RESEARCH ARTICLE

Quantum Search Approaches to Sampling-Based Motion Planning

PAUL LATHROP[1,2], **(Student Member, IEEE), BETH BOARDMAN**[2],
AND SONIA MARTÍNEZ[1], **(Fellow, IEEE)**
[1]Department of Mechanical and Aerospace Engineering, University of California at San Diego, La Jolla, CA 92092, USA
[2]Los Alamos National Laboratory, Los Alamos, NM 87545, USA
Corresponding author: Paul Lathrop (pdlathrop@gmail.com)

ABSTRACT In this paper, 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 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.

INDEX TERMS Sampling based motion planning, quantum computing, probability and statistical methods.

I. INTRODUCTION

The emergence of digital electronic computing in the 1940s and 1950s brought widespread changes to virtually every area of human life. More recently, in 1980, Paul Benioff presented the quantum Turing machine [1], which outlined a simple computer using the principles of quantum mechanics to represent mixed states. The concept of quantum gates [2], 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

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

and awaits the development of reliable physical mediums to be used in practice [3]. Candidates for physical implementation of quantum bits (qubits) include superconducting circuits [4] (with information storage in harmonic oscillations between energy levels of an inductor-capacitor circuit), the trapped ion quantum computer [5] (with information storage in stable electronic ion states), and the semiconductor quantum dot quantum computer [6] (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 [7], [8] and chemistry [9] simulations, cryptography [10], [11], optimization [12], [13], and machine learning [14].

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 [15], [16]. We believe this property allows quantum algorithms to parallelize computationally heavy steps in motion planning. 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.

A. 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 [17], [18], [19], and [20]. Quantum methods have been shown to increase speed [17] and robustness [18] of state-action pair learning algorithms in gridded environments when compared to temporal difference epsilon-greedy and softmax choice strategies. Quantum reinforcement learning [19] 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 [21]. 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 [22]. 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 [23].

Simple robotic trajectory planning is addressed in the work [24], which uses the Quantum Evolutionary Algorithm [25], 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 [26].

Quantum methods have been applied to several other motion-planning-adjacent areas within robotics. The work [27] 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, [27] 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 [27], it aligns with the general proposed research agenda. The review [28] 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 [29]. In particular, Grover's Quantum Search Algorithm has been used in [30] 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 [31]. 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 [32] 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 [33]. Similarly, quantum walks have been applied to search over more abstract spaces; see [34] 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 [23]. We provide an introductory set of references, and readers are encouraged to consult the textbook [23] for further reading. In sampling based motion planning, the most commonly used algorithms are Probabilistic Roadmaps (PRM) [35] and the Rapidly-exploring Random Trees (RRT) [36], both of which provide samples to grow

graphs and trees respectively. These algorithms have been extended and modified in their sampling strategies [37], [38], exploration [39], [40], collision checking [41], [42], speed and optimality [43], [44], [45], and kinodynamic constraint satisfaction [46], [47], 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 [48]. In this work, 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 [3], [49]. Motion planning algorithms have been rewritten for multi-threading [50], parallel tree creation [51], and parallel computation with GPUs [52]. In [50], 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 [52] 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.

B. CONTRIBUTIONS

In this work, 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.

C. NOTATION

The general notation used throughout this work is as follows. Let $d \in \mathbb{N}$, we denote by $\mathbb{R}^d$ the d-dimensional real vector space, and by $x \in \mathbb{R}^d$ a vector in it. We denote the Euclidean norm in $\mathbb{R}^d$ as $\| \cdot \|_2$. 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, and let $\mathbb{C}$ be the space of complex numbers.

II. QUANTUM COMPUTING BASICS

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). An extended introduction can be found at [16] and [49]. 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* [53]. 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 [54]).

Qubits are placed in superpositions using the Walsh-Hadamard transform, a multidimensional Fourier operator which forms the quantum Hadamard gate [55]. This is a

unitary operator mapping a quantum state to an equal super-position 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 [19]. Although the probability amplitudes α and β of the system cannot be known explicitly [56], quantum algorithms use quantum parallelism to manipulate the amplitudes. Planning algorithms written for quantum methods can be thought of as fully parallelized. In this paper, 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.

This succinct description on how to apply QAA to a specific problem is inspired by the work [57], 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 [58]. They are widely used in quantum algorithms to study complexity and runtime comparisons [59]. 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 local planning is included in Section IV. 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 [16], 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 [16].

III. 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. 1 (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 $x_G \in C_{\text{free}}$. The path is denoted as an ordered set of states $\gamma : x_0, x_1, \ldots, x_G$. For the path to be considered safe, $x_i \in C_{\text{free}}$, $\forall i$. Continuous path curves can also be considered.

