimization problem (discrete optimal transport reduces to a linear program.) The W-method is more robust in the face of an adversarial path check.

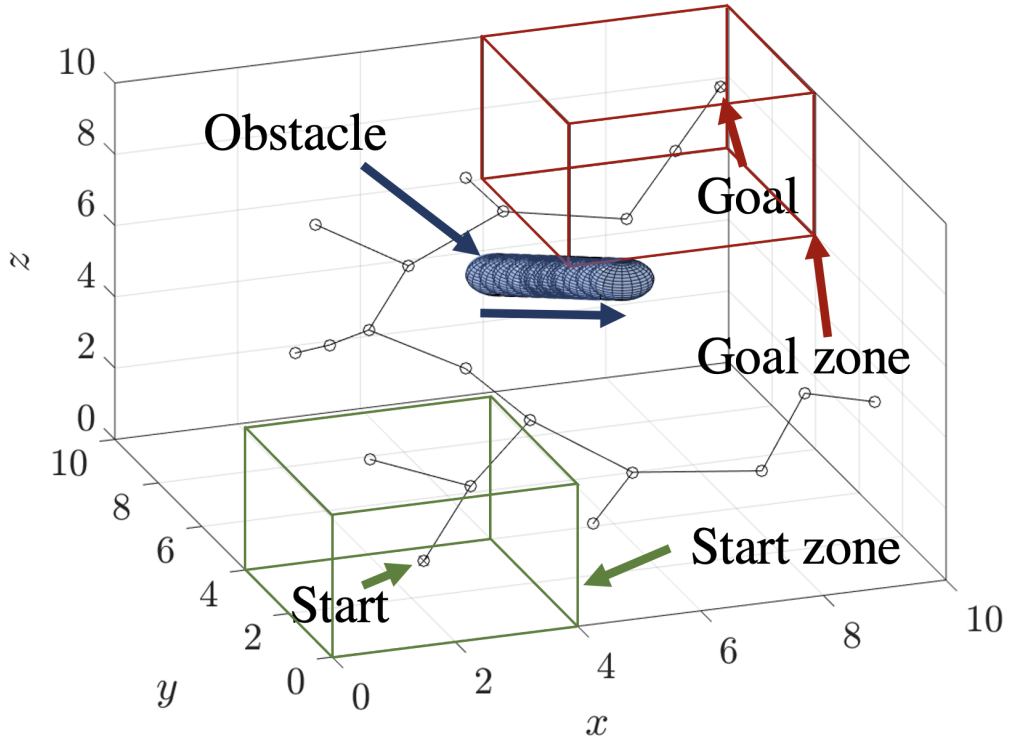

Figure 3.3. Tree of probabilistically feasible states found by W-Safe RRT shown within a random opposite-corner start and goal location path planning problem.

Table 3.2. W-Safe RRT performance comparison in the three dimensional three obstacle case.

	Ball Method	W-Safe RRT
Average Time	16.53 s	45.12 s
Ave. Path Length	14.65 u	16.27 u
Ave. No. Nodes	39.41	51.46
Nominal Check	935/1000	1000/1000
Adversarial Check	344/1000	990/1000

Additionally, Method 1 allows W-Safe RRT to return a path with the probabilistic guarantee on distributional uncertainty given in Proposition 1, while Method 2 does not. The adversarial check analyzes algorithm resiliency to adversarial distributional modeling errors, where obstacles exist outside of possible distribution sampling ranges and toward the state in question. W-Safe RRT outperforms the ball method when returned paths are checked adversarially.

Any finitely supported distribution can be used to represent uncertainty, and poor distribution choice is accounted for by W-Safe RRT. Additionally, when a known state is projected forward with a dynamic model, unknown future events impact the state. W-Safe RRT accounts for distributional uncertainty that results from dynamic model errors, noise model errors, and unpredictable future disturbances.

3.6.2 Multi-Obstacle Performance

We compare the algorithms in a random start and goal environment with 3 randomly placed and drifting obstacles, as shown in Fig. 3.4. Obstacle drift information is known to each algorithm with 10% error. Table 3.2 records the number of correctly returned paths out of 1000 as well as the averages of relevant metrics for both algorithms. With the adversarial check on obstacles, when compared to the single obstacle case, the ball method dropped from 681 safe paths to 344 while W-Safe RRT dropped from 998 to 990, and both algorithms take about three times as long to run. W-Safe RRT shows vastly improved performance over the baseline in uncertain multi-obstacle environments.

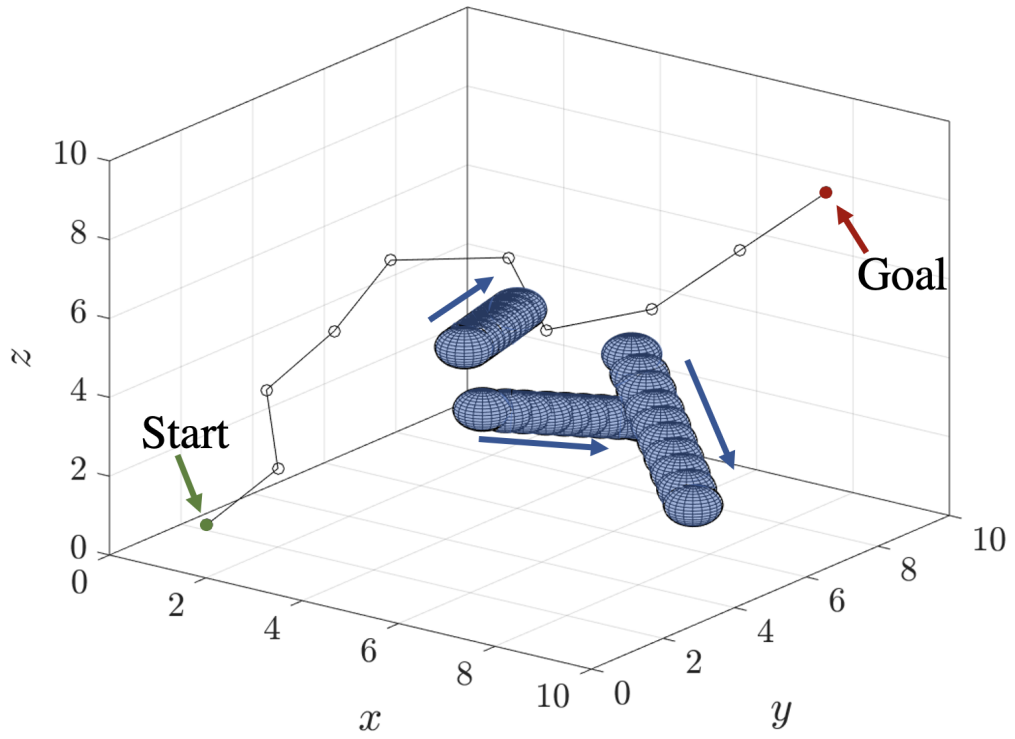

Figure 3.4. Returned path from W-Safe RRT shown navigating a three obstacle environment with random constant drift values.

3.6.3 Non-convex and Rotating Multiple Obstacle Case

We compare the algorithms in a random start and goal environment with three randomly placed, drifting, and rotating non-convex obstacles. Obstacles tested were *L*-shaped, rotate on multiple world-frame axes, and rotate between a fifth and a half of a full rotation during each simulation, as shown in Fig. 3.5. Obstacle drift information is again known to each algorithm with 10% error and no information on obstacle shape and rotation is known. As shown in Table 3.3, W-Safe RRT returned 867 out of 1000 paths as safe paths, at a cost of approximately three times the computation time, while the comparison algorithm returned 663 safe paths. W-Safe RRT outperforms the comparison method in high uncertainty environments with rotating non-convex obstacles.

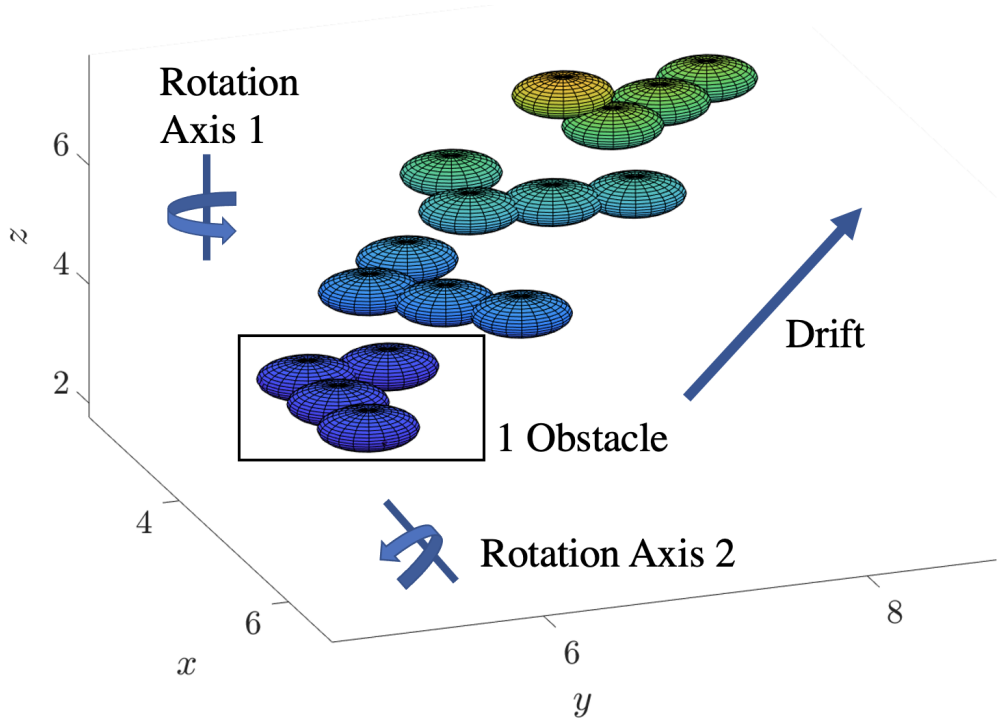

Figure 3.5. A single L-shaped non-convex randomly rotating and translating obstacle moving across a 3D environment.

3.6.4 Variance of Risk Parameter

The W-distance threshold value θ can be used as a risk parameter to change the margin of safety, as shown in Fig. 3.6. Risk is defined as the likelihood that returned paths intersect with obstacles. Risk parameter performance is measured by the nominal and adversarial obstacle checks on large numbers of produced paths at each parameter level. Since the Ball Method is independent of θ, its performance is represented as a line at 68.1% adversarially and 99.7%

Table 3.3. W-Safe RRT performance comparison in the three dimensional three rotating nonconvex obstacle case.

	Ball Method	W-Safe RRT
Average Time	20.48 s	62.71 s
Ave. Path Length	14.77 u	16.36 u
Ave. No. Nodes	40.25	47.03
No. Safe Paths	663/1000	867/1000

nominally. As shown in Fig. 3.6, reducing θ decreases the distributional safety and causes the percentage of safe paths to drop. With a low enough θ, W-Safe RRT can be outperformed by the comparison method. The confidence $1-\beta$ can also act as a risk parameter, and parameter value choice is left as a supervisory decision for the appropriate trade-off between performance and cost in the specific application.

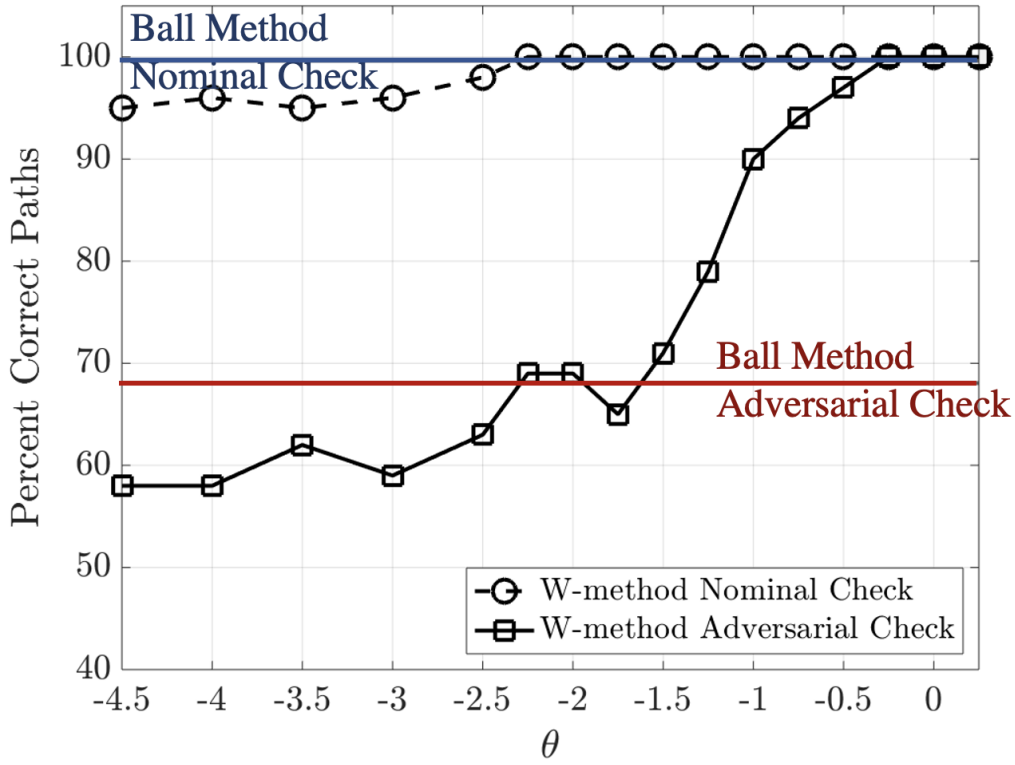

Figure 3.6. Illustration of variance to the risk parameter θ and its effect on percent safe paths.

3.6.5 Variance of Drift Informational Error

When vehicle observations and assumptions toward obstacle drift vector are erroneous, W-Safe RRT performance and comparison algorithm performance both drop. As the drift error percentage is larger, the performance of W-Safe RRT drops from 99% correct paths to 80% correct paths, as shown in Fig. 3.7. The performance of the baseline method also sees an overall downward trend, but it is bounded from below at 60%, which could represent the proportion of paths that do not interact with the obstacle at all. With highly erroneous information, W-Safe

RRT still outperforms the baseline algorithm.

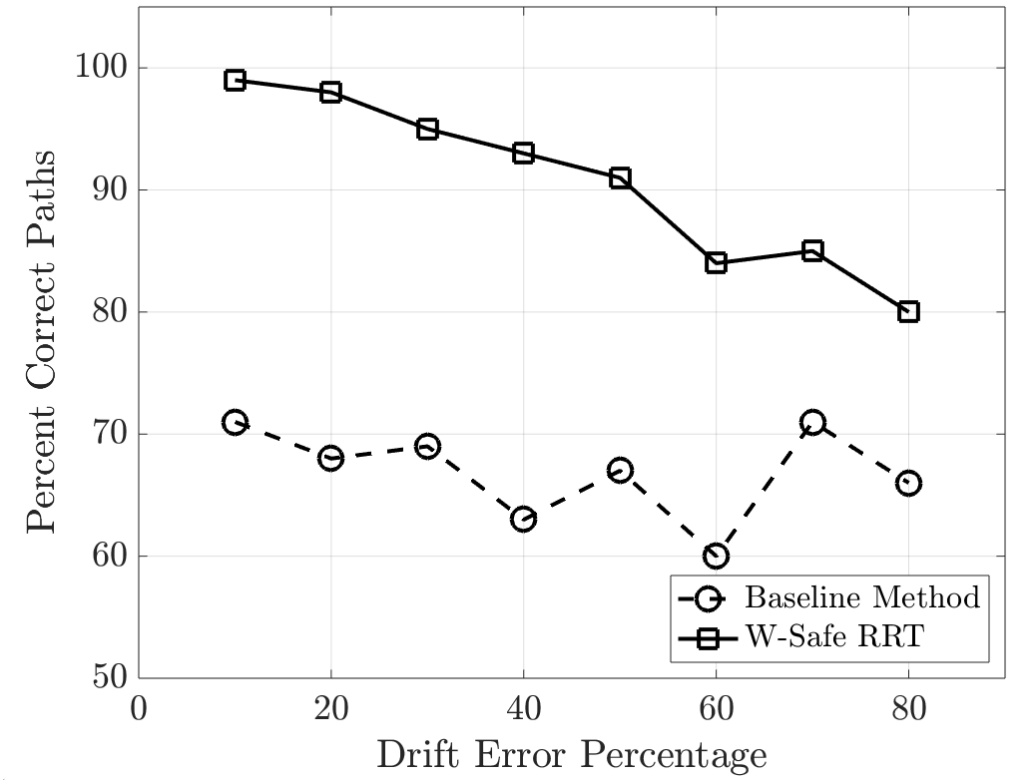

Figure 3.7. Illustration of variance to the drift error percentage and its effect on percent safe paths

3.6.6 Comparison with Similar Algorithms and Extensions

The work [66] describes a state-of-the-art path planning tool that uses Wasserstein ambiguity sets around obstacle drift values. The main approach consists of constraining a model predictive control problem to pick state control $u(t)$ that can closely follow an un-safety-constrained reference trajectory. In contrast, W-Safe RRT is an offline pre-planning state estimation algorithm that applies uncertainty models to the vehicle state as well as the obstacle states and only admits nodes to the RRT tree that satisfy Wasserstein distance safety constraints. W-Safe RRT can handle rotating non-convex and convex obstacles with non-linear boundaries and substantially drifting obstacles that are not fully known to the algorithm. In the spirit of sampling-based motion planners, the explicit boundary representation of the obstacles

and bounding uncertainty sets is not required; this makes the algorithm generalizable to higher-dimensional configuration spaces. Crucially, W-Safe RRT is tested to return paths with a probabilistic guarantee on distributional sampling error. Lastly, because the MPC constraints in [66] scale with number of samples, the number of obstacles, and time horizon, it becomes computationally more expensive in the face of a large number of samples, substantial drift, and offline use over long time horizons.

Because of the assumptions in [66] that include that (i) the robot state and a convex polytopes of each obstacle are known, and (ii) the uncertainty satisfies known polytope constraints; the fact that W-Safe RRT simulations are performed with rotating non-convex and substantially drifting obstacles; and the fact that [66] considers a MPC decision making solver that is not readily adaptable to be a pre-planning path finder; a direct comparison in simulation is neither plausible nor meaningful. Performing fair comparisons between algorithms that rely on differing assumptions is an important open question in safe learning and control [73]. As a result, for a state of the art comparison algorithm, we have extended PCC-RRT [57] into the associated minimum encompassing-ball algorithm. This extension takes the same assumptions and information as W-Safe RRT, and can compare multiple uncertainty distributions in a direct way.

W-Safe RRT can be extended to handle known time-varying discrete linear systems by substituting the appropriate equations into $\mathcal{M}_r$. However, known nonlinear systems will result in nonlinear distributional evolutions, which can be handled by linearizing dynamics at each time step. W-Safe RRT can handle continuous-time dynamical systems by taking a small enough discretization step when performing planning and safety checks.

3.7 Conclusion

In this chapter, we develop the safety-aware algorithm Wasserstein-Safe RRT, which leverages probabilistic guarantees on discrete sampling error from [62] and [68] to build a

probabilistically complete RRT-style tree that accounts for distributional uncertainty. When the vehicle and obstacles are both represented with distributions, W-Safe RRT outperforms the PCC-RRT-inspired minimum encompassing ball method in both simple convex obstacle environments and highly uncertain environments with rotating non-convex obstacles. W-Safe RRT carries probabilistic guarantees on returned paths being a certain Wasserstein distance away from obstacles at the cost of a time penalty, and shows robustness in the face of distributional error. The method demonstrates that use of model uncertainty in situations where such uncertainty is warranted, such as with uncertain obstacle shape and dynamics, leads to safer path planning behavior. Future work includes testing W-Safe RRT with linear approximations of nonlinear dynamics, planning with continuous time dynamics, and planning in crowded static environments.

Chapter 3, in full, is a reprint of the material "Distributionally Safe Path Planning: Wasserstein Safe RRT" as it appears in IEEE Robotics and Automation Letters [74]. Lathrop, Paul, Beth Boardman, and Sonia Martínez. IEEE Robotics and Automation Letters 7.1 (2021): 430-437. The dissertation author was the primary investigator and author of this paper.

Explicit use of uncertainty in state representation and evolution assists in the generation of safer motion plans, especially when true obstacle shape, dynamics, or behavior is not known exactly. Additionally, as is demonstrated by the variance in risk parameter section, changes to risk parameters themselves have a profound impact on algorithm safety performance. However, risk parameter values are often chosen heuristically (and somewhat arbitrarily) by a supervisor for particular applications, and are not automatically adjusted. In the following chapter, we are interested in exploring a few ideas in conjunction: closing the safety parameter feedback loop with an algorithm that allows automatic safety margin adjustment, employing a human-debuggable method that classifies obstacle behavior in a more abstract *feature* space, and how *classification uncertainty* can be used in higher order representations of state and behavior.

Chapter 4

Mobile Robot Behavioral Classification with Uncertainty for Adaptive Safety

4.1 Introduction

Safety is an essential component of autonomous path planning algorithms. The ability to assess an environment, classify environmental objects, and adapt safety margins enables both safer and more efficient path planning, as shown in Fig. 4.1. Numerous trajectory planning algorithms exist to operate dynamic and often dense obstacle environments. The intersection between trajectory planning and classification typically takes the form of computer vision and image classification, especially in autonomous driving scenarios, where trained machine learning models predict, with uncertainty, what elements of an image correspond with particular agents. On the other hand, in the field of spatio-temporal data mining, there is often a desire to classify trajectories to infer semantic information. Examples of this include parsing what type of ship created a particular GPS trail within a harbor, or predicting whether a GPS track is from a bicyclist or pedestrian.

Trajectory analysis in motion planning is used in model-heavy trajectory prediction techniques, but existing trajectory analysis stops short of using inferences for robotic safety purposes. To improve human-understandable robotic behaviors, methods that employ semantic models may help robots adhere to common sense safety principals in high-stakes applications such as driving [75], especially compared to black-box input-output deep learning methods.

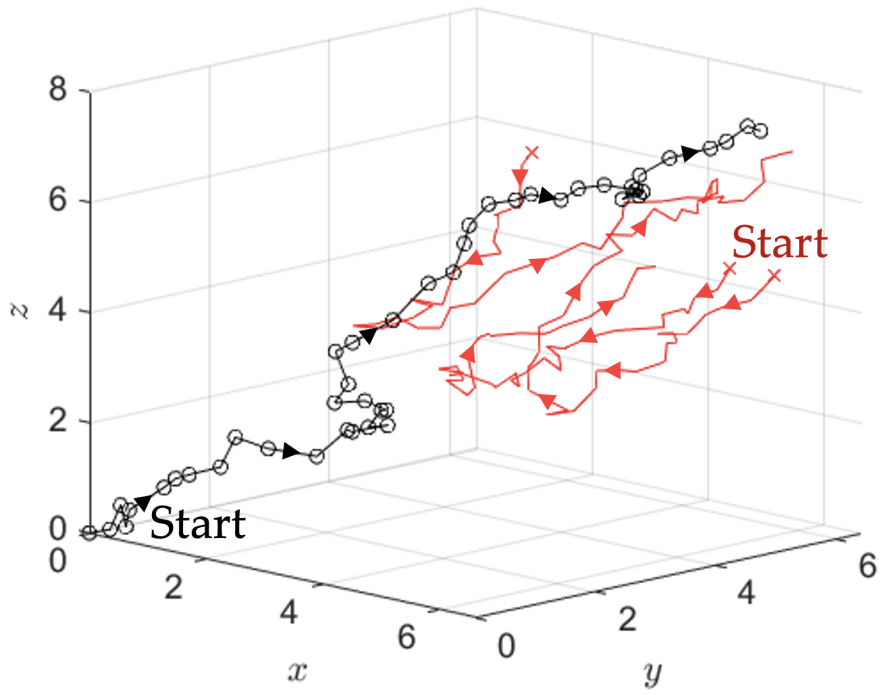

Figure 4.1. Path planning problem solved with time varying safety margins in a 3D environment with 3 adversarial agents.

Transparency of how safety-related intuitions are used in autonomous algorithms can further public understanding of robotic behavior. The methods in this chapter follow this thought process through creating human-understandable semantic behavioral classes, and employing a method with uncertainty to benefit safe path planning. Thus, we are motivated to investigate safety tuning strategies that employ behavioral states to directly influence agent-specific safety values.

4.2 Literature Review

Model-based methods offer adaptive safety margins, but methods perform best with restricted (and sometimes static) agent behaviors, and struggle to adapt to online situations. Authors in [76] achieves provable safety in the face of adversarial agents with guarantees on goal satisfaction by using Hamilton-Jacobi reachability to define sequential trajectory planning, creating a channel-carving effect of reachable guaranteed-safe states. However, agent complexity and processing time are sacrificed to maintain guarantees on goal satisfaction and safety – which

can be considered crucial for close human-robot interaction [73]. In general, agents are few, have known and restricted dynamics, and do not move in a significant way.

Several Reinforcement Learning (RL) approaches integrate learning and planning methods which balance exploration and exploitation to learn high reward actions within an environment. Q-Learning [77] and extensions such as Speedy Q-Learning [78] approximate the optimal action-value function from visits to particular states and actions. However, these RL approaches function best in stationary environments where environmental states are approximately constant. Approaches to learning of optimal actions over continuous robot and environmental states with a continuous action space consist of either function approximation (such as with linear [79] or neural network [80] functions) combined with a large number of examples [81, 82] or policy-gradient methods such as Actor-Critic [83] and other hybrid asynchronous methods [84]. The latter approaches can evaluate policies over continuous spaces, but still suffer from the fact that they are on-policy and largely only learn about the policy being followed. RL approaches suffer from the need for a relatively large number of learning runs, which is coupled with the inability to handle multiple dynamic, adversarial agents. Additionally, we note that behavior-following agents in general do not satisfy a n-th order Markov assumption, and therefore fall into a non-Markovian state space because time-series data differentiates agent attitudes. [85] uses high-level *environmental* semantics to enable a deep RL approach for mobile autonomous robots, but RL over high-level semantic *trajectory* information remains unexplored. RL techniques for online motion planning fall short in reacting to dynamic agents with high dimension state and action spaces, and approaches with high level semantic analysis do not address trajectory differentiation.

In autonomous driving, obstacle classification is focused on vision based classification [86, 87] rather than trajectory based. The works [88, 89] predict pedestrian trajectory and behavior via a model-based probabilistic approach and a deep learning approach respectively. The former work probabilistically informs a vehicle operator of predicted dangerous situations. Despite the more in-depth analysis of trajectory data over previous pedestrian detection stud-

ies, the circle is not fully closed to quickly react by modifying autonomous safety parameters. Additionally, [90] uses car trajectories and an uncertainty model to predict probabilistic representations of future possible trajectories. However, the work is heavily model-based with uncertain real-time performance and the loop is not closed to how trajectory analysis can be used to ensure safer performance.

In the field of signal processing, expectation maximization (EM) forms a core method for performing clustering of unlabeled data [91]. EM relies on two steps, iterated multiple times, to result in a description of a generative model for observed data points. In expectation, the likelihood of a particular model to generate the observed data is calculated. In maximization, the model parameters are adjusted in a way that maximizes the likelihood of observed data generation. EM has been extended into different machine learning clustering algorithms to accomplish different goals, reweighted EM [92] for improved learning, hybrid and variational EM [93] that views model parameters as stochastic variables to be estimated, and incremental EM [94] to calculate only one parameter in each step, to name a few. Evidence lower bound (ELBO) is a lower bound on log likelihood of observed data, used in clustering as a metric to maximize to create an accurate generative model. A relaxed version of ELBO was found in [95] to allow a less computationally intensive calculation.

The analysis and clustering of trajectory data is referred to as spatio-temporal data mining, and is well explored in signal processing and machine learning literature. The survey paper [96] summarizes methods and applications of trajectory analysis. Notably (with respect to this work), [97] uses motion characteristics to cluster trajectories by typical behavior, and offers comparisons across clustering methods. In [98], authors employ a mixture model approach to cluster trajectories and allow probabilistic representations of class assignment, with deterministic decisions deferred until assignment is required, but class uncertainty is not discussed. Uncertainty is studied with respect to incompleteness of data and artificial noise [99], but again not with respect to class uncertainty. Focus is placed on geospatial trajectory analysis [100] and classification [101] for semantic information assignment, but these works focus on class

identification and not on analyzing autonomous agents from the perspective of robotic planning and safety.

Classification uncertainty is well addressed in machine learning literature, especially with respect to deep learning classifiers. [102] explicitly models distributions over class probabilities, trains a neural network to learn the function that creates the class predictions, and creates a predictor whose distribution parameters are set by the neural network. The work [103] trains classifiers on human-like uncertainty and finds better out-of-training-distribution performance and increased robustness to adversarial attacks. While uncertainty of machine learning classifiers is studied in the general case, there is a lack of study of uncertainty as it relates to trajectory classification.

For robotic trajectory prediction, in [104] the authors address trajectory prediction, which lies downstream of class prediction, with a deep learning setup based on the Trajectron++ [105] method. The work addresses how classification uncertainty can benefit the predictive abilities of trajectory generation and verifies performance on real world self driving vehicle datasets. Class uncertainty (in the form of car vs bike vs pedestrian) is fed (alongside state information) into a contractive variational autoencoder to generate trajectories for autonomous driving scenarios. The work largely reasons about correctly labeling classes for prediction purposes and methods are evaluated on whether correct labels are made. Safety, and how class uncertainty relates to safety in online planning is not directly addressed.

In [106], an obstacle avoidance Model Predictive Control algorithm is characterized that uses a three-class obstacle trajectory classification to predict future paths based on discrete dynamics. The three classes of trajectories in the system are static, linear, and trajectory, which allows trajectory prediction with preset dynamic models. While this work analyzes obstacle trajectories for motion planning purposes, clustering all nonlinear trajectories into a single class can miss out on further insights into obstacle behavior.

Trajectory classification uncertainty with respect to safety remains seemingly unexplored in literature. We aim to incorporate class uncertainty for safety-based applications rather than

for trajectory prediction itself. A further goal is to create human-understandable notions of semantic behavior that enables a human-debuggable classification scheme. To do this, we avoid a deep learning encoder architecture to at least guarantee, for a particular problem, that high level behavioral qualities of interest are human-selected and not abstractly chosen by deep learning.

We propose a data-driven classification and safety margin adjustment method that categorizes environmental agents with class uncertainty. The approach relies on an abstract created *feature space*, which encodes semantic notions of agent behavior. To acknowledge feature-space uncertainty, we employ a novel integration-based classifier to assign classes to agent observations with the goal of enabling safer online planning. Class definitions and safety values are based upon environment observations during an offline phase. Observed agents are classified and safety margins are tuned during an online phase. The semantic uncertainty-aware classification scheme is tested against state-of-the-art classification methods in simulation and experimentation.

4.3 Contributions

In this chapter, we consider an online motion planning and classification problem where we use observations of environmental agents to define semantic attitudes, then adjust safety margins for an online path planning algorithm. The two main problems this chapter addresses are: 1) how can we enable hands-off safety parameter adaptation in a way that acknowledges uncertainty, and 2) how can agent attitudes toward a robot be defined to encompass safety-related semantic notions.

The main contributions are the following:

- The proposal of semantic trajectory classes for environmental agent classification
- An uncertainty-aware integrating Bayesian classifier
- The use of the above methods for real time adaptive safety margin adjustment

4.4 Problem Formulation

We consider a robot motion planning problem with an offline training phase and an online execution phase. The robot, at time t, is described by state $x(t) \in \mathbb{R}^d$ and is constrained within a configuration space, $Q \subseteq \mathbb{R}^d$. The robot must avoid N_O agents within the environment, described respectively by $y_1(t), \ldots, y_{N_O}(t) \in Q$. Let the free space be,

$$Q_{\text{free}}(t) := Q \setminus (B_r(y_1(t)) \cup \cdots \cup B_r(y_{N_O}(t))),$$

where $B_r(y(t))$ refers to a ball of radius r centered at $y(t)$. Finally, let $\mathbb{X}, \mathbb{O}_n \in (\mathbb{R}^d)^T$ represent trajectories of the robot and n^{th} agent respectively, each made of a T-long time series of observed states. To ensure that $\|x(t) - y_n(t)\| > r$, $\forall\, n \in \mathbb{N}_{N_O}$, $\forall\, t$, the robot maintains a N_O long vector of safety margins $\theta(t)$, where the n^{th} entry of $\theta(t)$ represents the safety margin with respect to the n^{th} agent. The goal of the robot is to navigate a path from initial condition $x(0) \in \mathbb{R}^d$, to a goal state $x_{\text{goal}} \in \mathbb{R}^d$, such that, $\forall t$, $x(t) \in Q_{\text{free}}(t)$.

When this goal cannot be met due to multiple adversarial agents and planning uncertainty, the goal is to minimize hits while maintaining as low a safety margin as possible and approaching the goal. This goal can be expressed by,

$$\min_{\gamma} J = \alpha_1 \|\gamma\| + \alpha_2 \sum_{t,N_O} \theta + \alpha_3 \sum_{N_O} H,$$

where $\|\gamma\|$ refers to the length of path γ, $\sum_{t,N_O} \theta$ is the sum of safety values $\forall t, \forall N_O$ obstacles, and $\sum_{N_O} H$ is the total of the running hit counter that is defined as follows: for a success metric (and feedback purposes), let a hit be defined as an unsafe encounter, at a single time t, between the robot and an agent n such that

$$\|x(t) - y_n(t)\| \leq r.$$

We seek to evaluate the proposed classification strategy against several alternatives in a variety of dynamic adversarial scenarios.

4.5 Semantic Classification Algorithm

In this section, we define a novel path planning formulation for safety margin adjustment based on agent classifications. The algorithm contains an offline unsupervised learning period and an online path planning execution period.

4.5.1 Unsupervised Learning Problem

The goal of the offline learning section shown in Alg. 4 is to analyze environmental conditions by characterizing observed agents. The word feature describes semantic characterizations derived from state trajectories that are condensed into single descriptive values for analysis. Feature values are calculated from time-series trajectory data in order to perform feature-space classifications of each agent $y_1(t), \ldots, y_{N_O}(t)$ at each time step t. Classification consists of a class assignment in a space created by multiple feature values, which we define as the feature space.

The offline process is shown in Alg. 4, where on line 2, the map f defines features of interest in particular planning scenarios, and may include notions of differential behavior between an agent and the controlled robot, past safety observations, and future trajectory predictions.

On lines 3-6, a feature-space dataset $D \in \mathbb{R}^{N_D \times d_f}$ of N_D agent feature values is collected, where d_f is the dimension of the feature space. Observations of an environment yield T-long trajectory data $\mathbb{O}_m \in (\mathbb{R}^d)^T$, which is condensed into feature data, represented by the symbol Z, through a user-created map $f : \mathbb{X}, \mathbb{O} \mapsto$ feature point p on line 5, such that for agent m and time t,

$$Z(m,t) = f(\mathbb{X}, \mathbb{O}).$$

On line 7, the objective of EM in this formulation (and of the unsupervised learning

problem) is to find k^*-variate Gaussian parameter Φ^* such that,

$$\Phi^* = \arg\min_{\Phi} \frac{1}{N_D} \sum_{i=1}^{N_D} \mathcal{L}(\mathcal{N}(\Phi), i),$$

where $\mathcal{L}$ is the log likelihood that the model $\mathcal{N}(\Phi)$ produced the piece of data i (from the dataset D). The parameter Φ is made of the weights, means, and variances $\Phi = \{W_k \in \mathbb{R} | 0 \leq W_k \leq 1, M_k \in \mathbb{R}^{d_f}, V_k \in \mathbb{R}^{d_f \times d_f}\}, \forall k \in \{1, 2, \ldots, k^*\}$ to make the multivariate model. The Gaussian Mixture Model (GMM) described by Φ^* has a maximal likelihood of producing D, as shown in the example in Fig. 4.2. EM is chosen over k-medians and k-means clustering (based on Lloyd's Algorithm [107]) due to the susceptibility of the latter techniques to arrive at locally, rather than globally, optimal clustering solutions in closely packed scenarios. EM with a GMM was also observationally more resilient to input output perturbation.

