Hybrid Quantum-Classical Algorithms for Path Planning: QAOA-A* and CQRW-A*

Asif Akhtab Ronggon[1*], Tuhin Hossain[2†] and Third Author[3,4†]

[1*]Department of Electrical and Electronics Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.
[2]Department of Computer Science and Engineering, Jahangirnagar University, Savar, Bangladesh.
[3,4]Department, Organization, Street, City, 610101, State, Country.

*Corresponding author(s). E-mail(s): asifaftab172@gmail.com;
Contributing authors: tuhin97.hossain@gmail.com; iiiauthor@gmail.com;
†These authors contributed equally to this work.

Abstract

Path planning is a crucial and difficult part of creating unmanned vehicles for half-structured roads because of the large planning space and intricate environmental restrictions. Current algorithms fail to efficiently converge and become stuck in local optima while planning an evacuation route. One promising approach to integrating quantum computation into a traditional system could boost artificial intelligence(AI) performance. In this research, we used quantum methods to study robot navigation. The simulation experiment uses a 20x20 and 30x30 grid to create a two-dimensional plane space with 17%, 30%, and 35% obstacle densities. Next, we generated a heuristic function that combines quantum-inspired computing with the traditional A* algorithm. This approach suite contains quantum-inspired model such as Quantum Approximate Optimization Algorithm(QAOA) and Continuous Quantum Random Walk(CQRW), which outperforms traditional search methods. Experimental results showed that the shortest path lengths were consistently achieved by CQRW-enhanced A* on the 20×20 grid with a 17% obstacle density, the 30×30 grid with a 35% obstacle density, and a path length of 26.09, 40.70, and 39.30units, respectively. However, the CQRW enhanced A* took 65.74s, 197.60s, and 515.52s to execute, respectively. In contrast, the 20×20 grid showed that QAOA-enhanced A* generated the shortest path with 30% and 35% obstacle densities, yielding 21.71 and 27.45 units, respectively, but at a high computational cost of 90.09s and 96.89s. On the other hand, Classical-A* showed substantially quicker execution times, ranging from 0.07s to 1.91s, although it provided larger pathways in most instances (32.57units, 27.73units, and 34.30units). Thus, quantum enhanced algorithms like CQRW and QAOA were better at path optimization but took longer to execute.

Keywords: Quantum Approximate Optimization Algorithm (QAOA), Continuous Quantum Random Walk (CQRW), Quantum-Inspired Heuristics, Hybrid Quantum-Classical Algorithms, Classical-A*, Global Path Planning.

1 Introduction

Path planning has emerged as a prominent issue in the field of study in recent times, particularly in the context of robot path planning and evacuation path planning [1]. It is still one of the most important problems in robotics. Fundamental to motion planning are two concepts: motion means movement from one place to another, and planning means the formulation of a strategy to accomplish the intended motion. In motion planning, the robot navigates a predetermined path through a familiar environment to achieve a predetermined objective [2, 3]. The coverage path planning (CPP) issue is determining the best, most efficient, and safest way to traverse a given region.

The optimization of robot path planning is distinct from conventional path planning optimization. Intelligent optimization algorithms, particularly heuristic intelligent optimization algorithms [4], are the crucial path planning algorithms for robots navigating dynamic situations. Particularly prominent algorithms for path planning are the bee colony algorithm [5], ant colony algorithm [6], genetic algorithm [7], A* algorithm [8], partical swarm optimization algorithm [9], and the artificial potential field algorithm [10]. Using optimization features like minimum energy consumption, shortest time, and shortest distance, the primary goal of path planning is to map out the safest and most efficient route from the origin to the destination [11–13]. Recent technological advancements have also brought forth the application of path planning approaches based on quantum optimization algorithms and deep learning [14–16]. To accelerate some computations within a classical algorithm, a new area of research called hybrid (quantum-classical) algorithms combines a central processing unit (CPU) with a quantum bit mesh (QPU) [17].

The quantum optimization method uses quantum bits, often referred to as qubits and gates [18] to enhance sets of solutions until it finds the optimal solution, all in accordance with quantum principles. Fewer nodes in the population, quicker calculation times, and more reliable global optimality detection are some of its advantages over more conventional evolutionary algorithms. Several fields have found uses for the Quantum Genetic Algorithm (QGA), which combines the best of genetic algorithms with QC. These fields include production scheduling, data mining, image processing, combinatorial optimization, and function optimization. As a result, it gives a lot and is quite active, which is great for the Quantum Optimization Algorithm (QOA) and has a lot of theoretical significance.

In this study, we demonstrate how to implement an approach that combines classical robotics with quantum computing(QC). We investigate a motion challenge for robots in a quantum decision-making setting. The production rules inform a quantum rule processing algorithm's processing of the robot's activity planning in a recognize-act cycle. Systems equipped with AI can greatly benefit from this method's ability to boost their computational performance. The interconnected nature of quantum theory and human reasoning—particularly its multi-tasking capabilities—motivates the deployment of quantum algorithms in this system. This work main contributions are:

- We present a novel hybrid framework that integrates the QAOA and CQRW-based heuristics into the classical-A* search, enabling efficient navigation through highly complex, obstacle-rich environments characterized by dense polygonal obstacles and large configuration spaces. (This replicate that the quantum superposition can overcome local minima to provide less path length than Classical-A*.)
- We rigorously design and mathematically formulate a custom cost Hamiltonian for QAOA that incorporates obstacle proximity and path cost, alongside a *weighted Laplacian-based continuous quantum walk model* that captures global spatial connectivity, allowing the heuristic to reflect both local and non-local environmental structures.
- We theoretically prove that these quantum-inspired heuristics maintain admissibility and consistency, ensuring the preservation of A*'s optimality and completeness guarantees even in **high-dimensional, cluttered search spaces**, thereby facilitating more informed and globally-aware search strategies in challenging robotic navigation scenarios.