Algorithm 1 Quantum Full Path Search (q-FPS)

Input: x_0, x_G, n, oracle function $\mathcal{X}$
Output: $\gamma : x_0, x_1, \ldots, x_G$
1: Init Database D
2: **for** $i = 1$ to 2^n **do**
3:    $D(i) \leftarrow$ 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:    $|\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
11: **end for**
12: $\gamma \leftarrow F(\text{measure}(|\Psi\rangle))$
13: Return γ

Algorithm 1, 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 1, on line 5, QCA refers to the Quantum Counting Algorithm [60], 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 IV. 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,

$$i_{\max} = \left\lfloor \frac{\pi}{4} \sqrt{2^n/m} \right\rfloor ; \qquad (1)$$

see [61]. If Q is further applied, the amplitudes of correct solutions will start to decrease, as shown in Fig. 1. 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.

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. 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. (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. 1. 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 $(N/m)/2 = 102.4$.

IV. QUANTUM RRT

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) [62] are devised to produce successful collision free-paths more quickly in cluttered environments. In this section, we outline the q-RRT Algorithm (Alg. 2), 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:

- 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.
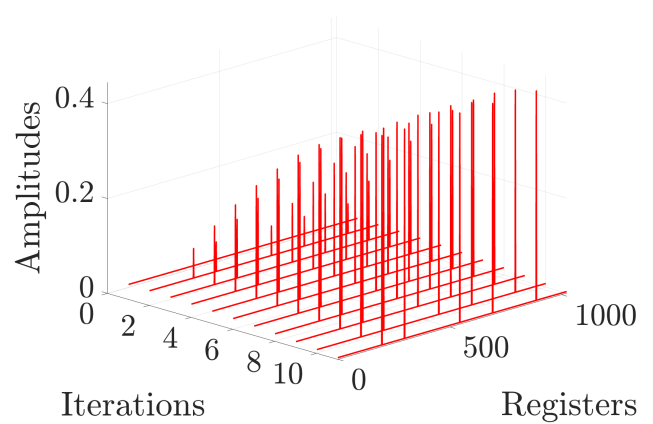

FIGURE 1. Effect of repeated applications of operator Q on probability amplitudes of a 2^{10} qubit representing a database with 5 free paths. Each register corresponds with a database element. The amplitudes of non-collision-free paths is shown as a small (non-zero) magnitude line that decreases with increased iterations. Further applications of Q decrease amplitudes of free paths.

- 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 [63].

The lattice environment is shown in Fig. 3 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 ($d > 3$) 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 $\gamma \subseteq Z$.

A. QUANTUM RRT ALGORITHM
The q-RRT algorithm, Alg. 2, 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. (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 IV-D, and a more general discussion on reachability estimations can be found in Sec. V. 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.

Algorithm 2 Quantum RRT (q-RRT)

Input: x_0, n, M, oracle $\mathcal{X}$, r, L
Output: Tree T
1: Init T with root at x_0
2: **while** size$(T) < M$ **do**
3:   **for** $i = 1$ to 2^n **do**
4:     $t =$ random point
5:     $P =$ closest parent of t in T
6:     $D(i) = [t; \ P]$
7:   **end for**
8:   Enumerate D via $F : \{0, 1\}^n \rightarrow D$
9:   Init n qubit register $|z\rangle \leftarrow |0\rangle^{\otimes n}$
10:   $|\Psi\rangle \leftarrow \mathbf{W} |z\rangle$
11:   $p_1^* = p^*(r, L), \quad p_2^* = p^*\left(r, L/\sqrt{\text{size}(T)}\right)$ from Eq. (14)
12:   **for** $i = 1$ to $\left\lfloor \frac{\pi}{4}\sqrt{1/p_1^*} \right\rfloor$ **do**
13:     $|\Psi\rangle \leftarrow Q(\mathcal{X})|\Psi\rangle$
14:   **end for**
15:   $[x_{\text{last}}, P] \leftarrow F(\text{measure}(|\Psi\rangle))$
16:   Add $[x_{\text{last}}, P]$ to T
17: **end while**
18: Return T

B. 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 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. 1. 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 [62]. The output of RRT, T_{RRT}, satisfies $\mathbb{P}(x_2 \in T_{\text{RRT}}) \rightarrow 1$ as the number of samples $\rightarrow \infty$. 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 $\mathbb{P}(x_2 \in T_{\text{RRT-m}}) \rightarrow 1$. 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. 2 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. 2. 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. 2 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. 2 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. 2 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) [64]. 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 1: 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,

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

where 2^n is the current database size and m is the current number of solutions within the database. Eq. (2) is the minimum value that is achieved when Q is applied exactly according to Eq. (6).[1] As the number of nodes $M \to \infty$, $\mathbb{P}(E)$ monotonically increases to $\lim_{M\to\infty} \mathbb{P}(E) \equiv \mathbb{P}(E_{\lim})$,

$$\mathbb{P}(E_{\lim}) = 1 - \sin^2\left(\left(\frac{\pi}{2}\sqrt{1/r} + 1\right)\arcsin\left(\sqrt{r}\right)\right), \quad (3)$$