Algorithm. 4 Offline Data Analysis

Input: k^*
Output: maps f, g, $\{W, M, V, L\}$
1: Select representative agent features
2: Define feature map $f : \mathbb{X}, \mathbb{O} \mapsto$ feature point $p \in \mathbb{R}^{d_f}$
3: **for** $i = 1 : N$ **do**
4: Observe $\mathbb{X}, \mathbb{O}$
5: Dataset $D(i) \leftarrow f(\mathbb{X}, \mathbb{O})$
6: **end for**
7: $\{W \in \mathbb{R}^{k^*}, M \in \mathbb{R}^{k^* \times d_f}, V \in \mathbb{R}^{k^* \times d_f \times d_f}, L \in \mathbb{R}\} \leftarrow \text{EM}(D, k^*)$
8: **for** $i = 1 : N$ **do**
9: **for** $k = 1 : k^*$ **do**
10: $P(i,k) \leftarrow$MVNPDF$(D(i), M(k), V(k))$
11: **end for**
12: $T(i) \leftarrow \arg\max_{\mathbb{N}_{k^*}}(P(i,:))$
13: **end for**
14: Observe structure of (D, T) and (M, V)
15: Design safety margin loss function J
16: Define map g : class k $\mapsto \theta \, \forall k^*$ classes by $\text{argmin}_\theta J$

On lines 8-13, the model is used alongside the multivariate normal probability density function (MVNPDF) to classify existing data. The model is also used to classify new observations

in the online planning portion shown in Alg. 5. Tagging the dataset allows observations of the structure of the feature space and clustering model. Observations of representative tagged data allows for construction of the safety margin loss function on line 15, which is minimized using gradient descent to select safety margins that smoothly mix safety, runtime, and path optimality.

A sample model, trained on naturally clustered data, is shown in Fig. 4.2 to illustrate the trained model's ability to classify observations based on differences in behavior. Observations placed in the feature space can be classified by the mixture model.

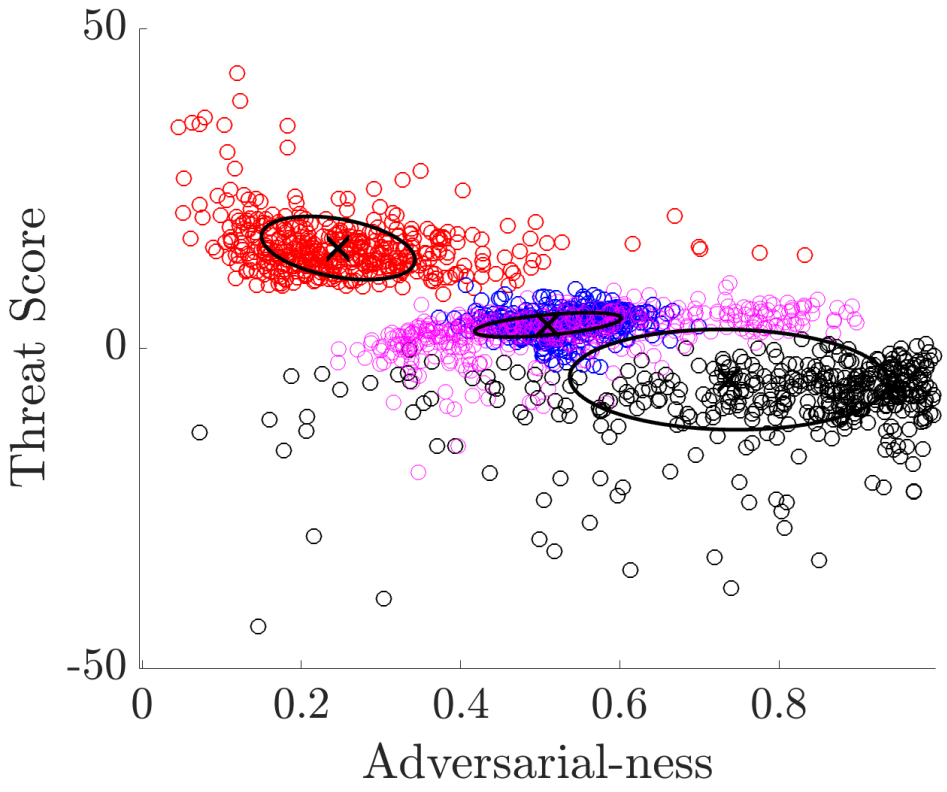

Figure 4.2. Feature-space simulated data is clustered using EM with three components. Data, category means, and data labels are denoted with ∘, X, and colors, respectively. Data in red refers to agents behaving adversarially, blue to static, purple to drifting, and black to avoidant. Covariance rings are depicted as ovals.

4.5.2 Online Planning Algorithm

The goal of this section is to perform safety-aware online planning using the clustering model produced by the offline portion detailed above, where the GMM is used to classify new agent observations. The algorithm takes as inputs start and goal locations x_{start} and x_{goal}, a feature map f and safety margin assignments g, and the mixture model. Of particular interest is mitigation of noise introduced and amplified by the use of a feature map that turns trajectory information into more abstract behavioral notions.

In Alg. 5, on lines 3 and 6, the robot state $x(t)$ and all N_O agent states are observed (at each time step t) and appended to the trajectories $\mathbb{X}$ and $\mathbb{O}$ respectively. If no observation is available at time t for agent m, a dynamic projection can be appended and corrected in future time steps. The trajectory sets $\mathbb{X}$ and $\mathbb{O}$ can contain all prior state information or can be truncated. In order to observe and compile trajectory data, observations of each agent in the environment need to be separable from other agents.

Algorithm. 5 Online Planning

Input: $x_{\text{start}}, x_{\text{goal}}$, maps f, g, $\{W, M, V\}$

1: $x \leftarrow x_{\text{start}}$
2: **while** $\|x - x_{\text{goal}}\| > \delta$ **do**
3: $\quad \mathbb{X}$.append(x)
4: $\quad$ **for** $m = 1 : M$ agents **do**
5: $\quad\quad x_{\text{obs}}(m) \leftarrow$ observe agent m
6: $\quad\quad \mathbb{O}(m)$.append$(x_{\text{obs}}(m))$
7: $\quad\quad Z(m) \leftarrow f(\mathbb{X}, \mathbb{O}(m))$
8: $\quad\quad T(m) \leftarrow \text{Classify}(Z(m), W, M, V)$
9: $\quad\quad \theta(m) \leftarrow g(T(m))$
10: $\quad$ **end for**
11: $\quad x \leftarrow \text{WaypointFinder}(x, x_{\text{goal}}, x_{\text{obs}}, d, \{\theta_1, \ldots, \theta_M\})$
12: **end while**

On line 7, the feature map f transfers trajectories $\mathbb{O}$ into feature-space points $Z(m), m \in \{1, \ldots, N_O\}, t \geq 0$. On line 8, the feature-space points are classified with the novel uncertainty-aware integral method shown in Alg. 6, which outputs a class $T(m)$ for the feature point $Z(m)$. The integral classification algorithm is detailed next. On line 9, the tag $T(m)$ maps from feature

classes to safety margin values, which are updated. After each agent is observed, classified, and has undergone a safety margin update, state information and safety margins are passed to an intermediate waypoint finding routine on line 11.

In order to close the loop from classification to safe set determination, the waypoint finding routine outputs the next intermediate control point that satisfies all safety constraints (as given by safety margin assignments on line 9 of Alg. 5). In the event that all constraints cannot be met due to spatial limitations and/or number of nearby agents, a point is output that violates the fewest number of safety constraints. The waypoint finder uses safety values to minimize hits in the event that staying within the defined safe set is impossible. Any safety-aware controller/waypoint generator can be used, and applications of particular interest include situations where, due to agent number and behavior, guaranteed safe-set behavior (and associated safety guarantees) are impossible.

Algorithm. 6 Uncertainty Integral Classification

Input: Z, W, M, V
Output: T
1: Define feature-space region R around Z
2: **for** $k = 1 : k^*$ **do**
3: $\quad P(k) \leftarrow \int \cdots \int_R \mathbb{P}_{W(k),M(k),V(k)}(Z)\, df_1 \ldots df_{d_f}$
4: **end for**
5: $T \leftarrow \arg\max_{\mathbb{N}_{k^*}}(P)$

The integral classification algorithm shown in Alg. 6 is an alternative to simpler probability-density-function-based classifiers such as the naive Bayes classifier. Instead of making assignments as per the highest likelihood class at point Z, Alg. 6 uses a sense of classification uncertainty to make the assignment as the highest *local* likelihood (rather than *point*).

Alg. 6 takes as inputs a feature-space point Z and the mixture model (here represented by the weights, means, and variances W, M, L). It outputs a class assignment T corresponding to the feature point Z. The classifier uses a novel feature-space integration that allows a sense of uncertainty on the actual feature values of a particular agent/trajectory. Integration is performed

over a region R centered at the feature point of interest Z, as shown on line 1 and in Fig. 4.3. On line 3, for each class represented in $W, M,$ and V, an integration over R of $\mathbb{P}_{W(k),M(k),V(k)}$ at the point Z is performed, where $\mathbb{P}_{W(k),M(k),V(k)}$ is the weighted probability density function of the kth class of the mixture model. The integration variables are each of the d_f variables of the feature space.

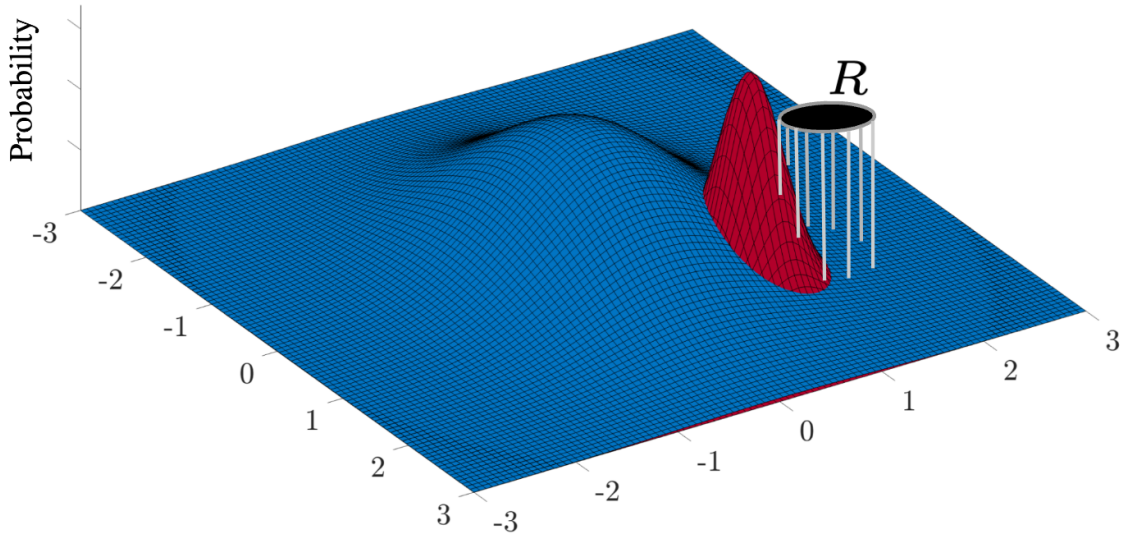

Figure 4.3. The action of the integral classification method is depicted for a region R (depicted floating above the feature space, in black), centered at a feature space point Z on a two component Gaussian Mixture Model (shown in blue and red). The method integrates the region of each component within R and selects the maximum integrated likelihood as the class assignment.

The regional class likelihood is calculated,instead of the point-likelihood which is calculated with a naive Bayesian classifier. This allows the integral classifier to admit locational uncertainty of the feature-space point, enabling robustness to trajectory perturbations that are exacerbated by the map f from trajectory to features, leading to noisy time-series behavior in the feature space. Lastly, on line 5, the maximum regional likelihood over the classes is chosen as the tag T for Z, and the tag is passed back to Alg. 5 for planning.

4.5.3 Divergent Case

To understand a capability of the integral region method, we explore behavior compared to a highest-likelihood classifier in a one-dimensional two-peak Gaussian case with divergent covariances. The probability density function (pdf) of $\mathcal{N}(\mu,\sigma)$ at a point x is,

$$\varphi(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}.$$

Let the generative model contain $\mathcal{N}_1(\mu_1,\sigma_1)$ and $\mathcal{N}_2(\mu_2,\sigma_2)$. The action of a naive Bayesian classifier is given by,

$$f_{\text{pdf}}(x,\mathcal{N}_1,\mathcal{N}_2) = \arg\max_n \ \varphi(x,\mu_n,\sigma_n),$$

and in the two-peak case, $n \in \{1,2\}$.

The integral classification method compares a definite integral of the probability density function through a difference in the cumulative density function (cdf), which for Gaussians can be expressed as,

$$\Phi(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right),$$

where erf is the standard error function given by,

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

The action of the integral classification method is given by

$$f_{\text{cdf}}(x,\mathcal{N}_1,\mathcal{N}_2) =$$
$$\arg\max_n \left(\text{erf}\left(\frac{x+r-\mu_n}{\sigma_n\sqrt{2}}\right) - \text{erf}\left(\frac{x-r-\mu_n}{\sigma_n\sqrt{2}}\right)\right),$$

where again in the two-peak case, $n \in \{1,2\}$. As $\sigma_1 \to 0$, the behavior of $\varphi(x,\mu_1,\sigma_1)$ approaches

the Dirac delta function, with all mass concentrated at μ_1,

$$\lim_{\sigma_1 \to 0} \varphi(x, \mu_1, \sigma_1) = \delta(x - \mu_1).$$

As $\sigma_2 \to \infty$, the behavior of $\varphi(x, \mu_2, \sigma_2)$ approaches a constant function of infinitesimal but nonzero height ε,

$$\lim_{\sigma_2 \to \infty} \varphi(x, \mu_2, \sigma_2) = \varepsilon.$$

A 2D visualization of the divergent covariance cases for a Gaussian distribution is shown in Fig. 4.4.

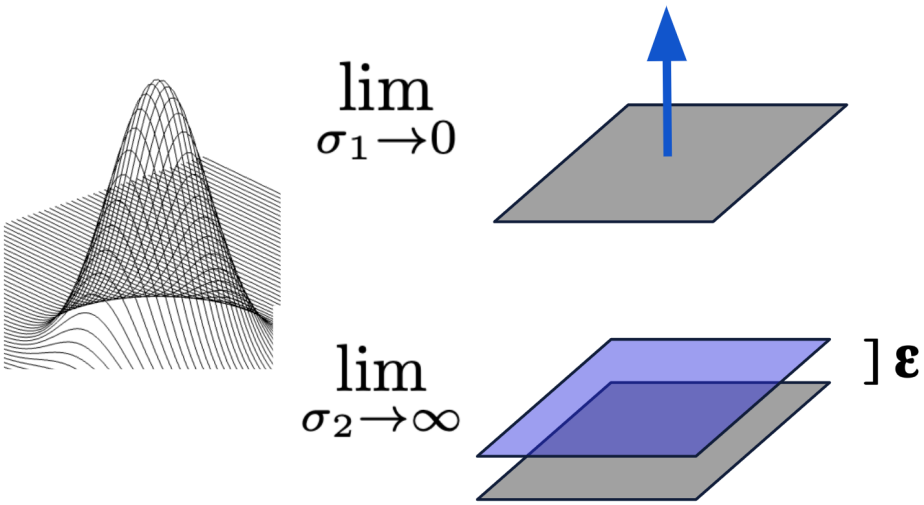

Figure 4.4. Visualization of the divergent Gaussian analysis is shown for the two dimensional case. At left, a generic Gaussian is shown. At top right, a Dirac delta function is shown for the approaching zero covariance case, and at bottom right, a nonzero constant function is shown for the approaching infinity covariance case.

Consider the case where a piece of data x is generated from $\mathcal{N}_1$ and noise is injected into the measurement so as to perturb the data point from $x = \mu_1$ to $x = \mu_1 + \Delta, \Delta > 0$. Under these conditions the pdf of model 1 evaluated at $x = \mu_1 + \Delta$ goes to 0,

$$\lim_{\sigma_1 \to 0} \varphi(\mu_1 + \Delta, \mu_1, \sigma_1) = 0,$$

while the pdf of model 2 evaluated at $x = \mu_1 + \Delta$ goes to ε,

$$\lim_{\sigma_2 \to \infty} \varphi(\mu_1 + \Delta, \mu_2, \sigma_2) = \varepsilon,$$

causing $f_{\text{pdf}}(x, \mathcal{N}_1, \mathcal{N}_2)$ to incorrectly return model 2. This error is caused by the perturbation/noise Δ.

However, when the integration region radius $r > \Delta$, the cdf difference of model 1 evaluated at $x = \mu_1 + \Delta$ goes to 1,

$$\lim_{\sigma_1 \to 0} \left(\text{erf}\left(\frac{x + r - \mu_1}{\sigma_1 \sqrt{2}} \right) - \text{erf}\left(\frac{x - r - \mu_1}{\sigma_1 \sqrt{2}} \right) \right) = 1,$$

while the cdf difference of model 2 evaluated at $x = \mu_1 + \Delta$ goes to $2r\varepsilon$,

$$\lim_{\sigma_2 \to \infty} \left(\text{erf}\left(\frac{x + r - \mu_2}{\sigma_2 \sqrt{2}} \right) - \text{erf}\left(\frac{x - r - \mu_2}{\sigma_2 \sqrt{2}} \right) \right) = 2r\varepsilon,$$

causing $f_{\text{cdf}}(x, \mathcal{N}_1, \mathcal{N}_2)$ to correctly return model 1 as the generative model of the perturbed point x. This is provided that $\varepsilon < \frac{1}{2r}$, a condition that is easily met for two reasons: 1. ε is small even in non-divergent cases when σ_2 is *relatively* large (but finite), and 2. r is a chosen parameter of the integrating method, and should be chosen such that it contains possible disturbances ($r > \Delta$) and does not approach $\frac{1}{\varepsilon}$, which in non-divergent cases can be described as the model 2 *local* height.

The thought process of the preceding example can be extended into the non-divergent case where $\sigma_1 << \sigma_2$, such as is shown in Fig. 4.3, to explain the qualitative advantage of the integral method over a pdf-based method. For $r > \Delta$, the bulk of the model 1 curve can be included in the model 1 integral as compared to the model 2 integral. In the convergent case, when $\sigma_1 = \sigma_2$ but $\mu_1 \neq \mu_2$, the classification choice of the pdf and integral methods are the same. The above can also be extended into higher dimensional representations, albeit with less explicit notation (the cdf of multivariate Gaussians, for example, is not well defined).

The ability of the integral method to correct perturbed and noisy sensor readings and localizations in the feature space allows it to be an effective uncertainty-aware classification method, especially in cases of a highly concentrated (with respect to possible noise) but relatively heavily weighted class (with respect to other classes).

4.6 Results and Discussion

In this section, we show chasing-game simulation results that were performed in a simulated three-dimensional environment run with Matlab v2023a on an 8-core MacBook Pro with M2 chip, followed by real-world chasing-game results with noisy GPS data. Algorithm 5 using Algorithm 6 as a classifier is compared against two similar algorithms using alternative classifiers, a naive Bayesian classifier (NBC) and an aggregated naive Bayes optimal ensemble classifier. The ensemble classifiers are trained on random subsets of the original data, and make class assignments by vote. We defer a more in-depth analysis of ensemble methods to [108]. Comparisons are performed in multiple scenarios to evaluate the ability of the integral classification method as compared to state-of-the-art alternatives.

All simulations are performed against multiple agents, with a two-dimensional feature-space for enhanced visualization and coherence. The used features encode 'adversarialness' of the differential trajectory between robot and agent, and a past hit count combined with a time-to-collision prediction into a single metric. For each cluster, the safety margins are found through gradient descent using a weighted cost function in offline training. The cost function includes number of hits, runtime, number of steps, and the safety value itself in an attempt to balance safety and optimality in a way that does not admit maximal safety values for all clusters.

4.6.1 General Comparison Results

The goal of the chasing and avoiding game simulations is to present a sufficiently difficult and complex scenario where hits from adversarial agents are unavoidable, so that performance can be tested rigorously. To accomplish this, both the robot and each adversarial agent are

performing path planning using the same control algorithm. All of the cases presented in Table 4.1 are run with eight agents for 1000 simulations each. Each simulation consists of a robot that is tasked with moving from one corner of the environment to the opposite, with randomly placed agents. The agent behaviors during this task vary by simulation as described below. The comparison methods are a naive Bayesian classifier (NBC) performing feature-point maximum probability classification over clusters, and an aggregated ensemble classifier with three separately (and randomly) trained NBC modules performing classification by vote. A bootstrap aggregated classifier can reduce classification variance and the tendency to overfit that single classifiers can exhibit.

Table 4.1. Classifier Comparison

	Method	Correct	Hits	Time	Steps
Mixture	NBC	0.459	10.4	0.016 s	39.2
	Ensemble	0.458	9.9	0.041 s	38.3
	Integral	0.656	8.4	0.039 s	39.9
Many Adversary	NBC	0.585	23.3	0.020 s	42.6
	Ensemble	0.583	21.5	0.047 s	41.3
	Integral	0.719	17.7	0.046 s	43.1
No Adversary	NBC	0.436	0.3	0.015 s	37.9
	Ensemble	0.442	0.3	0.040 s	38.1
	Integral	0.533	0.0	0.038 s	38.3

Mixture cases are run with two adversarial agents, two passively and randomly drifting agents, two static agents, and two avoidant agents. Many adversary cases are run with five agents pursuing the robot and one of each other type. No adversary cases are run with two static agents, three passively drifting agents, and three avoidant agents that actively try to run away from the robot. In the correct column, the proportion of correct agent classifications at each time step is shown. For each algorithm, there are on average $320,000$ time step opportunities within the $1,000$ simulations of 8 agents to correctly classify an agent. According to the selected feature metrics, passively drifting and static agents occupy the same place in the feature space and are

therefore clustered together. The three behaviors (adversarial, drifting/static, avoidant) form three clusters in the feature space and correctness is determined by a correct classification of an agent at a time step.

Hits are counted as defined in the problem formulation with $r = 0.7$, and the average total number (from all agents) of hits per simulation is shown in the table. The average wall clock run-time and average number of steps each algorithm took to reach the goal location are both shown in the table for a cost performance in terms of computation time and path optimality.

Table 4.1 highlights the advantages of the uncertainty-aware integral method over the naive Bayes classifier and the ensemble method. The integral method is able to make a significantly larger proportion of correct classifications, with 65.6% correct as compared to 45.9% and 45.8% for the naive classifier and ensemble method respectively. We believe this is due to the integral method's ability to model uncertainty and handle the effect of noise propagation from raw trajectory data to the feature space.

Additionally, the integral method was able to reduce the average number of hits from all agents per problem, from 10.4 for the naive classifier and 9.9 for the ensemble method to 8.4 for the integral method. This is likely stemming from the increased ability to classify correctly, meaning correct classification is closely tied to a notion of safety when safety margins are assigned to classes. The integral method comes at a computation time cost over the naive classifier, with each problem average 0.039s vs 0.016s for the NBC. However, runtime is not a concern as each 40-step simulation is still performed in under a hundrenth of a second, as each step of the integral method is fairly computationally light. Cost in the form of path optimality shows no significant changes.

4.6.2 Categorization Comparison

In this section we compare the behavioral trajectory categorization scheme with integrating classifier against the closest agent trajectory categorization scheme available in literature, the Static-Linear-Trajectory (SLT) method, in a three dimensional obstacle environment with eight

agents. The SLT method breaks trajectories into three classes based on behavior (rather than differential behavior between robot and agent), with largely immobile agents classes as static, agents drifting roughly in lines as linear, and all else as trajectory.

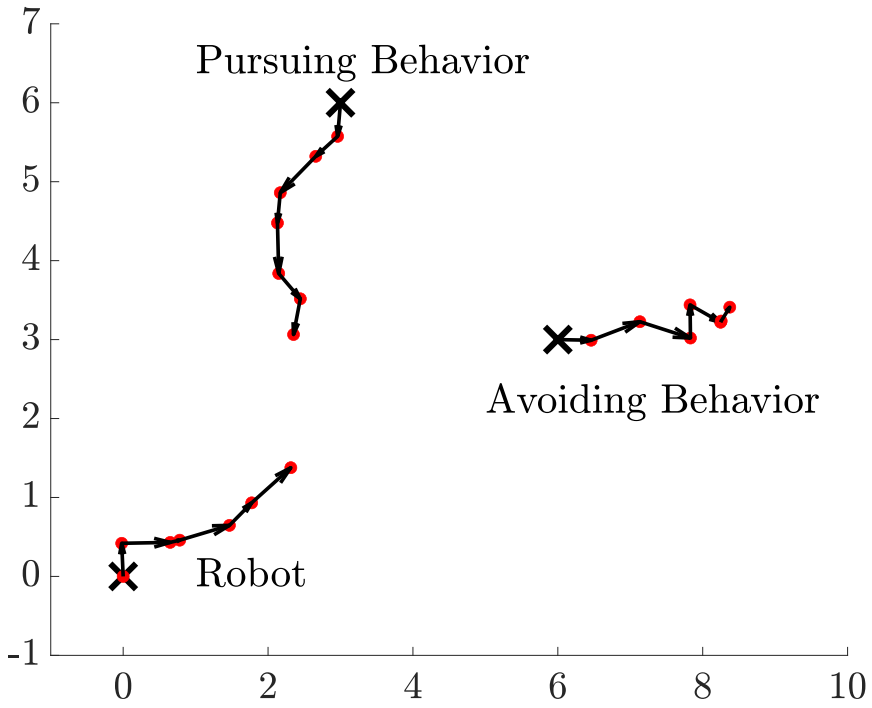

Figure 4.5. Two environmental agents following 'trajectories' yet exhibiting vastly differing behaviors toward a robot in a 2D environment. Each object begins at the *X*.

In Table 4.2, we show comparison results of the two trajectory classification methods in three scenarios: mixture, many adversary, and no adversary, averaged over 1,000 simulations per line. With a mixture of agent behaviors (two adversarial, two avoidant, two static, and two drifting), our behavioral class method averaged 8.5 hits as compared to 15.2 for the SLT method. A similar safety performance advantage is shown in the "many adversary scenario", with five adversarial agents, and one each of avoidant, static, and drifting.

The correct column again measures the proportion of total classification opportunities (per agent, per time step, per simulation) where agents are correctly classified. For the multi-class behavioral method, classifications are counted as correct according to the three behaviors

Table 4.2. Categorization Comparison

	Method	Correct	Hits	Time	Steps
Mixture	Multi-class	0.640	8.5	0.041 s	40.6
	Stat/Lin/Traj	0.910	15.2	0.007 s	37.0
Many Adversary	Multi-class	0.749	17.2	0.047 s	42.9
	Stat/Lin/Traj	0.866	31.7	0.008 s	36.9
No Adversary	Multi-class	0.625	0.0	0.039 s	38.3
	Stat/Lin/Traj	0.939	0.6	0.007 s	37.8

adversarial, drifting/static, and avoidant as explained in the previous section. For the SLT method, both adversarial and avoidant behaviors are correctly identified if the method classifies the agent as trajectory. This greatly increases the ability of the method to "correctly" identify the simplified class, and this is reflected in the near 90% correct identification performance of the SLT method as compared to $60-70\%$ for the multi-class method analyzing a more difficult problem.

Adding behavioral nuance to trajectory feature classification allows more detailed and therefore more safe performance around dynamic agents and adversarial conditions.

4.6.3 Categorization Comparison Experiments

To assess the integrating classifier's ability to correct noisy real-world data, we performed comparative testing on GPS tracks from single-actor chasing-game simulations in a large field, as shown in Fig. 4.6. In table 4.3, the proportion of correct actor tags is shown as a decimal for the three methods, over 450 classification opportunities, split between adversarial behavior, drifting, and avoidant. The NBC was able to correctly identify classes 0.487 of the time, the ensemble method 0.501 of the time, and the integrating method 0.555 of the time.

The relatively lower classification ability of all methods (compared from simulation data to experimental data) is due to both noisier data and smaller sample sizes. This example showcases the ability of the integrating method to filter out noise uncertainty present in the GPS data, which is combined with noise associated with the mapping to the feature space.

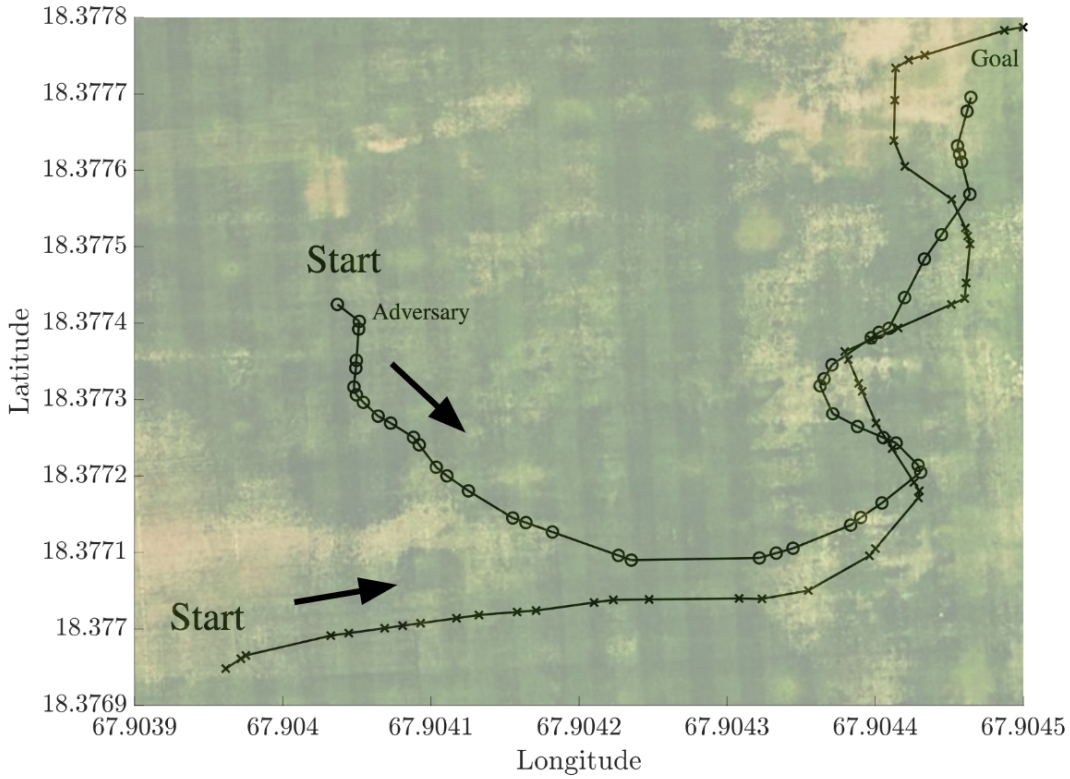

Figure 4.6. Time-series trajectories of a chasing game experiment in a field with a single adversarial actor. The subject robot is shown with the symbol $\times$ and the adversary with the symbol $\circ$.

4.7 Conclusion

The presented algorithm forms safety attitudes toward each agent in a path planning environment and uses those attitudes to automatically adapt safety margins for safer path planning. We present an integrating uncertainty classification method, which makes class decisions for actors and takes uncertainty into account. The algorithm increases safety in simulated environments when compared to alternative classification methods, and uses abstract behavioral characterizations to outperform simpler trajectory classification algorithms in adversarial and unstructured environments. Chapter 4, or portion thereof, is currently being prepared for publication of the material. Lathrop, Paul, Beth Boardman, and Sonia Martínez. "Mobile Robot Behavioral Classification with Uncertainty for Adaptive Safety." In preparation. (2023). The

Table 4.3. GPS Experiment Comparison

Method	Correctness
NBC	0.487
Ensemble	0.501
Integral	0.555

dissertation author was the primary investigator and author of this paper. Future work includes planning with missing trajectory points, multi-robot coordination planning, and implementing a feature-space error flow-map for enhanced predictive classification.

While some motion planning methods, such as the safety margin adaptation scheme presented in this chapter, are amenable to online execution, safety-related algorithmic development has also trended in another direction: that of provable safety, reinforcement learning (RL), and optimization. Efficient computation of actions in RL schemes, optimal motion plans, and provably reachable and safe trajectories can be difficult at best on GPU-enabled parallel-computing hardware. Computing time, computing device metrics, and symbolic comparative runtime analysis are reported in a fraction of cases, and failure to report or analyze algorithm computational costs represents a failure to properly frame work in existing literature. The rise of parallel computing, especially through multi-core computers and through the use of GPUs, has opened a new landscape in algorithmic development: that of parallelization. In the next chapter, we turn our attention to an even newer landscape of computational efficiency, through the lens of sampling-based motion planning algorithms.

Quantum computing represents a fully-parallelized architecture, with idyllic potential computational advantages ripe for algorithmic adaptation. But, as is true of many things, with inordinate advantages comes a fresh Pandora's box of complications to address, questions to ponder, and drawbacks to mitigate.

Chapter 5

Quantum Computing for Motion Planning: Quantum RRT

5.1 Introduction

The emergence of digital electronic computing in the 1940s and 50s brought widespread changes to virtually every area of human life. More recently, in 1980, Paul Benioff presented the quantum Turing machine [109], which outlined a simple computer using the principles of quantum mechanics to represent mixed states. The concept of quantum gates [110], which fulfill a similar function to the binary logic gates of classical computing, paved the way for the emerging field of quantum computing. Physically different from traditional computing, quantum computers leverage the quantum mechanical properties of physical matter to perform calculations simultaneously. Quantum computation is on the horizon and awaits the development of reliable physical mediums to be used in practice [111]. Candidates for physical implementation of quantum bits (qubits) include superconducting circuits [112] (with information storage in harmonic oscillations between energy levels of an inductor-capacitor circuit), the trapped ion quantum computer [113] (with information storage in stable electronic ion states), and the semiconductor quantum dot quantum computer [114] (with information storage in nuclei spin states). However, the theory behind quantum computing is well established and has shown the potential to dramatically impact the solutions to many complex problems, such as in physics [115, 116] and chemistry [117] simulations, cryptography [118, 119], optimization [120, 121], and

machine learning [122].

Quantum algorithms such as Grover's Algorithm and its generalization, Quantum Amplitude Amplification (QAA), have a proven quadratic speedup in unstructured database searches when compared to classical algorithms [39, 123]. We believe this property allows quantum algorithms to parallelize computationally heavy steps in motion planning. Quantum search algorithms have been applied to several areas within robotics, such as machine learning and estimation, but have yet to be applied to sampling-based motion planning algorithms. Motivated by this, we seek to explore how quantum algorithms and quantum speedup can be applied to sampling-based motion planning algorithms in complex spaces with dynamic constraints.

5.2 Literature Review

In this section, we provide a brief account of related works employing quantum computation in incidental problems in robotics, planning, and control theory. This is followed by a brief overview on sampling-based motion planning.

With respect to motion planning, quantum algorithms have been applied to reinforcement learning in [29, 124–126]. Quantum methods have been shown to increase speed [124] and robustness [125] of state-action pair learning algorithms in gridded environments when compared to temporal difference epsilon-greedy and softmax choice strategies. Quantum reinforcement learning [29] relies on encoding the state-action set as an eigen-state eigen-action set, with probability amplitudes characterized by quantum states in order to update the value function [127]. As is well known, exact reinforcement learning does not scale well to high-dimensional discrete state and action spaces. Even when using neural-network function approximations, the identification of the best reward functions for planning tasks in complex environments is an open question [128]. Instead, we seek to apply quantum computing methods to sampling-based motion planners to solve simpler path feasibility problems. This has the advantage to provide fast solutions in multi-dimensional environments with probabilistic completeness guarantees [6].