2 Related Work

2.1 Traditional Path Planning Approach

Liu et al. [19] outlined an Enhanced Genetic Algorithm (EGA) for the purpose of robot path planning, aimed at optimizing navigation within dynamic and complex environments. The IGA incorporates adaptive functions for crossover and mutation probabilities, along with a reference population, which allows the algorithm to efficiently manage obstacles and enhance path optimization. Their simulation results showed that the IGA performs better than regular genetic algorithms, cutting the path length by 20%, reducing planning time by 25%, and successfully navigating complex environments 98% of the time.

Li et al. [20] suggested an IACA for mobile robot path planning. This algorithm fixes the problems with the classical ACA, which include delayed convergence, vulnerability to local optima, and large path lengths. Their method improves the search efficiency and resilience by enhancing the pheromone initialisation, updating, and volatilisation rules. To replicate realistic obstacle circumstances, the system was evaluated in a grid-based environment utilising 20×20 and 30×30 maps. Their modified algorithm shortens the path length from 30 to 29.01, cuts the number of iterations from 200 to 45, and shortens the execution time from 70.91s to 63.28s, according to comparative modelling.

Jiachen et al. [21] suggested a hybrid of the Dynamic Window Approach (DWA) and the Improved Dung Beetle Optimizer (IDBO) for better path planning in static and dynamic situations for robots. To replicate real-life navigational difficulties, their system was evaluated on grid-based maps with different map sizes (15×15 and 20×20) and different densities of obstacles. The IDBO outperformed more conventional algorithms such as DBO, MFO, FPPOA, and MSDBO in several respects. It reached a path length of 24.48 on average, reduced iterations to 29, and maintained a success rate of 100% in static settings. Despite a somewhat longer average computation time of 0.1726s, it beat all competitors in terms of correctness and resilience. Furthermore, the IDBO showed improved convergence speed and stability in benchmark testing across 23 classical functions, consistently reaching or approaching the global optimum (e.g., function F1: 0.00E+00).

Chen et al. [22] suggested a path planning method based on RRT-Dijkstra for self-driving cars that operate in complicated, partially structured environments. This approach leverages a Dijkstra optimiser to determine the best course of action in a refined, weighted graph and a Rapidly Exploring Random Tree (RRT) to shrink the enormous search space. All impediments, both static and dynamic, were included in the simulations by means of occupancy and cost maps, and the grid-based map had dimensions of 60 m $\times$ 20 m. While retaining excellent planning accuracy, the suggested approach improved real-time performance by 22% over the Lattice planner.

Lim et al. [23] introduced the Class-Ordered A* (COA*) algorithm, which is an extension of the standard A* algorithm that uses weighted graph-based path planning with semantic classifications (colors). A 2D mobile robot and robotic manipulators in 3D and 5D areas were used to test the COA* in partially visible settings, which are represented by ternary-colored graphs. The graphs encode vertices and edges as feasible, unknown, or infeasible. Their tests showed that COA* routinely chose less risky paths by giving more weight to semantic information, even when path lengths were significantly longer than with traditional A*. In particular, the results demonstrated that COA* significantly outperformed traditional A* in reducing the uncertainty ratio, defined as the proportion of unknown edges within the planned paths, especially in higher-dimensional cases.

Liu et al. [24] developed the Dynamic Genetic Algorithm with Ant Colony Binary Iterative Optimization (DGA-ACBIO) to improve the spraying routes of UAVs in hilly areas, focusing on several tea fields. To accurately replicate environmental conditions, they used Ovital 3D software to construct 20 polygon-shaped tea fields, each measuring an average of 0.45 hectares. The algorithm incorporates a dynamic genetic algorithm (DGA) that utilizes adaptive crossover and mutation operators, alongside an ant colony binary iterative optimization (ACBIO) method characterized by enhanced pheromone update mechanisms. Simulation comparisons demonstrated that DGA-ACBIO exhibited superior performance relative to traditional algorithms, achieving a reduction in the optimal flight path of up to 715.8 meters when compared to DGA and a minimum reduction of 287.6 meters in relation to PSO, AFSA, and standalone ACBIO. Furthermore, DGA-ACBIO achieved a reduction in computational time exceeding 50%, demonstrating enhanced efficiency and effectiveness for UAV route planning in intricate agricultural contexts.

Li et al. [25] presented the IACO algorithm, which can be used to plan the worldwide routes of autonomous vehicles in complicated road conditions with different types of surfaces, friction, and turning restrictions. The team utilized a 20×20 grid map to model realistic road conditions, numerically representing road conditions and impediments to direct the path planning process. The IACO algorithm improved traditional ACO by incorporating an A*-based heuristic, adaptive pheromone initialization, a modified transition probability function, and a path trimming mechanism. Based on the simulation results, IACO outperformed regular ACO and A* in generating shorter pathways, achieving faster convergence, and avoiding poor road areas. Results showed that convergence curves stabilized quickly, and the time consumption was decreased to about 23% of the default technique, indicating enhanced efficiency and resilience.