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,

$$\mathbb{P}(F) \le 1 - (1 - \mathbb{P}(E_{\lim}))^M, \quad (4)$$

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

$$\mathbb{P}(F_\gamma) \le 1 - (1 - \mathbb{P}(E_{\lim}))^{|\gamma|}, \quad (5)$$

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

We remark that there is no way of finding lower bounds similar to Eq. (4) and Eq. (5), as the expected lower bound

---

[1]Functionally, Eq. (2) will be modified by the fact that Q is applied an integer number of times.

value of Eq. (2) depends on the local planner. In this case, Eq. (4) and Eq. (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,

$$i_{\max} = \frac{\pi}{4}\sqrt{\frac{2^n}{m}}, \quad (6)$$

as given in [16]. We further note that, after $i_{\max}$ iterations, the probability of measuring a good state is,

$$\mathbb{P}(E^c) = \sin^2((2i_{\max} + 1)\theta), \quad (7)$$

where θ is defined such that $\sin^2(\theta) = \frac{m}{2^n}$ [16], and where $\frac{m}{2^n}$ is the success probability of the database. Thm. 1 Eq. (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. 1 Eq. (3).

Lastly, we observe that $\mathbb{P}(E)$ is upper bounded by Eq. (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. 1 Eq. (4) follows by substituting Eq. (3) into the probability formula of at least one $\mathbb{P}(E)$ occurring over M events. Eq. (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). □

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 IV-D.

The above quantum measurement error analysis is modified in Prop. 2 to additionally account for false positive and false negative errors by the oracle.

Proposition 2: 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,

$$\mathbb{P}(G) = (-1 + q + v)\,\mathbb{P}(E) + 1 - q, \quad (8)$$

where $\mathbb{P}(E)$ is given by Eq. (2). As the number of nodes $M \to \infty$, the probability of event G, denoted as $\lim_{M\to\infty} \mathbb{P}(G) \equiv \mathbb{P}(G_{\lim})$, is given by,

$$\mathbb{P}(G_{\lim}) = (-1 + q + v)\,\mathbb{P}(E_{\lim}) + 1 - q, \quad (9)$$
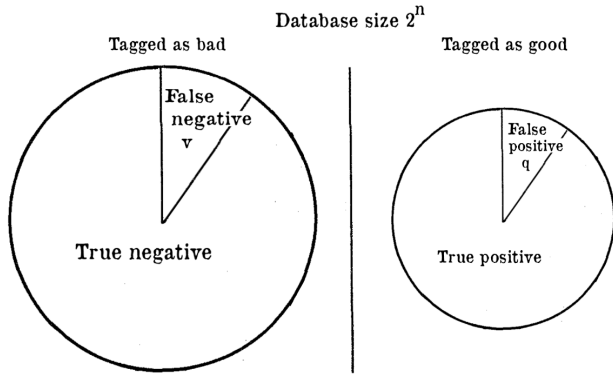
FIGURE 2. A visual depiction of the false positive and false negative regions of the good and bad tags by an oracle.

where $\mathbb{P}(E_{\text{lim}})$ is given by Eq. (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,

$$\mathbb{P}(F^*) \leq 1 - (\mathbb{P}(G_{\text{lim}}))^M, \qquad (10)$$

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

$$\mathbb{P}(F^*_\gamma) \leq 1 - (\mathbb{P}(G_{\text{lim}}))^{|\gamma|}, \qquad (11)$$

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

Proof: The proof stems from modifications made to the statement of Eq. (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. (7)) must be added to $\mathbb{P}(E)$ from Eq. (2), and $\mathbb{P}(E)v$ (fraction of false negative error) must be subtracted from $\mathbb{P}(E)$, as shown in Fig. 2. This yields,

$$\mathbb{P}(\bar{G}) = \mathbb{P}(E) + \mathbb{P}(E^c)q - \mathbb{P}(E)v, \qquad (12)$$

where $\bar{G}$ denotes the complement of event G. Eq. (12) can be simplified, and the complement taken, to give Eq. (8). Eq. (9) is found by taking Eq. (8) and substituting $\mathbb{P}(G)$ and $\mathbb{P}(E)$ with $\mathbb{P}(G_{\text{lim}})$ and $\mathbb{P}(E_{\text{lim}})$, respectively. Eq. (10) and Eq. (11) are found with the same process as Eq. (4) and Eq. (5) with the complement of event G. □

A tree with as many good states as possible is achieved with the lower bounds of error in Thm. 1. Attaining this bound requires applying QAA an optimal number of times, which is what we estimate next.