Simple robotic trajectory planning is addressed in the work [129], which uses the Quantum Evolutionary Algorithm [130], to obtain optimal trajectories with respect to an obstacle-distance-based objective function. A quantum genetic evolutionary algorithm is shown to compute trajectories in a two dimensional obstacle environment using a population-crossover-mutation workflow. This is enabled via particle swarm optimization (PSO); however, it is known that PSO approaches to motion planning suffer from a host of problems, including premature convergence, the inability to adapt to high dimensional search spaces (due to local optima traps and the potential to be restricted to a sub-plane of the entire search hyperplane), ambiguity in optimizer form (to yield both useful motion plans and solutions via PSO), and ad-hoc parameter tuning [131].

Quantum methods have been applied to several other motion-planning-adjacent areas within robotics. [40] outlines the state of the art of quantum computation (in terms of quantum algorithms) in robotic science and helps frame open future research topics on sensing and perception, "traditional artificial intelligence" such as graph search algorithms, the integration of quantum computers into robotic and distributed systems, and testing frameworks for quantum computation. In particular, combinatorial graph search algorithms may be amenable to quantum speedup through the application of Grover's Algorithm, quantum annealing, or quantum random walks. Additionally, [40] outlines applications of quantum algorithms to inverse kinematics and optimal planning problems for manipulators, by means of static optimization and model predictive control approaches. Here, we evaluate the integration of Grover's Algorithm and its extension, QAA, with sampling-based motion planners. While this is unaddressed in [40], it aligns with the general proposed research agenda. The review [132] outlines the state of the art of quantum mechanics and quantum control algorithms, addressing questions of controllability, open and closed loop control, and feedback control methods through the lens of quantum computing. The work at hand focuses on the computation of motion plans in obstacle environments with the help of quantum algorithms, rather than on the computation of feedback controls for quantum systems.

The speed up of search algorithms via quantum computation has also received attention from other application areas; see the textbook [133]. In particular, Grover's Quantum Search Algorithm has been used in [134] to search a physical region, with special focus on 2D grids, with the goal of addressing information storage constraints. The authors define quantum query algorithms on predefined graphs, which could in theory be applied to algorithms such as the A* graph search algorithm [5]. However, a proven advantage of sampling-based motion planners over A* approaches is that they automatically tune their resolution as the number of samples increases.

Quantum walks are used in [135] to find a marked element in a discrete and finite state space. If the quantum walk is ergodic and symmetric, quadratic speedup is achieved with respect to classical Markov-chain counterparts [136]. Similarly, quantum walks have been applied to search over more abstract spaces; see [137] on search engine network navigation. Quantum walks are an extension of classical random walks, and they require state space discretization. Instead, we seek to extend quantum speedup to tree-based planners that use randomness to find samples in continuous spaces, rather than performing motion planning over a discrete graph with random walks. This approach has been proven to efficiently solve difficult planning problems compared to methods based on discrete counterparts, and can also better handle robot dynamics.

Compared to other motion planning paradigms, sampling-based motion planning avoids explicit construction of obstacle spaces in favor of performing collision checks on generated samples [6]. We provide an introductory set of references, and readers are encouraged to consult the textbook [6] for further reading. In sampling-based motion planning, the most commonly used algorithms are Probabilistic Roadmaps (PRM) [8] and the Rapidly-exploring Random Trees (RRT) [9], both of which provide samples to grow graphs and trees respectively. These algorithms have been extended and modified in their sampling strategies [138, 139], exploration [140, 141], collision checking [142, 143], speed and optimality [14, 144, 145], and kinodynamic constraint satisfaction [17, 146], among other parameters and heuristics. An extended review of the field of sampling-based motion planning and the relative merits and advantages of extending motion

planning algorithms to satisfy certain parameters can be found at [27]. In this chapter, we apply quantum algorithms to basic RRTs specifically as they are able to find fast solutions in multi-dimensional systems, with no discretization required, and can account for robot dynamic constraints. This has made possible their widespread application in autonomous vehicle motion planning and complex object manipulation. Moving forward, the benefits of this approach can only be enhanced by integration with quantum computing tools. To the best of our knowledge, this work takes a first step in this direction.

Algorithm parallelization is related to quantum computation, as the heart of quantum speedup lies in the ability to perform simultaneous calculations on superpositions of states [38, 111]. Motion planning algorithms have been rewritten for multi-threading [147], parallel tree creation [148], and parallel computation with GPUs [28]. In [147], the authors devise a message passing scheme and compare performance of several parallel RRT schemes, such as OR Parallel RRT, Distributed RRT, and Manager-Worker RRT. The work [28] identifies the collision checking procedure as the computationally expensive portion of sampling-based motion planning and seeks to parallelize it. We therefore target the collision checking procedure as the main candidate for quantum computing speedup. Although parallel computation is not always a tractable solution, as with single tree creation, path planning in dense spaces with dynamic constraints can benefit from parallelization for quantum algorithm application.

As is detailed above, quantum search algorithms have been applied to several areas within and adjacent to robotics, such as optimization, machine learning, and estimation, but have yet to be directly applied to sampling-based motion planning algorithms, which is what we seek to accomplish here.

In this chapter, we present a novel formulation of traditional sampling-based motion planners as database-oracle structures that can be solved via quantum search algorithms. We consider two complementary scenarios: for simpler sparse environments, we formulate the Quantum Full Path Search Algorithm (q-FPS), which creates a superposition of full random path solutions, manipulates probability amplitudes with Quantum Amplitude Amplification

(QAA), and quantum measures a single obstacle free full path solution. For dense unstructured environments, we formulate the the Quantum Rapidly Exploring Random Tree algorithm, q-RRT, that creates quantum superpositions of possible parent-child connections, manipulates probability amplitudes with QAA, and quantum measures a single reachable state, which is added to a tree. As performance depends on the number of oracle calls and the probability of measuring good quantum states, we quantify how these errors factor into the probabilistic completeness properties of the algorithm. We then numerically estimate the expected number of database solutions to provide an approximation of the optimal number of oracle calls in the algorithm. We compare the q-RRT algorithm with a classical implementation and verify quadratic run-time speedup in the largest connected component of a 2D dense random lattice. We conclude by evaluating a proposed approach to limit the expected number of database solutions and thus limit the optimal number of oracle calls to a given number.

5.3 Contributions

We introduce two novel formulations of path planning algorithms using QAA. In Quantum Full Path Search (q-FPS), we describe a quantum search over a database of randomly generated paths from a start to a goal configuration over sparse environments. Next, we describe a Quantum Rapidly Exploring Random Tree (q-RRT) algorithm that admits reachable states to the tree through a quantum search of a randomly constructed database of points.

The main contributions of this work are the following.

- Creation of a strategy for achieving path planning using quantum computing in sparse environments with Quantum Full Path Search (q-FPS);
- Re-framing of RRTs for quantum computation with the algorithm Quantum RRT (q-RRT);
- Analysis of the probabilistic completeness (PC) properties and derivations of key probability values of interest with respect to adding unreachable tree elements;

- Characterization of oracle and measurement errors, how these errors affect PC, and how to ensure PC properties remain intact;
- Simulations of the use of quantum algorithms for sampling-based motion planning and verification of quadratic speedup;
- Numerical simulations regarding connectivity within 2D square random lattices for optimal QAA application and the creation of a sampling method for selecting (rather than estimating) the optimal number of QAA applications.

5.4 Full Path Database Search with Quantum Amplitude Amplification

In this section, we outline a first algorithm for path planning based on a direct application of QAA, with an illustration of its advantages over classical methods in a particular example.

We outline a path planning algorithm, Quantum Full Path Search, Alg. 7 (q-FPS), which uses QAA to search a database D of completed paths. The robot is described by state $x \in \mathbb{R}^d$ which is constrained within a compact configuration space, $C \subseteq \mathbb{R}^d$. Let C_{free} denote the free space, or the space within C outside of all static obstacles. The goal is for the robot to navigate a path, in C_{free}, from the initial state $x_0 \in C_{\text{free}}$ to a goal state $x_{\text{G}} \in C_{\text{free}}$. The path is denoted as an ordered set of states $\gamma : x_0, x_1, \ldots, x_{\text{G}}$. For the path to be considered safe, $x_i \in C_{\text{free}}, \ \forall\, i$. Continuous path curves can also be considered.

Algorithm 7, the Quantum Full Path Search (q-FPS) takes as input the initial and goal states, the desired number of quantum registers n (for database size 2^n), and an oracle function $\mathcal{X}$. The algorithm output is a path $\gamma \in C_{\text{free}}$ from x_0 to x_{G}.

The q-FPS algorithm relies on the creation of a database of full length path solutions on line 3. In order to create a database that is likely to contain solutions, random paths should deviate from straight line behavior. In more complex or blocked environments, higher deviation alongside larger database sizing n can lead to a higher likelihood of a valid solution. In Alg 7,

Algorithm. 7 Quantum Full Path Search (q-FPS)

Input: x_0, x_G, n, oracle function $\mathcal{X}$
Output: γ: $x_0, x_1, \ldots, x_G$

1: Init Database D
2: **for** $i = 1$ to 2^n **do**
3: $\quad D(i) \leftarrow$ random path from x_0 to x_G
4: **end for**
5: $m = \text{QCA}(\mathcal{X}, D)$
6: Enumerate D via $F : \{0,1\}^n \rightarrow D$
7: Init n qubit register $|z\rangle \leftarrow |0\rangle^{\otimes n}$
8: $|\Psi\rangle \leftarrow \mathbf{W}|z\rangle$
9: **for** $i = 1$ to $\left\lfloor \frac{\pi}{4}\sqrt{2^n/m} \right\rfloor$ **do**
10: $\quad |\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
11: **end for**
12: $\gamma \leftarrow F(\text{measure}(|\Psi\rangle))$
13: Return γ

on line 5, QCA refers to the Quantum Counting Algorithm [149], an extension of Grover's algorithm and the quantum phase estimation algorithm that estimates directly the number of solutions within the database. Line 6 refers to a $1-$to-1 mapping from the elements of database D to states of a qubit register. It can also be thought of as a numbering scheme. Let $\mathbf{W}$ be the Walsh-Hadamard transform.

In the loop, from lines 9 to 11, we apply the QAA operator (combined with oracle $\mathcal{X}$) to the qubit multiple times to increase the amplitude of correct database entries. The exact number of iterations depends on the database size 2^n and the number of solutions m in D, as discussed in Section 5.5. In this application, the oracle $\mathcal{X}$ functions as a black box indicating whether a path is obstacle collision-free. If m is known, then the number of applications of Q that maximize the feasible paths amplitudes is,

$$i_{\max} = \left\lfloor \frac{\pi}{4}\sqrt{2^n/m} \right\rfloor \; ; \tag{5.1}$$

see [150]. If Q is further applied, the amplitudes of correct solutions will start to decrease, as shown in Fig. 5.2. Lines 12 to 13 refer to the process of measuring the qubit, retrieving the database path element, and returning said path.

This method provides us with a quantum algorithm approach to motion planning problems with a quadratic speedup over the same method using classical search algorithms. Speedup is effected on path collision-checking, which is the most computationally heavy portion of path planning.

5.4.1 q-FPS Example

We illustrate the algorithm and speedup on the following example. The probabilities are known because we simulate the quantum computer on a classical device. Consider a randomized database in a $2-$dimensional obstacle environment using a $n = 10$ register qubit corresponding to a database with 1024 random paths. A visualization of a sample full path databse is shown in Fig. 5.1. Let there be a total of $m = 5$ obstacle free solutions within D (as measured by QCA) and Q will be applied to the equal-superposition qubit $|\Psi\rangle$ a total of $i = \lfloor 11.24 \rfloor$ times, calculated using Eq. (5.1). After 11 iterations, the total probability of measuring one of the 5 correct solutions is 99.86% and the total probability of measuring one of the 1019 incorrect solutions is 0.14%, as shown in Fig. 5.2. Classically (on a non-quantum computer), the expected value of oracle calls to find one of five solutions in a database of size $N = 1024$ with $m = 5$ solutions is $(N/m)/2 = 102.4$.

5.5 Quantum Rapidly Exploring Random Tree Algorithm

The approach of the previous section only works successfully for obstacle-sparse environments, as randomly generated full paths are very unlikely to find a valid, obstacle free path when the density of obstacles is high. Instead, RRTs and Probabilistic Roadmaps (PRMs) [67] are devised to produce successful collision free-paths more quickly in cluttered environments. In this section, we outline the q-RRT Algorithm (Alg. 8), an RRT-like path planning algorithm, , which is based on RRTs. The q-RRT algorithm uses QAA on a database of individual points during tree creation to only admit reachable points that are within the same connected component. The main algorithmic differences between the q-RRT algorithm and RRT are as follows:

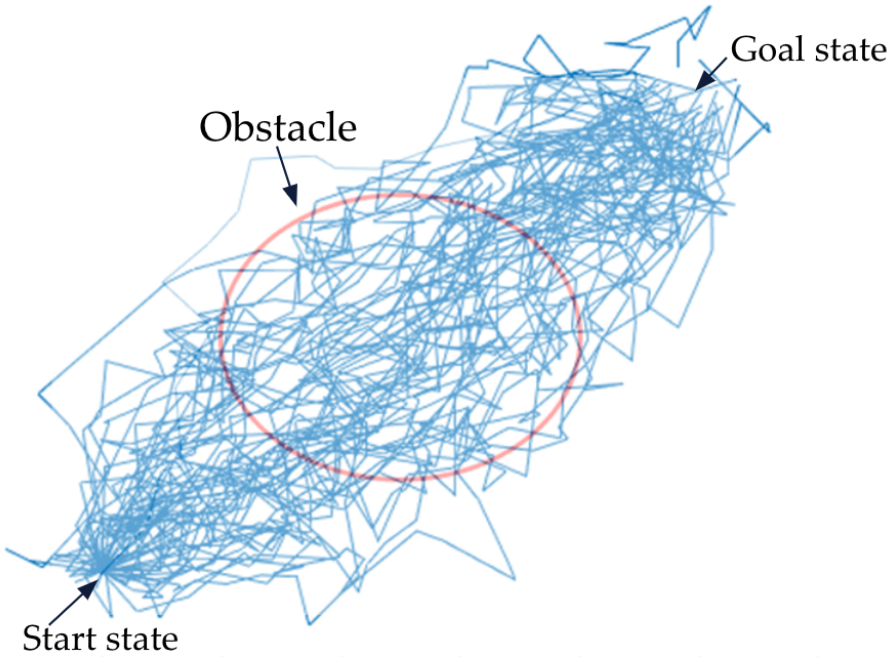

Figure 5.1. A visualization of a full path search database in a two dimensional environment. Randomly generated full path solutions are depicted in blue, and a single circular obstacle is depicted in red.

- q-RRT creates databases of possible states to analyze simultaneously, rather than single states.
- States are assessed simultaneously for addition to the tree using quantum algorithms and measurement.
- A metric, p^*, is used to estimate the number of correct database solutions.

We analyze the algorithm performance in a d-dimensional finite square (lattice) environment $C \subseteq \mathbb{R}^d$. The reason for this choice is twofold: firstly, there are established tools, methods, and theory regarding them, and secondly, they can yield sufficiently dense and scattered environments to provide an interesting study. Related applications include cave exploring or search and rescue efforts in collapsed structures [151].

The lattice environment is shown in Fig. 5.4 and is defined as a square region $C = \bigcup_{i\in\mathbb{N}} S_i \subseteq \mathbb{R}^d$ that is partitioned into equal sized squares ($d = 2$), cubes ($d = 3$), or hypercubes

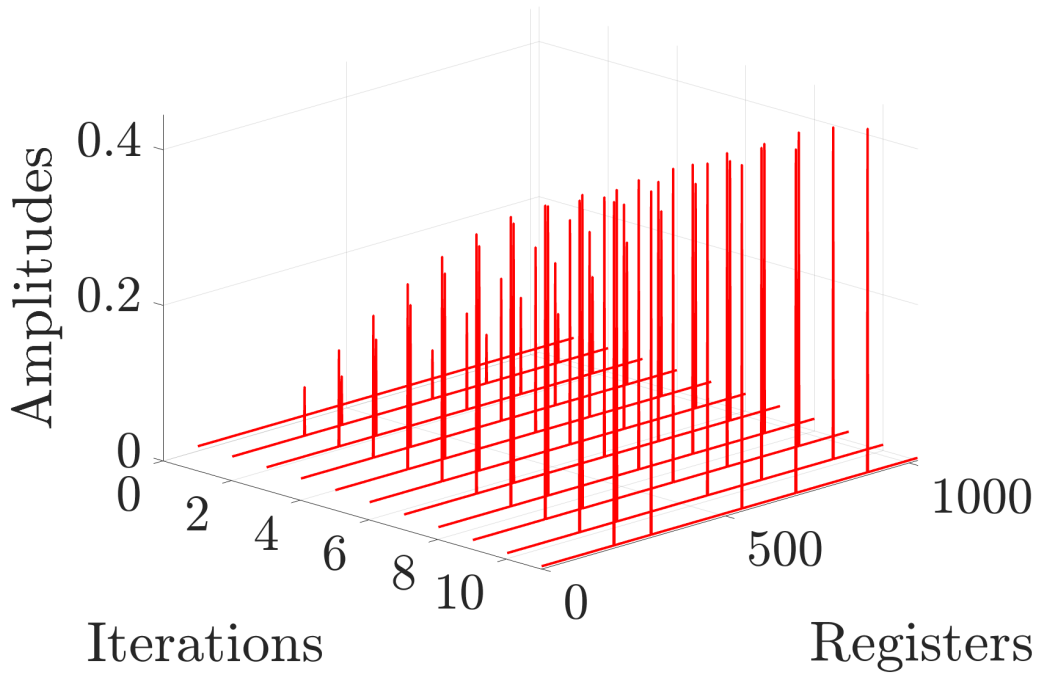

Figure 5.2. Effect of repeated applications of operator Q on probability amplitudes of a 2^{10} qubit representing a database with 5 free paths. Each register corresponds with a database element. The amplitudes of non-collision-free paths is shown as a small (non-zero) magnitude line that decreases with increased iterations. Further applications of Q decrease amplitudes of free paths.

($d > 3$) S_i, $i \in \mathbb{N}$. Each element is either obstacle free with probability $1-r$ or occupied with probability (or concentration) r. Obstacle free elements are denoted by white in our figures, and form $C_{\text{free}} \subseteq \mathbb{R}^d$, and occupied elements are denoted by black and form $C_{\text{obs}} \subseteq \mathbb{R}^d$. The characteristic length L is the ratio of the side length of C to the square increment spacing. In this section, we allow the lattice spacing to be defined as size one and the side length of C to be L.

Two d-dimensional elements are adjacent in $\mathbb{R}^d$ if and only if they share a $d-1$ edge. For $d = 2$, adjacency is defined for edges and not corners. Let a connected component Z be a set of adjacent grid cells $\bigcup_{i=1} S_i$ such that, $S_i \subseteq C_{\text{free}}, \forall i$, and any two points $x_1, x_2 \in Z$ be connected by a continuous path $\gamma \subseteq Z$.

5.5.1 Quantum RRT Algorithm

The q-RRT algorithm, Alg. 8, takes as inputs an initial point $x_0 \in C_{\text{free}}$, the number of qubit registers n, a number of nodes M, the oracle function $\mathcal{X}$, a concentration r, and the characteristic length L. It outputs a connected tree T of M reachable states (or tree nodes) from x_0. We note that, traditionally, RRTs end when a goal is found and return a path. Instead, the goal is to construct an RRT that ends when the given number of nodes M are added successfully to the tree, providing a type of PRM.

To add a node, q-RRT creates a size 2^n database D of random states-nearest parent pairs, as shown in lines 3 through 7. The nearest parent in this context is defined using the d-dimensional Euclidean distance. On lines 8 to 10, a $1-$to-1 database-element-to-qubit mapping is created and an equal superposition is created across all qubit states. Recall that $\mathbf{W}$ is the Walsh-Hadamard transform, the equal superposition operator. On lines 12 through 14, QAA is performed on $|\Psi\rangle$ a repeated number of times (as per Eq. (5.1)) based on an estimate of number of solutions m on Line 11, where p^* refers to estimates of $m/2^n$. A single database element is added to the tree on line 16 based upon the quantum measurement on line 15. The oracle function performs a reachability check (within the operator Q) with a local planner on the random point t from the proposed parent point P to certify that the returned tree is fully reachable. Our specific local planner for simulation is explained in Section 5.5.4, and a more general discussion on reachability estimations can be found in Sec. 5.6.1. We note that the method is defined as RRT (but can be extended to RRT* through the addition of standard rewiring after line 17) in order to apply quantum algorithms to the most broadly applicable sampling-based motion planner.

5.5.2 Probabilistic Completeness and Probability Results

This section analyzes the effect of two sources of error that can affect probabilistic completeness (PC) and the admission of unreachable states to the tree in q-RRT, leading to wrong solutions. These are imperfect oracles and the measurement process. The following discussion

Algorithm. 8 Quantum RRT (q-RRT)

Input: x_0, n, M, oracle $\mathcal{X}$, r, L
Output: Tree T

Init T with root at x_0
while size$(T) < M$ **do**
  for $i = 1$ to 2^n **do**
    $t =$ random point
    $P =$ closest parent of t in T
    $D(i) = [t; P]$
  end for
  Enumerate D via $F : \{0,1\}^n \rightarrow D$
  Init n qubit register $|z\rangle \leftarrow |0\rangle^{\otimes n}$
  $|\Psi\rangle \leftarrow \mathbf{W}|z\rangle$
  $p_1^* = p^*(r,L),\ p_2^* = p^*\left(r, L/\sqrt{\text{size}(T)}\right)$ from Eq. (5.14)
  for $i = 1$ to $\left\lfloor \frac{\pi}{4}\sqrt{1/p_1^*} \right\rfloor$ **do**
    $|\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
  end for
  $[x_{\text{last}}, P] \leftarrow F(\text{measure}(|\Psi\rangle))$
  Add $[x_{\text{last}}, P]$ to T
end while
Return T

and statements apply to any path planner with similar inaccuracies.

In what follows, we define PC with respect to q-RRT. For any x_1 and x_2 that belong to the same connected component $Z \subseteq C_{\text{free}}$, it requires that:

A: Eventually $x_2 \in T$, for T rooted at x_1 with probability 1.

B: $\exists$ a good path from x_1 to x_2 in T with probability 1.

We relax this standard definition to just A for the following Lemma and we address B in Thm. 3. When there are no errors, A is sufficient because every node admitted to T is reachable. We show how q-RRT can meet these criteria in Lemma 1.

Lemma 1. *For every $x_1, x_2 \in Z$, where $Z \subseteq C_{\text{free}}$ is a connected component, the output tree T of q-RRT with a final check, with root x_1 satisfies $\mathbb{P}(x_2 \in T) \rightarrow 1$, as the number of tested samples goes to ∞.*

Proof. The proof follows from the probabilistic completeness of RRTs [67]. The output of RRT,

T_{RRT}, satisfies $\mathbb{P}(x_2 \in T_{\text{RRT}}) \to 1$ as the number of samples $\to \infty$. All points in C will be tested for addition to $T_{\text{q-RRT}}$, similar to T_{RRT}, and reachable states will be admitted to $T_{\text{q-RRT}}$. This result holds for the output of q-RRT, $T_{\text{q-RRT}}$, because the sampling distribution (and process for selecting and admitting states) and configuration space satisfy the same conditions as the proof for RRT, as explained next.

In database creation, q-RRT uses independent uniform sampling of points from within C, where C is a nonconvex bounded open n-dimensional configuration space. This distribution is multiplied by the probability of tagging each of these states as good by the oracle process (regardless of whether they are good or bad as ground truth), and by the probability of measuring one of these states to be added to the tree. It holds that $T_{\text{q-RRT}}$ contains a tree $T_{\text{RRT-m}}$, which is created with only correctly identified samples (generated by a uniform distribution over C) that have been measured. The latter net distribution satisfies the necessary conditions for the RRT result, namely that it is a smooth strictly positive probability density function over the connected component $Z \subseteq C_{\text{free}}$ of interest. Then, $T_{\text{RRT-m}}$ satisfies the theorem of RRT, and $\mathbb{P}(x_2 \in T_{\text{RRT-m}}) \to 1$. Since we have $\mathbb{P}(x_2 \in T_{\text{RRT-m}}) \leq \mathbb{P}(x_2 \in T_{\text{q-RRT}}) \leq 1$, the result follows. $\square$

If the oracle in Alg. 8 Line 13 is imperfect, reachable states may be tagged as unreachable (false negative oracle error) and vice versa, unreachable states may be tagged as reachable (false positive oracle error), as shown in Fig. 5.3. An "imperfect oracle" is one that admits any type of error. False negative errors reduce efficiency and have the potential to remove PC properties, as good states may not be added to the tree. False positive errors serve to increase the likelihood that unreachable states are admitted to the tree. The local planner employed does not make repeatable false negative errors, as reachability is defined with respect to a current state, and as the current state approaches the target state (as discussed later), if the target state is reachable, the oracle will identify it as such. Therefore, oracle false negative errors do not affect PC properties.

These errors are compounded with those introduced by the measurement step on Alg. 8 Line 15, which may admit unreachable states to the tree (additional false positive measurement

error), but because the measurement produces a reachable output (and not a tag like the oracle), additional false negative measurement error is not possible.

We analyze these error-measurement likelihoods next, and their impact on property B. First, we note that the false positive measurement error can be mitigated through a final deterministic oracle check before a state is added to T. We call this the "final check", to be applied after Alg. 8 Line 15, to verify that the measured node is indeed reachable with an obstacle-free path before it is added to T, allowing us to use the PC definition according to solely criteria A. However, this final check comes at a cost of additional oracle calls.

Measurement error stems from the probabilistic nature of the qubit measurement process (Alg. 8 Line 15). In general, there is a nonzero probability that a database element marked (by the oracle) as bad is selected for addition to T (false positive measurement error). The quantum measurement process takes a qubit and returns a deterministic state, where the returned state probability of selection is the square of the probability amplitude (Born's rule) [152]. In general, the probability amplitude of bad states after successive applications of Q is nonzero, and the following theorem provides a characterization of this probability and its impact on criterion B.

Theorem 3. *Let E be the event of a bad state, as tagged by the oracle (regardless of ground truth), being added to T on a particular qubit measurement (false positive measurement error). Let database sampling be uniform over C and let the database be optimally amplified. The probability of E is,*

$$\mathbb{P}(E) = 1 - \sin^2\left(\left(\frac{\pi}{2}\sqrt{\frac{2^n}{m}} + 1\right)\arcsin\left(\sqrt{\frac{m}{2^n}}\right)\right), \tag{5.2}$$

where 2^n is the current database size and m is the current number of solutions within the database. Eq. (5.2) *is the minimum value that is achieved when Q is applied exactly according to Eq.* (5.6)[1].

[1]Functionally, Eq. (5.2) will be modified by the fact that Q is applied an integer number of times.

As the number of nodes $M \to \infty$, $\mathbb{P}(E)$ monotonically increases to $\lim_{M\to\infty}\mathbb{P}(E) \equiv \mathbb{P}(E_{\text{lim}})$,

$$\mathbb{P}(E_{\text{lim}}) = 1 - \sin^2\left(\left(\frac{\pi}{2}\sqrt{1/r}+1\right)\arcsin\left(\sqrt{r}\right)\right), \tag{5.3}$$

where r is the environment concentration. Lastly, let F be the event that at least one bad state exists within T. When M nodes are in T, an upper bound on the probability of F is,

$$\mathbb{P}(F) \leq 1 - (1 - \mathbb{P}(E_{\text{lim}}))^M, \tag{5.4}$$

and an upper bound on the probability that at least one bad state is part of a given path γ,

$$\mathbb{P}(F_\gamma) \leq 1 - (1 - \mathbb{P}(E_{\text{lim}}))^{|\gamma|}, \tag{5.5}$$

where $|\gamma|$ is the number of nodes in γ.

We remark that there is no way of finding lower bounds similar to Eq. (5.4) and Eq. (5.5), as the expected lower bound value of Eq. (5.2) depends on the local planner. In this case, Eq. (5.4) and Eq. (5.5) form expected worst-case estimates to tree and path errors, respectively, when using q-RRT.

Proof. First, we note that the optimal number of applications of Q to maximize the chance of a good measurement is,

$$i_{\max} = \frac{\pi}{4}\sqrt{\frac{2^n}{m}}\,, \tag{5.6}$$

as given in [39]. We further note that, after $i_{\max}$ iterations, the probability of measuring a good state is,

$$\mathbb{P}(E^c) = \sin^2((2i_{\max}+1)\theta), \tag{5.7}$$

where θ is defined such that $\sin^2(\theta) = \frac{m}{2^n}$ [39], and where $\frac{m}{2^n}$ is the success probability of the database. Thm. 3 Eq. (5.2) follows via substitution. For local planners testing reachability, as

$M \to \infty$, in the maximal case the entirety of C_{free} becomes locally reachable. Therefore, the ratio of correct database solutions $2^n/m$ approaches the environment concentration r, yielding Thm. 3 Eq. (5.3).

Lastly, we observe that $\mathbb{P}(E)$ is upper bounded by Eq. (5.3) and $\mathbb{P}(E)$ is strictly increasing as a function of m, over our entire effective domain of $m/2^n = (0.04 \;\; 0.75)$. If we assume the upper bound for each node in the tree, Thm. 3 Eq. (5.4) follows by substituting Eq. (5.3) into the probability formula of at least one $\mathbb{P}(E)$ occurring over M events. Eq. (5.4) is an upper bound over the number of nodes M, as it is found by assuming an upper bound value occurs in every case. This is modified by replacing the power M for $|\gamma|$ for the case of a path γ with node length $|\gamma|$, yielding Eq. (5.5). □

We note that for databases with less than 75% solutions, $0 < m < 0.75 * 2^n$, $\mathbb{P}(E)$ is strictly increasing as the fraction of solutions in the database $m/2^n$ increases. It is also well approximated by a linear function, $\hat{\mathbb{P}}(E) = 1.251\frac{m}{2^n} - 0.0159$, achieved with linear least squares on $m/2^n = (0.04 \;\; 0.75)$ with coefficient of determination $R^2 > 0.999$. With a local planner testing reachability, in general, as $M \uparrow$, the number of database solutions $m \uparrow$. We defer this discussion to Section 5.5.4.

The above quantum measurement error analysis is modified in Prop. 4 to additionally account for false positive and false negative errors by the oracle.

Proposition 4. *Let G be the event that a good state, with respect to ground truth (rather than as tagged by the oracle), is measured for addition to the tree. Let the probability of a state marked incorrectly as good be given by $q \in [0,1]$ (false positive), and let the probability of a state marked incorrectly as bad be given by $v \in [0,1]$ (false negative). Let the database be optimally amplified. Then, the event G has probability,*

$$\mathbb{P}(G) = (-1 + q + v)\,\mathbb{P}(E) + 1 - q, \tag{5.8}$$

where $\mathbb{P}(E)$ is given by Eq. (5.2)*. As the number of nodes $M \to \infty$, the probability of event G, denoted as $\lim_{M\to\infty}\mathbb{P}(G) \equiv \mathbb{P}(G_{\text{lim}})$, is given by,*

$$\mathbb{P}(G_{\text{lim}}) = (-1+q+v)\,\mathbb{P}(E_{\text{lim}}) + 1 - q, \tag{5.9}$$

where $\mathbb{P}(E_{\text{lim}})$ is given by Eq. (5.3) *and where the maximum value is again achieved when the database is optimally amplified. Let F^* be the event that at least one bad state exists within T, when oracle errors are considered. When M nodes are in T, an upper bound on the probability of F^* is,*

$$\mathbb{P}(F^*) \leq 1 - (\mathbb{P}(G_{\text{lim}}))^M, \tag{5.10}$$

and an upper bound on the probability that at least one bad state is part of a given path γ is given by,

$$\mathbb{P}(F^*_\gamma) \leq 1 - (\mathbb{P}(G_{\text{lim}}))^{|\gamma|}, \tag{5.11}$$

where $|\gamma|$ is the number of nodes in γ.

Proof. The proof stems from modifications made to the statement of Eq. (5.2) to move from measurement probability with respect to the oracle to measurement probability with respect to ground truth. To factor in both types of error, $\mathbb{P}(E^c)q$ (fraction of false positive error, as given in Eq. (5.7)) must be added to $\mathbb{P}(E)$ from Eq. (5.2), and $\mathbb{P}(E)v$ (fraction of false negative error) must be subtracted from $\mathbb{P}(E)$, as shown in Fig. 5.3. This yields,

$$\mathbb{P}(\bar{G}) = \mathbb{P}(E) + \mathbb{P}(E^c)q - \mathbb{P}(E)v, \tag{5.12}$$

where $\bar{G}$ denotes the complement of event G. Eq. (5.12) can be simplified, and the complement taken, to give Eq. (5.8). Eq. (5.9) is found by taking Eq. (5.8) and substituting $\mathbb{P}(G)$ and $\mathbb{P}(E)$ with $\mathbb{P}(G_{\text{lim}})$ and $\mathbb{P}(E_{\text{lim}})$, respectively. Eq. (5.10) and Eq. (5.11) are found with the same process as Eq. (5.4) and Eq. (5.5) with the complement of event G. □

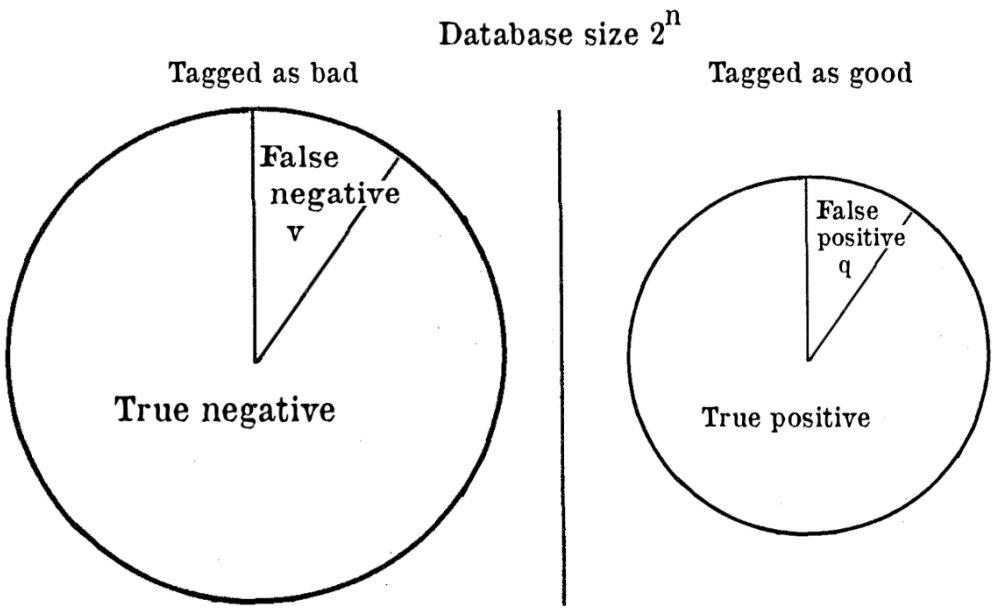

Figure 5.3. A visual depiction of the false positive and false negative regions of the good and bad tags by an oracle.

A tree with as many good states as possible is achieved with the lower bounds of error in Thm. 3. Attaining this bound requires applying QAA an optimal number of times, which is what we estimate next.