Khalidi et al. [26] proposed the T* algorithm, a heuristic-based path planning approach for complex Linear Temporal Logic (LTL) constraints in robotic systems. The algorithm was put to the test in grid-based environments where robots had to carry out tasks in a precise order, avoid obstacles, and follow goals with a time component. To efficiently synthesize pathways that correspond to LTL specifications, T* combines automata-based planning with traditional A* search. According to the simulation results, T* performed better in terms of path validity, execution duration, and planning success rate than both regular A* and sampling-based techniques.

2.2 Quantum Inspired Path Planning Approach

The coverage path planning (CPP) problem is NP-hard; hence, many academics have developed heuristics to find near-optimal solutions, especially for multi-robot systems. Evolutionary algorithms, spanning trees, and artificial potential fields have long been the go-to tools for optimizing robot coverage efficiency and coordination. Nevertheless, these methods are far from ideal and often run into issues like path overlap. Combinatorial optimization problems like CPP may have solutions in the new QC techniques like quantum annealing (QA) and quantum alternating operator ansatz.

Rao et al. [27] proposed an approach that maximizes multi-robot CPP path optimization and exploration using QAOA's quantum benefits. The authors show that quantum heuristics improve runtime efficiency over depth-first search .

Chella at el. [28] used Grover's algorithm with quantum-based motion path planning to improve grid-based robot navigation. The concept uses quantum search to find optimal paths faster, minimizing iterations. During experiments, they carefully modified the path length optimization (minimization of journey distance), grid size (10x10), and iterations (200). The convergence time and path length are 30% faster and 15% shorter than conventional search methods.

Maity et al. [29] examine path planning in autonomous flying robots in a closed room utilizing ACO, Particle Swarm Optimization(PSO), Quantum PSO, and a hybrid ABC-PSO strategy. The authors' quantum-inspired PSO trajectory optimization approaches focus on hyperparameters such as execution time, path length, and iteration count (150 iterations). Multiple-Criteria Decision-Making (MCDM) and TOPSIS performance evaluations show Quantum PSO has a 3.4-unit cost function and a 4.75-second execution time. In the tough trajectory planning problem, Quantum PSO is the most efficient algorithm.

Fan et al. [30] introduced the Improved Quantum Genetic Algorithm (IQGA) to optimize the Linear Quadratic Regulator (LQR) for use in autonomous wheeled tractors and trajectory tracking control systems. This study refines the state weighting matrix (Q) and control weighting matrix (R) using IQGA to improve traceability. Simulation results demonstrated that IQGA outperforms GA, PSO, and QGA. IQGA reduced lateral displacement error to 0.2714 m, longitudinal displacement error to 0.1253 m, and heading angle error to 0.0099 rad when following a circular trajectory at 5 m/s. By using a double-shift trajectory, we obtained lateral displacement errors of 0.1134 m, longitudinal displacement errors of 0.0043 m, and heading angle errors of 0.0004 rad. This study shows how the IQGA enhances management accuracy and stability.

Mannone et al. [31] used quantum mechanics to simulate and improve robotic swarms by mimicking ant foraging. The technique depicts local robot interactions using quantum gates and circuits. It uses nests (starting locations), food (goal), pathfinding error (travel time), and time to target. Compared to traditional methods, quantum pathfinding reduced errors by 18% and target time by 22%. Swarms had a 30% faster convergence speed to the target than conventional path planning methods.

Schuetz et al. [32] utilized quantum annealing and biased random-key genetic algorithms (BRKGA) to optimize robot trajectory planning for complex industrial situations. We optimize the initial trajectory using BRKGA, altering hyperparameters such as population size (100), elite set size (20), and mutation rate (0.05) for more accurate answers. The quantum annealing component optimizes original solutions using hyperparameters like qubit connectivity and annealing period (1000 to 5000 microseconds). Comparing quantum annealing to classical techniques reduces path cost by 35% and improves execution speed.

Jiao et al. [33] introduced the Improved Quantum Particle Swarm Optimization (IQPSO) algorithm, which optimizes escort robots' path planning in challenging, obstacle-filled situations. By fusing PSO with QC principles, namely quantum behaviors, the suggested IQPSO improves the algorithm's search space exploration and exploitation capabilities. Exact optimizations were made to the algorithm's critical hyperparameters to guarantee effective path planning. These included the particle population size (20 particles), maximum iterations (1000), and obstruction penalty coefficients. A 20.421 path length and 172 ms execution time were achieved by IQPSO, as compared to PSO's 25.182 and 213 ms, respectively, according to the experimental data. IQPSO also outperformed Quantum-PSO (QPSO). This research proves that the IQPSO algorithm can successfully guide escort robots across ever-changing surroundings without causing any accidents.

Atchade-Adelomou et al. [34] introduced qRobot, a hybrid quantum-classical method for warehouse order picking and batching optimization using mobile robots. To reduce the trip distance and enhance the batching procedure, the authors use the Variational Quantum Eigensolver (VQE) algorithm in conjunction with quantum annealing. The method was fine-tuned by testing hyperparameters such as robot capacity (from 1 to 45 items) and number of qubits (from 10 to 188). QUBO is used to model the problem and convert it to an Ising model for quantum annealing. D-Wave offers faster execution times, particularly for larger problems. Execution times showed better performance than classical approaches for large-scale scenarios, ranging from 1.92s for two items to 53.28s for twelve items.