C. 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
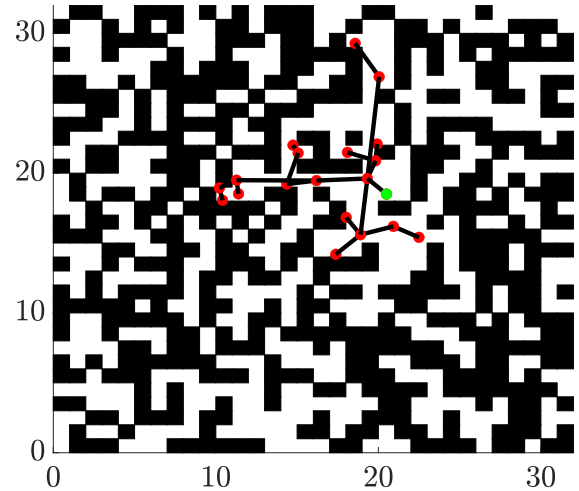

FIGURE 3. A sample random square lattice with $L = 32$ and $r = 0.5$ spanned by a 20 node tree with x_0 in green.

estimation will guide the algorithm in applying QAA the optimal number of times (Eq. (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 IV-D, 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,

$$p^* = \mathbb{E}_{\pi(x_1, x_2)}(p(x_1, x_2)), \qquad (13)$$

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 [65], [66], [67]. The work [68] uses results from [69] 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 [68], 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 [68].

We calculate the connection probability, Eq. (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^*,

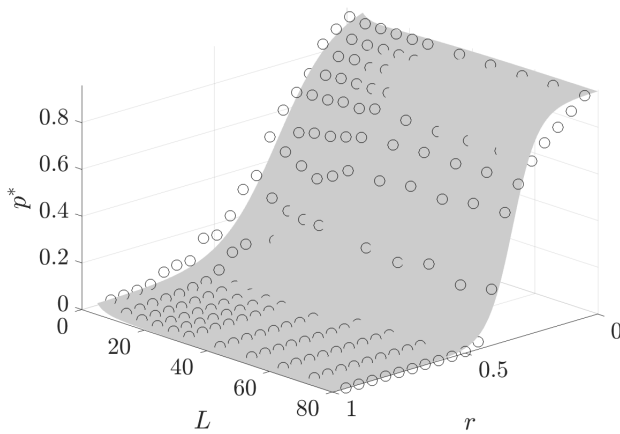$$p^*(r, L) = \frac{f}{1 + e^{-a(L-b)(r-c)}} + dL^{-2}, \qquad (14)$$

FIGURE 4. Numerically generated data points (○) estimating p^* (free-random point connectivity) as a function of concentration r and length L. Eq. (14) is depicted as the gray surface, with a coefficient of determination $R^2 = 0.9957$.

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 [70]. The model was chosen as a logistic function due to observations on matching function data in [68, 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 with a quadratic.

Each point ○ in Fig. 4 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 V-A for an evaluation of this metric.

D. LOCAL PLANNERS AND UPPER BOUND LIMIT
The choice of local planner affects the accuracy and, therefore, the relevance of Section IV-C. 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. 4 can be tweaked to yield a second estimate. We also note that considering dynamic models in the estimation in Eq. (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 [72], 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. (6). An upper bound p_2^* on the average reachability to a set of nodes in random square lattices is defined in Thm. 3.

Theorem 3: 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. (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. (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) [73] 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 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. (14). $\square$

This is similar to the noted result in [72] 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. 2, p_2^* serves to lower bound the number of times

---

[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 [71].

QAA must be applied to the database qubit. The intuition behind Thm. 3 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.

V. Q-RRT RESULTS AND DISCUSSION

A. 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 [60], 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},

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

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

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

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

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

In Fig. 5, 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 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 $M = 5$.

B. 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 IV), as shown in Fig. 3. 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
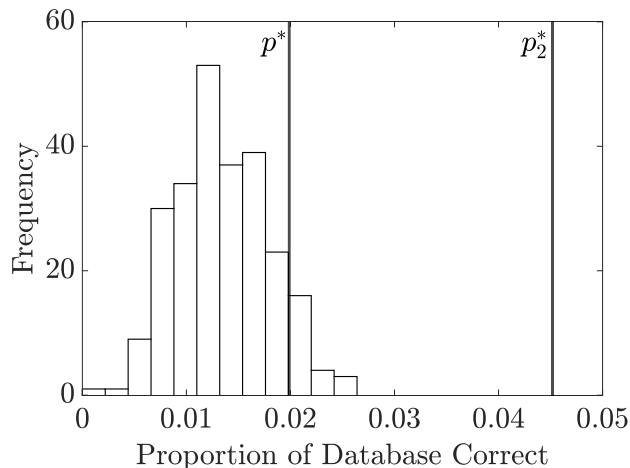