5.5.3 Estimate of the Number of Correct Solutions

In this section, we explore methods for estimating the number of tree-admittable states out of a database of uniformly random points inside of a 2-dimensional periodic finite square lattice of characteristic length L and concentration r. This estimation will guide the algorithm in applying QAA the optimal number of times (Eq. (5.1)). Let the function,

$$p(x_1,x_2) = \begin{cases} 1, & \text{if } x_1,x_2 \in Z, \\ 0, & \text{otherwise,} \end{cases}$$

represent connectivity, for a connected component $Z \subseteq C_{\text{free}}$. Initially, we are concerned with whether or not the two states are within the same connected component. In Section 5.5.4, we

discuss local planners and reachability. We estimate the average connectivity p^* of 2 random points within the square lattice as an estimator of the number of correct solutions to the database D,

$$p^* = \mathbb{E}_{\pi(x_1,x_2)}(p(x_1,x_2)), \tag{5.13}$$

where $\pi(x_1,x_2) = \mathbb{U}(C_{\text{free}}) \times \mathbb{U}(C)$.

Several results from Percolation Theory provide insight as to average connectivity of finite square lattices [153], [154], [155]. The work [156] uses results from [157] to calculate and estimate wrapping probabilities of 2D square lattices. Wrapping probabilities refer to the probability that there exists a giant connected component from one edge of the 2D square lattice to the opposite edge. In the context of q-RRT, since each parent is assumed to be in C_{free}, wrapping probabilities, as presented in [156], cannot be directly used. Additionally, our desired estimation is with respect to individual points and not a set of points representing an edge, as in [156].

We calculate the connection probability, Eq. (5.13), from a state $x_1 \in C_{\text{free}}$ to a random state $x_2 \in C$. This reflects an estimate for correct solutions to the database in the case where all nodes of the tree reduce to the root x_0. In the next section, we evaluate the case where trees that are maximally spread in the environment. We fit a model to numerical simulations over concentration r and characteristic length L to estimate connection probability p^*,

$$p^*(r,L) = \frac{f}{1+e^{-a(L-b)(r-c)}} + dL^{-2}, \tag{5.14}$$

with $a = -0.1597$, $b = -54.59$, $c = 0.3212$, $d = 1.195$, $f = 0.9542$, found with nonlinear least squares and with an ordinary coefficient of determination $R^2 = 0.9957$; see [158]. The model was chosen as a logistic function due to observations on matching function data in [156, Fig. 5]. While $(p^*,\ r)$ slices of data exhibit a logistic relation, it is not independent of L based on inspection of level sets in L, which is therefore modeled as a scaling parameter of the logistic function. It is observed that $(p^*,\ L)$ slices exhibit a negative nonlinear relation which is modeled

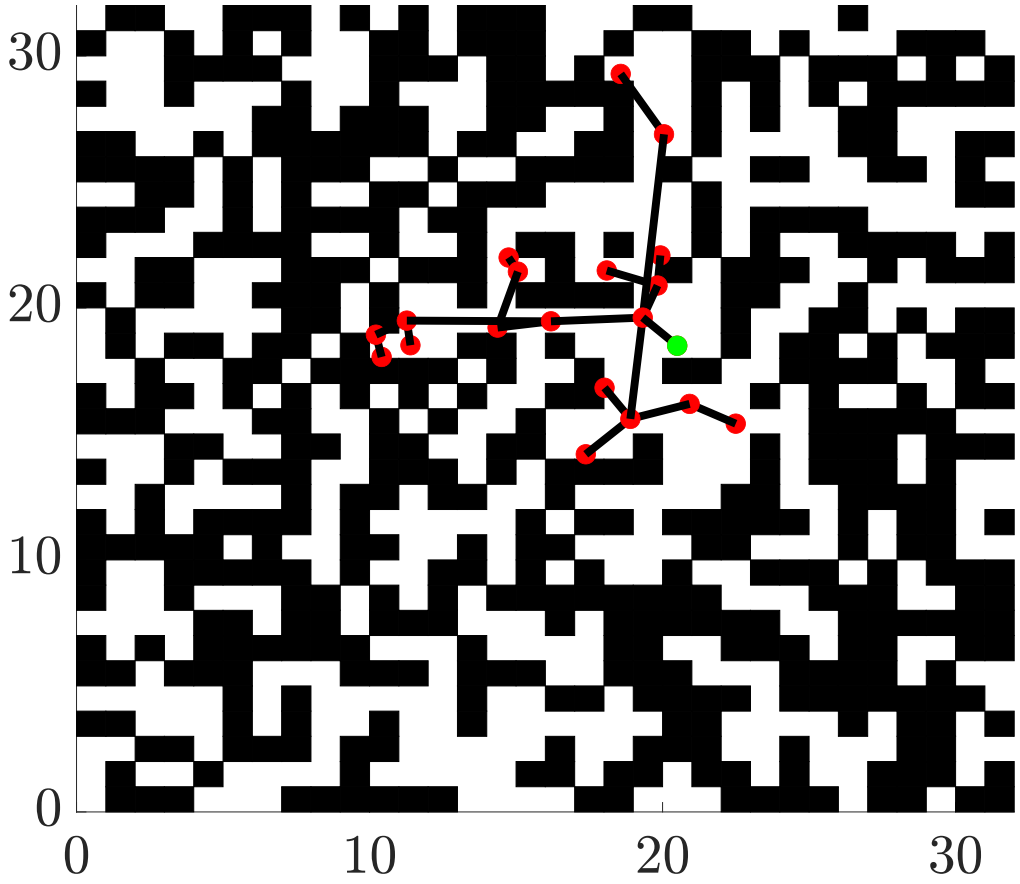

Figure 5.4. A sample random square lattice with $L = 32$ and $r = 0.5$ spanned by a 20 node tree with x_0 in green.

with a quadratic.

Each point $\circ$ in Fig. 5.5 in the parameter space (r, L) represents the average of $1,000$ random connectivity tests over 25 different random lattices each, totaling $25,000$ points. In aggregate, data was collected over 209 parameter-space points, totaling 5.225 million data points. The total dataset was condensed, and the model was trained on averages because we seek to estimate averages. Since the number of data points (209 averages) is large compared to the number of parameters (5), we are not concerned with over-fitting and therefore report the coefficient of determination R^2 and do not split the data into training and validation sets. We refer the reader to Section 5.6.1 for an evaluation of this metric.

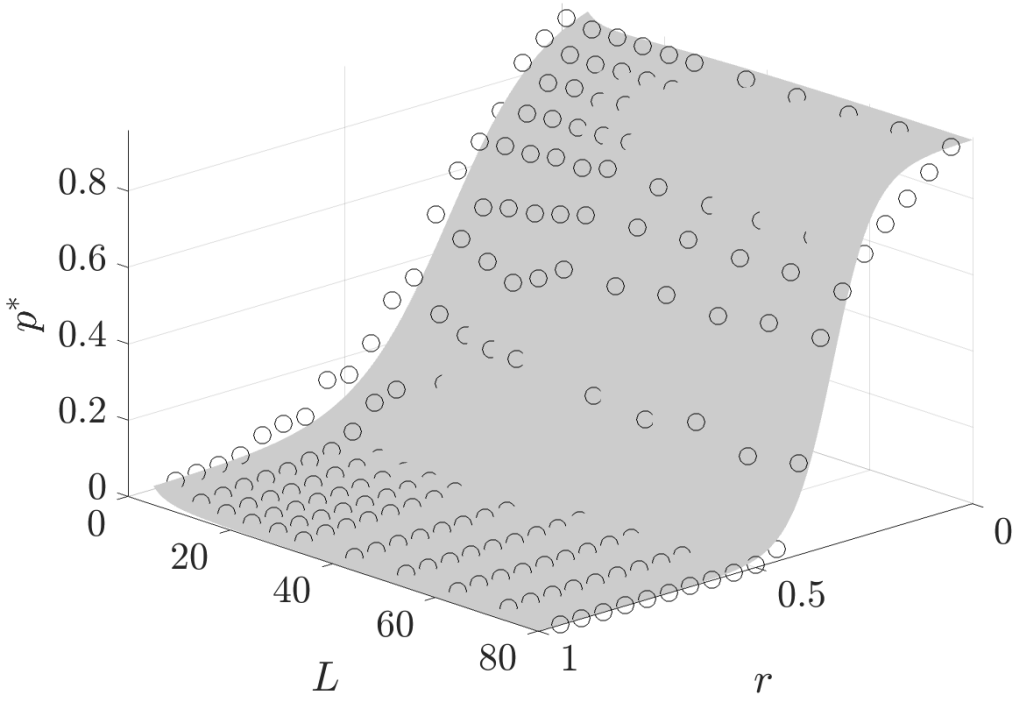

Figure 5.5. Numerically generated data points (∘) estimating p^* (free-random point connectivity) as a function of concentration r and length L. The model is depicted as the gray surface, with a coefficient of determination $R^2 = 0.9957$.

5.5.4 Local Planners and Upper Bound Limit

The choice of local planner affects the accuracy and, therefore, the relevance of Section 5.5.3. Previously, we sought to add points to the tree that are connectable to the tree, i.e. within the same connected component, with no restrictions on the connecting path. If we instead desire the local planner admit *reachable* points to the tree (which account for some dynamics), the model of Fig. 5.5 can be tweaked to yield a second estimate. We also note that considering dynamic models in the estimation in Eq. (5.14) leads to an expansion of the parameter space in an unmanageable way, so the model estimates connectivity sans dynamics.

Given $x_1 \in C_{\text{free}}$, we define the reachable set from x_1 as the states $x_2 \in C_{\text{free}}$ that can be connected to x_1 by a dynamic, obstacle-free path generated by a predefined type of control. We

choose this type of restricted reachability[2], so we can factor in system dynamics and remain according to [160], who note that it is preferable to use a very fast local planner even if it is not too powerful. The oracle marks dynamic paths as not reachable if they are not obstacle-free, as we are more concerned with testing many solutions quickly rather than every solution rigorously, even if it may reachable be with a modified controller or intermediate references.

Next, we provide an upper bound characterization to reachability from a tree of M nodes in free space by considering the case where the tree is maximally spread. This case gives the minimum effective characteristic length because new samples are connected to the nearest node, and for a maximally spread tree, the Euclidean distance of that node to the nearest one in the tree should be the smallest. The minimum characteristic length maximizes reachability, maximizing the proportion of the database marked as correct, which enables us to lower bound the number of applications of Q as per Eq. (5.6). An upper bound p_2^* on the average reachability to a set of nodes in random square lattices is defined in Thm. 5.

Theorem 5. *For a random square lattice C characterized by length L, with concentration r and an arbitrary set T of M nodes in free space, an upper bound $p_2^*(r,L)$ with $x_1 \in T \subseteq C_{\text{free}}$ and $x_2 \sim \mathbb{U}(C)$ is given by Eq.* (5.14) *with characteristic length $L^* = \frac{3L}{\sqrt{M}}$. p_2^* is the absolute upper bound of the number of correct database solutions, which is related to the number of times to apply QAA by Eq.* (5.1).

Proof. The proof follows by considering the best case of a maximally spread tree T of $\widetilde{M}$ nodes (and M feasible nodes) within lattice C. A tree T with nodes placed according to a centroidal Voronoi tessellation (CVT) [161] of C with $\widetilde{M}$ nodes and regions, is one that minimizes the expected distance of every node in C to the closest generator. Assume that $\widetilde{M}$ is sufficiently large so that there M feasible nodes in C_{free}. A random point will attempt to connect with the closest parent. Each existing node, when placed according to a CVT in a convex region, creates a region of connection characterized (in 2D) by length $\frac{L}{\sqrt{\widetilde{M}}}$ for a C of area L^2 with $\widetilde{M}$ regions. A CVT, by

[2]More generally, a reachable state from x_1 is x_2 for which there exists a control $u(t)$ that connects these states by a dynamic path. [159]

definition, creates Voronoi regions of connectivity of expected minimal characteristic length. If a certain node turns out to be infeasible, the distance of a point to the nearest feasible generator is $\frac{3L}{\sqrt{\widetilde{M}}}$, which can be upper bounded by $\frac{3L}{\sqrt{M}}$. This minimal characteristic length yields a maximum connectivity estimate by substituting $L^* = \frac{3L}{\sqrt{M}}$ for L into Eq. (5.14). □

This is similar to the noted result in [160] regarding the restriction of new test nodes to sufficiently close existing nodes in the tree to maximize the connection likelihood. In the q-RRT Alg. 8, p_2^* serves to lower bound the number of times QAA must be applied to the database qubit. The intuition behind Thm. 5 is that as the tree grows in number of nodes, it is easier to prove reachability to the tree. The bounding case is when the tree is maximally spread within C, as given by a CVT. In that case, the characteristic length can be thought of as $\frac{L}{\sqrt{M}}$, or the original environment size split into M equal sized and roughly convex regions.

5.6 q-RRT Results and Discussion

5.6.1 Comparison With Ground Truth

In the following, we evaluate p^* and p_2^* on a particular example. Quantum computers must find the number of correct solutions within the database using the Quantum Counting Algorithm [149], which is a mix of quantum phase estimation and Grover's Algorithm. Due to the use of a quantum computing simulation on a classical computer, this value is knowable. To ascertain reachability, the oracle $\mathcal{X}$ uses the following robot dynamics and control law and performs reference tracking from an x_{parent} to a x_{new},

$$x(t+1) = Ax(t) + Bu(t)\,,\, x(0) = x_{\text{parent}},$$

$$A = \begin{bmatrix} -1.5 & -2 \\ 1 & 3 \end{bmatrix}, B = \begin{bmatrix} 0.5 & 0.25 \\ 0 & 1 \end{bmatrix},$$

$$u(t) = -Kx(t),$$

$$K = \begin{bmatrix} 1.9 & -7.5 \\ 1 & 7 \end{bmatrix}.$$

The constant gain matrix K can be any matrix such that the closed loop system is stable.

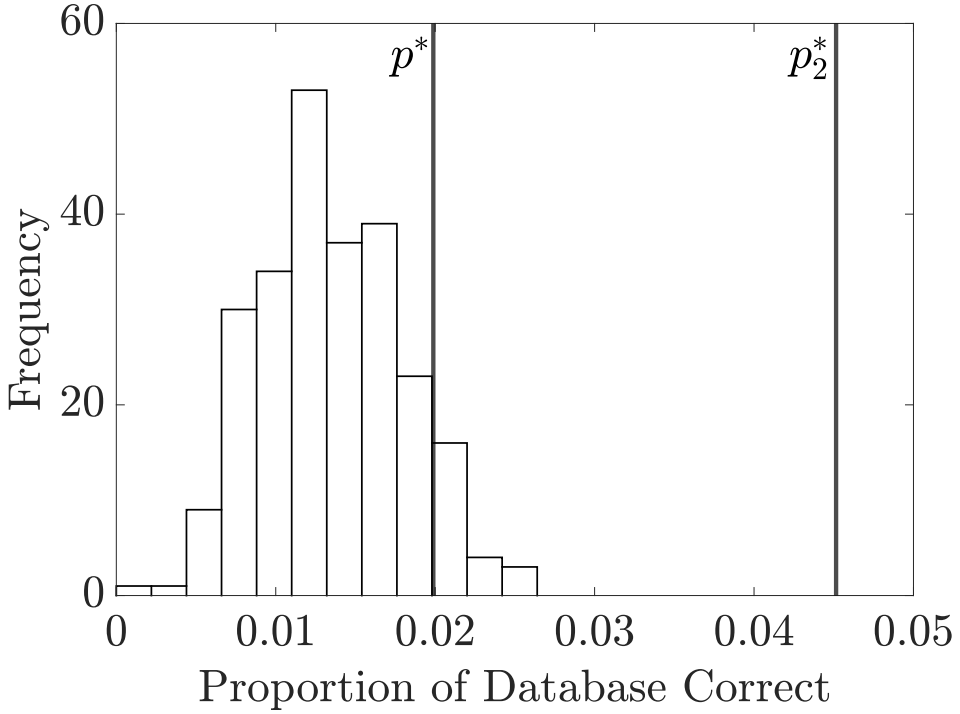

Figure 5.6. Comparison of p^* and p_2^* with a histogram of 250 random cases of database size 2^{11} with $L = 32$, $r = 0.6$, and a tree of $M = 5$ nodes.

In Fig. 5.6, we compare p^* and p_2^* against a histogram of 250 simulations, in randomized environments, of a 2^{11} size database. Each of the 250 simulations are grouped according to the proportion of correct database solutions they provide. Cases are run with $L = 32$, $r = 0.6$, and with a tree of $M = 5$ nodes. From the figure, observe that p^* forms a slightly high estimation, while p_2^* is validated to form an upper bound. Since p^* refers to mean connectivity, and not reachability, the proportion of correct solutions, when we factor in provable reachability, is generally less than p^*. This concept is shown in Fig. 5.7, where the forward reachable set is shown as a subset of the connected component. This explains why p^* forms a slightly high estimation of the mean. On the other hand, p_2^* correctly upper bounds the proportion in the case

where $M = 5$.

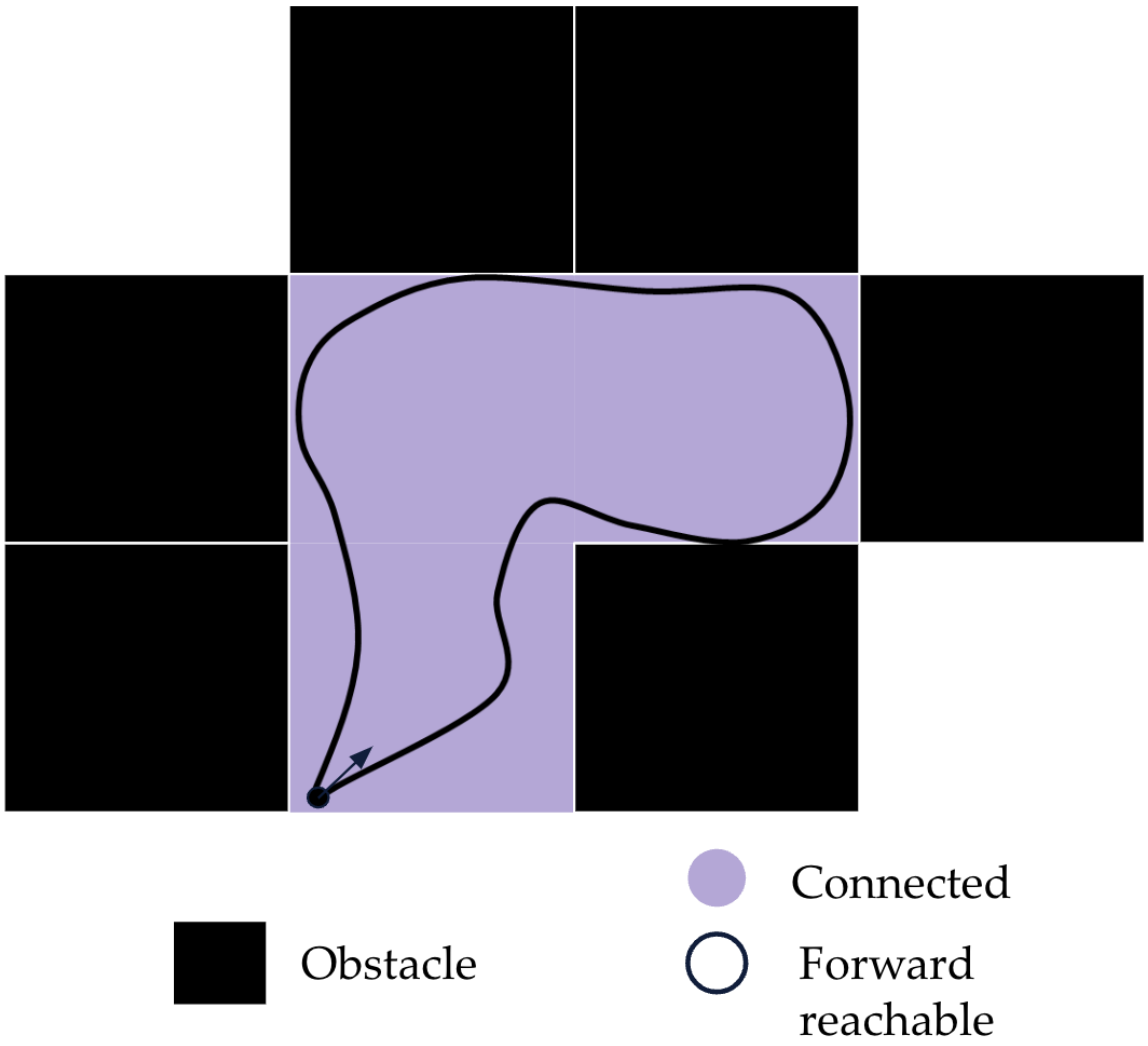

Figure 5.7. An illustration of why p^* forms a slightly high estimation: the forward reachable set is generally a subset of the connected component. Reachability tests will return a lower estimate of database correctness than connectability, which is what p^* is based upon.

5.6.2 Results in Dense Random Lattices

In this section, we show the results of q-RRT creating a tree within large connected components of random 2D lattices (explained in Section 5.5), as shown in Fig. 5.4. We compare algorithm performance with a classical, and largely identical, version of RRT attempting to span the same connected component. The classical version of RRT replaces the quantum

database search with a classical oracle check on a single point. All path planning simulations are performed in a 2D environment run with Matlab v2022b on a desktop computer with an Intel i5-4690K CPU and an AMD RX 6600XT GPU. A selection of Matlab code is available at github.com/pdlathrop/QRRT. The quantum states and algorithms are simulated using the Matlab Quantum Computing Functions library [150]. Simulations are run in a random square lattice of size $L = 72$, where each method is given a random start node within the largest connected component. In the simulations, r varies because concentration creates large differences in performance of both algorithms. Fig. 5.8 and Fig. 5.9 show the average number of oracle calls and average real run-time of each algorithm to create an 11 node tree, which is an arbitrary number chosen to showcase average performance. Each data point is averaged over 50 planning problems, in 50 random environments. Both algorithms are tested against the same environments.

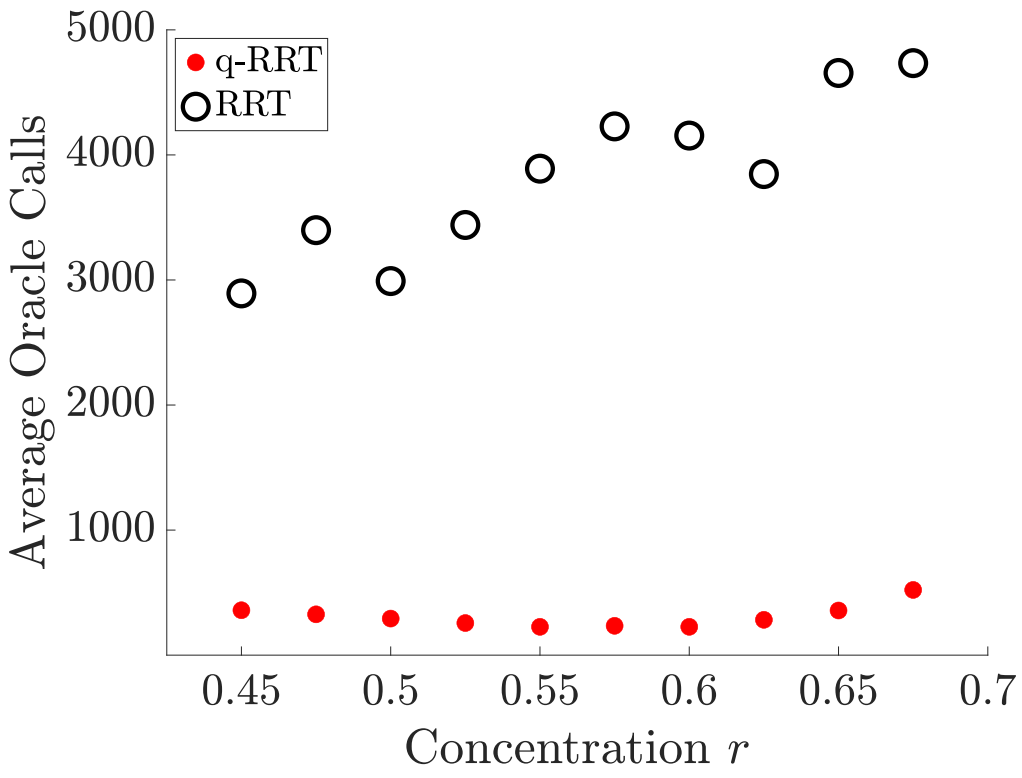

Figure 5.8. Comparison of the average number of oracle calls by q-RRT and RRT as concentration varies, for $L = 72$.

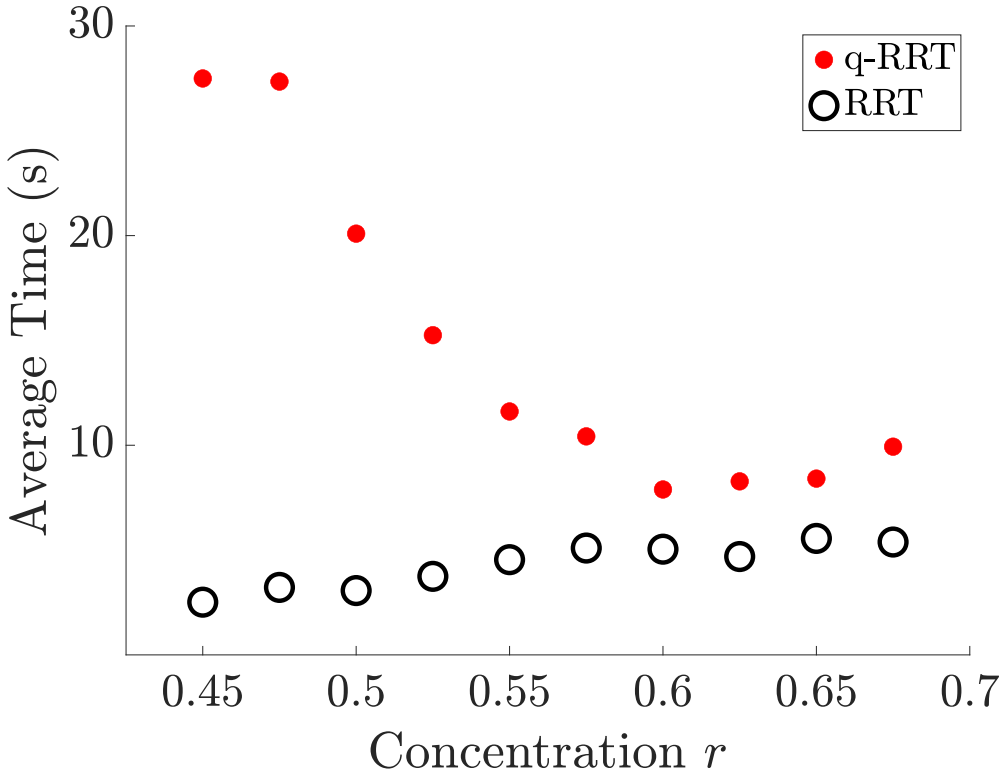

Figure 5.9. Comparison of the average real run-time of q-RRT and RRT as concentration varies, for $L = 72$.

Over the range $[0.45, 0.7]$ in concentration r, q-RRT averaged 308 oracle calls, while the classical RRT averaged 3820 oracle calls, as a result of a quadratic performance increase. Algorithm q-RRT averaged 14.7 seconds per case, compared to RRT's average of 4.3 seconds, and this is due to the implementation of quantum algorithms via arrays on a classical computer. On a quantum computer, the actual run-time advantage would be proportional to the average oracle call advantage. As r increases, the average number of oracle calls also generally increases due to the increased difficulty in making connections in denser environments. For RRT, the average time per case shows this increase because most of the algorithm run-time is in performing reachability tests. For q-RRT, as r increases, there is an initial decrease in average run-time, possibly because at low r, the largest connected component tends to be very non-convex and widely spread. This causes additional reachability tests to be performed because of a run-time

optimization where points are excluded first based on whether they are not in the same connected component, then based on reachability.

Minimizing oracle calls is useful in situations where admitting points to the tree has high computational cost, or where reachability checks carry a cost. In our method, the oracle tests experimentally whether a possible point (or database of points) is within the reachable set, which is a complex problem to solve analytically for non-simple systems [162]. This can result in significant time savings, and in some cases may allow offline algorithms to become online. In large dense environments where most random points are not admittable to the tree, many reachability tests must be performed to admit even a single valid state. In such situations, q-RRT far outperforms RRT in the ability to admit new nodes to the tree (per oracle call).

5.6.3 Database Size Comparison

In this section, we show the effect of variance of total database size 2^n on the performance of q-RRT as compared to classical RRT. In Fig. 5.10 and Fig. 5.11 we show the average number of oracle calls and the average real run-time, respectively, of q-RRT with databases sized 2^8 and 2^9 for $L = 72$ while concentration varies. Again, we compared q-RRT with RRT in creating a tree with 11 nodes, and each data point is averaged over 50 planning problems, in 50 random environments.

We verify that changes to database sizing does not effect the overall trend of average number of oracle calls or average run-time over varying concentration. The larger 2^9 sized database resulted in lowered average oracle calls, especially at higher r, when compared to 2^8 sizing. This is consistent with the main reason the quantum algorithm provides a reduction at all, which is the ability to perform reachability tests on many possible states simultaneously. Predictably, with respect to real run-time on a classical computer, larger database versions of q-RPM take longer across all r, as more reachability tests need to be performed with the quantum computing simulation. However, on a quantum device, we expect the run-time to be analogous to the number of oracle calls.

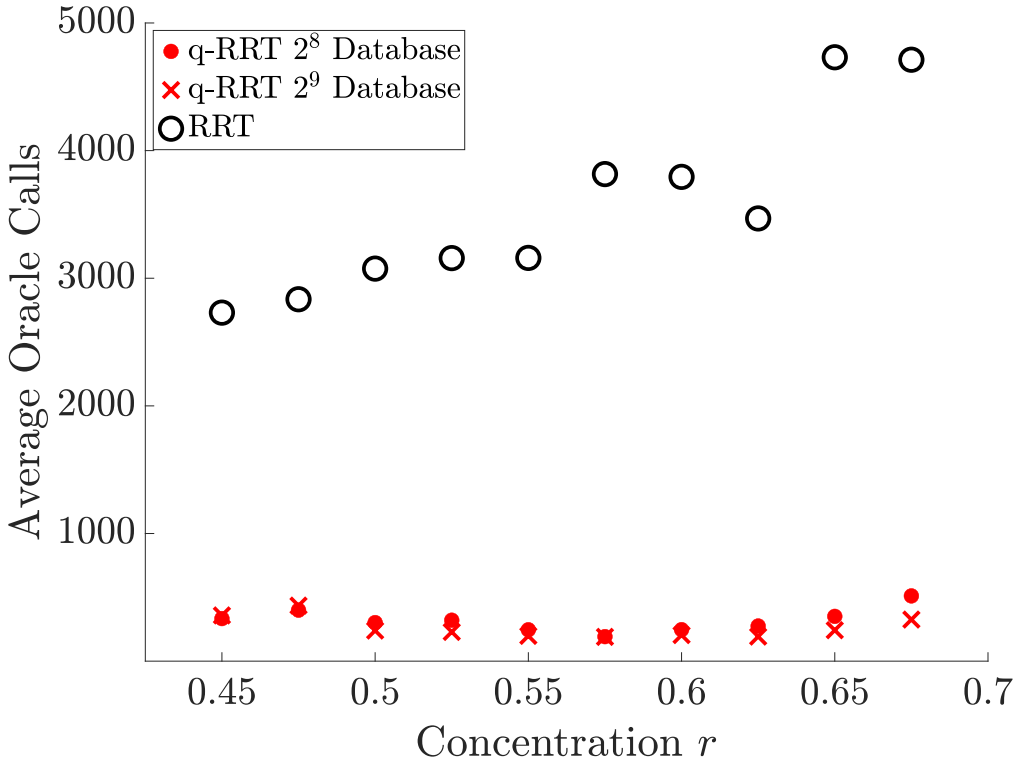

Figure 5.10. Comparison of the average number of oracle calls by q-RRT and RRT as concentration varies, for $L = 72$ and Database sizes 2^8 and 2^9.

5.6.4 Oracle Call Constraint

In this section, we identify an approach for tree construction that limits the optimal number of oracle calls to a maximum of $N_{\mathcal{X}}$ per node added to the tree. We may want to create databases of correctness proportion p, rather than just predict p from the environmental parameters, especially in time limited cases. In order to limit the number of optimal oracle calls to $N_{\mathcal{X}}$, we constrain the L_1 (Manhattan) sampling distance to evaluate reachability to be equal to a certain value, which we call D_{L1}. L1 'rings' around existing nodes in a lattice environment is shown in Fig. 5.12, and sampling for new points to add to the database is restricted to the edge/surface of the rings (or, alternatively, to within the rings). This will ensure that the number of successful reachable connections becomes higher in cluttered environments, thus requiring a smaller number of oracle calls. We consider an environment with a fixed L value and measure

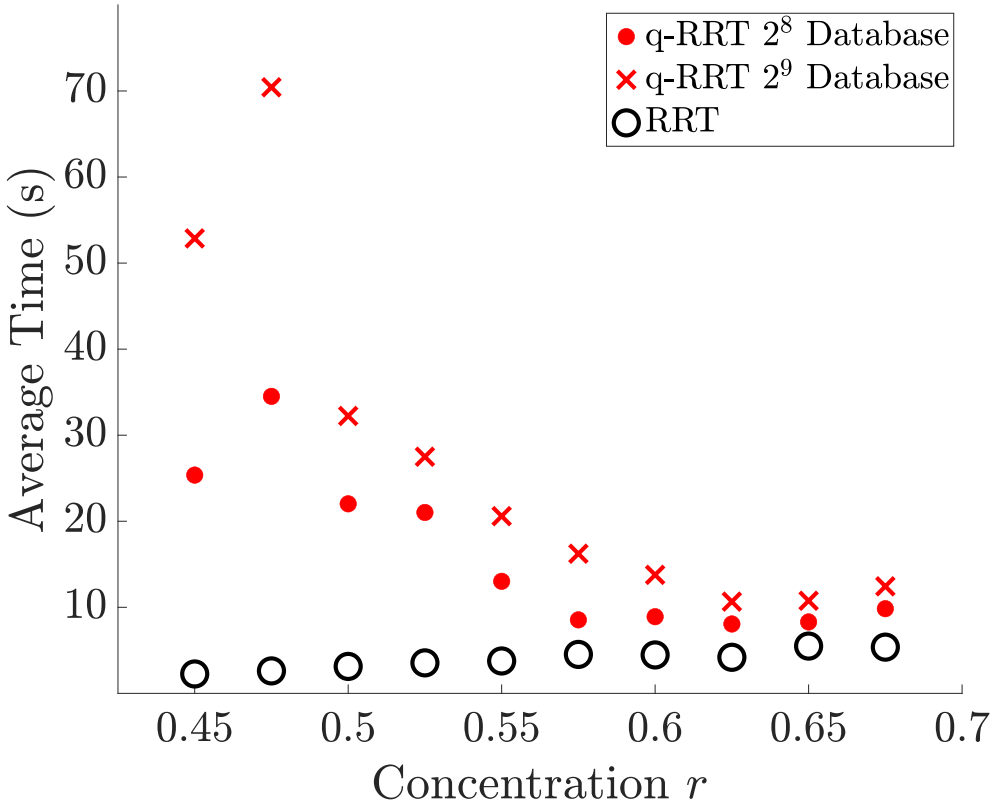

Figure 5.11. Comparison of the average run-time of q-RRT and RRT as r varies, for $L = 72$ and Database sizes 2^8 and 2^9.

distance in terms of the 1-norm or Manhattan distance. The 1-norm is chosen over the Euclidean distance as, intuitively, it can yield a superior parameter for estimating connectivity within a square lattice. In this context, the word optimum refers to the number of oracle calls that maximizes the likelihood of measuring a correct database element.