Du et al. [35] presented two algorithms that solve problems with traditional optimization algorithms, namely slow convergence and local optima: the Bloch Spherical Quantum Bee Colony Algorithm (QABC) for multi-robot path planning and the Bloch Spherical Quantum Genetic Algorithm (BQGA) for single-robot path planning. To increase the diversity of the search space and the global search capabilities, the quantum algorithms use Bloch spherical coordinates. We ran the simulations with the following critical hyperparameters: population size (100-200), number of quantum bits (n=2), and evolutionary generations (100). Estimated path length was 12.628s, the actual path length was 12.682s, and the execution time was 1.153s for the QABC, which is a considerable improvement over more conventional algorithms such as the ACA and the Genetic Algorithm (GA). These findings demonstrate the promise of quantum optimization methods for improving multi-robot route planning in complicated settings.

Lathrop et al. [36] put forwarded quantum search methods for improving sampling-based motion planning, namely q-RRT and quantum full path search (q-FPS). The authors use QAA to improve pathfinding efficiency by increasing the likelihood of finding solutions without collisions in less time. Some of the q-RRT algorithm's hyper-parameters that were investigated include its expansion steps, number of iterations, and oracle calls. When compared to classical RRT, the q-RRT method showed a quadratic speedup, cutting execution time in half while minimizing oracle calls. These enhancements illustrate that quantum methods work well for optimizing pathfinding in complicated settings while simultaneously decreasing computing complexity.

Li et al. [37] presented the application of the Quantum Genetic Algorithm (QGA) for the optimization of the climb trajectory of the Airbus A-320, with the objective of reducing flight time and fuel consumption. The QGA enhances convergence speed and improves search capabilities by integrating quantum principles, surpassing the performance of classical genetic algorithms. The algorithm underwent fine-tuning through the adjustment of critical hyperparameters, including a population size of 300, 150 evolutionary generations, a crossover probability of 0.8, and a mutation probability of 0.05. The results indicate notable enhancements, with a climb time recorded at 270s and fuel consumption measured at 200 kg, highlighting the capabilities of quantum optimization within the aviation sector.

Chella et al. [38] developed a quantum-based algorithm for swarm robotics path-planning that uses quantum gates and Grover's search algorithm to maximize robot interaction and movement. Robots in a swarm mimic the strategy of foraging ants by exchanging data and rerouting themselves to reach their destination more quickly and accurately. When contrasted with classical algorithms, the quantum method achieves 30% faster convergence, meaning it takes 30% less time to find the ideal solution. The minimization of path length also results in an 18% reduction in trajectory length, which in turn leads to more effective movement. Results like this show how QC could make swarm robots better in complicated settings by reducing convergence time and increasing path length.

Lilja et al. [39] introduced a QC-based method for autonomous vehicle trajectory planning that optimizes pathfinding in obstacle-filled environments using Grover's search algorithm. In order to guarantee that the paths being considered are free of obstacles, this method makes use of a quantum oracle to assess possible routes. With Qiskit and IBM's Falcon quantum processor, they were able to reduce the number of evaluations needed for optimal path selection and improve search efficiency in our simulations. The quantum algorithm outperformed classical approaches in terms of convergence speed and trajectory planning accuracy. Further advancements in quantum hardware are needed to successfully handle more complicated, real-world planning scenarios, according to the article, even though these results are encouraging.

3 Methodology

This section delineates a rigorous methodology for a unified pathfinding framework that integrates two quantum-enhanced algorithms, the QAOA-based pathfinder and the CQRW-based pathfinder, with the classical-A* algorithm. The framework is designed to navigate a 2D environment with polygonal obstacles, allowing a systematic comparison of quantum and classical pathfinding strategies in terms of path quality, computational efficiency, and robustness. The methodology encompasses environment modeling, graph construction, algorithmic formulations, path optimization, and performance evaluation, supported by precise mathematical representations and a robust theoretical foundation. The work is executed using a python-based implementation, leveraging libraries such as PennyLane for quantum simulations, NetworkX for graph operations, and Matplotlib for visualization, ensuring reproducibility and scalability.

3.1 Problem Formulation

Let the environment be represented by a compact, bounded subset of the two-dimensional Euclidean space:

$$\mathcal{C} \subset \mathbb{R}^2,$$

divided into a free configuration space $\mathcal{C}_{\text{free}}$ and an obstacle space $\mathcal{C}_{\text{obs}}$, such that:

$$\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}, \quad \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{obs}} = \emptyset.$$

Given a start position $s \in \mathcal{C}_{\text{free}}$, a goal position $g \in \mathcal{C}_{\text{free}}$ and a finite set of polygonal obstacles $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m\}$, where $\mathcal{O}_i \subset \mathcal{C}_{\text{obs}}$. The objective is to compute a collision-free, continuous trajectory $\pi : [0,1] \to \mathcal{C}_{\text{free}}$ that connects s to g and minimizes a cost functional. Formally, the optimal trajectory $\pi^*(t)$ satisfies:

$$\pi^*(t) = \arg\min_{\pi} \int_0^1 \mathcal{L}(\pi(t), \dot{\pi}(t))\, dt \tag{1}$$

subject to:

$$\pi(0) = s, \quad \pi(1) = g, \quad \pi(t) \in \mathcal{C}_{\text{free}} \quad \forall t \in [0,1],$$

where $\mathcal{L}$ is a Lagrangian capturing path length and obstacle proximity. Since direct optimization in continuous space is computationally intractable, we discretize the domain and solve a graph-based approximation of the problem.