FIGURE 5. Comparison of p^* and p_2^* with a histogram of 250 random cases of database size 2^{11} with $L = 32$, $r = 0.6$, and a tree of $M = 5$ nodes.
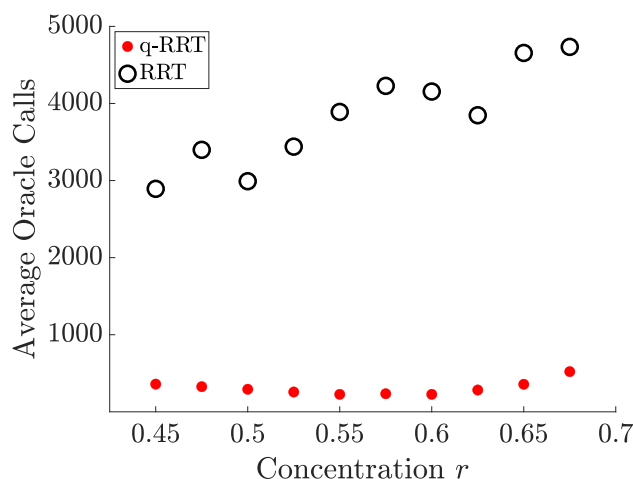

FIGURE 6. Comparison of the average number of oracle calls by q-RRT and RRT as concentration varies, for $L = 72$.

algorithms are simulated using the Matlab Quantum Computing Functions library [61]. 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. 6 and Fig. 7 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.

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
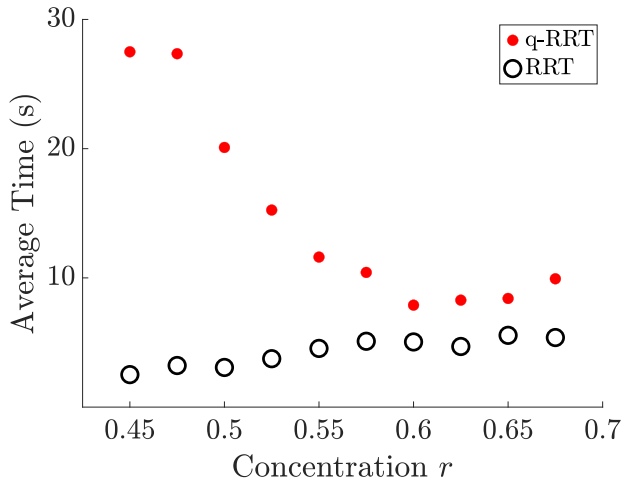
FIGURE 7. Comparison of the average real run-time of q-RRT and RRT as concentration varies, for *L* = 72.

FIGURE 8. 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.

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 [74]. 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).

C. 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. 8 and Fig. 9 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.
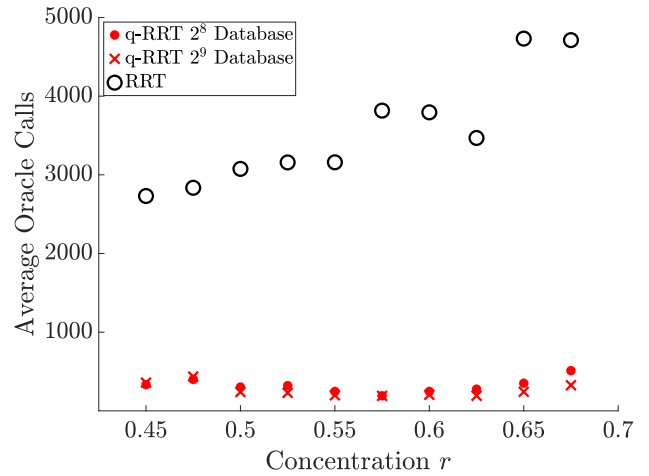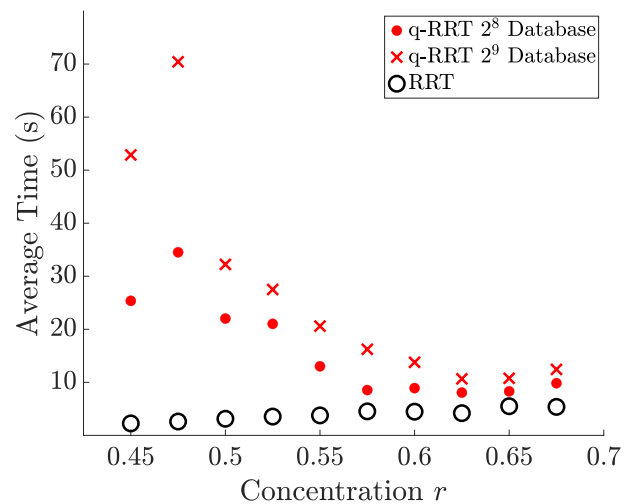

FIGURE 9. Comparison of the average run-time of q-RRT and RRT as r varies, for $L = 72$ and Database sizes 2^8 and 2^9.

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.