In Fig. 5.13, we show how average connectivity p scales according to a negative exponential with increasing L_1 distance between parent and child. Values spread at the larger L_1 distances due to smaller sample sizes. For a given concentration r and an oracle call constraint $N_{\mathcal{X}}$, Fig. 5.13 can be used to select the maximum D_{L_1} that will select $N_{\mathcal{X}}$ as the approximate number of optimal oracle calls. To exemplify how such a tool can be used, we fit a model using

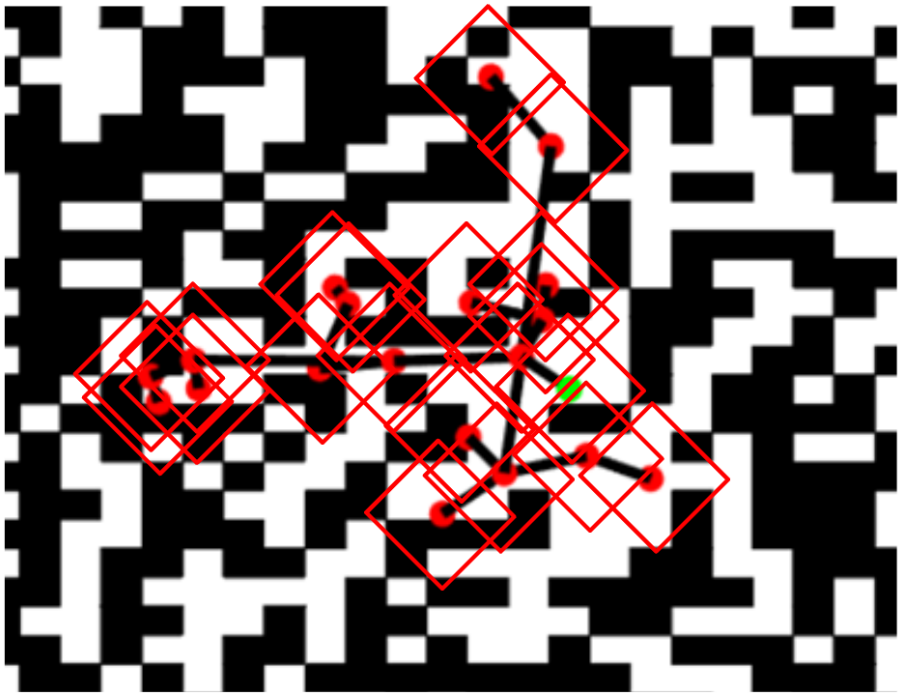

Figure 5.12. An illustration of the constrained sampling idea in a two dimensional lattice with L1 'rings' shown around each existing node in the tree. New samples are restricted to the edge of the rings.

nonlinear least squares to the numerical $L = 72$ data shown in Fig. 5.13, which takes the form,

$$p = a\, e^{(br+c)\, D_{L_1}}, \tag{5.15}$$

with $a = 0.479$, $b = -1.72$, and $c = 0.674$ with coefficient of determination $R^2 = 0.981$. Again, over-fitting is not a concern for three parameters modeling 336 data points. Equivalently,

$$D_{L_1} = \ln\left(\frac{\pi^2}{16 N_{\mathcal{X}}^2 a}\right) / (br + c), \tag{5.16}$$

when Eq. (5.15) is solved for D_{L_1} and p is related to $N_{\mathcal{X}}$ via Eq. (5.1). Eq. (5.16) allows an algorithmic distance constraint to be found from an oracle call constraint and the environment concentration. From here, when q-RRT is building a database, states should be randomly selected from the boundary of a ball at radius D_{L_1}. In order to further restrict oracle calls, q-RRT can

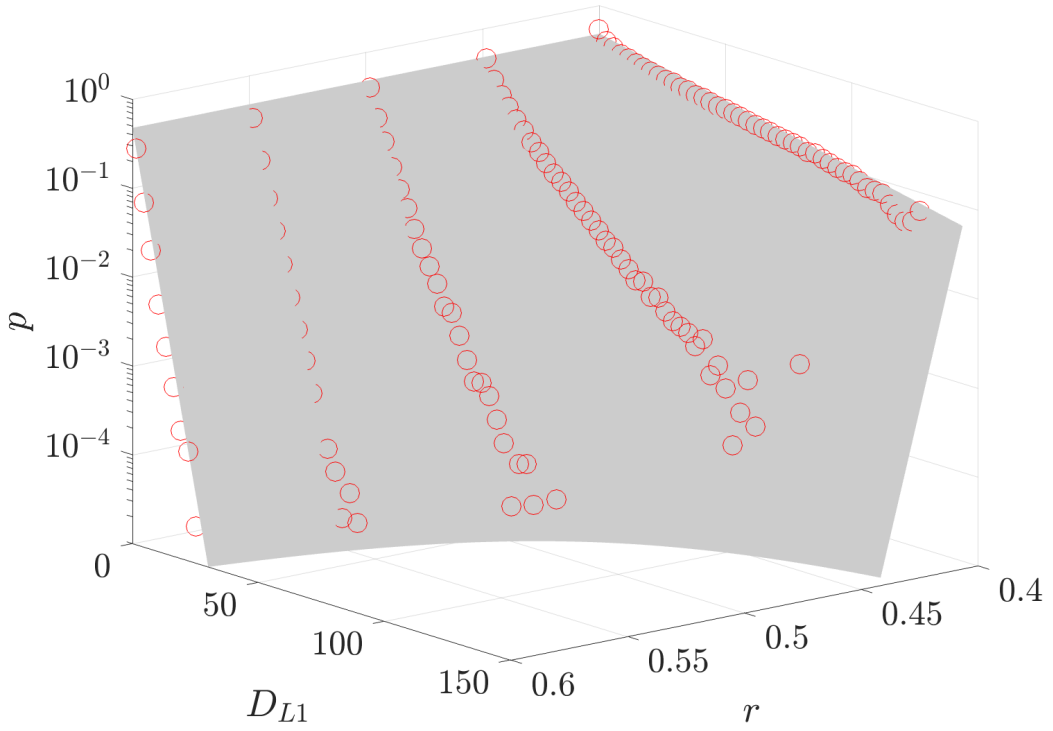

Figure 5.13. Semilog plot of numerically generated data points (○) estimating p, the likelihood of free-random point connectivity as a function of D_{L1}, the L_1 distance between parent and child, for various concentrations r. The model is depicted as the gray surface. Data is generated with $L = 72$.

instead randomly sample within a ball of radius D_{L_1}, which results in a lower mean L_1 distance, and therefore higher p and lower number of oracle calls.

An analysis of the ability to select p given a concentration r and oracle call constraint $N_{\mathcal{X}}$ is given in Fig. 5.14. Datasets of size 2^{14} are constructed in a random square lattice of $r = 0.5$ and $L = 72$ (chosen for discretizability) for various $N_{\mathcal{X}}$. The goal points are the p that correspond with an optimum number of oracle calls $N_{\mathcal{X}}$ to admit one node to the RRT. Therefore, the number of oracle calls which yields the maximum likelihood of adding M nodes to the RRT from M database creations is $MN_{\mathcal{X}}$. The use of an L_1 restricting version of q-RRT alongside Eq. (5.16) allows the creation of an M node RRT where $N_{\mathcal{X}}$ has been approximately chosen as the optimum number of oracle calls per node. Fig. 5.14 shows that, as $N_{\mathcal{X}}$ increases, we have a more accurate

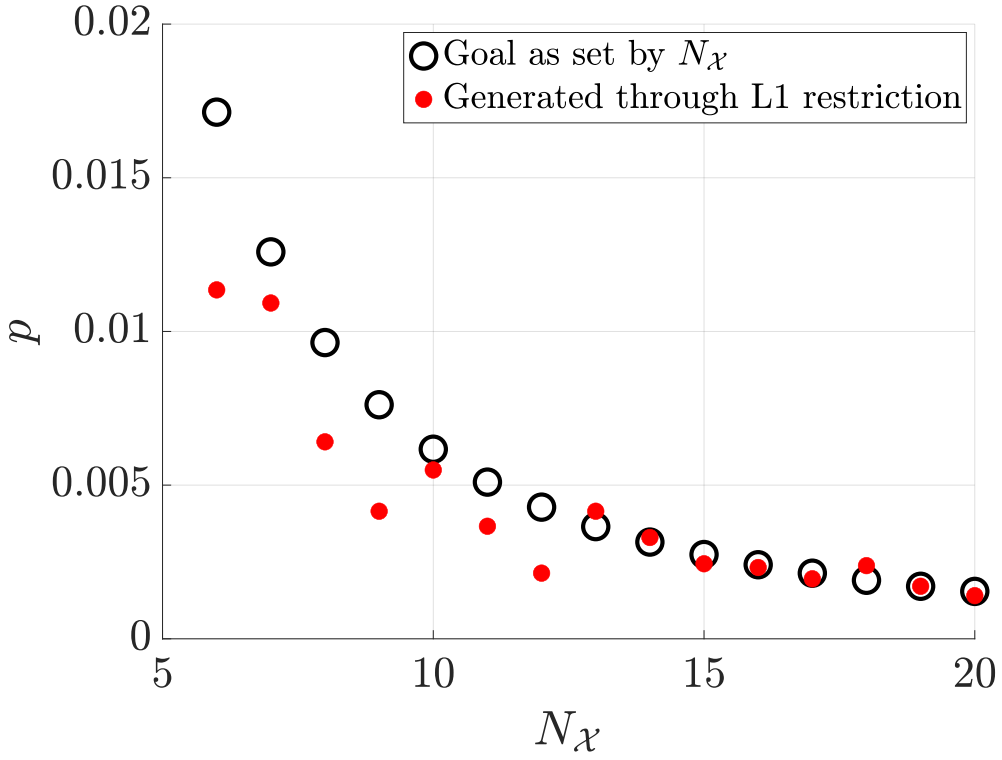

Figure 5.14. Evaluation of the ability to select p given $N_{\mathcal{X}}$ using the model. Data is generated with $L = 72$, $r = 0.5$, and the database size 2^{14}.

ability to select p.

5.7 Conclusion

The goal of this work was to provide a first study of the application of quantum algorithms to sampling-based robotic motion planning. We developed a full path quantum search algorithm for sparse environments and a Quantum RRT algorithm for dense random square lattices. The q-RRT algorithm uses Quantum Amplitude Amplification to search a database of possible parent-child relationships for reachable states to add to the tree. The q-RRT algorithm, tested on a simulated quantum device, successfully employs a quadratic speedup of database searches to reduce oracle reachability calls when constructing a tree. We also provide key quantum measurement probability results, and tools for estimating and selecting the number of correct

database entries using numerical modeling and guided sampling. Future work includes path planning employing quantum mean estimation for uncertainty modeling [163], implementing q-RRT on disjoint trees [164] for planning over multiple disconnected components, and exploring path-optimality based algorithms in the context of quantum computing.

Chapter 5, in full, is a reprint of material "Quantum Search Approaches to Sampling-Based Motion Planning" as it appears in IEEE Access. Lathrop, Paul, Beth Boardman, and Sonia Martínez. IEEE Access, vol. 11, pp. 89506-89519, 2023, doi: 10.1109/ACCESS.2023.3307316. The dissertation author was the primary investigator and author of this paper.

Initial exploration into sampling-based motion planning for quantum computing yielded several distinct problems. The nature of the probabilistic measurement process and the possibility of incorrect oracles yielded two sources of error, which we symbolically quantified with respect to the growing tree. In order to optimally amplify a database, knowledge of what proportion of that database is correct was needed, for which we introduced a particular environment (random square lattices) that enabled numerical estimations of that quantity. In order to extend this work to more general obstacle environments, we seek to move on from addressing the question of database correctness, and focus on addressing the problem of measurement collapse.

When the amplified quantum state is measured to return a single database element, probability information on the database is lost due to wave function collapse, an inescapable reality of the quantum world. The focus of the following chapter is on this subject, which is highlighted in Fig. 5.15. However, even though qubits in superpositions cannot be copied, introducing a parallel quantum structure to the architecture can enable multiple qubit superpositions to be created in parallel, allowing multiple database solutions to be found simultaneously, albeit by different workers.

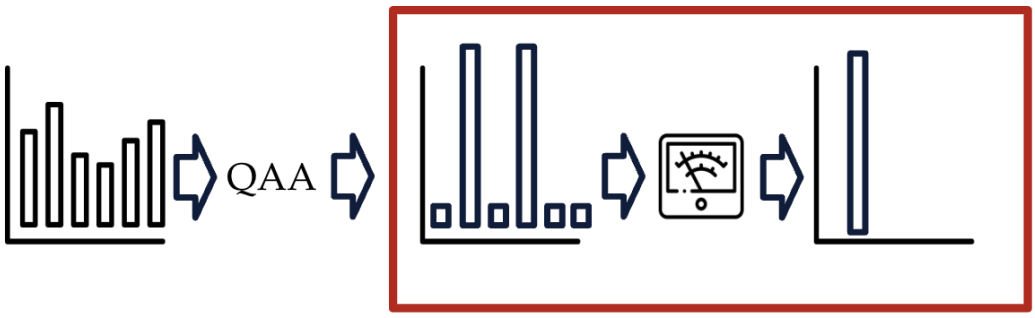

Figure 5.15. Highlighted in red is the measurement collapse problem in a nutshell: probability amplitude information on multiple solutions is lost when the superposition qubit is measured and collapses to a deterministic state. This is one of the motivations of the following chapter.

Chapter 6

Parallel Quantum Rapidly-Exploring Random Trees

6.1 Introduction

Quantum computing algorithms have shown promise in accelerating the search for solutions when applied to computationally intensive, complex problems. More efficient solutions have been found and proven with quantum computing in fields such as pure science simulations [165], cryptography [166], and machine learning [167].

With respect to robotics and motion planning, quantum algorithms have also been found to increase speed and efficiency. The heart of quantum advantage is derived from quantum parallelism and the ability to perform simultaneous computations on superpositions of states, which motivated our work in [168]. To aid in sampling-based motion planning, the key efficiency bottleneck is the search for dynamically-reachable, obstacle-free states to connect to the search tree. Unstructured databases of possible next states can be searched simultaneously with Quantum Amplitude Amplification (QAA) to efficiently find an amenable state, but the quantum measurement process forces information loss through the collapse of the superposition. Although all database states are searched in parallel, the process can only return one state. In this chapter we are motivated by efforts in traditional motion planning to parallelize sampling-based planners for efficient path generation using multi-core computers and GPUs, and the ability of quantum algorithms to perform parallel computations. We seek to explore ways for quantum

motion planning algorithms to allow multiple, parallel quantum computers to more efficiently search a database of states and return multiple possible solutions.

6.2 Literature Review

In this section, we provide an overview of quantum computing as it applies to robotic applications, non-quantum efforts to parallelize sampling-based motion planning algorithms, the use of annealing and temperature constructs as they applies to motion planning, and how this is related to our efforts to increase the efficiency of q-RRT.

Quantum algorithms have been applied to a range of robotic and motion planning-related applications. They have been used to solve generalized optimization problems [169], estimate stochastic processes [170], and provide speedup to Monte Carlo methods [171]. They have also performed quantum searches [133] of physical regions [134], found marked elements of a Markov chain [135], and searched more abstract spaces such as a search engine network [137]. A more detailed overview on how quantum computing has been applied to robotics, along with open questions, can be found at [40].

Quantum computing has also been used to improve motion planning specifically. Quantum reinforcement learning [127] has increased the speed and robustness (when compared to classical, non-quantum algorithms) of robotic reinforcement learning algorithms when learning optimal policies in gridded environments [29, 124]. An additional use of quantum computing for robotic trajectory planning is addressed in [129], which uses a Quantum Evolutionary Algorithm to search for optimal trajectories with a population dynamics/mutation quantum algorithm. Lastly, the review [132] examines quantum control algorithms, and the topic of feedback control accomplished using quantum computing. The work at hand is distinct from the state of the art of quantum computing as applied to motion planners, as we apply such methods to sampling-based planners, which have the advantage of providing fast solutions in high dimensional environments alongside providing probabilistic completeness guarantees [6]. Besides our previous q-RRT algo-

rithm, quantum computing has not, to the best of our knowledge, been applied to sampling-based planning algorithms. A further overview on how quantum computing has been applied to motion planning and robotics can be found in [168].

In the field of non-quantum motion planning, sampling-based planning algorithms such as Probabilistic Roadmaps (PRMs) [8] and Rapidly-exploring Random Trees (RRT) [9] have taken the forefront due to their efficiencies in high-dimensional planning spaces and ability to handle complex robot dynamic constraints [27], such as robotic grasping tasks, autonomous vehicle planning, and UAV navigation. RRTs and PRMs have been extended in many ways to improve their sampling speed, exploration ability, and collision-checking subroutine. RRT* is an important extension regarding path optimality through rewiring [14]. An overview of sampling-based motion planning can be found in the textbook [6]. Three ways to increase motion planning efficiency are the use of quantum computing, the use of parallel algorithms and architectures, and the use of sampling strategies. In this chapter we consider the combinations of the three approaches through parallel quantum computing and database construction strategies, which is akin to sampling strategies in classical algorithms.

Motion planning algorithms have been written for parallel tree creation [148] and parallel computation with GPUs [28]. In [148], local trees are built in parallel to explore difficult regions, and guidelines on when to create and grow local trees are made. In [28], the authors parallelize the collision-checking procedure using GPUs to increase optimal planning speed in high-dimensional spaces. The work [172] outlines Parallel RRT and Parallel RRT*, which uses lock free parallelism and partition based sampling to provide superlinear speedup to RRT and RRT*. The work [37] compares parallel versions of RRTs on large scale distributed memory architectures, including *or*-Parallel RRT (multiple simultaneous individual RRT's) and two methods of collaborative single RRT, Distributed RRT and Manager-Worker RRT. The work [37] also includes a succinct literature review regarding parallel motion planning and Parallel RRT. For comparison purposes, in the work at hand we consider a class of Manager-Worker Parallel RRTs, focusing on the parallelization of the collision-checking procedure of RRT. This is justified

by the findings in [37], which concludes that for variable expansion cost cases, where the communication load is insignificant compared to the computation load, Manager-Worker RRT outperforms, or nearly matches, Distributed RRT in studied passage, corridor, and roundabout environments. Several prominent architectures are visualized in Fig. 6.1.

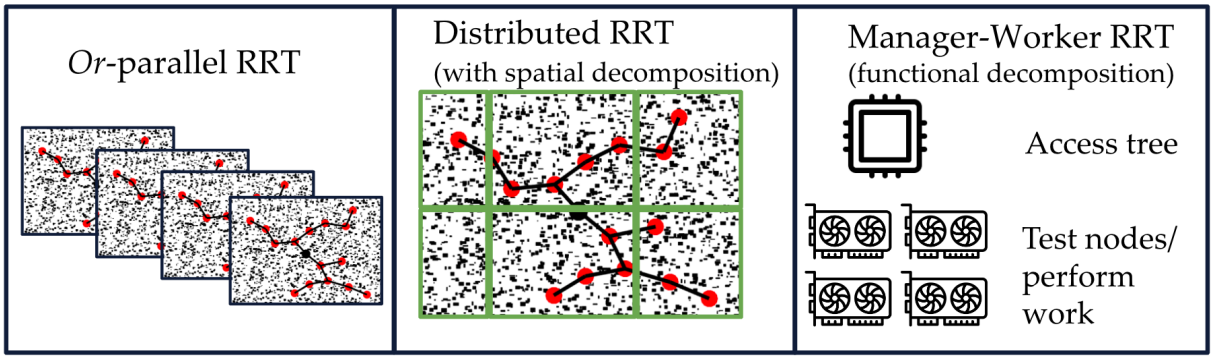

Figure 6.1. Several prominent high level architectures for parallel tree construction. At left, *Or*-parallel architecture, with different workers/cores performing individual tree construction until one finds a path. In the middle, a distributed collaborative tree construction with spatial decomposition is shown where different workers/cores work on expanding into different regions of the configuration space. At right, a functional decomposition is shown with Manager-Worker RRT where a manager interacts with a central tree and workers test add additional nodes. Icons provided from Flaticon.

The work [173] discusses parallel quantum computing architectures and control strategies for distributed quantum machines, noting that multiple few-qubit devices may be more technologically feasible than single many-qubit devices. In this chapter, we consider parallel quantum computers to be multiple simultaneous identical quantum devices governed by a classical device in order to perform parallel computation.

A second extension to increase the efficiency of q-RRT relies on database construction, where we employ a method inspired by simulated annealing. Simulated annealing [174] is an optimization technique that relies on decreasing a temperature parameter to find global maxima and minima of a nonconvex optimization problem, which is somewhat robust to local features. Temperature acts as a guide to the probability of accepting a worse state, allowing an optimizer to explore past local features while eventually settling on estimates of global optima when

temperature falls. An overview can be found at [175] and [176].

Although a temperature construct is mainly used in the context of optimization, similar annealing ideas have also been applied to motion planning, and we intend to apply them to guiding the exploration vs exploitation trade-off of the planning algorithm. In a manner somewhat reminiscent of annealing, the work [177] uses a dynamic reaction-diffusion process to propagate, then contract, a search area for a goal location. Additionally, the covariant Hamiltonian optimization for motion planning (CHOMP) method [178] uses gradient techniques to improve trajectories and solve motion planning queries. CHOMP uses simulated annealing to avoid local minima in trajectory optimization and not to guide sampling-based motion planning itself. The work [179] uses simulated annealing to balance exploitation of Sampling-Based A* (SBA*) and exploration of Rapidly-exploring Random Tree* (RRT*). As cooling occurs, the probability of choosing the exploration strategy drops and the probability of choosing the exploitation strategy increases. Similarly, the transition based RRT [180, 181] method uses a temperature quantity inspired by simulated annealing to define the difficulty level of transition tests to accept higher cost configurations in an effort to explore a configuration space. Similar to this work, we use a temperature quantity to guide the level of exploration, but because we are using quantum computing with q-RRT to perform motion planning, temperature factors into database construction rather than individual samples themselves.

In [168], we introduced how quantum computing methods can be applied to sampling-based motion planning in two ways, a full path database search and an RRT-based single-state database search q-RRT. The Quantum Rapidly-Exploring Random Trees algorithm, q-RRT, uses Quantum Amplitude Amplification (QAA) to search databases of possible reachable states. A focus of our work [168] was in estimating solution likelihood (so QAA could be performed an optimal number of times) through the use of random square lattice environments and numerical simulations. We chose this approach over quantum counting in an effort to keep oracle efficiency high. In the work at hand, we shift focus to address a particular shortcoming of quantum computers and qubits: this approach suffers from the limits of quantum mechanics, as qubits

cannot be copied and the quantum measurement process admits a single solution. We study how quantum devices, working in parallel, can efficiently solve motion planning problems, while generalizing environments away from random square lattices. Instead of focusing on how many solutions exist within a database (which can be found with the Quantum Counting Algorithm (QCA)), we focus on how multiple solutions can efficiently be found from a single database, as is depicted in Fig. 6.2.

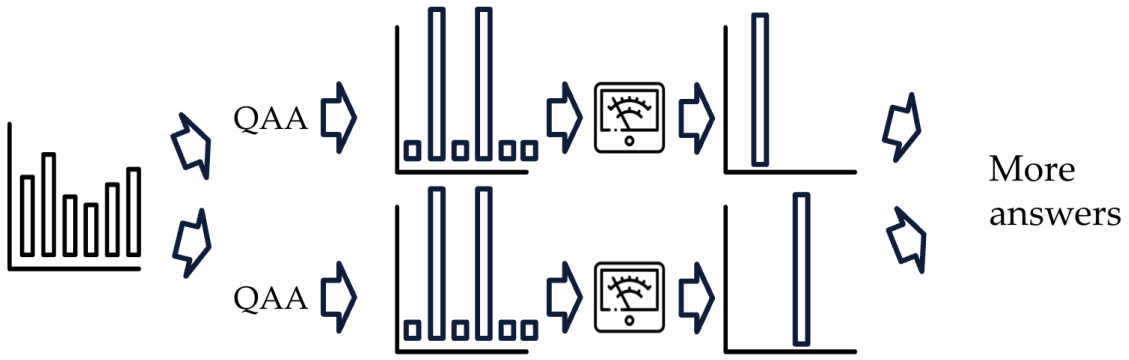

Figure 6.2. An illustration into how a parallel approach to amplification and measurement can result in multiple deterministic solutions attained from a single superposition, created and manipulated in parallel. Probability amplitudes of a sample qubit are shown amplified, indicating two distinct solutions, which are both found by parallel processes. Icons from Flaticon.

6.3 Contributions

The main contributions of this work are the following.

- Creation of a parallel quantum computing variation to q-RRT, called Parallel q-RRT (Pq-RRT), which uses a parallel quantum computing structure to allow multiple solutions from a single database in general obstacle environments;
- Symbolic runtime analysis of shared and unshared database formulations of a Manager-Worker parallel architecture;
- Characterization of key probability values for multiple quantum workers searching a

shared database, with and without false positive and false negative oracle errors in order to minimize efficiency loss;

- Creation of a construction strategy for quantum-search databases, called Quantum Database Annealing, which uses a temperature construct to select sample distances and balance exploration vs exploitation;
- Demonstration (through simulation) of the increased efficiency of Pq-RRT over q-RRT, as compared to the efficiency increase of Parallel RRT over RRT;
- Simulations of faster tree exploration with Quantum Database Annealing as compared to standard uniform-sampling database construction.

6.4 Organization

In Section 6.5.1, for reference purposes we provide a working definition of q-RRT from [168]. In Sections 6.5.2 and 6.5.4, we define Parallel q-RRT then key probability results for Pq-RRT, respectively. In Section 6.5.5, we define the database construction strategy Quantum Database Annealing. In Section 6.6.1, we provide runtime and efficiency results for q-RRT and Pq-RRT as compared to RRT and Parallel RRT. In section 6.6.2, we provide heatmaps of q-RRT's node placement speed over RRT, and in Section 6.6.3 we provide narrow corridor results for q-RRT. In Section 6.6.4, we provide tree comparisons between Quantum Database Annealing and standard database construction.

6.5 Parallel Quantum RRT and Quantum Database Annealing

In Section 6.5.1, we outline the q-RRT Algorithm from [168], followed by a presentation of the Parallel Quantum RRT Algorithm in Section 6.5.2, and then probability results for Pq-RRT in Section 6.5.4. Lastly, in Section 6.5.5, we present the Quantum Database Annealing strategy.

6.5.1 Quantum RRT Algorithm

The q-RRT algorithm from our previous work [168] is a tree-based search algorithm based on RRTs [67]. Quantum RRT uses QAA on a database of possible parent-child pairs to admit reachable points to the tree. In this work, q-RRT returns a path (as opposed to the full tree in [168]) and has the end condition of finding a goal. The line-by-line algorithm can be found at [168], and because of q-RRT's similarities with Pq-RRT Manager (Alg. 9) and Pq-RRT Worker (Alg. 10), we omit the line-by-line and instead reference lines of the latter two enumerated algorithm descriptions.

The q-RRT Algorithm takes as input an initial state x_0 and a goal state x_G in a compact configuration space $C \subseteq \mathbb{R}^d$, a number of qubit registers n, and a quantum oracle function $\mathcal{X}$. It returns a dynamically feasible obstacle free path γ. The q-RRT algorithm adds nodes to graph T until there is a node within distance δ of the goal x_G. To add a node, q-RRT creates a 2^n sized database D of random possible nodes and the nearest parent in T to the random node, as shown in Alg. 9 on lines 4-8. A $1-$to-1 mapping F is created between database D and qubit $|\Psi\rangle$ (shown in Alg. 10, line 1). Then, the qubit is initialized and an equal superposition between states set (shown in Alg. 10 on lines 2 and 3 respectively). Let $\mathbf{W}$ be the Walsh-Hadamard transform, the operator which maps a qubit to an equal superposition of all qubit states. On lines 4-6 of Alg. 10, the operator Q performs QAA to amplify the probability amplitudes of correct states as defined by oracle $\mathcal{X}$, which tests the reachability of random samples to the nearest proposed parent P of the existing graph T, thus ensuring that T is fully reachable. For an analysis into selecting $i_{\max}$ we refer readers to [168].

Measurement is performed and the correct database element selected in Alg. 10 on line 7. After measurement is performed, the quantum state has collapsed and no further information (beyond the selected state) can be gained from the qubit. Lines 14-16 of Alg. 9 allow a node placed within δ of x_G to be admitted to T as x_G, ending the algorithm. The path is returned after successful loop execution on Alg. 9, line 19.

6.5.2 Parallel Quantum RRT

In this section, we define the Parallel Quantum RRT (Pq-RRT) algorithm as a manager (Alg. 9) worker (Alg. 10) formulation. The Pq-RRT algorithm performs reachability tests using a parallel pool of quantum computers, and is a direct extension of q-RRT inspired by parallel motion planning. The manager algorithm assigns work to the parallel pool and adds results to the tree T. The assigned work consists of each quantum worker performing a reachability check on a database D using QAA with a quantum oracle, and returning a single database element. The specific parallelization architecture is chosen for a few reasons. We consider scenarios where generally worker runtime cost dominates the message passing cost (as per [37]). This rules out such architectures as disjoint workers independently searching for a solution by growing separate trees, which have relatively little message passing but are much less runtime-efficient in finding a solution.

In the chosen manager-worker scheme, instead of discretizing a search space to allow workers to each grow a separate part of a tree, each worker is tasked with adding a single element to the tree (anywhere). This removes the idleness aspect of workers, as workers do not have to be actively listening for tree updates and do not rely on the work of others to perform their own search. Additionally, because of the probabilistic nature of the quantum search process, the parallel quantum routine generally can have all workers complete work simultaneously. This feature is not possible in non-quantum parallel architectures, as each worker is performing a stochastic search for a solution, which generally takes differing times between workers. In the quantum architecture, however, the runtime to amplify a database is much more consistent, and if each quantum worker is performing the same number of amplifications before measurement, they should complete a search nearly simultaneously. The key difference lies in the goal of the work performed. A solution does not need to be deterministically found for work to be completed (as in the non-quantum case). Work is instead completed when a solution is more likely to be found, which can be standardized for runtime across the workers.

Alg. 9, the manager algorithm, has the same inputs and outputs as q-RRT. The worker algorithm, Alg. 10, admits as inputs the current tree T, the number of qubit registers n, the quantum oracle function $\mathcal{X}$, and a copy of the shared database D, and returns a selected element of the database $[x_{\text{add}}, P]$.

Algorithm. 9 Pq-RRT Manager, shared database

Input: $x_0, x_{\text{G}}\ n$, oracle $\mathcal{X}$
Output: Path γ

1: Init p-worker pool
2: Init tree T with root at x_0
3: **while** $x_{\text{G}} \notin T$ **do**
4: **for** $i = 1$ to 2^n **do**
5: $t =$ random point
6: $P =$ closest parent of t in T
7: $D(i) = [t; P]$
8: **end for**
9: $[x_{\text{add}}, P](k) =$ Worker$(T, n, \mathcal{X}, D)$, for $k \in p$
10: **for** $k = 1$ to p **do**
11: **if** $[x_{\text{add}}, P](k) \notin T$ **then**
12: Add $[x_{\text{add}}, P](k)$ to T
13: **end if**
14: **if** $\|x_{\text{add}}(k) - x_{\text{G}}\| < \delta$ **then**
15: $x_{\text{add}}(k) = x_{\text{G}}$
16: **end if**
17: **end for**
18: **end while**
19: Return path γ from T

Two versions of this parallel formulation are possible, shared and unshared database. The fundamental difference is whether parallel pool workers create a database or perform QAA on copies of the same database D, which is created by the manager. For the shared database version, as shown in Fig. 6.3, in Alg. 9 Lines 4-8 the manager creates the database D and passes copies to each worker $k \in p$, as shown by the inputs to Alg. 10. In this way, the workers would "share" and search (copies of) the same database. The manager ignores additional identical solutions returned by different workers, which is a fast process given that the workers essentially are returning an index to a database element.

Algorithm. 10 Pq-RRT Worker, shared database

Input: $T, n, \mathcal{X}, D$
Output: $[x_{\text{add}}, P]$
1: Enumerate D via $F : \{0,1\}^n \rightarrow D$
2: Init n qubit register $|z\rangle \leftarrow |0\rangle^{\otimes n}$
3: $|\Psi\rangle \leftarrow \mathbf{W}|z\rangle$
4: **for** $i = 1$ to $i_{\max}$ **do**
5: $\quad |\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
6: **end for**
7: $[x_{\text{add}}, P] \leftarrow F(\text{measure}(|\Psi\rangle))$
8: Return $[x_{\text{add}}, P]$

For an unshared database, as shown in Fig. 6.4, the database construction step is performed within the worker algorithm (Alg. 10), which can be a classical worker until QAA is performed.

The advantage of the shared database approach is an increase in database-use-efficiency due to extracting multiple possible reachable states per database construction. This aligns with the main motivation behind this work, which is to mitigate the probability information loss due to quantum measurement collapse. Because quantum computers are reducing the time spent on the computationally intensive portion of the algorithm (state collision/reachability checks), steps such as database construction will become a larger proportion of algorithm runtime, so it is advantageous to have high database-use-efficiency. We reserve additional runtime analysis for future work. However, the shared database approach can be slightly less oracle-call efficient (compared to unshared database), with fewer reachable states are admitted per oracle call because repeated identical solutions are discarded. This is shown in Section 6.6.1 Fig. 6.14, and we discuss how important that efficiency loss is and how to mitigate it in the next section.

6.5.3 Runtime Analysis

We consider runtime and computational cost comparisons of the shared and unshared database formulations to characterize database-use efficiency and discuss potential trade-offs.

Since databases are made from parent–child connections, adding an element to a database

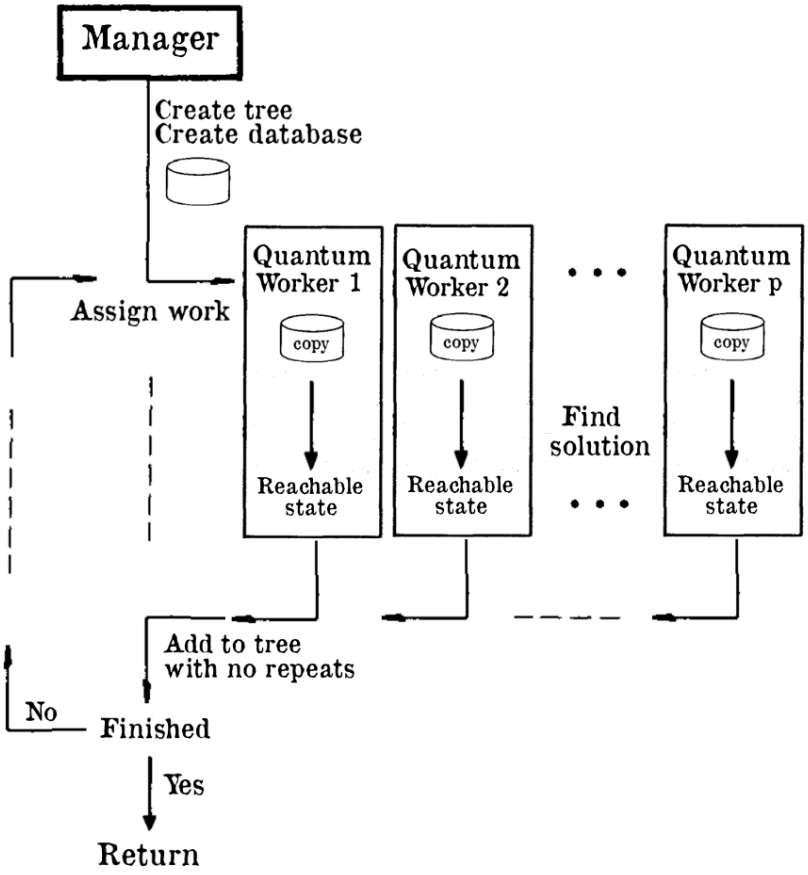

Figure 6.3. A graphical depiction of the shared database Quantum Parallel RRT algorithm. The manager (Alg. 9) creates a database and passes copies to the quantum workers (Alg 10).

requires a tree search. Let the cost of randomly generating a possible node be τ_0 (generally small), and let the unit search cost of finding a parent be τ_s (generally computing some distance metric). Then, when the tree has M nodes, building a database of size 2^n (where n is the number of qubits in use) has cost $C_D(M)$,

$$C_D(M) = (\tau_0 + \tau_s M)2^n.$$

Remark. We note that this search cost of $\tau_s M$ can be minimized through a spatial tree discretization scheme where only partial *local* tree searches are necessary, but generally this cost still scales as a function of M.