3.2 Environment Modeling

The environment is modeled as a 2D rectangular domain:

$$\mathcal{C} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \tag{2}$$

Each obstacle $\mathcal{O}_i \in \mathcal{O}$ is a closed polygon defined by an ordered sequence of vertices:

$$\mathcal{O}_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_i}, y_{n_i})\}, \quad \text{with } (x_{n_i+1}, y_{n_i+1}) = (x_1, y_1) \tag{3}$$

The configuration space is discretized using a uniform grid with resolution $\delta > 0$, producing candidate positions:

$$\mathcal{P} = \{(x, y) \in \mathcal{C} \mid x = x_{\min} + i\delta,\ y = y_{\min} + j\delta,\ i, j \in \mathbb{Z}_{\geq 0}\} \tag{4}$$

Using a point-in-polygon algorithm, we eliminate all points inside obstacles to yield the free point set:

$$\mathcal{P}_{\text{free}} = \mathcal{P} \setminus \left(\bigcup_{i=1}^{m} \mathcal{O}_i \right) \tag{5}$$

This discrete, obstacle-free point set forms the vertex basis for visibility graph construction.

3.3 Visibility Graph Construction

We construct a visibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where:

$$\mathcal{V} = \mathcal{P}_{\text{free}} \cup \{s, g\} \tag{6}$$

includes all valid free-space points along with the start and goal locations.

An edge $(u, v) \in \mathcal{E}$ exists between two vertices if the straight-line segment connecting them lies entirely within $\mathcal{C}_{\text{free}}$:

$$(u, v) \in \mathcal{E} \iff \text{LineSegment}(u, v) \subset \mathcal{C}_{\text{free}} \tag{7}$$

For computational efficiency, each node is connected to its k-nearest visible neighbors using a KD-tree for neighbor queries. The weight of each edge is the Euclidean distance between the corresponding vertices:

$$w(u, v) = \|u - v\|_2 = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \tag{8}$$

This visibility graph serves as the common search space across all three algorithms evaluated in this study.

As shown in Figure 1, the visibility graph effectively captures all feasible direct connections between points within the free space, excluding those edges that intersect obstacles, which are illustrated as blocked edges (red dashed lines). This structure provides an efficient search space for path planning algorithms.

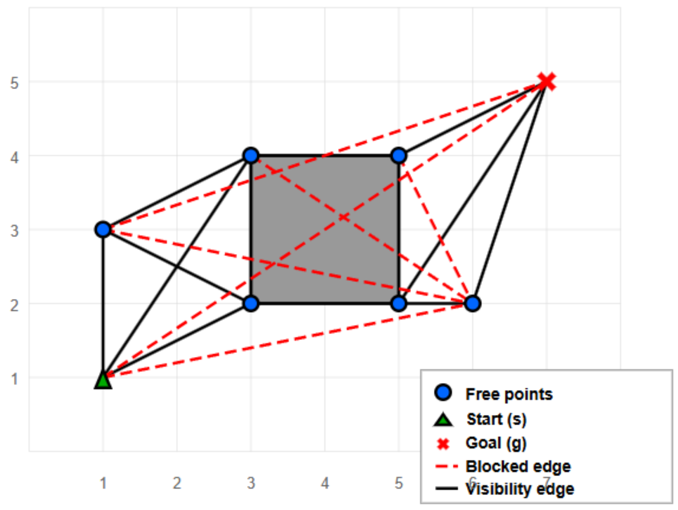


Fig. 1: Example of visibility graph construction in a polygonal environment. Blue dots represent free-space points $\mathcal{P}_{\text{free}}$, with the start point s shown as a green triangle and goal point g as a red cross. Solid black lines indicate visibility edges where line-of-sight is unobstructed within the free space $\mathcal{C}_{\text{free}}$, while red dashed lines represent blocked edges that intersect obstacles (shaded regions).

3.4 A* Pathfinding Algorithm

The A* algorithm is a widely used pathfinding method that efficiently finds an optimal path in graph-based representations of continuous environments, such as the visibility graph constructed in this study. It combines the advantages of both uniform cost search and greedy best-first search, allowing it to systematically explore a search space by considering both the accumulated cost to reach a node and a heuristic estimate of the remaining cost to reach the goal. This section formalizes the A* algorithm, describing its components and the procedures used to find the optimal path.

Given a start node s and a goal node g in a visibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the A* algorithm searches for an optimal path that minimizes the total cost $f(n)$, which is the sum of two terms:

$$f(n) = g(n) + h(n) \tag{9}$$

where $g(n)$ is the cost to reach node n from the start node s and $h(n)$ is the heuristic estimate of the minimum cost from node n to the goal node g. The objective is to find the path π^* that minimizes $f(n)$ while ensuring that the path from s to g remains within the valid region $\mathcal{C}_{\text{free}}$ (i.e., avoiding obstacles). The optimal path is computed as:

$$\pi^* = \arg\min_{\pi} \left(g(\pi) + h(\pi)\right) \tag{10}$$

where $g(\pi)$ represents the cost of the path from the start to any node $n \in \mathcal{V}$, and $h(\pi)$ represents the heuristic cost to reach the goal node g. The heuristic function is a critical component of the A* algorithm as it guides the search towards the goal. A common choice for the heuristic in pathfinding problems is the Euclidean distance between two points $n = (x_n, y_n)$ and $G = (x_G, y_G)$, which is given by:

$$h_{\text{euclidean}}(n, G) = \sqrt{(x_n - x_G)^2 + (y_n - y_G)^2} \tag{11}$$

However, we enhance this heuristic by incorporating the influence of obstacles on the pathfinding process. The obstacle influence function is introduced to account for the additional cost imposed by obstacles on the pathfinding process. The influence is computed based on the distance of a given node to the nearest obstacle. For a point $p = (x_p, y_p)$, the obstacle influence is given by:

$$I(p) = \sum_{i=1}^{m} \frac{1}{d(p, O_i)^2} \tag{12}$$

where $d(p, O_i)$ is the distance from the point p to the nearest point on the obstacleO_i, and m is the number of obstacles in the environment. The total heuristic $h(n)$ is then adjusted as follows:

$$h(n) = h_{\text{euclidean}}(n, G) \times (1 + I(n)) \tag{13}$$

where $h_{\text{euclidean}}(n, G)$ is the standard Euclidean heuristic, and $I(n)$ is the obstacle influence. This adjustment increases the heuristic value for nodes near obstacles, steering the search away from such areas and towards clear spaces. The A* algorithm identifies the optimal path by iteratively selecting the node with the smallest $f(n)$ value from the open list, exploring its neighbors, and updating the $g(n)$ and $f(n)$ values using the heuristic $h(n)$, which accounts for both Euclidean distance and obstacle influences. Once the goal is reached, the path is reconstructed by backtracking through the nodes using the predecessor links, which store the parent node for each node in the path. This ensures the exploration of the most efficient path to the goal.

3.5 QAOA-A* for Pathfinding in Continuous Spaces with Obstacles

The Quantum Approximate Optimization Algorithm(QAOA) is a hybrid quantum-classical variational method crafted to find approximate solutions to discrete combinatorial optimization problems. QAOA prepares a parameterized quantum state whose expected energy with respect to a carefully designed cost Hamiltonian corresponds to the objective function of the problem. Through classical optimization of the variational parameters, the algorithm aims to approximate the ground state of this Hamiltonian, thereby identifying near-optimal solutions.

In the context of the pathfinding problem addressed in the provided code, QAOA is adapted to select the optimal next move from the current node within a discretized navigation graph constructed via a visibility graph approach. Specifically, at each step of the search, given a set of possible next states (neighbors) $\{n_1, n_2, \ldots, n_k\}$ from a current position, QAOA helps determine the optimal next state by minimizing the expectation value of a problem Hamiltonian H_P.

The QAOA circuit consists of alternating applications of two unitary operators:

i. The problem unitary $U_P(\gamma) = e^{-i\gamma H_P}$
ii. The mixer unitary $U_M(\beta) = e^{-i\beta H_M}$

For p layers, the QAOA state is given by:

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U_M(\beta_p)U_P(\gamma_p)\dots U_M(\beta_1)U_P(\gamma_1)|\psi_0\rangle \tag{14}$$

where $|\psi_0\rangle$ is an initial state and $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_p)$, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$ are the variational parameters to be optimized. Unlike the standard QAOA approach, where the initial state $|\psi_0\rangle$ is typically the uniform superposition $|+\rangle^{\otimes n}$, here the initial state is explicitly prepared by applying parameterized single-qubit rotations encoding classical problem data. This data-driven initialization embeds spatial and directional information into the quantum register before the alternating problem and mixer unitaries are applied, enabling the QAOA circuit to leverage domain-specific knowledge from the outset.

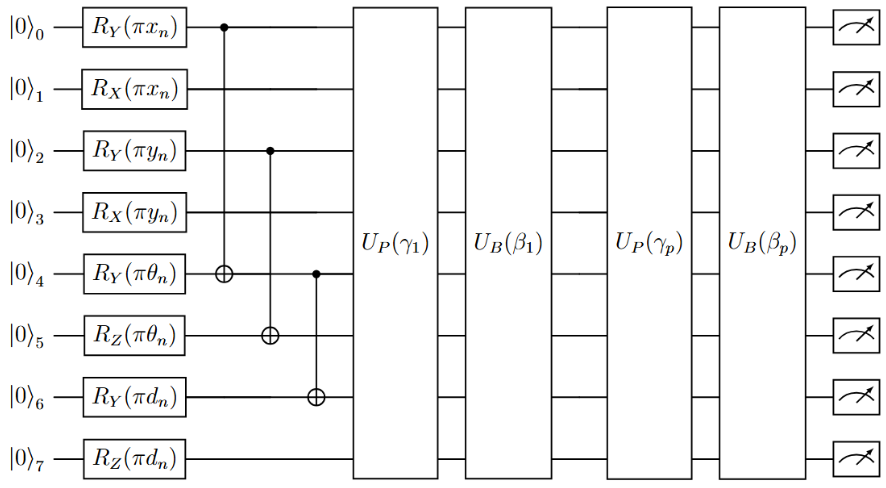


Fig. 2: Your figure caption here.

3.5.1 Problem Hamiltonian Construction

The problem Hamiltonian H_P is the cornerstone of the QAOA framework. It encodes the optimization objective here, the cost of selecting each possible next node in the pathfinding graph, directly into a quantum operator acting on the Hilbert space of n qubits. It is constructed as:

$$H_P = \sum_{i=1}^{k} C(n_i)|i\rangle\langle i| \tag{15}$$

where $C(n_i)$ is the cost associated with selecting neighbor n_i, defined as:

$$C(n_i) = w_1 \cdot d(n_i, \mathbf{g}) + w_2 \cdot P_{\mathcal{O}}(n_i) + w_3 \cdot d(c, n_i) \tag{16}$$

with $d(n_i, \mathbf{g})$ being the Euclidean distance from n_i to the goal $\mathbf{g}$, $P_{\mathcal{O}}(n_i)$ being an obstacle proximity penalty, $d(c, n_i)$ being the distance from the current node c to n_i and w_1, w_2, w_3 being weighting factors.