D. ORACLE CALL CONSTRAINT
In this section, we identify an approach for tree construction that limits the optimal number of oracle calls to a maximum
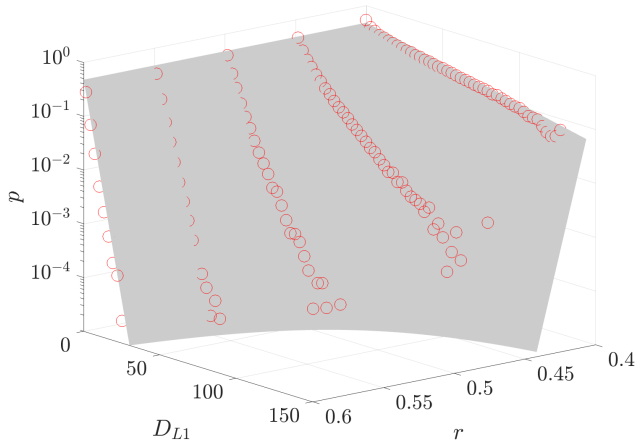
FIGURE 10. Semilog plot of numerically generated data points (○) estimating *p*, the likelihood of free-random point connectivity as a function of D_{L_1}, the L_1 distance between parent and child, for various concentrations *r*. Eq. (15) is depicted as the gray surface. Data is generated with $L = 72$.

FIGURE 11. Evaluation of the ability to select *p* given $N_{\mathcal{X}}$ using Eq. (15). Data is generated with $L = 72$, $r = 0.5$, and the database size 2^{14}.

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) distance to evaluate reachability to be equal to a certain value, which we call D_{L1}. 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 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. 10, 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. 10 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 nonlinear least squares to the numerical $L = 72$ data shown in Fig. 10, which takes the form,

$$p = a\, e^{(br+c)\, D_{L_1}}, \qquad (15)$$

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,

$$D_{L_1} = \ln\left(\frac{\pi^2}{16 N_{\mathcal{X}}^2 a}\right)/(br + c), \qquad (16)$$
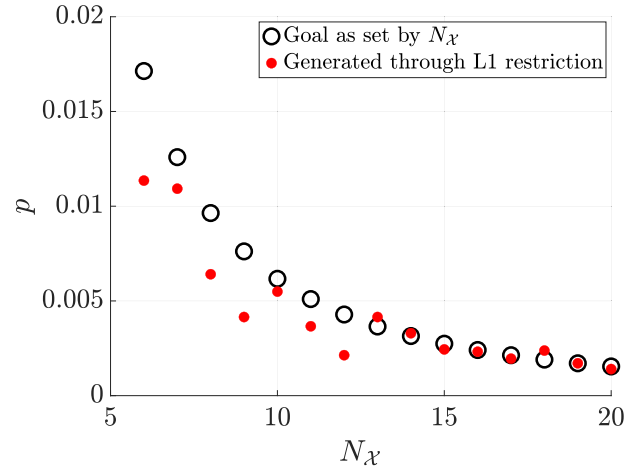
when Eq. (15) is solved for D_{L_1} and *p* is related to $N_{\mathcal{X}}$ via Eq. (1). Eq. (16) allows an algorithmic distance constraint to be found from an oracle call constraint and the environment concentration. From here, when q-RRT is building a database, states should be randomly selected from the boundary of a ball at radius D_{L_1}. In order to further restrict oracle calls, q-RRT can 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. 11. 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. (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. 11 shows that, as $N_{\mathcal{X}}$ increases, we have a more accurate ability to select *p*.

VI. 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. q-RRT, 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 [75], implementing q-RRT on disjoint trees [76] for planning over multiple disconnected components, employing a parallel quantum computing structure to q-RRT, and exploring path-optimality based algorithms in the context of quantum computing.

REFERENCES

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *J. Stat. Phys.*, vol. 22, no. 5, pp. 563–591, May 1980.

[2] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proc. Roy. Soc. London A, Math. Phys. Sci.*, vol. 400, no. 1818, pp. 97–117, 1985.

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

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

[5] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett.*, vol. 74, no. 20, pp. 4091–4094, May 1995.

[6] D. Loss and D. P. DiVincenzo, "Quantum computation with quantum dots," *Phys. Rev. A, Gen. Phys.*, vol. 57, no. 1, pp. 120–126, Jan. 1998.

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

[8] M. C. Bañuls, R. Blatt, J. Catani, A. Celi, J. I. Cirac, M. Dalmonte, L. Fallani, K. Jansen, M. Lewenstein, S. Montangero, C. A. Muschik, B. Reznik, E. Rico, L. Tagliacozzo, K. Van Acoleyen, F. Verstraete, U.-J. Wiese, M. Wingate, J. Zakrzewski, and P. Zoller, "Simulating lattice gauge theories within quantum technologies," *Eur. Phys. J. D*, vol. 74, no. 8, pp. 1–42, Aug. 2020.

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

[10] 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. N. Cleland, and J. M. Martinis, "Computing prime factors with a Josephson phase qubit quantum processor," *Nature Phys.*, vol. 8, no. 10, pp. 719–723, Oct. 2012.

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

[12] C. C. Chang, A. Gambhir, T. S. Humble, and S. Sota, "Quantum annealing for systems of polynomial equations," *Sci. Rep.*, vol. 9, no. 1, p. 10258, Jul. 2019.