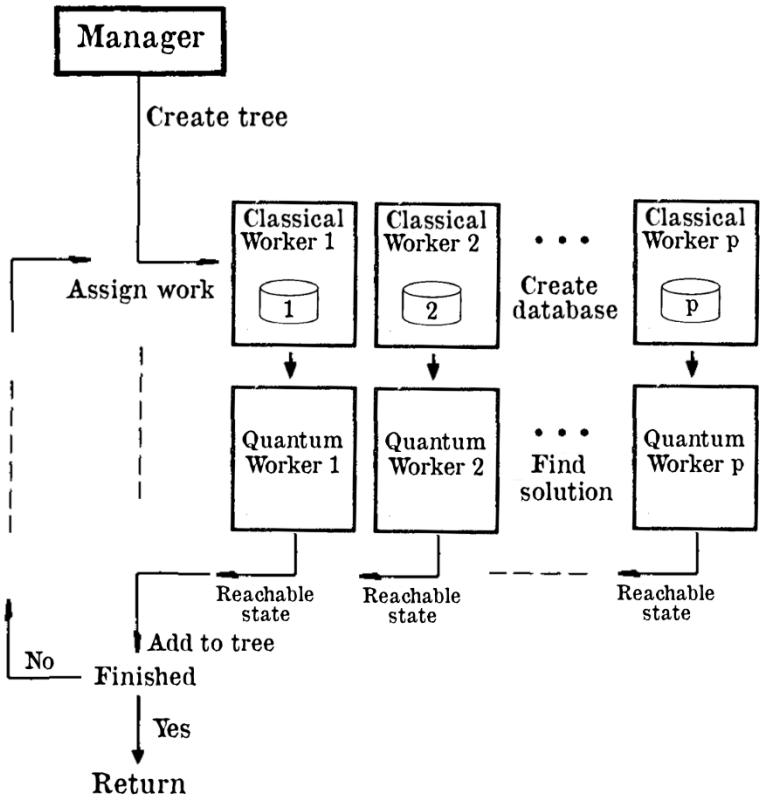

Figure 6.4. A graphical depiction of the unshared database Quantum Parallel RRT algorithm. The manager prompts p classical workers to create p different databases, which are passed to p quantum workers to find solutions, which are returned to the manager.

Let the message passing costs of passing a node between manager and worker be τ_v. Then, the cost of passing an M-node tree is $M\tau_v$ and the cost of passing a 2^n-sized database is $2^{n+1}\tau_v$.

Remark. The $+1$ appears in the exponent of the database-passing cost due to database elements being a parent–child pair, but this can be reduced to be an *index* rather than node.

Let the cost of one application of QAA be τ_Q. Since optimal amplification is achieved by $\frac{\pi}{4}\sqrt{2^n/m}$ applications of QAA, we define the optimal amplification cost as C_Q,

$$C_Q = \left\lfloor \frac{\pi}{4}\sqrt{\frac{2^n}{m}} \right\rfloor \tau_Q,$$

where m is the number of solutions within the database.

Remark. In general, for traditional (non-quantum) sampling-based planning algorithms, costs like $\tau_Q >> \tau_{0,s,v}$. Connection costs far outweigh sampling and tree-traversal costs. Additionally, the function is floored, as QAA can only be applied a discrete number of times.

In the following, we derive two types of costs: runtime and total cost. Runtime cost, for p-core parallel algorithms, is the analog to computation time, where centralized components are counted fully and decentralized components (that run in parallel) are discounted by a factor of p. Total cost has two analogous scenarios: total power consumption and sequential algorithm runtime (to accomplish the same task), where decentralized components lose the p-fold discount factor. Terms in Props. 6 and 7 are presented in the order with which they are incurred when the algorithm is running.

Proposition 6. *The runtime cost of the unshared database algorithm shown in Fig. 6.4, with p workers, is,*

$$C_{Unsh}^{Run} = p \sum_{k=1::p}^{M} k\tau_v + \frac{M}{p} 2^n \tau_0 + 2^n \sum_{k=1::p}^{M} k\tau_s + \frac{M}{p} C_Q + M\tau_v, \tag{6.1}$$

and the total cost is,

$$C_{Unsh}^{Tot} = p \sum_{k=1::p}^{M} k\tau_v + M2^n \tau_0 + 2^n p \sum_{k=1::p}^{M} k\tau_s + MC_Q + M\tau_v, \tag{6.2}$$

where the notation $\sum_{k=1::p}^{M}$ refers to the sum from $k = 1$, stepping by p, to M, while $k < M$.

For the runtime cost in Eq. (6.1), the first term, $p\sum_{k=1::p}^{M} k\tau_v$, is the message passing cost of the current tree, of growing size, being sent to the p workers. Since the tree is growing by p nodes each iteration, the message passing cost is made of a sum and must be incurred p times, as the tree must be sent to each worker each iteration. The second term, $\frac{M}{p}2^n\tau_0$, is the random sampling cost incurred in database construction but each worker, and since this process is distributed, at runtime the M multiplier is scaled by $\frac{1}{p}$. The third term, $2^n\sum_{k=1::p}^{M} k\tau_s$, is the parent search cost, which grows as the tree grows by p nodes each iteration. The fourth term, $\frac{M}{p}C_Q$, is the cost of applying QAA (and testing for reachability), and is distributed at runtime (divided

over p). The last term, $M\tau_v$, is the cost of sending all M found nodes back to the manager.

To attain the total cost in Eq. (6.2), the distributed terms of Eq. (6.1) (terms 2, 3, and 4) are multiplied by p to find sequential runtime cost.

Proposition 7. *The runtime cost of the shared database algorithm shown in Fig. 6.3, with p workers, is,*

$$C_{Sh}^{Run} = \frac{M}{p}2^n\tau_0 + 2^n \sum_{k=1::p}^{M} k\tau_s + M2^{n+1}\tau_v + \frac{M}{p}C_Q + M\tau_v, \tag{6.3}$$

and the total cost is,

$$C_{Sh}^{Tot} = \frac{M}{p}2^n\tau_0 + 2^n \sum_{k=1::p}^{M} k\tau_s + M2^{n+1}\tau_v + MC_Q + M\tau_v, \tag{6.4}$$

where the notation $\sum_{k=1::p}^{M}$ refers to the same as previous.

For the runtime cost in Eq. (6.3), the first term, $\frac{M}{p}2^n\tau_0$, is the random sampling cost incurred by the manager in database construction, which occurs $\frac{M}{p}$ times in the shared database case. The second term, $2^n\sum_{k=1::p}^{M} k\tau_s$, is the parent search cost, which is identical to term 2 of Eq. (6.1). The third term, $M2^{n+1}\tau_v$, is the message passing cost of sending each worker a database (for a total of M passes). The fourth term, $\frac{M}{p}C_Q$, is the cost of applying QAA and is distributed at runtime. The last term, $M\tau_v$, is the cost of sending all M found nodes back to the manager. To attain the total cost in Eq. (6.4), the distributed term of Eq. (6.3) (term 4) is multiplied by p to find sequential runtime cost.

There are two main cost differences between the unshared and shared database formulations in creating an M-node tree. In both the runtime cost, in Eq. (6.1), and the total cost, in Eq. (6.2), the unshared algorithm incurs a message passing cost of $p\sum_{k=1::p}^{M} k\tau_v$ in passing the growing tree (from 1 to M nodes) to p workers. In Eq. (6.3) and (6.4), the shared algorithm incurs a message passing cost of $M2^{n+1}\tau_v$ instead, in passing the database to workers (M times). The other terms in the runtime costs between the versions are identical. Whether the difference in message passing costs favors one scheme over the other is dependent on the relative sizes of

M and 2^n. For small trees and large databases, the unshared database method incurs a smaller cost. For large trees and small databases, the shared database method incurs a smaller cost.

The second difference is in the total cost. In Eq. (6.2), the unshared algorithm incurs a total database creation cost of $M2^n\tau_0 + 2^n p\sum_{k=1::p}^{M} k\tau_s$ to create an M node tree. In Eq. (6.4), the shared algorithm incurs a total database creation cost of $\frac{M}{p}2^n\tau_0 + 2^n\sum_{k=1::p}^{M} k\tau_s$, effectively giving a p-fold reduction in total database creation cost. We refer to this as *increased database efficiency*, since an M-node tree is created with fewer database constructions.

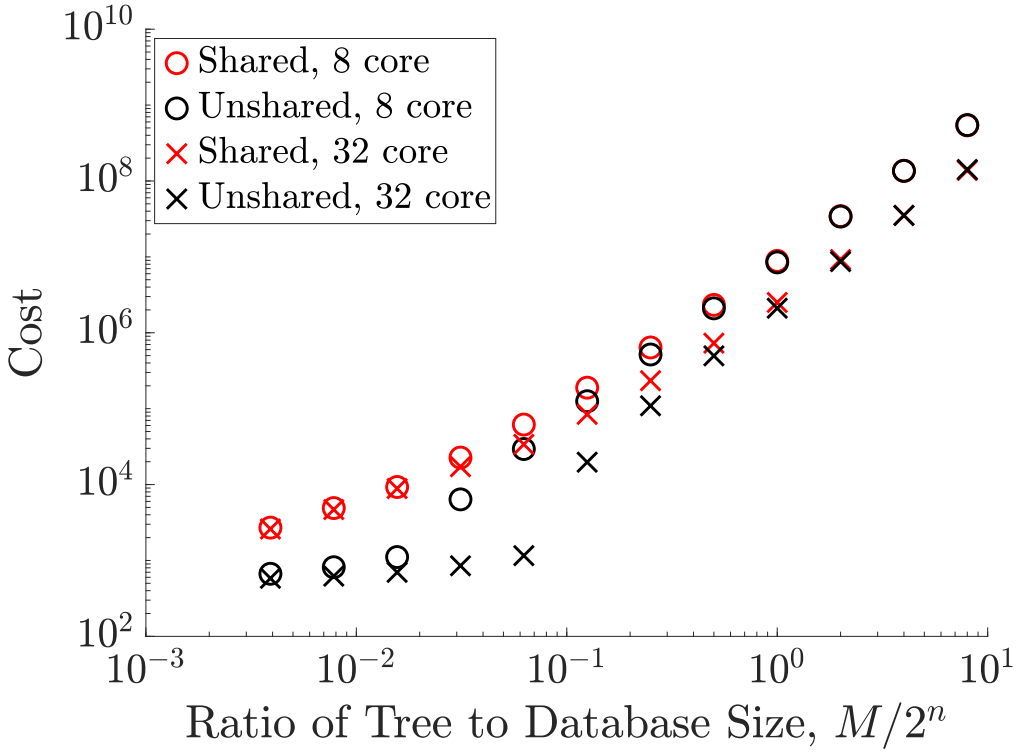

Figure 6.5. A comparison in loglog space of the cost differences between the unshared and shared database architectures as the relative final tree size M varies with respect to database size 2^n.

In Fig. 6.5, we compare runtime costs of the two architectures when weights are assigned to each cost. As noted above, at runtime, the cost difference comes in message passing either the tree or the database, so the ratio of number of desired nodes M to total database size is plotted against cost. We show cost curves for the 8 and 32 core cases for both architectures to further

understanding on how costs scale with number of cores. As is expected, 32-core cases minimize costs across the entire plotted domain for both architectures. Additional cores seem to minimize the unshared database cost to a greater extent than the shared database cost. At small tree sizes relative to the database, the unshared architecture minimizes cost, and the curves appear to converge as the end tree size eclipses the database. The visualization is possible through the log scaling of the cost, but this scaling obscures the fact that the shared formulation actually becomes lower cost at high ratios of $\frac{M}{2^n}$.

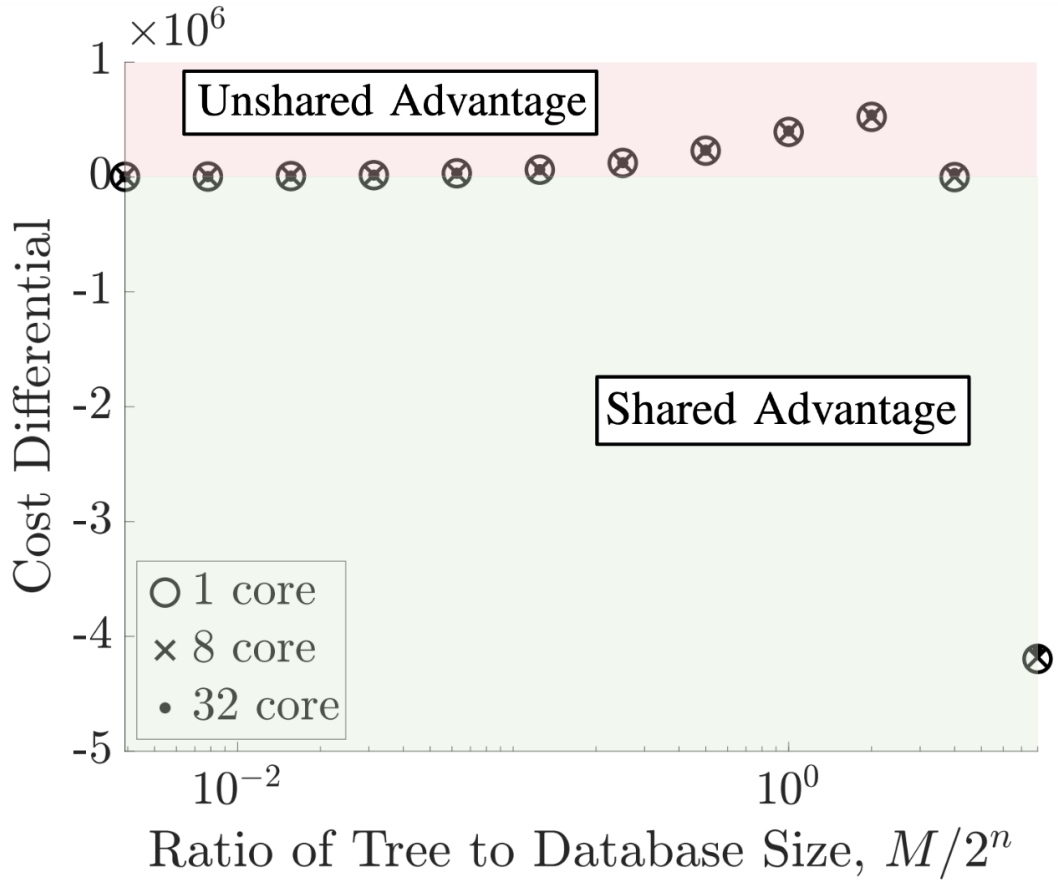

Figure 6.6. The differential cost between the shared and unshared database architectures as the relative final tree size M varies with respect to database size 2^n.

In Fig. 6.6, we depict a semilog plot of the cost differential between the shared and unshared architectures over the saim domain of ratios of tree to database size. For smaller final trees, the unshared formulation maintains a cost advantage, but the advantage skews strongly

to the shared database as the tree size grows. This effect is obscured in Fig. 6.5 by the loglog nature of the scaling and apparent equal performance on the upper end of the domain. The cost differential over the entire domain appears relatively similar between 1, 8, and 32 core cases.

A key assumption in the above analysis is that for both algorithm architectures, in each iteration, the p cores add p nodes to the tree. For both architectures this is overlooking the fact that the database cannot be optimally amplified, as applying an operator can only be done an integer number of times, and the optimal amplification number is exactly defined using an irrational number (π). Furthermore, even if optimal amplification was possible, in general there is still a nonzero probability of measuring a 'bad' (unamplified; as defined by the oracle) element, which should not be admitted to the tree (and can be caught with a final deterministic check). This simple fact eliminates the possibility that in each iteration, the p cores can always add p nodes to the tree.

A further complication for the shared database setup is that nodes may find identical solutions, further lowering efficiency. In the following section we explore to what extent this is likely, and guidelines for minimizing this effect to allow high database use efficiency.

A different possible solution is to use a partitioned "shared" database, where a single larger database is passed to each worker with a partition rule such that each worker searches different and non-overlapping portions of the database. This would eliminate the repeated solution problem, but is more database efficient (if each partitioned database is considered part of one database), but would require larger databases and is functionally equivalent to the unshared database architecture, with the manager creating databases rather than the workers. A larger database with more solutions also helps to eliminate the repeated solution problem. We also note that although classically connection costs $\tau_Q >> \tau_{0,s,v}$, this may no longer be the case when quantum computers are used to determine reachability.

6.5.4 Pq-RRT Probability Results

In this section, we characterize p parallel quantum workers finding multiple solutions when Pq-RRT is operating in a shared database setup. Since all the workers are independently analyzing the same 2^n-sized database D, with m oracle-marked solutions, in general multiple workers may arrive at the same solution. This represents an efficiency loss to the shared database setup, so in what follows we characterize the worst and best case events. The worst case event is all workers arriving at the same solution, which has the runtime performance as non-parallel q-RRT but is p-times less oracle call efficient. The best case event is each worker finding a different solution, which has no runtime or oracle call efficiency loss, and ends with p solutions. Only when $p \geq m$ can all solutions be found in a single parallel pass. To build fast solutions, an understanding of the effects of choices of p (and to some extent m) is necessary to maximize efficiency.

We assume that the database is optimally amplified according to $i_{\max}$ applications of Q, where $i_{\max}$ is given by,

$$i_{\max} = \frac{\pi}{4}\sqrt{2^n/m}\,;$$

see [150]. We note that our connectivity analysis on estimating database correctness in [168] can be applied to the section at hand in order to attain optimal amplification (to maximize chances of measuring a solution). In what follows, let G be the event that a good state, as defined by the oracle, is measured by a worker after $i_{\max}$ iterations of Q.

Without Oracle Errors

Lemma 2. *Let there be a parallel quantum process with p workers and a shared 2^n-sized database with m solutions. The probability that all p workers find the same solution, as shown in Fig. 6.7, is,*

$$\mathbb{P}(\textit{same solution}) = \mathbb{P}(G)^p m^{1-p}, \tag{6.5}$$

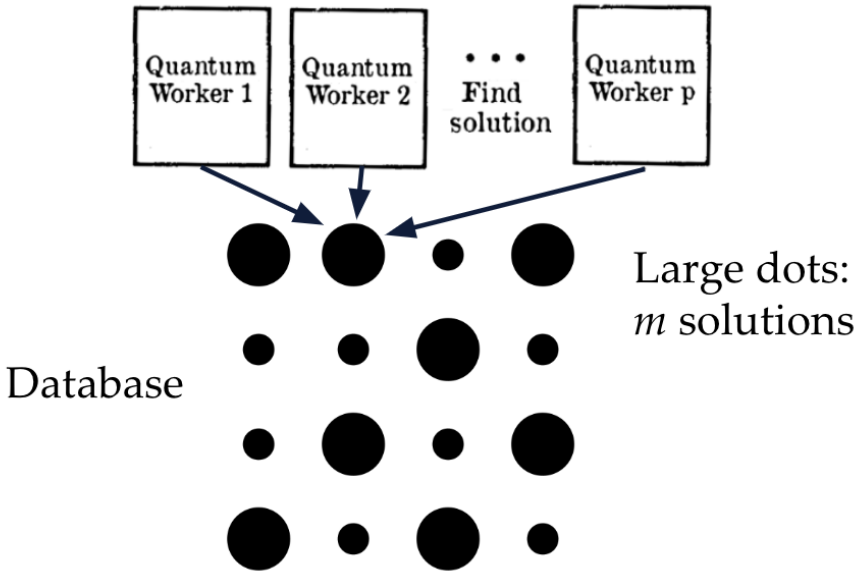

Figure 6.7. A visual depiction of Lemma 2, where each quantum worker finds the same solution in the database. This is the worst case scenario (besides all quantum workers finding incorrect solutions, which are depicted as small dots) in terms of efficiency. The database is depicted as dots, with larger dots indicating the solutions. The arrows indicate measurement of a particular solution, to be passed back to the manager.

where $\mathbb{P}(G)$ *is the total probability of event G,*

$$\mathbb{P}(G) = \sin^2((2i_{\max}+1)\theta),$$

and where θ *is defined such that* $\sin^2(\theta) = \frac{m}{2^n}$.

Proof. To attain this result, we observe that after $i_{\max}$ iterations of Q, all m solutions have equal probability of measurement given by $\mathbb{P}(G)/m$. The probability that all p workers measure a *particular* solution i is,

$$\mathbb{P}(\text{particular solution}) = \left(\frac{\mathbb{P}(G)}{m}\right)^p,$$

and this is multiplied by m to generalize to finding *any* same solution, yielding Eq. (6.5). The total good measurement probability and the definition of θ can be found at [39]. □

Lemma 3. *For* $m \geq p$ *and* $m, p \in \mathbb{N}$*, the probability that all workers find different solutions, as*

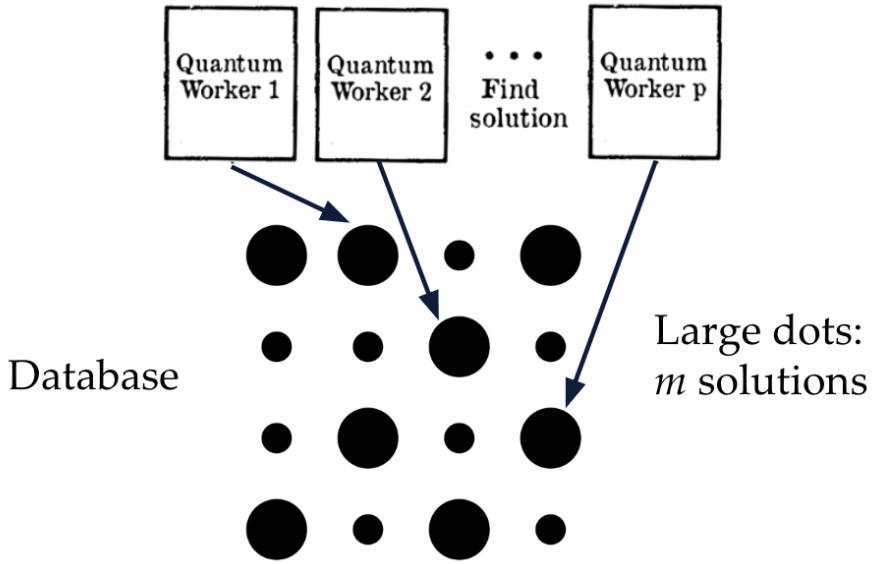

Figure 6.8. A visual depiction of Lemma 3, where each quantum worker finds a different solution in the database. This is the best case scenario in terms of efficiency.

shown in Fig. 6.8, is given by,

$$\mathbb{P}(\textit{different solutions}) = \frac{\mathbb{P}(G)^p m!}{m^p (m-p)!}, \tag{6.6}$$

Proof. This result follows from m permute p (the number of possible ways p objects can be selected, without replacement, from m possibilities) over the total number of possible outcomes m^p. This is scaled by the likelihood that all workers find a correct solution, $\mathbb{P}(G)^p$, to yield Eq. (6.6). □

For the worst case scenario in Eq. (6.5), as $m \to \infty$, $\mathbb{P}(\text{same solution}) \to 0$. This makes intuitive sense: as the number of available solutions increases, the likelihood of all workers finding the same solution decreases as a function with power $1-p$, where $p \geq 2$, as it is only sensible to conjecture about the parallel behavior of 2 or more workers.

For the best case scenario in Eq. (6.6), as $m \to \infty$, $\mathbb{P}(\text{different solutions}) \to \mathbb{P}(G)^p$. This also makes intuitive sense: as the number of solutions increases, the likelihood of all workers finding different solutions approaches the total likelihood of all workers finding a solution. We

note that for $m \geq p$ and $m, p \in \mathbb{N}$:

$$\lim_{m \to \infty} \frac{m!}{m^p (m-p)!} = 1.$$

The number of solutions m should be as large as practicable to reduce efficiency loss through oracle overlap.

Lemma 4. *For $p \geq m$, the expected number of workers p, to find all m solutions within D in one pass, is given by,*

$$E(p) = \frac{mH_m}{\mathbb{P}(G)},$$

where H_m is the m^{th} harmonic number. Equivalently, $\frac{E(p)}{p_2}$ also describes the expected number of passes at a single database that the set number of workers p_2 must make to find all solutions.

Proof. We reach this result from the application of the Coupon Collector's problem [182], with a minor modification, to the independent quantum computing worker processes. Briefly, the coupon collector's problem concerns questions about the "collect all coupons from cereal boxes to win" contest. In this context, solutions in the database represent coupons (to be found with a certain probability), and number of workers represents (the expectation of) how many cereal boxes must be opened to find one of each solution/coupon. The result takes into account that workers may return the same solution. This application is scaled by the total probability of correct solutions, $\mathbb{P}(G)$, to account for the proportion of the time when a good solution is not measured. □

Lemma 4 allows a parallel (and repeated) architecture to be chosen based upon knowledge of m, such as from our connectivity analysis on estimating database correctness in [168]. Additionally, this leads to the database construction tool described in the following section that allows the proportion $m/2^n$ to be made larger or smaller. We remark that in general, it is possible to calculate the probability of n workers coinciding on the same exact solution, as it relates to the multinomial distribution.

With Oracle Errors

We also consider the case where the oracle is making repeated false positive and false negative errors. Let the probability that a state is marked by the oracle incorrectly as good be given by $q \in [0,1]$ (false positive), and let the probability that a state is marked by the oracle incorrectly as bad be given by $v \in [0,1]$ (false negative). Let the number of ground-truth solutions in the database tagged by the oracle as a solution be $m_1 \leq m$. Let the number of actual ground truth solutions in the database, which were mistakenly tagged by the oracle as bad be m_2, such that,

$$q = \frac{1-m_1}{m},\ v = \frac{m_2}{2^n - m},$$

as shown in Fig. 6.9.

First, we derive the likelihood that a single worker finds a real solution. Let G^* be the event that a real, ground truth solution (not according to the oracle) is measured for addition to the tree after optimal amplification with QAA.

Lemma 5. *The total probability of measuring real, ground truth solutions is,*

$$\mathbb{P}(G^*) = \frac{m_1}{m}\mathbb{P}(G) + \frac{m_2}{2^n - m}(1 - \mathbb{P}(G)).$$

Proof. We attain this result by adding the probability of measuring a correctly tagged good solution, $\frac{m_1}{m}\mathbb{P}(G)$ to the probability of measuring an incorrectly tagged bad solution, $\frac{m_2}{2^n-m}(1 - \mathbb{P}(G))$. □

The following lemma is a modification of Lemma 2 to include oracle errors.

Lemma 6. *The probability that all p workers find the same ground truth solution, adjusted for oracle mistakes, is,*

$$\mathbb{P}(\textit{same solution}) = m_1 \left(\frac{\mathbb{P}(G)}{m}\right)^p + m_2 \left(\frac{1-\mathbb{P}(G)}{2^n - m}\right)^p. \tag{6.7}$$

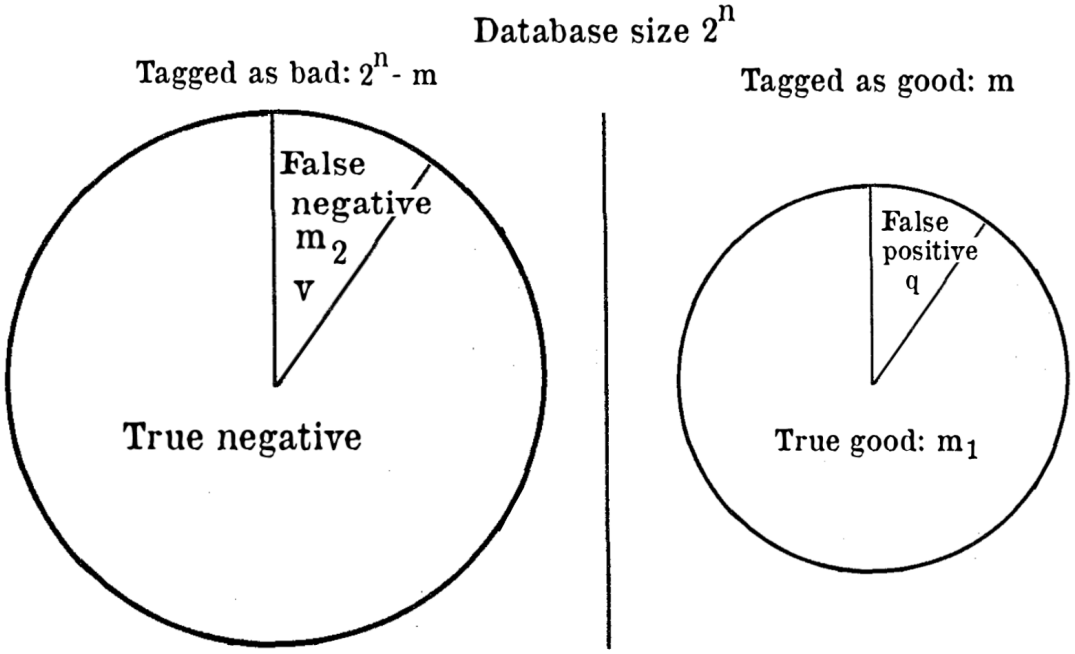

Figure 6.9. A graphical depiction of the false positive and false negative regions of a database with good and bad tags by an oracle. Of the m good tags, m_1 are true good tags, with a false positive probability of q. Of the $2^n - m$ bad tags, m_2 are actually good elements, with a false negative probability of v. After optimal QAA, the probability of a tagged-as-good state being measured is $\mathbb{P}(G)$.

Proof. This result follows from the addition of the probability of all p workers measuring a *particular* correctly tagged good solution, $\left(\frac{\mathbb{P}(G)}{m}\right)^p$, and the probability of all p workers measuring a *particular* incorrectly tagged bad solution, $\left(\frac{1-\mathbb{P}(G)}{2^n-m}\right)^p$. Each of these is multiplied by m_1 and m_2 respectively, to consider *any* real solution, then added together to yield Eq. (6.7). □

For the worst case scenario in Lemma 6, again, as $m \to \infty$, $\mathbb{P}(\text{same solution}) \to 0$. The following lemma is a modification of Lemma 4, adjusted for oracle false positive errors.

Lemma 7. *The expected number of workers p to find all m_1 ground truth solutions, when false positive oracle errors are taken into account, is given by,*

$$E(p^*) = \frac{m_1 H_{m_1}}{\frac{m_1}{m}\mathbb{P}(G)},$$

where H_{m_1} is the $m_1{}^{th}$ harmonic number.

Proof. This result follows similarly to Lemma 4, with m_1 rather than m, and where the expectation from the Coupon Collector's Problem is scaled by the proportion of the time that a ground truth good solution is found, $\frac{m_1}{m}\mathbb{P}(G)$. □

Remark. Oracle false negative errors, when considered in Lemma 7, transform the problem into a simple version of the Weighted Coupon Collector's Problem [183] (also known as McDonald's Monopoly), with some "rare coupons" to find (represented by the m_2 false negative solutions), and some "common coupons" (represented by the m_1 real positive solutions). In reality, after optimal amplification, the probability of measuring the m_2 good solutions incorrectly tagged as bad, $\frac{m_2}{2^n-m}(1-\mathbb{P}(G))$, is small, and additionally we care more about finding correctly identified good solutions than determining incorrectly tagged bad solutions from a database.

6.5.5 Quantum Database Annealing

In this section, we define the Quantum Database Annealing (QDA) strategy, shown in Alg. 11. QDA builds databases with elements constrained to a certain distance from the parent node, as defined by a temperature matrix H and iterator h. The QDA strategy is an alternative to standard database construction and is inspired by the optimization technique simulated annealing and our investigations into oracle call constraints in [168]. It represents a possible way to guide database construction to achieve a particular algorithmic goal, such as approximately selecting m (with regard to the previous section), or in this case, initial fast expansion via spread node placement followed by increasing density through closer node placement.

In a broad sense, sampling strategies for motion planning have been explored since the beginning, with strategies such as medial axis sampling, boundary sampling, Gaussian (obstacle) sampling, goal biasing, and hybrid schemes [27]. QDA is distinct from current classical computing approaches because the initial goal in this quantum formulation is to make sample connections less likely. When paired with a large database, QDA exploits quantum computing's ability to quickly find unlikely solutions, as shown in Fig. 6.10. This results in a motion planner that can explore very quickly when measured on oracle calls.

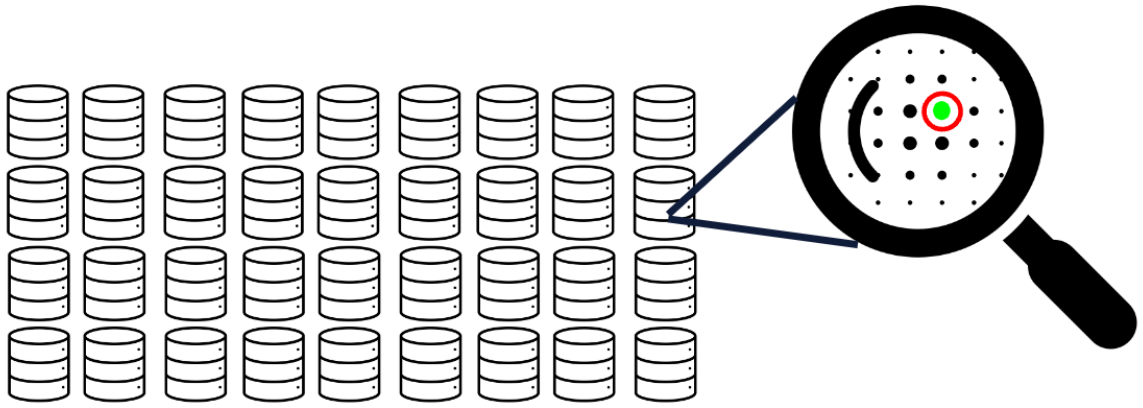

Figure 6.10. An illustration of a larger database with fewer but more desirable solutions, to be exploited by quantum computing to provide farther away solutions, resulting in faster exploration. Icons from Flaticon.

QDA first samples according to a uniform distribution over the configuration space C. The nearest (Euclidean distance) existing node to the sample is chosen as the parent. Next, the resulting parent-child relationship is constrained to be within a ball of radius $H(h)$ (with iterator h) while maintaining child-sample direction. The resultant pair is added to the database. An alternative is to sample initially over a disc or boundary at a distance constrained by $H(h)$.

In the beginning, with high temperature (when $H(h)$ is large), QDA will build a large database of further away and therefore less likely solutions. This allows further reachable solutions to be found quickly as compared to q-RRT and RRT. As the path planning problem continues (as h increases), the temperature ($H(h)$) may drop to account for the addition of new nodes and to increase the ratio of good solutions in the database. The database size 2^n may also drop throughout the problem to increase efficiency, as when there are more solutions, smaller databases function as well as larger. When no additional information is known to guide sampling region, an alternative but similar sampling method in very large bounded configuration spaces is to build extremely large databases of unlikely solutions in an attempt to span long obstacle free channels quickly.