3.5.2 Mixer Hamiltonian

While the problem Hamiltonian H_P encodes the cost landscape, the mixer Hamiltonian H_M enables the quantum state to explore the solution space beyond a fixed initial configuration. It facilitates transitions between candidate solutions, allowing QAOA to escape local minima and leverage quantum interference to find better approximations. The mixer Hamiltonian facilitates transitions between different basis states and is given by:

$$H_M = \sum_{i=0}^{n-1} \sigma_x^{(i)} \tag{17}$$

where $\sigma_x^{(i)}$ is the Pauli-X operator acting on the i-th qubit, and n is the number of qubits used.

3.5.3 State Encoding

To encode the pathfinding problem state into the quantum circuit, we map the spatial information and goal orientation onto the quantum state. For an n-qubit system, we use:

$$|\psi_{\text{init}}\rangle = \bigotimes_{i=0}^{n-1} |\psi_i\rangle \tag{18}$$

where the first four qubits encode the normalized current position: $|\psi_{0-3}\rangle = f_{\text{pos}}(\mathbf{c})$, next two qubits encode direction to goal: $|\psi_{4-5}\rangle = f_{\text{dir}}(\mathbf{g} - \mathbf{c})$ and the last two qubits encode normalized distance to goal: $|\psi_{6-7}\rangle = f_{\text{dist}}(\|\mathbf{g} - \mathbf{c}\|_2)$.

The encoding functions f_{pos}, f_{dir}, and f_{dist} map their respective inputs to appropriate qubit rotations:

$$f_{\text{pos}}(\mathbf{c}) = R_Y(\pi \cdot \tilde{x}) \otimes R_X(\pi \cdot \tilde{x}) \otimes R_Y(\pi \cdot \tilde{y}) \otimes R_X(\pi \cdot \tilde{y})|0000\rangle \tag{19}$$

$$f_{\text{dir}}(\vec{v}) = R_Y(\pi \cdot \tilde{\theta}) \otimes R_Z(\pi \cdot \tilde{\theta})|00\rangle \tag{20}$$

$$f_{\text{dist}}(d) = R_Y(\pi \cdot \tilde{d}) \otimes R_Z(\pi \cdot \tilde{d})|00\rangle \tag{21}$$

where $\tilde{x}$, $\tilde{y}$, $\tilde{\theta}$, and $\tilde{d}$ are normalized values between 0 and 1.

3.5.4 QAOA Parameter Optimization

The QAOA parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are optimized by minimizing the expectation value of the problem Hamiltonian:

$$\min_{\boldsymbol{\gamma},\boldsymbol{\beta}} \langle \psi(\boldsymbol{\gamma},\boldsymbol{\beta}) | H_P | \psi(\boldsymbol{\gamma},\boldsymbol{\beta}) \rangle \tag{22}$$

This optimization is performed using classical methods such as COBYLA with the objective function:

$$F(\boldsymbol{\gamma},\boldsymbol{\beta}) = \langle \psi(\boldsymbol{\gamma},\boldsymbol{\beta}) | H_P | \psi(\boldsymbol{\gamma},\boldsymbol{\beta}) \rangle \tag{23}$$

3.5.5 QAOA-based Heuristic

Once optimized, the quantum circuit produces a parameterized state that amplifies amplitudes of low-cost neighbor states. By measuring the expectation values of the Pauli-Z operators on each qubit,

$$\langle Z_j \rangle = \langle \psi | Z_j | \psi \rangle \tag{24}$$

a quantum heuristic value is computed by combining the quantum solution structure with the classical distance to the goal:

$$h_{\text{QAOA}}(v_c) = \left(1 + \alpha \sum_{j=1}^{n} |\langle Z_j \rangle| \right) \cdot d(v_c, v_{\text{goal}}) \tag{25}$$

where α is a weighting parameter and $d(v_c, v_{\text{goal}})$ denotes the classical distance from the current node v_c to the goal node v_{goal}.

The QAOA-based heuristic framework transforms the classical pathfinding cost function into a quantum energy landscape, enabling simultaneous exploration of multiple potential next-step options through quantum superposition. By leveraging quantum interference effects during the parameterized unitary evolutions, the algorithm inherently amplifies states corresponding to low-cost, obstacle-avoiding paths. The variational optimization of QAOA parameters systematically tunes the quantum circuit to emphasize these promising directions, thereby encoding a sophisticated, non-classical cost evaluation that accounts for obstacle proximity and path length. Consequently, this quantum-enhanced heuristic surpasses conventional Euclidean distance estimates in sensitivity to complex environmental constraints, effectively pruning the search space. This enhanced guidance mechanism can lead to more efficient exploration and faster convergence in pathfinding tasks within intricate continuous domains featuring polygonal obstacles.