[13] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Frontiers ICT*, vol. 4, p. 29, Dec. 2017.

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

[15] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. ACM Symp. Theory Comput.*, vol. 28, 1996, pp. 212–219.

[16] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," in *Contemporary Mathematics*, vol. 305. Providence, RI, USA: American Mathematical Society, 1999, pp. 53–74.

[17] D. Daoyi, C. Chunlin, and L. Hanxiong, "Reinforcement strategy using quantum amplitude amplification for robot learning," in *Proc. Chin. Control Conf.*, Jul. 2006, pp. 571–575.

[18] D. Dong, C. Chen, J. Chu, and T.-J. Tarn, "Robust quantum-inspired reinforcement learning for robot navigation," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 1, pp. 86–97, Feb. 2012.

[19] D. Dong, C. Chen, H. Li, and T. Tarn, "Quantum reinforcement learning," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1207–1220, Oct. 2008.

[20] D. Wang, A. Sundaram, R. Kothari, A. Kapoor, and M. Roetteler, "Quantum algorithms for reinforcement learning with a generative model," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10916–10926.

[21] K. Rajagopal, Q. Zhang, S. Balakrishnan, P. Fakhari, and J. Busemeyer, "Quantum amplitude amplification for reinforcement learning," in *Handbook of Reinforcement Learning and Control* (Studies in Systems, Decision and Control). Cham, Switzerland: Springer, 2021, pp. 819–833.

[22] L. Chen, Z. Jiang, L. Cheng, A. Knoll, and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers Neurorobotics*, vol. 16, May 2022, Art. no. 883562.

[23] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[24] L. Ming, "An adaptive quantum evolutionary algorithm and its application to path planning," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 2067–2071.

[25] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 580–593, Dec. 2002.

[26] A. G. Gad, "Particle swarm optimization algorithm and its applications: A systematic review," *Arch. Comput. Methods Eng.*, vol. 29, no. 5, pp. 2531–2561, Aug. 2022.

[27] C. Petschnigg, M. Brandstötter, H. Pichler, M. Hofbaur, and B. Dieber, "Quantum computation in robotic science and applications," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 803–810.

[28] N. B. Dehaghani, F. L. Pereira, and A. P. Aguiar, "Quantum control modelling, methods, and applications," *Extensive Rev.*, vol. 2, no. 1, pp. 75–126, Nov. 2022.

[29] R. Portugal, *Quantum Walks and Search Algorithms*, vol. 19. Berlin, Germany: Springer, 2013.

[30] S. Aaronson and A. Ambainis, "Quantum search of spatial regions," in *Proc. 44th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2003, pp. 200–209.

[31] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

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

[33] M. Szegedy, "Quantum speed-up of Markov chain based algorithms," in *Proc. 45th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2004, pp. 32–41.

[34] E. Sánchez-Burillo, J. Duch, J. Gómez-Gardeñes, and D. Zueco, "Quantum navigation and ranking in complex networks," *Sci. Rep.*, vol. 2, no. 1, pp. 1–8, Aug. 2012.

[35] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Apr. 1996, pp. 113–120.

[36] J. Barraquand, L. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," in *Proc. 7th Int. Symp. Robot. Res.* London, U.K.: Springer, 1996, pp. 249–264.

[37] D. Hsu and Z. Sun, "Adaptively combining multiple sampling strategies for probabilistic roadmap planning," in *Proc. IEEE Conf. Robot., Autom. Mechatronics*, vol. 2, Dec. 2004, pp. 774–779.

[38] S. Thomas, M. Morales, X. Tang, and N. M. Amato, "Biasing samplers to improve motion planning performance," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1625–1630.

[39] L. Jaillet, A. Yershova, S. M. La Valle, and T. Simeon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2005, pp. 2851–2856.

[40] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3307–3312.

[41] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Sep./Oct. 2002, pp. 2383–2388.

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

[43] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[44] 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," *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 501–513, Apr. 2016.

[45] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: An improved path planning algorithm for mobile robots," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113425.

[46] J. Canny, B. Donald, J. Reif, and P. Xavier, "On the complexity of kino-dynamic planning," Cornell Univ., Ithaca, NY, USA. Tech. Rep. 88-929, 1988.

[47] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *J. Guid., Control, Dyn.*, vol. 25, no. 1, pp. 116–129, Jan. 2002.

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

[49] J. Preskill, "Lecture notes for physics 229: Quantum information and computation," *California Inst. Technol.*, vol. 16, no. 1, pp. 1–8, 1998.

[50] D. Devaurs, T. Siméon, and J. Cortés, "Parallelizing RRT on distributed-memory architectures," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2261–2266.

[51] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 4, Apr./May 2004, pp. 3258–3262.

[52] J. Bialkowski, S. Karaman, and E. Frazzoli, "Massively parallelizing the RRT and the RRT*," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 3513–3518.