The differences between q-RRT and q-RRT with QDA, Alg. 11, lie in the latter algo-

Algorithm. 11 q-RRT with Quantum Database Annealing

Input: x_0, x_G n, oracle $\mathcal{X}$
Output: Path γ

Init tree T with root at x_0
Define temperature array H, index $h = 0$
while $x_G \notin T$ **do**
  for $i = 1$ to 2^n **do**
    $t =$ random point
    $P =$ closest parent of t in T
    Constrain t to disc of dist. $H(h)$ from P
    $D(i) = [t; P]$
  end for
  Enumerate D via $F : \{0,1\}^n \rightarrow D$
  Init n qubit register $|z\rangle \leftarrow |0\rangle^{\otimes n}$
  $|\Psi\rangle \leftarrow \mathbf{W}|z\rangle$
  for $i = 1$ to 2 **do**
    $|\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
  end for
  $[x_{\text{add}}, P] \leftarrow F(\text{measure}(|\Psi\rangle))$
  if $\|x_{\text{add}} - x_G\| < \delta$ **then**
    $x_{\text{add}} = x_G$
  end if
  Add $[x_{\text{add}}, P]$ to T
  $h{+}{+}$
end while
Return path γ from T

rithm's lines 2, 7, and 21. Alg. 11 line 2 is where the temperature array H is defined and iterator h initialized. On Alg. 11 line 7, the temperature constraint is carried out by modifying the random point t with respect to P, the closest parent of t in T. On Alg. 11 line 21, the iterator h is incremented to allow different temperatures on future database constructions. In the defined formalism, the database size 2^n is set and not decreased.

This approach to guided sampling is distinct from efforts within guided sampling in non-quantum literature. Generally, the point of guided sampling is to make the search for additional states and solutions easier. Additionally, there is no reason (until the introduction of quantum computers to motion planning) to construct *databases* of possible solutions, the algorithms always progress by testing one possible state at a time. With QDA, however, we design a guided sampling scheme with a polar opposite goal to existing efforts: making the solution process more difficult. This is because we are able to exploit the quantum advantage, and by making the solution process more difficult, we are able to find 'better' solutions in the sense of solutions that allow vigorous (and provable with respect to metrics such as reachability and safety) expansion into an environment. When constructing databases, we also think of guided sampling in a different lens.

6.6 Results and Discussion

In this section, we show tree creation comparison results within two dimensional obstacle environments for Pq-RRT, q-RRT, Parallel RRT, and RRT. Direct comparisons highlighting the simulated quadratic runtime advantage of q-RRT over standard RRT are shown in detail in our previous work at [168]. Unless otherwise stated, results are presented comparing algorithm performance for solving the same problem in the same randomized obstacle environments. Both Pq-RRT and Parallel RRT are implemented with eight cores (workers). Both quantum algorithms use databases of size 2^8, and the classical versions of the algorithms (RRT and Parallel RRT) replace the database construction and quantum search process with single reachability tests.

The specific version of Parallel RRT is a Manager-Worker formulation (outlined in [37] under Manager-Worker RRT), where a manager processes the tree and assigns single-node expansion work to workers, as expansion is the computationally expensive part of planning.

All path planning simulations are run with Matlab v2022b on an eight core MacBook Pro with M2 chip. Quantum states and algorithms are simulated with the Matlab Quantum Computing Functions library [150]. All algorithms use the following arbitrary dynamics and reference tracking controller to test reachability for node admittance to the tree,

$$x(t+1) = Ax(t) + Bu(t)\,,\, x(0) = x_{\text{parent}},$$

$$A = \begin{bmatrix} -1.5 & -2 \\ 1 & 3 \end{bmatrix}, B = \begin{bmatrix} 0.5 & 0.25 \\ 0 & 1 \end{bmatrix},$$

$$u(t) = -Kx(t),$$

$$K = \begin{bmatrix} 1.9 & -7.5 \\ 1 & 7 \end{bmatrix}.$$

The constant gain matrix K can be any matrix such that the closed loop system is stable.

6.6.1 Node Placement and Oracle Calls

In this section, we highlight the advantages and disadvantages of q-RRT and Pq-RRT over RRT and Parallel RRT in being able to add nodes to the tree. We show performance compared to runtime in seconds, which we call wall-clock time to highlight that this is the "real" runtime of the simulations, and then to number of oracle calls, which functions as the projected runtime improvement if algorithms are run on quantum devices. For performance metrics, we consider two quantities: number of oracle calls and number of nodes. Number of oracle calls is the metric for comparing how much Parallel q-RRT is able to speed up computation as compared to q-RRT. For all other comparisons, number of nodes is the chosen metric, as it signifies on a functional

level the ability of each algorithm to search for a solution. Each point in Fig. 6.11- 6.14 represents one 30-node tree creation, chosen to showcase average performance.

First, in Fig. 6.11 we show the wall-clock speed of Pq-RRT and q-RRT in being able to perform oracle calls to analyze the amount of computation speedup achieved. Next, we compare the wall clock speed of the two classical algorithms (in Fig. 6.12) and the two quantum algorithms (in Fig. 6.13) to study the relative performance gain in parallelizing the quantum routines compared to classical parallel advantage. Lastly, in Fig. 6.14, we change to an oracle call (our quantum time surrogate) vs node creation comparison to show the efficiency of the quantum algorithms in admitting nodes to the tree. The performance advantage of Pq-RRT is inferred to combine the advantage shown over q-RRT in Fig. 6.13 and the advantage shown over the classical algorithms shown in Fig. 6.14.

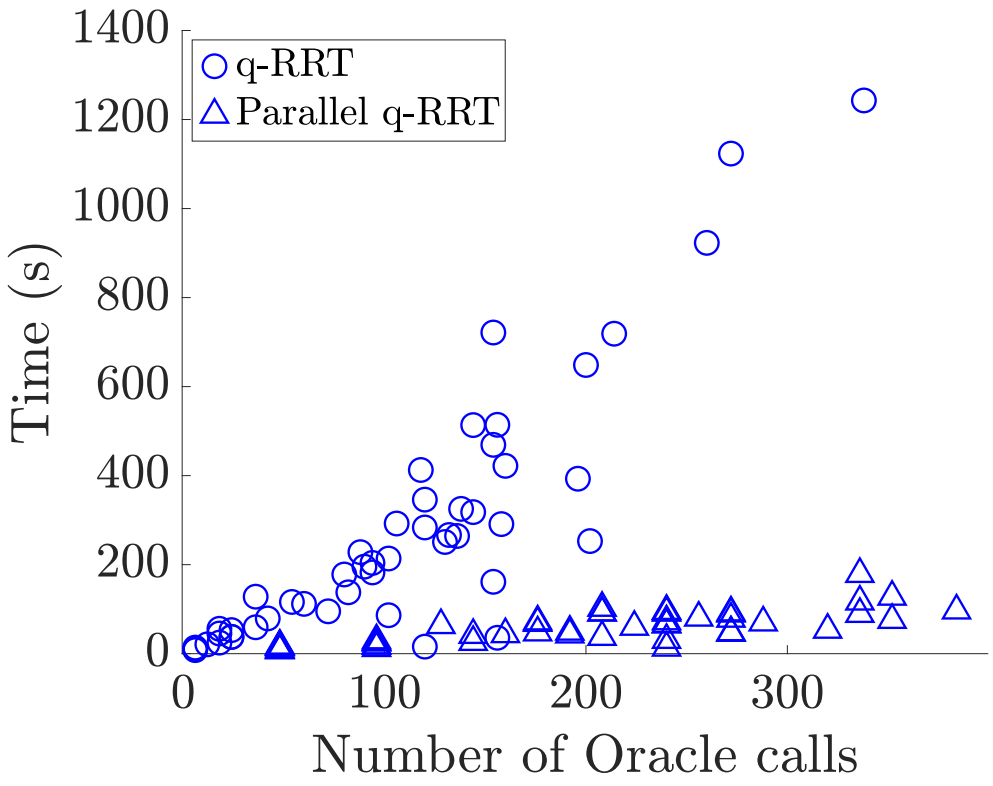

Figure 6.11. Comparison of the wall-clock speed (in seconds) of q-RRT, and shared database Pq-RRT in performing oracle calls or reachability tests.

Figure 6.11 depicts the wall-clock speed of q-RRT and parallel q-RRT in performing oracle calls. Pq-RRT performs oracle calls in less time than q-RRT, as it more efficiently uses multiple workers to retrieve information from created databases. The relationship between oracle calls and time is approximately linear (as expected), and with a linear fit (using linear least squares), Pq-RRT shows a smaller slope (0.20) compared to q-RRT (3.52), with slope referring to seconds per oracle call (lower is more efficient). A classical computing shortcut is used which allows Pq-RRT to be 17.6 times more efficient in performing work. In quantum computing simulation, a single created database is analyzed for reachability once, then amplified once, and each worker then measures a solution. This shortcut would not be possible on a quantum device, as qubits, once they are created, cannot be copied, so each worker would need to perform the reachability analysis separately. This would change the expected slope difference to be approximately 8 times less than 17.6 (as 8 cores are used), for a total work (oracle call) efficiency gain of 2.2.

Figure 6.12 depicts the wall-clock speed of RRT and Parallel RRT in admitting nodes to the graph, as opposed to performing oracle calls. The same data for q-RRT and Pq-RRT is shown in Fig. 6.13. This comparison factors in differing node-admission oracle call efficiencies. The parallel versions of both algorithms, Parallel RRT and Pq-RRT, each are on average more time efficient than the non-parallel versions in admitting reachable states to the tree.

The intuition behind slope in Figures 6.12 and 6.13 is seconds per node, with lower numbers meaning more efficient. Pq-RRT (slope 3.17) in particular shows greater improvement over q-RRT (slope 25.3) than Parallel RRT (slope 0.58) over RRT (slope 1.23), as is evidenced by a larger difference in slope (8.0-fold efficiency increase compared to 2.1-fold), as calculated with a linear fit and linear least squares. The quantum algorithms, when measured by real time, lag behind both non-quantum RRT versions because they are not benchmarked on quantum computers (see the y-axis label differences between Fig. 6.12 and Fig. 6.13). The quantum computing simulations are performed via large arrays on classical devices. On a quantum device, we expect the run-time to be analogous to oracle calls, as discussed next.

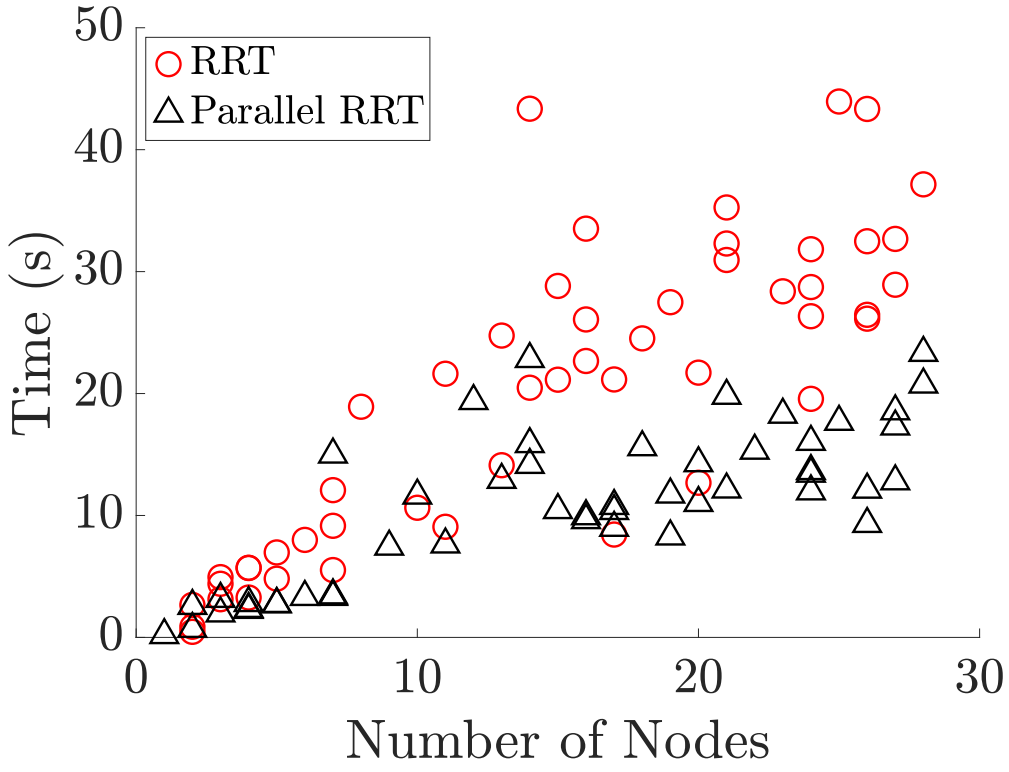

Figure 6.12. Comparison of the wall-clock speed (in seconds) of RRT and Parallel RRT in admitting reachable states to the tree.

Figure 6.14 depicts the oracle call efficiency of all four algorithms in admitting reachable states to the tree as a function of the number of oracle calls it takes. This figure is analogous to expected run-time when the quantum algorithms are executed on a quantum device. Slopes are found with a linear fit using linear least squares and represent the number of oracle calls per node, with lower being more efficient. The efficiency advantage of q-RRT (slope 7.6) and Pq-RRT (slope 10.7) in admitting reachable states is shown over RRT (slope 21.5) and Parallel RRT (slope 19.4). The q-RRT algorithm is more efficient than Pq-RRT due to the fact that multiple workers can simultaneously return the same solution from a database (as explored in Props. 2 to 7), and repeat solutions are discarded. However, Pq-RRT is capable of making simultaneous oracle calls with different workers, so for parallel vs not parallel time comparisons we refer the reader to Fig. 6.12 and Fig. 6.13.

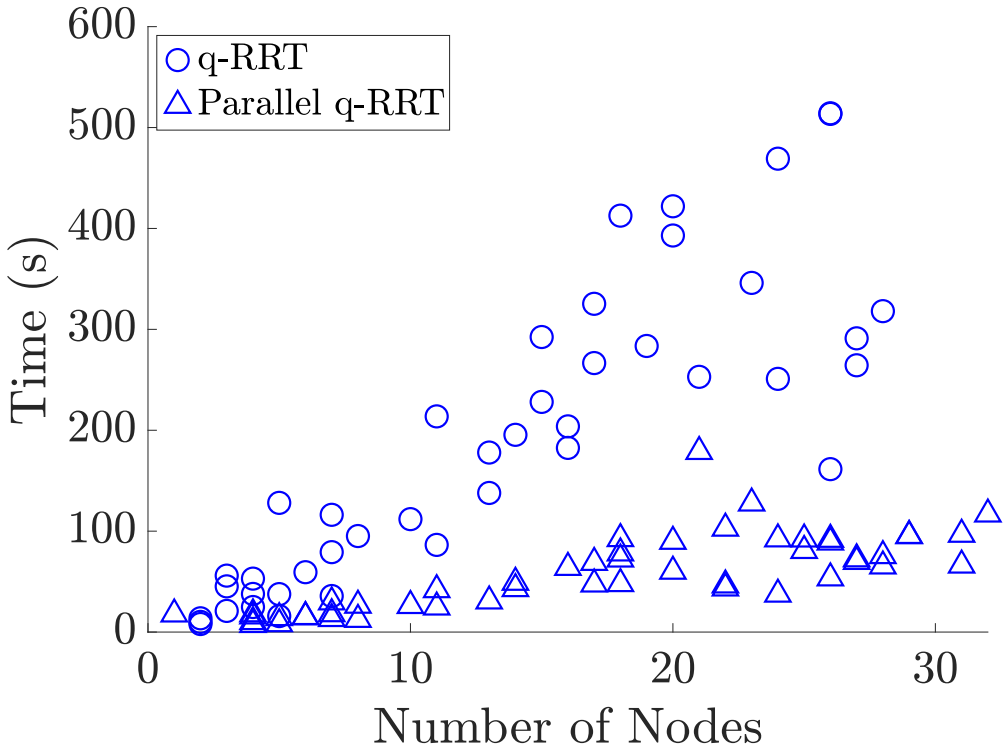

Figure 6.13. Comparison of the wall-clock speed (in seconds) of q-RRT and Pq-RRT in admitting reachable states to the tree.

The conclusions of the above analysis are the following: Pq-RRT is more time-efficient than q-RRT in performing work and placing nodes, Pq-RRT shows a greater time efficiency increase over q-RRT than Parallel RRT does over RRT, and q-RRT is slightly more oracle-call-efficient than Pq-RRT, but both quantum algorithms are more oracle-call-efficient than the classical algorithms.

6.6.2 Exploration Speed

The results of this section are extensions to our results in [168] between q-RRT and RRT, showing q-RRT's ability to explore quickly and in a generalized environment. We show a heat-map of state space nodes placed within a certain number of oracle calls. Oracle calls are chosen as a substitute to time because the quantum computer simulation performs slowly on classical devices. Actual runtime is expected to be analogous to the number of oracle calls, as

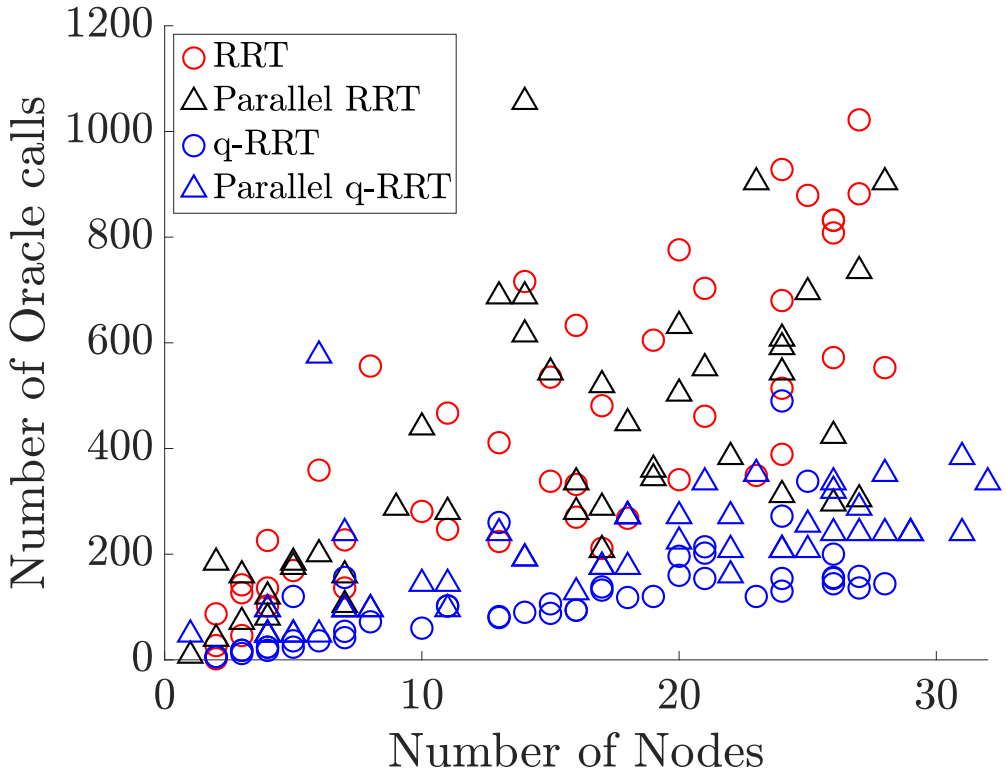

Figure 6.14. Comparison of the oracle call efficiency of RRT, Parallel RRT, q-RRT, and Pq-RRT in admitting reachable states to the tree.

reachability tests for the local planner consume the majority of the algorithm runtimes. Each algorithm is tested over 100 trials. Each trial is cut off after a certain number of oracle calls to show each algorithm's speed of node placement. Let oracle efficiency be the ratio of total nodes placed over total oracle calls. In each figure, the red circle refers to a goal zone.

Figure 6.15 depicts the initial exploration speeds, from 0 to 10 oracle calls, of RRT and q-RRT. Each path planning problem is cut off after 10 oracle calls and a heatmap is created of the total node placement in the state space over 100 trials. The q-RRT method shows much faster initial node placements over RRT, admitting 372 nodes with an oracle efficiency of 31.2%. The q-RRT method has more than a thousand total oracle calls due to the inclusion of a finalizer line before nodes are admitted to the tree. RRT admitted 125 nodes with an oracle efficiency of 12.5%. Node placement is more dense both in the initial node pocket and along lines exploring

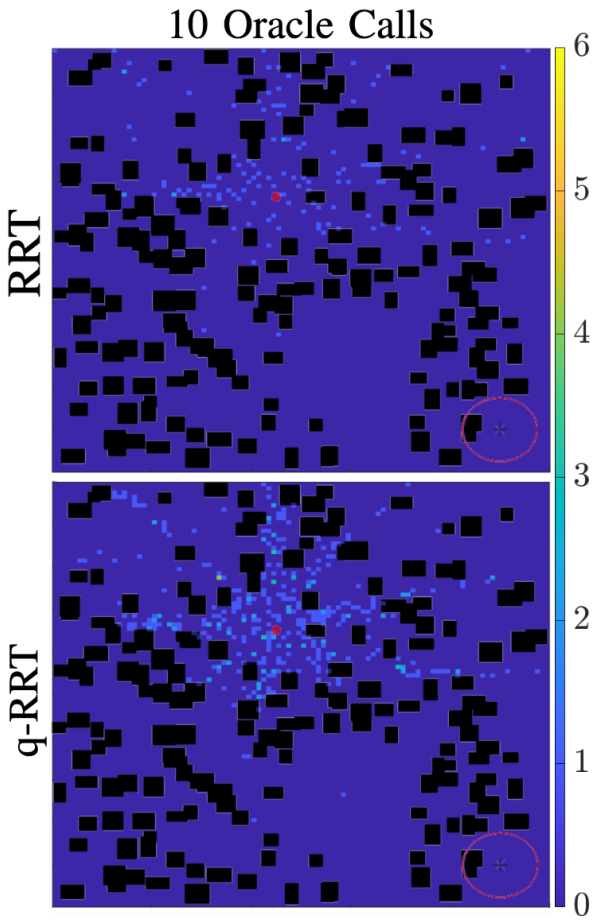

Figure 6.15. Comparison of initial exploration speeds (up to 10 oracle calls) of RRT and q-RRT. Data is shown as a state space heat-map of node placements over 100 trials of each algorithm in the shown obstacle environment. A goal zone is shown as a red ring in the bottom right, and the heatmap color key is shown on the right of the graph.

outward between obstacles away from the initial node.

Figure 6.16 depicts the middle-time exploration speed, from 0 to 20 oracle calls, of RRT and q-RRT. Similarly, each path planning problem is cut off after 20 oracle calls and a heatmap created from the total node placement of each algorithm over 100 trials. The q-RRT method shows much faster and more full middle-time node placement, admitting 650 nodes with an oracle efficiency of 31.0%. RRT admitted 231 nodes with an oracle efficiency of 11.6%. Node placement is more "full" in the initial pcket, and is much more dense along lines exploring out between obstacles from the initial node.

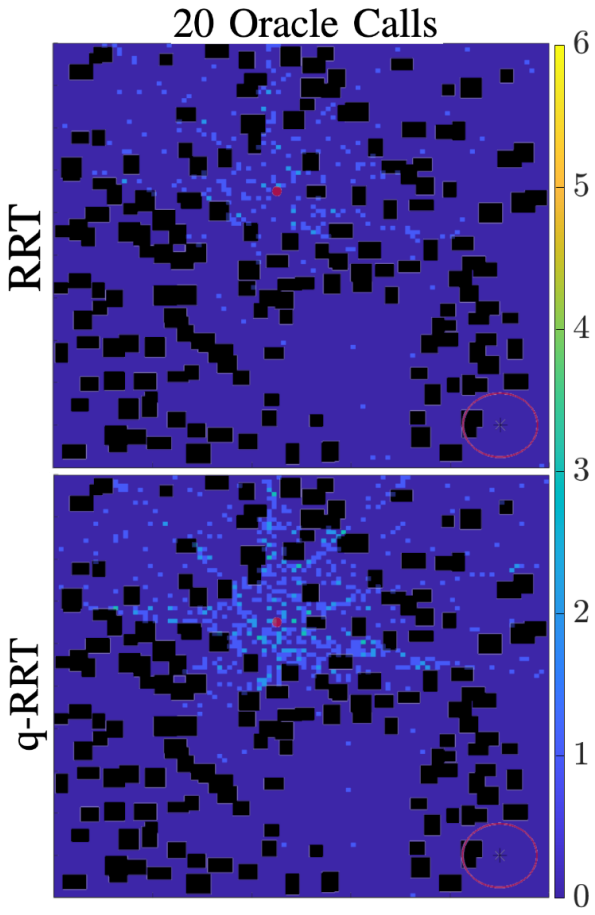

Figure 6.16. Comparison of middle-time exploration speeds (up to 20 oracle calls) of RRT and q-RRT. Data is shown as a state space heat-map of node placements over 100 trials of each algorithm in the shown obstacle environment.

Figure 6.17 depicts the "late-time" exploration speed, from 0 to 40 oracle calls, of RRT and q-RRT. Heatmap creation is again similar to previous. The q-RRT method has admitted more nodes in nearby pockets, and has a more dense spread of nodes in further away regions, admitting 1091 nodes with an oracle efficiency of 26.0%, compared to RRT, which admitted 526 nodes with an oracle efficiency of 13.2%. The average oracle efficiency of q-RRT has dropped somewhat compared to the initial and middle time exploration, and this is due to the fact that, as the existing tree grows, new random points are more likely to be reachable to the existing graph. This serves to allow RRT to catch up in terms of efficiency, and q-RRT's created database, on average, has allowed more solutions. The quantum version of the algorithms thrive

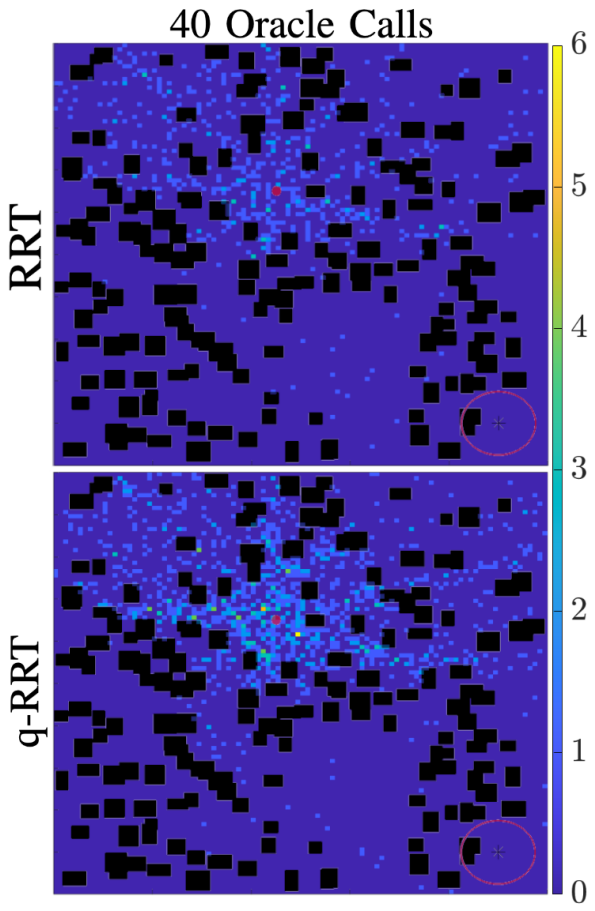

Figure 6.17. Comparison of late-time exploration speeds (up to 60 oracle calls) of RRT and q-RRT. Data is shown as a state space heat-map of node placements over 100 trials of each algorithm in the shown obstacle environment.

(in comparison) in situations where there are few solutions. Heatmaps for Pq-RRT were also created for 10, 20, and 40 oracle call cases as shown in Fig. 6.15- 6.17. The Pq-RRT heatmaps were omitted, as results were largely similar between q-RRT and Pq-RRT. This is similar to findings in Fig. 6.14 that, when compared over oracle calls (quantum time surrogate), Pq-RRT's advantage is not apparent, as Pq-RRT is able to make simultaneous oracle calls.

6.6.3 Narrow Corridor Exploration

The ability of motion planning algorithms to find paths through narrow corridor environments serves as a benchmark for the ability to find difficult solutions in narrow spaces. In

Figs. 6.18 and 6.19 we show, through a heatmap, the ability of q-RRT to find passage through a narrow corridor when compared to RRT. The figures depict a heatmap of node placements of 50 trials of each algorithm in the overlaid environment, where each method is cut off after 25 oracle calls to analyze ability to quickly place nodes in the narrow corridor. Obstacles are depicted in black, and are distributed randomly on both sides of the narrow corridor, which is created by 2 large obstacles.

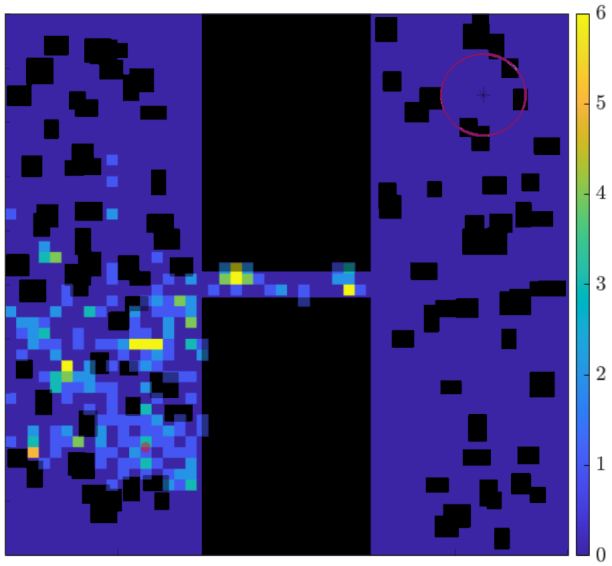

Figure 6.18. A heatmap of q-RRT's node placement in a narrow corridor environment (up to 25 oracle calls) over 50 trial runs, with 47 nodes in the channel. Obstacles are depicted in black, and a color key of node placements is shown to the right of the graph.

The q-RRT method placed 47 nodes in the narrow corridor, compared to RRT's 14 nodes. Results are presented with no guided sampling or known-goal direction to guide sampling. The q-RRT algorithm is quicker to find paths into narrow corridors toward possible goal locations.

6.6.4 Quantum Database Annealing

We compare the abilities of Quantum Database Annealing and standard q-RRT database construction to create trees that spread across larger configuration spaces with a large number (6025) of obstacles. In this formulation, Quantum Database Annealing is initially creating databases of points at a distance between 2.7 and 4.2 units from current nodes, then dropping that

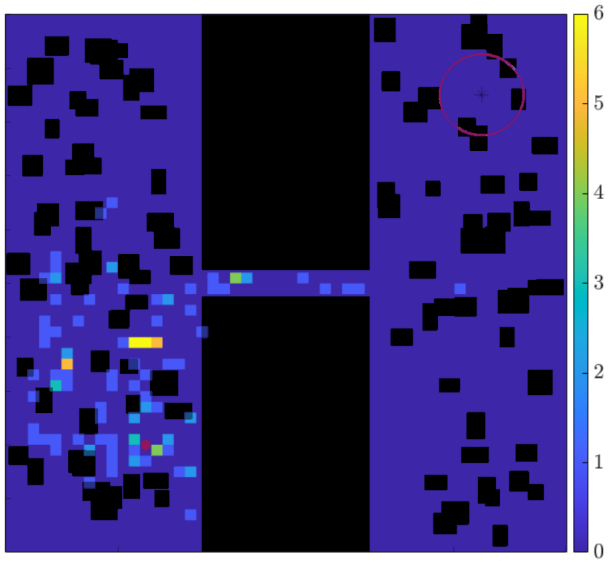

Figure 6.19. A heatmap of RRT's node placement in a narrow corridor environment (up to 25 oracle calls) over 50 trial runs, with 14 nodes in the channel.

range to between 0.8 and 2.0 units to fill in the space around the initially spread tree. On the other hand, q-RRT with standard database construction is sampling across the entire configuration space C. Both algorithms are using databases of size 2^9.

Fig. 6.20 depicts a 16-node tree with initial fast expansion made with Quantum Database Annealing, and Fig. 6.21 shows continued node addition to a 48-node tree with lower temperature to fill in the area around the initial spread tree. Fig. 6.22 depicts the standard q-RRT created 16-node tree in the same environment, to compare against Fig. 6.20. Obstacles are depicted as small black rectangles, the root node of each tree is shown as a black circle, nodes in each tree are shown as red circles, and parent child connections are shown as black lines.

The resulting trees differ in how spread they are for the same number of oracle calls (the quantum analog of runtime). Quantum Database Annealing initially creates nodes an average of 3.68 units away (with the above temperature setting) from their parent and standard q-RRT creates nodes an average of 1.70 units away from their parent. For a fixed number and size of obstacles, it should be noted that sampling parameters affect the average distance in the q-RRT tree, and different average distances can be obtained by varying the size of the database.

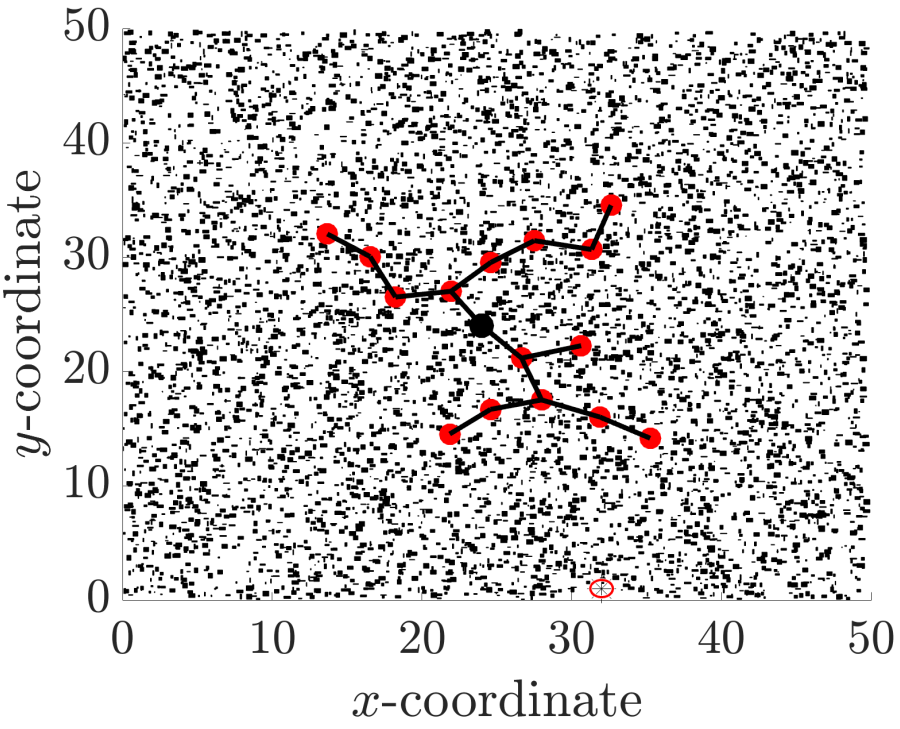

Figure 6.20. A 16 node tree created with Quantum Database Annealing with high temperature, showing fast initial exploration. The root node is a black circle and tree nodes are red circles. Parent-child relationships are shown via black lines, and obstacles depicted as small black rectangles.

For uniform sampling over C, as the database becomes larger, the average distance drops, as nodes are more frequently found near existing nodes. For equal-sized large databases, in the same amount of (quantum) time, QDA is able to create trees with more spread, as only further away nodes are admitted to the database. The temperature construct allows a balance between exploration and density of nodes, enabling a version of q-RRT that can connect distant regions of a configuration space very quickly before back-filling with lower temperature.