[53] B. Schumacher, "Quantum coding," *Phys. Rev. A, Gen. Phys.*, vol. 51, no. 4, pp. 2738–2747, 1995.

[54] M. Born, "Quantum mechanics of collision processes," *Physics-Uspekhi*, vol. 38, pp. 803–827, 1926.

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

[56] M. Schlosshauer, "Decoherence, the measurement problem, and interpretations of quantum mechanics," *Rev. Mod. Phys.*, vol. 76, no. 4, pp. 1267–1305, Feb. 2005.

[57] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Rev. Phys.*, vol. 4, Nov. 2019, Art. no. 100028.

[58] B. Schmitt, F. Mozafari, G. Meuli, H. Riener, and G. De Micheli, "From Boolean functions to quantum circuits: A scalable quantum compilation flow in C++," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1044–1049.

[59] A. Ambainis, "Understanding quantum algorithms via query complexity," in *Proc. Int. Congr. Mathematicians*. Singapore: World Scientific, 2018, pp. 3265–3285.

[60] G. Brassard, P. Høyer, and A. Tapp, "Quantum counting," in *Proc. 25th Int. Colloq. Automata, Lang., Program.*, Jul. 1998, pp. 820–831.

[61] C. Fox, "Quantum computing functions (QCF) for MATLAB," Robot. Res. Group, Oxford Univ., Oxford, U.K., Tech. Rep., 2003.

[62] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Apr. 2000, pp. 995–1001.

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

[64] W. Zurek, "Probabilities from entanglement, Born's rule $p_\kappa = |\psi_\kappa|^2$ from envariance," *Phys. Rev. A, Gen. Phys.*, vol. 71, May 2005, Art. no. 052105.

[65] K. Christensen, "Percolation theory," Imperial College London, London, U.K., Tech. Rep., 2002, vol. 1.

[66] R. M. Ziff, "Spanning probability in 2D percolation," *Phys. Rev. Lett.*, vol. 69, no. 18, pp. 2670–2673, Nov. 1992.

[67] N. Jan, "Large lattice random site percolation," *Phys. A, Stat. Mech. Appl.*, vol. 266, nos. 1–4, pp. 72–75, Apr. 1999.

[68] S. Mertens and R. M. Ziff, "Percolation in finite matching lattices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 94, no. 6, Dec. 2016, Art. no. 062152.

[69] M. F. Sykes and J. W. Essam, "Exact critical percolation probabilities for site and bond problems in two dimensions," *J. Math. Phys.*, vol. 5, no. 8, pp. 1117–1127, Aug. 1964.

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

[71] H. Trentelman, A. Stoorvogel, and M. Hautus, *Control Theory for Linear Systems*. Berlin, Germany: Springer, 2012.

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

[73] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637–676, Jan. 1999.

[74] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *Proc. IEEE IEEE Conf. Comput. Aided Control Syst. Design Int. Conf. Control Appl. Int. Symp. Intell. Control*, Oct. 2006, pp. 1582–1587.

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

[76] T. Lai, F. Ramos, and G. Francis, "Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5537–5543.

PAUL LATHROP (Student Member, IEEE) received the B.S. degree in aerospace engineering from the University of Maryland, College Park, in 2019, and the M.S. degree in aerospace engineering from the University of California at San Diego, in 2021, where he is currently pursuing the Ph.D. degree in mechanical and aerospace engineering. His research interests include robotic motion planning and quantum algorithms.

BETH BOARDMAN received the B.S. and M.S. degrees in aeronautics and astronautics from the University of Washington, Seattle, USA, in 2010 and 2012, respectively, and the Ph.D. degree in aerospace engineering from the University of California at San Diego, San Diego, USA, in 2017. She has been a Research and Development Engineer with the Los Alamos National Laboratory, since 2018. Her research interests include robotics and automation. She is currently the Team Leader for the Process Control and Robotics Team, Process Automation and Control Group.

SONIA MARTÍNEZ (Fellow, IEEE) received the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Spain, in May 2002. She is currently a Professor in mechanical and aerospace engineering with the University of California at San Diego, San Diego, CA, USA. She was a Visiting Assistant Professor in applied mathematics with the Technical University of Catalonia, Spain, from 2002 to 2003, and a Postdoctoral Fulbright Fellow of the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign, from 2003 to 2004, and the Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, Santa Barbara, from 2004 to 2005. Her research interests include the control of networked systems, multi-agent systems, nonlinear control theory, and planning algorithms in robotics. She is the coauthor (together with F. Bullo and J. Cortés) of *Distributed Control of Robotic Networks* (Princeton University Press, 2009). She is the coauthor (together with M. Zhu) of *Distributed Optimization-Based Control of Multi-agent Networks in Complex Environments* (Springer, 2015). She is the Editor-in-Chief of the recently launched *CSS IEEE Open Journal of Control Systems*.

● ● ●