6.7 Conclusion

To generalize and extend q-RRT, we provide analysis in more general obstacle environments, a formulation of q-RRT with parallel quantum computers, and a database building strategy based on simulated annealing. The Parallel Quantum RRT algorithm uses parallel

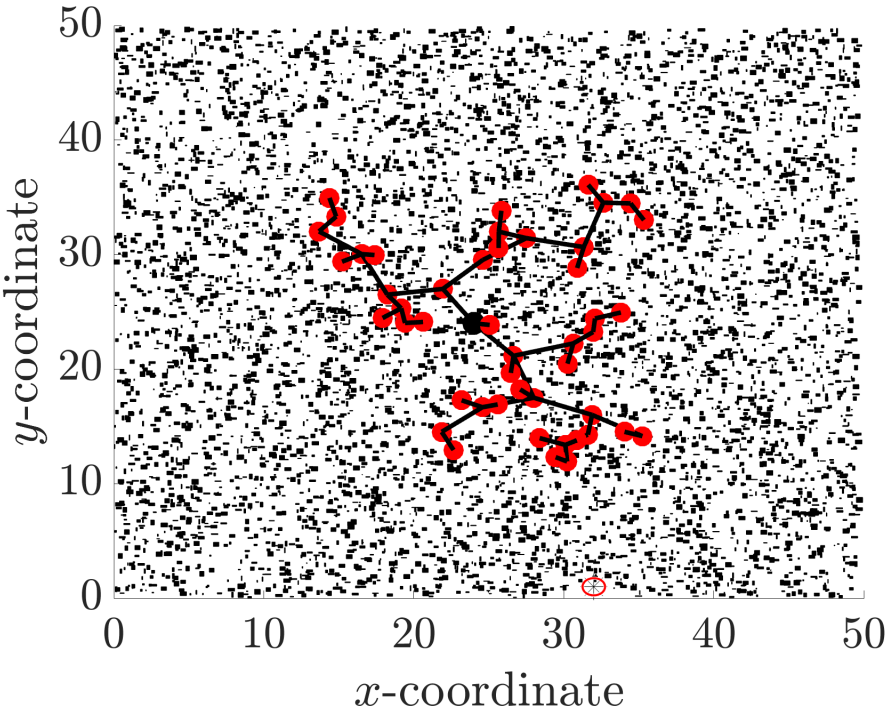

Figure 6.21. A 48 node tree created with Quantum Database Annealing with initial high temperature, then a dropping temperature, showing how temperature can be used to fill in the area around a spread tree. The root node is a black circle and tree nodes are red circles. Parent-child relationships are shown via black lines, and obstacles depicted as small black rectangles.

quantum computers in a manager-worker formulation to provide simultaneous measurements of a shared database, allowing more time-efficient tree construction with a higher exploration speed. We also provide key probability results for parallel quantum computers searching the same database. Quantum Database Annealing uses a temperature construct to guide database construction, providing trees that spread more quickly compared to those created with standard database construction. To support these claims, we provide analysis in the form of efficiency and run-time results, heatmaps for speed-of-exploration results, thin channel environment results, and database construction comparisons. Future work includes expanding on alternate methods of database construction and creating path planning algorithms that rely on alternate quantum algorithms to QAA.

Chapter 6, in full, has been submitted for publication as “Parallel Quantum Rapidly-

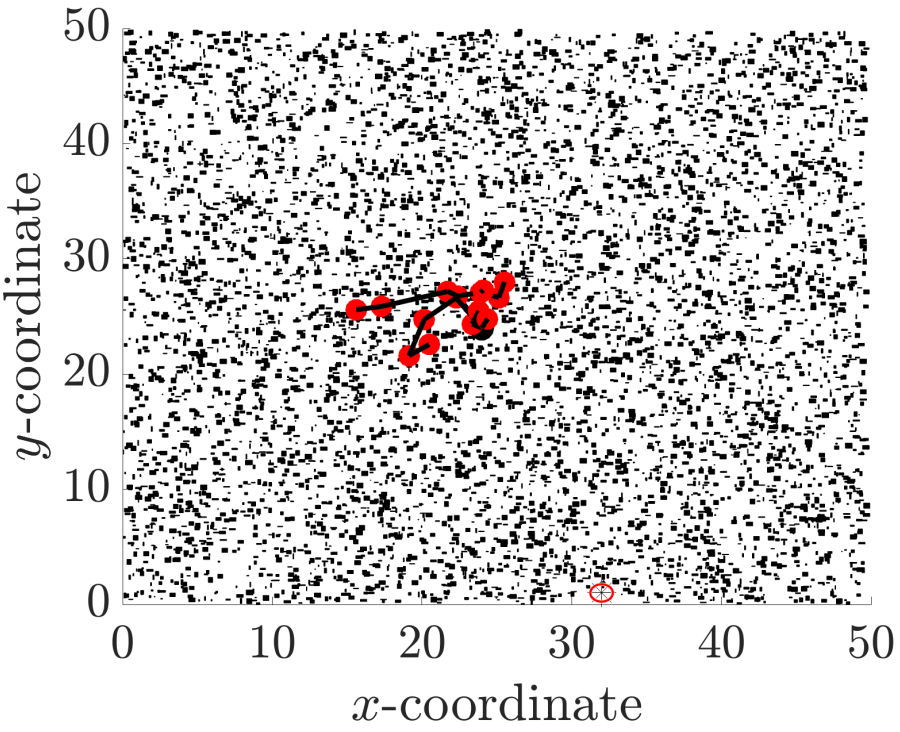

Figure 6.22. A 16 node tree created with q-RRT (with standard database construction) in the same 6025 obstacles environment.

Exploring Random Trees" as it may appear in IEEE Access. Lathrop, Paul, Beth Boardman, and Sonia Martínez. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Conclusion

This dissertation covers several areas within the field of sampling-based motion planning for autonomous robotic vehicles. The area of motion planning shows general active interest in the topic of safety with respect to environmental conditions and actors. A particularly potent method for addressing safety involves the use of uncertainty representations to characterize robots and environmental actors, alongside more abstract concepts such as behaviors and attitudes. The acknowledgement and use of uncertainty can address real-world conditions such as environmental disturbances (wind, unstable surfaces), poorly modeled or unknown dynamics, faulty or poor sensor readings, the inability to exactly localize oneself, and unknown changing future conditions.

In Chapter 3, in addition to using robotic and obstacle uncertainty representations, we create the algorithm W-Safe RRT that employs the Wasserstein metric to bound distances between distributions. This effectively acknowledges *uncertainty in uncertainty representations*, addressing the question 'what if robot and obstacle distributional shapes are poorly chosen or maintained'. To test this idea, we show simulated results in multi-obstacle situations where the selected distributional representation does not well represent the actual obstacle shape. W-Safe RRT shows robustness to poorly modeled distributions when compared to a state-of-the-art method inspired by Particle Chance Constrained RRT, but this performance increase comes with a time penalty. A future research direction is to extend W-Safe RRT in situations with continuous time dynamics and planning in more crowded environments. Additional emphasis

can also be placed on uncertainty-informed methods to adjust safety parameter values, which have a tendency to simply be chosen by a supervisor yet have the ability to drastically affect safety performance.

In situations with multiple classes of obstacles, where some may be adversarial in nature, some passive, and some static or drifting, the ability to quickly discern between obstacle classes is the key to efficient safe path planning. Environmental actor classification considering more abstract behavioral metrics can dually allow safer planning around adversarial actors and more efficient planning around harmless actors, all while ensuring an algorithm remains human understandable and therefore human debuggable. However, higher level notions of behavior introduce additional noise to the behavioral space, especially when trajectory information is noisy itself. Uncertainty representations in classification tasks in more abstract robotic spaces is addressed in Chapter 4, where we introduce an integral classification method in a space we dub the behavioral *feature space*. We find that the method allows for noisy data to be labeled correctly more often when compared to a naive Bayesian classifier and an ensemble classification-by-vote method. Future research directions include thoughtful definitions of abstract behavioral metrics and additional feature space analytical methods. Concepts such as feature space flow maps, history and future predictability analysis, and data-driven probabilistic behavioral guarantees can assist when actors exhibit *behavioral* dynamics, or switches between behaviors.

For motion planners, when the local planning procedure of connecting a new state to an existing state is made computationally expensive, such as planning in high dimensions, or when dynamics, safety, and optimality are desired, once-online planners can lose the functional ability to practically compute motion plans on-the-fly. This complication often goes unaddressed, especially in planners with strict performance or safety guarantees, and in cases where poorly-scaled optimization problems are presented with minimal evidence of practical use. To this point, we turn our attention to quantum computers, which use quantum parallelism and superpositions of states to efficiently perform simultaneous calculations. While practical and widespread quantum computers are still in active development, the mechanics and methodology of many

quantum algorithms are well known.

In Chapter 5, we present the first formulation of sampling-based motion planning built to be solved with quantum algorithms, on quantum devices. The algorithm q-RRT uses a database-search formulation alongside Quantum Amplitude Amplification to amplify the probability of measuring possible next states from the database. We address the effects of measurement error, present due to the probability-measurement process, and oracle errors, due to incorrectly performing connectivity or reachability tests on new states. The q-RRT algorithm is shown, in quantum computing simulation, to successfully employ a quadratic runtime improvement over classical RRT in building trees in dense random lattices.

Operating within the quantum realm offers tantalizing rewards in return for being at the mercy of the laws of quantum physics. One such law, dubbed quantum measurement collapse, ensures that probability information on a quantum state is lost when a single database element is desired and measurement performed. To this end, we establish two additional ideas that mimic the development of classical sampling-based motion planning: parallel structure and guided sampling. In Chapter 6, we introduce Pq-RRT: a parallel-quantum-computing extension to q-RRT. Using a classical-manager quantum-worker approach, we define a shared database architecture that allows additional efficiency increase over q-RRT in extracting simultaneous solutions. We provide symbolic runtime analysis of shared and unshared database formulations for the manager-worker architecture. The nature of the shared database leads us to derive probability characterizations of quantum workers searching the database, and finally the database construction strategy Quantum Database Annealing. With QDA, a temperature variable (inspired by simulated annealing) is used alongside a minimum sampling distance restriction to guide the database construction process in a way that is shown in simulation to effectively control the exploration vs exploitation tradeoff. We provide extensive simulations of Pq-RRT and q-RRT compared to Parallel RRT and RRT in oracle call efficiency, node placement efficiency, real runtime, thin channel environments, and heatmap exploration speed trials.

Future work related to q-RRT includes creating a sampling-based motion planning

structure to utilize alternative quantum algorithms, and comparing performance with q-RRT employing Quantum Amplitude Amplification. An additional area is to explore path-optimality based algorithms, and formal safety-guaranteeing planning methods, such as motion-planning-related safe reinforcement learning, in the context of quantum computing. The link between alternative quantum algorithms and motion planning and control is tenuously explored at best, and algorithms such as variational quantum eigensolvers with applications to quantum machine learning and quantum optimization has promise in solving existing issues in motion planning. Early applications of quantum computing to particle-level physics and chemistry simulations may also have some links to ideas in motion planning such as particle filters, Monte Carlo methods, and probabilistic uncertainty.

Future work also includes testing the presented methods on physical quantum devices, even in a limited way. An additional further research direction is a deeper exploration into database construction strategies to optimize various performance objectives and contexts, such as different construction strategies for different parallel architectures. From a broader perspective, adding in quantum computers as an additional tool in algorithmic architecture may have long reaching consequences across many areas of motion planning, including hierarchical planning, local planning, safe reinforcement learning, optimization, and neural networks. Combinations of these approaches have long been used to address the complex problem of motion planning. Efficiently exploiting quantum computation, balancing the use of classical computers, addressing quantum computing's downsides, and performing algorithmic architectural analysis with quantum components may prove to be a bountiful line of future research.

The adaptation of successful current methods and algorithms (such as RRT) to be assisted by possible tools of the future is of future interest to humanity, but the exercise also provides more practical insights to current problems and approaches. Exploration of out-of-the-box (and out-of-the-field) solutions to key issues (such as scaling and runtime) in provably safe, optimal, and/or reachable algorithms is of paramount importance for practical and real world adoption of such algorithms and general advancement of the field of autonomous vehicles. Quantum

computing represents one possible line of approach, and countless others lie dormant, awaiting unconventional application to the problems of motion planning.

Bibliography

[1] R. Orus, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, no. 100028, 2019.

[2] C. Alexopoulos and P. Griffin, "Path planning for a mobile robot," *IEEE Transactions on Systems, Man, & Cybernetics*, vol. 22, no. 2, pp. 318–322, 1992.

[3] O. Takahashi and R. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 1989.

[4] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math. 1*, pp. 269–271, 1959.

[5] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Transactions on Systems, Science, and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[6] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[8] N. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, 1996, pp. 113–120.

[9] J. Barraquand, L. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," in *Robotics Research: The Seventh International Symposium*. Springer, 1996, pp. 249–264.

[10] A. Yershova, L. Jaillet, T. Siméon, and S. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 2005, pp. 3856–3861.

[11] L. Guibas, C. Holleman, and L. Kavraki, "A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, vol. 1, 1999, pp. 254–259.

[12] V. Boor, M. Overmars, and A. V. D. Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2. IEEE, 1999, pp. 1018–1023.

[13] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2011, p. 2640–2645.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[15] ——, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *IEEE Int. Conf. on Decision and Control*, 2010, pp. 7681–7687.

[16] ——, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics: Science and Systems*, Zaragoza, Spain, 2010.

[17] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[18] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2006, pp. 5369–5375.

[19] Z. Kingston, M. Moll, and L. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 159–185, 2018.

[20] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[21] G. Aoude, B. Luders, J. Joseph, N. Roy, and J. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, pp. 51–76, 2013.

[22] T. H. Summers, "Distributionally robust sampling-based motion planning under uncertainty," *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pp. 6518–6523, 2018.

[23] B. D. Luders, M. Kothari, and J. P. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," *AIAA Conf. on Guidance, Navigation and Control*, 2010.

[24] B. Luders, S. Karaman, and J. How, "Robust sampling-based motion planning with asymptotic optimality guarantees," in *AIAA Conf. on Guidance, Navigation and Control*, August 2013.

[25] D. Webb and J. V. D. Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 2013, pp. 5054–5061.

[26] A. Orthey, C. Chamzas, and L. Kavraki, "Sampling-based motion planning: A comparative review," *arXiv preprint arXiv:2309.13119*, 2023.

[27] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[28] J. Bialkowski, S. Karaman, and E. Frazzoli, "Massively parallelizing the RRT and the RRT*," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2011, p. 3513–3518.

[29] D. Dong, C. Chen, H. Li, and T. Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, & Cybernetics. Part B: Cybernetics*, vol. 38, no. 5, pp. 1207–1220, 2008.

[30] N. Amato and L. Dale, "Probabilistic roadmap methods are embarrassingly parallel," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1. IEEE, 1999, pp. 688–694.

[31] D. Challou, M. Gini, and V. Kumar, "Parallel search algorithms for robot motion planning," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 1993, pp. 46–51.

[32] E. Plaku, K. Bekris, B. Chen, A. Ladd, and L. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, 2005.

[33] D. Henrich, C. Wurll, and H. Wörn, "Multi-directional search with goal switching for robot path planning," in *International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Springer, 1998, pp. 75–84.

[34] S. Carpin and E. Pagello, "On parallel RRTs for multi-robot systems," in *International Conference of the Italian Association for Artificial Intelligence*, 2002, pp. 834–841.

[35] I. Aguinaga, D. Borro, and L. Matey, "Parallel RRT-based path planning for selective disassembly planning," *The International Journal of Advanced Manufacturing Technology*, vol. 36, pp. 1221–1233, 2008.

[36] D. Devalarazu and D. Watson, "Path planning for altruistically negotiating processes," in *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005.* IEEE, 2005, pp. 196–202.

[37] D. Devaurs, T. Siméon, and J. Cortés, "Parallelizing RRT on large-scale distributed-memory architectures," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 571–579, 2013.

[38] J. Preskill, "Lecture notes for physics 229: Quantum information and computation," *California Institute of Technology*, vol. 16, no. 1, pp. 1–8, 1998.

[39] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[40] C. Petschnigg, M. Brandstötter, H. Pichler, M. Hofbaur, and B. Dieber, "Quantum computation in robotic science and applications," in *IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 803–810.

[41] B. Schumacher, "Quantum coding," *Physical Review A*, vol. 51, no. 4, pp. 2738–2747, 1995.

[42] M. Born, "Quantum mechanics of collision processes," *Physics-Uspekhi*, 1926.

[43] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, 1995.

[44] M. Schlosshauer, "Decoherence, the measurement problem, and interpretations of quantum mechanics," *Reviews of Modern Physics*, vol. 76, no. 4, pp. 1267–1305, 2005.

[45] B. Schmitt, F. Mozafari, G. Meuli, H. Riener, and G. D. Micheli, "From boolean functions to quantum circuits: A scalable quantum compilation flow in C++," in *2021 Design, Automation & Test in Europe Conference*. IEEE, 2021, pp. 1044–1049.

[46] A. Ambainis, "Understanding quantum algorithms via query complexity," in *Proc. Int. Congress of Mathematicians*. World Scientific, 2018, pp. 3265–3285.

[47] R. Feynman, "Quantum mechanical computers," *Optics News*, vol. 11, no. 2, pp. 11–20, 1985.

[48] T. Toffoli, "Reversible computing," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1980, pp. 632–644.

[49] A. Fedorov, L. Steffen, M. Baur, M. da Silva, and A. Wallraff, "Implementation of a toffoli gate with superconducting circuits," *Nature*, vol. 481, no. 7380, pp. 170–172, 2012.

[50] T. Monz, K. Kim, W. Hänsel, M. Riebe, A. Villar, P. Schindler, M. Chwalla, M. Hennrich, and R. Blatt, "Realization of the quantum toffoli gate with trapped ions," *Physical Review Letters*, vol. 102, no. 4, p. 040501, 2009.

[51] M. Wilde, *Quantum information theory*. Cambridge university press, 2013.

[52] S. Aaronson, D. Grier, and L. Schaeffer, "The classification of reversible bit operations," *arXiv preprint arXiv:1504.05155*, 2015.

[53] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," *American Control Conference*, pp. 4023–4028, 2017.

[54] L. Blackmore, H. Li, and B. C. Williams, "A probabilistic approach to optimal robust path planning with obstacles," *American Control Conference*, p. 7, 2006.

[55] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[56] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 723—-730.

[57] B. D. Luders and J. P. How, "Probabilistic feasibility for non-linear systems with non-gaussian uncertainty using RRT," *AIAA Infotech@Aerospace*, p. 1589, 2011.

[58] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "On the probabilistic completeness of the sampling-based feedback motion planners in belief space," *IEEE Int. Conf. on Robotics and Automation*, 2012.

[59] P. Mohajerin Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, no. 1-2, pp. 115–166, 2018.

[60] I. Yang, "Wasserstein distributionally robust stochastic control: A data-driven approach," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3863–3870, 2020.

[61] A. Cherukuri and J. Cortés, "Cooperative data-driven distributionally robust optimization," *IEEE Transactions on Automatic Control*, vol. 65, no. 10, pp. 4400–4407, 2020.

[62] D. Boskos, J. Cortés, and S. Martínez, "High-confidence data-driven ambiguity sets for time-varying linear systems," *IEEE Transactions on Automatic Control*, 2023, to appear.

[63] A. Hota, A. Cherukuri, and J. Lygeros, "Data-driven chance constrained optimization under Wasserstein ambiguity sets," in *American Control Conference*, Philadelphia, PA, USA, 2019, pp. 1501–1506.

[64] A. Jasour, W. Han, and B. Williams, "Convex risk bounded continuous-time trajectory planning in uncertain nonconvex environments," *arXiv preprint arXiv:2106.05489*, 2021.

[65] S. Singh, Y. Chow, A. Majumdar, and M. Pavone, "A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2905–2912, 2019.

[66] A. Hakobyan and I. Yang, "Wasserstein distributionally robust motion planning and control with safety constraints using conditional value-at-risk," *IEEE Int. Conf. on Robotics and Automation*, pp. 490–496, 2020.

[67] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2000, p. 995–1001.

[68] D. Boskos, J. Cortés, and S. Martínez, "Data-driven ambiguity sets with probabilistic guarantees for dynamic processes," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 2991–3006, 2021.

[69] Z. Littlefield, D. Klimenko, H. Kurniawati, and K. Bekris, "The importance of a suitable distance function in belief-space planning," *International Journal of Robotics Research*, pp. 683–700, 2018.

[70] U. Yilmaz. (2009) The Earth Mover's Distance. MATLAB Central File Exchange https://www.mathworks.com/matlabcentral/fileexchange/22962-the-earth-movers-distance.

[71] L. G. Y. Rubner, C. Tomasi, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[72] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *New results and new trends in computer science*. Springer, 1991, pp. 359–370.

[73] L. Brunke, M. Greeff, A. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. Schoellig, "Safe learning in robotics: from learning-based control to safe reinforcement learning," *arXiv preprint ArXiv:2108.06266*, 2021.

[74] P. Lathrop, B. Boardman, and S. Martínez, "Distributionally safe path planning: Wasserstein Safe RRT," *IEEE Robotics and Automation Letters*, vol. 7, pp. 430–437, 2021.

[75] J. Freitas, A. Censi, B. Smith, L. Lillo, S. Anthony, and E. Frazzoli, "From driverless dilemmas to more practical commonsense tests for automated vehicles." *Proceedings of the National Academy of Sciences*, vol. 118, no. 11, 2021.

[76] M. Chen, S. Bansal, J. Fisac, and C. Tomlin, "Robust sequential trajectory planning under disturbances and adversarial intruder," *IEEE Transactions on Control Systems Technology*, 2018.

[77] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3-4, p. 279–292, May 1992.

[78] M. Azar, R. Munos, M. Ghavamzadaeh, and H. Kappen, "Speedy Q-learning," 2011.

[79] D. Carvalho, F. Melo, and P.Santos, "A new convergent variant of Q-learning with linear function approximation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 412–19 421, 2020.

[80] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[81] Z. Chen, S. Zhang, T. Doan, S. Maguluri, and J. Clarke, "Performance of Q-learning with linear function approximation: Stability and finite-time analysis," *arXiv preprint arXiv:1905.11425*, p. 4, 2019.

[82] P. Xu and Q. Gu, "A finite-time analysis of Q-learning with neural network function approximation," in *Int. Conf. on Machine Learning*. PMLR, 2020, pp. 10 555–10 565.

[83] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," vol. 12, 1999.

[84] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Int. Conf. on Machine Learning*. PMLR, 2016, pp. 1928–1937.

[85] L. Kästner, C. Marx, and J. Lambrecht, "Deep-reinforcement-learning-based semantic navigation of mobile robots in dynamic environments," in *IEEE Conference on Automation Sciences and Engineering*. IEEE, 2020, pp. 1110–1115.

[86] N. Deepika and V. Variyar, "Obstacle classification and detection for vision based navigation for autonomous driving," *Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2092–2097, 2017.

[87] V. Losing, B. Hammer, and H. Wersing, "Interactive online learning for obstacle classification on a mobile robot," *Int. Joint Conf. on Neural Networks*, pp. 1–8, 2015.

[88] A. Møgelmose, M. Trivedi, and T. Moeslund, "Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations." *IEEE Intelligent Vehicles Symposium*, pp. 330–335, 2015.

[89] P. Kothari, S. Kreiss, and A. Alahi, "Human trajectory forecasting in crowds: A deep learning perspective." *IEEE Int. Conf. on Intelligent Transportation Systems*, 2021.

[90] C. Wissing, T. Nattermann, K. Glander, and T. Bertram, "Trajectory prediction for safety critical maneuvers in automated highway driving," *IEEE Int. Conf. on Intelligent Transportation Systems*, vol. 21, pp. 131–136, 2018.

[91] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 47–60, 1996.

[92] A. Dieng and J. Paisley, "Reweighted expectation maximization," *arXiv preprint arXiv:1906.05850*, 2019.

[93] G. Tian, Y. Xia, Y. Zhang, and D. Feng, "Hybrid genetic and variational expectation-maximization algorithm for Gaussian-mixture-model-based brain MR image segmentation," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 3, pp. 373–380, 2011.

[94] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*. Springer, 1998, pp. 355–368.

[95] H. Duan, L. Yang, J. Fang, and H. Li, "Fast inverse-free sparse Bayesian learning via relaxed evidence lower bound maximization," *IEEE Signal Processing Letters*, vol. 24, no. 6, pp. 774–778, 2017.

[96] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–41, 2018.

[97] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 312–319.

[98] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2003, pp. I–I.

[99] Y. Chen, E. Wang, and A. Chen, "Mining user trajectories from smartphone data considering data uncertainty," in *Big Data Analytics and Knowledge Discovery: 18th International Conference*. Springer, 2016, pp. 51–67.

[100] C. Parent, S. Spaccapietra, C.Renso, G. Andrienko, V. Bogorny, M. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, and Y. Theodoridis, "Semantic trajectories modeling and analysis," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, pp. 1–32, 2013.

[101] J. Lee, J. Han, X. Li, and H. Gonzalez, "TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[102] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[103] J. Peterson, R. Battleday, T. Griffiths, and O. Russakovsky, "Human uncertainty makes classification more robust," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9617–9626.

[104] B. Ivanovic, K. Lee, P. Tokmakov, B. Wulfe, R. Mcllister, A. Gaidon, and M. Pavone, "Heterogeneous-agent trajectory forecasting incorporating class uncertainty," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*. IEEE, 2022, pp. 12 196–12 203.

[105] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision ECCV*. Springer, 2020, pp. 683–700.

[106] B. Lindqvist, S. Mansouri, A. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, 2020.

[107] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, p. 129–137, 1982.

[108] T. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*. Springer, 2000, pp. 1–15.

[109] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *Journal of Statistical Physics*, vol. 22, pp. 563–591, 1980.

[110] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[111] A. Fedorov, N. Gisin, S. Beloussov, and A. Lvovsky, "Quantum computing at the quantum advantage threshold: a down-to-business review," *arXiv preprint arXiv:2203.17181*, 2022.

[112] M. Devoret, A. Wallraff, and J. Martinis, "Superconducting qubits: A short review," *arXiv preprint cond-mat/0411174*, 2004.

[113] J. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Physical Review Letters*, vol. 74, no. 20, pp. 4091–4094, 1995.

[114] D. Loss and D. DiVincenzo, "Quantum computation with quantum dots," *Physical Review A*, vol. 57, no. 1, pp. 120–126, 1998.

[115] I. Bloch, J. Dalibard, and S. Nascimbene, "Quantum simulations with ultracold quantum gases," *Nature Physics*, vol. 8, no. 4, pp. 267–276, 2012.

[116] M. Banuls, R. Blatt, J. Catani, A. Celi, J. Cirac, M. Dalmonte, L. Fallani, K. Jansen, M. Lewenstein, S. Montangero, C. Muschik, B. Reznik, E. Rico, L. Tagliacozzo, K. Acoleyen, F. Verstraete, U. Wiese, M. Wingate, J. Zakrzewski, and P. Zoller, "Simulating lattice gauge theories within quantum technologies," *The European Physical Journal D*, vol. 74, pp. 1–42, 2020.

[117] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. Chow, and J. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[118] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. Cleland, and J. Martinis, "Computing prime factors with a Josephson phase qubit quantum processor," *Nature Physics*, vol. 8, no. 10, pp. 719–723, 2012.

[119] D. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[120] C. Chang, A. Gambhir, T. Humble, and S. Sota, "Quantum annealing for systems of polynomial equations," *Scientific Reports*, vol. 9, no. 10258, 2019.

[121] F. Neukart, G. Compostella, C. Seidel, D. V. Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Frontiers in ICT*, vol. 4, no. 29, 2017.

[122] I. Cong, S. Choi, and M. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019.

[123] L. Grover, "A fast quantum mechanical algorithm for database search," *ACM Symposium on Theory of Computing*, vol. 28, 1996.

[124] D. Dong, C. Chen, and H. Li, "Reinforcement strategy using quantum amplitude amplification for robot learning," in *Chinese Control Conference*. IEEE, 2007, pp. 571–575.

[125] D. Dong, C. Chen, J. Chu, and T. Tarn, "Robust quantum-inspired reinforcement learning for robot navigation," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 86–97, 2010.

[126] D. Wang, A. Sundaram, R. Kothari, A. Kapoor, and M. Roetteler, "Quantum algorithms for reinforcement learning with a generative model," in *Int. Conf. on Machine Learning*, 2021, pp. 10 916–10 926.

[127] K. Rajagopal, Q. Zhang, S. Balakrishnan, P. Fakhari, and J. Busemeyer, "Quantum amplitude amplification for reinforcement learning," *Springer Handbook of Reinforcement Learning and Control*, pp. 819–833, 2021.

[128] L. Chen, Z. Jiang, L. Cheng, A. Knoll, and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers in Neurorobotics*, vol. 16, 2022.

[129] L. Ming, "An adaptive quantum evolutionary algorithm and its application to path planning," 2015, pp. 2067–2071.

[130] K. Han and J. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[131] A. Gad, "Particle swarm optimization algorithm and its applications: a systematic review," *Archives of Computational Methods in Engineering*, vol. 29, no. 5, pp. 2531–2561, 2022.

[132] N. Dehaghani, F. Pereira, and A. Aguiar, "Quantum control modelling, methods, and applications," *Extensive Reviews*, vol. 2, no. 1, pp. 75–126, 2022.

[133] R. Portugal, *Quantum walks and search algorithms*. Springer, 2013, vol. 19.

[134] S. Aaronson and A. Ambainis, "Quantum search of spatial regions," 2003, pp. 200–209.

[135] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," *SIAM Journal on Computing*, vol. 40, no. 1, pp. 142–164, 2011.

[136] M. Szegedy, "Quantum speed-up of markov chain based algorithms," in *IEEE Symposium on Foundations of Computer Science*, 2004, pp. 32–41.

[137] E. Sánchez-Burillo, J. Duch, J. Gómez-Gardenes, and D. Zueco, "Quantum navigation and ranking in complex networks," *Scientific Reports*, vol. 2, no. 1, pp. 1–8, 2012.

[138] D. Hsu and Z. Sun, "Adaptively combining multiple sampling strategies for probabilistic roadmap planning," in *IEEE Conf. on Robotics, Automation and Mechatronics*, vol. 2, 2004, pp. 774–779.

[139] S. Thomas, M. Morales, X. Tang, and N. Amato, "Biasing samplers to improve motion planning performance," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1625–1630.

[140] L. Jaillet, A. Yershova, S. L. Valle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2005, pp. 2851–2856.

[141] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3307–3312.

[142] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, vol. 3, 2002, p. 2383–2388.

[143] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[144] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[145] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: An improved path planning algorithm for mobile robots," *Expert Systems with Applications*, vol. 152, no. 113425, 2020.

[146] J. Canny, B. Donald, J. Reif, and P. Xavier, "On the complexity of kinodynamic planning," Cornell University, Tech. Rep., 1988.

[147] D. Devaurs, T. Siméon, and J. Cortés, "Parallelizing RRT on distributed-memory architectures," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 2261–2266.

[148] M. Strandberg, "Augmenting RRT-planners with local trees," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2004, pp. 3258–3262.

[149] G. Brassard, P. Høyer, and A. Tapp, "Quantum counting," *Int. Colloquium on Automata, Languages, and Programming*, pp. 820–831, 1998.

[150] C. Fox, "Quantum computing functions (QCF) for matlab," 2003.

[151] B. Kaygisiz, M. Karahan, A. Erkmen, and I. Erkmen, "Robotic approaches at the crossroads of chaos, fractals and percolation theory," *Applications of Chaos and Nonlinear Dynamics in Engineering-Vol. 1*, pp. 167–199, 2011.

[152] W. Zurek, "Probabilities from entanglement, Born's rule from envariance," *Physical Review A*, vol. 71, no. 052105, 2005.

[153] K. Christensen, "Percolation theory," *Imperial College London*, vol. 1, 2002.

[154] R. M. Ziff, "Spanning probability in 2D percolation," *Physical Review Letters*, vol. 69, no. 18, pp. 2670–2673, 1992.

[155] N. Jan, "Large lattice random site percolation," *Physica A: Statistical Mechanics and its Applications*, vol. 266, no. 1-4, pp. 72–75, 1999.

[156] S. Mertens and R. M. Ziff, "Percolation in finite matching lattices," *Physical Review E*, vol. 94, no. 062152, 2016.

[157] M. Sykes and J. Essam, "Exact critical percolation probabilities for site and bond problems in two dimensions," *Journal of Mathematical Physics*, vol. 5, no. 8, pp. 1117–1127, 1964.

[158] N. Nagelkerke, "A note on a general definition of the coefficient of determination," *Biometrika*, vol. 78, no. 3, pp. 691–692, 1991.

[159] H. Trentelman, A. Stoorvogel, and M. Hautus, *Control theory for linear systems*. Springer Science & Business Media, 2012.

[160] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[161] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[162] E. Asarin, T. Dang, G. Frehse, A. Girard, C. L. Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pp. 1582–1587.

[163] P. Shyamsundar, "Non-boolean quantum amplitude amplification and quantum mean estimation," *arXiv preprint arXiv:2102.04975*, 2021.

[164] T. Lai, F. Ramos, and G. Francis, "Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees," *IEEE Int. Conf. on Robotics and Automation*, pp. 5537–5543, 2019.

[165] F. Tacchino, A. Chiesa, S. Carretta, and D. Gerace, "Quantum computers as universal quantum simulators: state-of-the-art and perspectives," *Advanced Quantum Technologies*, vol. 3, no. 3, p. 1900052, 2020.

[166] S. Pirandola, U. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. Pereira, M. Razavi, J. S. Shaari, M. Tomamichel, V. Usenko, G. Vallone, P. Villoresi, and P. Wallden, "Advances in quantum cryptography," *Advances in Optics and Photonics*, vol. 12, no. 4, pp. 1012–1236, 2020.

[167] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[168] P. Lathrop, B. Boardman, and S. Martínez, "Quantum search approaches to sampling-based motion planning," *IEEE Access*, vol. 11, pp. 89 506–89 519, 2023.

[169] M. Cerezo, A. Arrasmith, R. Babbush, S. Benjamin, S. Endo, K. Fujii, J. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[170] D. Abrams and C. Williams, "Fast quantum algorithms for numerical integrals and stochastic processes," *arXiv preprint quant-ph/9908083*, 1999.

[171] A. Montanaro, "Quantum speedup of monte carlo methods," *Royal Society of London. Proceedings Series A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2181, p. 20150301, 2015.

[172] J. Ichnowski and R. Alterovitz, "Parallel sampling-based motion planning with superlinear speedup." in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2012, pp. 1206–1212.

[173] R. Parekh, A. Ricciardi, A. Darwish, and S. DiAdamo, "Quantum algorithms and simulation for parallel and distributed quantum computing," in *IEEE/ACM Int. Workshop on Quantum Computing Software*, 2021, pp. 9–19.

[174] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, p. 671–680, May 1983.

[175] R. Eglese, "Simulated annealing: a tool for operational research," *European Journal of Operational Research*, vol. 46, no. 3, pp. 271–281, 1990.

[176] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated annealing: theory and applications*. D. Reidel Publishing Company, 1987.

[177] A. Vázquez-Otero, J. Faigl, and A. Munuzuri, "Path planning based on reaction-diffusion process," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2012, pp. 896–901.

[178] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. Bagnell, and S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[179] S. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[180] L. Jaillet, J. Cortés, and T. Siméon, "Transition-based RRT for path planning in continuous cost spaces," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2008, pp. 2145–2150.

[181] ——, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.

[182] P. Neal, "The generalised coupon collector problem," *Journal of Applied Probability*, vol. 45, no. 3, pp. 621–629, 2008.

[183] P. Berenbrink and T. Sauerwald, "The weighted coupon collector's problem and applications," in *Int. Conf. of Computing and Combinatorics*. Springer, 2009, pp. 449–458.