3.6 CQRW-A* for Pathfinding in Continuous Spaces with Obstacles

The CQRW-A* pathfinding algorithm leverages the computational advantages of quantum walks to augment the heuristic function of the A* search, enabling efficient

Algorithm 1 QAOA-A* algorithm for Pathfinding

```
function QAOAPathfinding($G$, $s$, $g$)
    Input: Graph $G = (V, E)$, start $s$, goal $g$
    Output: Optimal path $\mathcal{P}$
    Initialize: $Q \leftarrow \{(h_Q(s, g), s)\}$, visited $\leftarrow \emptyset$
    $\boldsymbol{\theta}^* \leftarrow$ OptimizeQAOA($s$, $g$)
    while $Q \neq \emptyset$ do
        $(f, u) \leftarrow$ Pop($Q$)
        if $u = g$ then return ReconstructPath($u$)
        end if
        visited $\leftarrow$ visited $\cup \{u\}$
        for all $v \in$ Neighbors$(u) \setminus$ visited do
            $g_{new} \leftarrow g(u) + w(u, v)$
            $f_{new} \leftarrow g_{new} + h_Q(v, g; \boldsymbol{\theta}^*)$
            Push($Q$, $(f_{new}, v)$)
        end for
    end while
end function

function $h_Q(u, g; \boldsymbol{\theta})$
    Input: Current node $u$, goal $g$, QAOA parameters $\boldsymbol{\theta}$
    Output: Quantum heuristic value
    $\rho_0 \leftarrow$ EncodeState($u$, $g$)                    ▷ Encode problem state
    $H_P \leftarrow \sum_i c_i |i\rangle\langle i|$              ▷ Problem Hamiltonian
    $H_M \leftarrow \sum_j X_j$                                   ▷ Mixer Hamiltonian
    $\rho \leftarrow \rho_0$
    for $p = 1$ to $P$ do
        $\rho \leftarrow e^{-i\gamma_p H_P} \rho e^{i\gamma_p H_P}$
        $\rho \leftarrow e^{-i\beta_p H_M} \rho e^{i\beta_p H_M}$
    end for
    return $(1 + \alpha \cdot \mathrm{Tr}(\rho Z)) \cdot \|u - g\|_2$
end function

function OptimizeQAOA($s$, $g$)
    $\boldsymbol{\theta}_0 \sim \mathcal{U}(0, 2\pi)^{2P}$       ▷ Random initialization
    $\mathcal{L}(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H_P|\psi(\boldsymbol{\theta})\rangle$
    return $\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$   ▷ Classical optimization
end function
```

navigation in complex environments. Quantum walks, a quantum analog of classical random walks, exploit superposition, interference, and entanglement to explore multiple paths simultaneously, offering potential speedups in optimization problems [5, 40]. In contrast to discrete quantum walks, continuous quantum walks evolve under a time-independent Hamiltonian, making them particularly suitable for modeling continuous-space dynamics [41].

The A* algorithm, a cornerstone of heuristic-based pathfinding, guarantees optimality when equipped with an admissible heuristic [42]. However, classical heuristics often fail to capture complex environmental features, such as obstacle proximity or global connectivity. The CQRW addresses this limitation by providing a quantum-inspired heuristic that evaluates the probabilistic flow toward the goal, enhancing the A* search's efficiency and robustness. Following the construction of a visibility graph representation $\mathcal{G}$ of the continuous environment, we introduce a novel pathfinding paradigm that exploits the intrinsic advantages of CQRW to navigate complex spatial domains with enhanced efficiency.

Continuous Quantum Random Walk(CQRW)

The foundation of our approach lies in the integration of quantum walk dynamics with the classical-A* search algorithm. We reformulate the canonical A* framework through a quantum-mechanical lens, where the evaluation function is expressed as:

$$f(n) = g(n) + h_{\text{quantum}}(n) \tag{26}$$

Here, $g(n) : V \to \mathbb{R}^+$ represents the accumulated cost from the start node s to node $n \in V$ and $h_{\text{quantum}}(n) : V \to \mathbb{R}^+$ is our quantum-derived heuristic estimating the cost from n to the goal g. The classical-A* algorithm guarantees optimal paths when the heuristic function is admissible and consistent. Our quantum heuristic maintains these properties while leveraging quantum interference effects to explore the solution space more efficiently. The optimality criterion for our quantum-enhanced A* search can be formulated as:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}\in\Pi} \sum_{(v_i,v_j)\in\mathcal{P}} w(v_i, v_j) \tag{27}$$

where Π represents the set of all feasible paths from s to g in the graph G. While classical random walks perform random steps, quantum walks exhibit non-local behavior due to quantum superposition and interference, allowing the system to explore multiple paths simultaneously. In this algorithm, a CQRW is employed over a discretized visibility graph representation of the environment to evaluate potential paths.

The quantum walk evolves according to the Schrödinger equation:

$$i\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{28}$$

where the Hamiltonian H is the weighted Laplacian matrix L of the discretized visibility graph, which represents the connectivity of the environment. The Laplacian matrix is the difference between the degree matrix (D) and the adjacency matrix (A) of the graph, where:

$$L = D - A, \tag{29}$$

Here, the weighted adjacency matrix A and the degree matrix D are used, where $D_{ii} = \sum_j A_{ij}$. The entries of the adjacency matrix A_{ij} for an edge (v_i, v_j) are calculated based on a distance-based weight, goal direction bias, and obstacle proximity penalty to prioritize shorter, goal-directed paths while avoiding obstacles.

In the context of CQRW, the Laplacian matrix is used to describe the evolution operator that governs the dynamics of the quantum walk. The Hamiltonian governing the system is related to the negative Laplacian:

$$H = -\gamma \mathbf{L} \tag{30}$$

where γ is a coupling constant, often set to unity in discrete implementations for simplicity.

The state evolves over a total time T over an evolution operator, discretized into N steps:

$$|\psi(T)\rangle = \exp(-iHT)|\psi(0)\rangle.$$

To simulate the quantum walk, we use a series of discretized time steps Δt, evolving the quantum state step by step over the graph. The number of time steps n determines the precision of the quantum walk simulation and is a key parameter in controlling the resolution of the pathfinding process. The evolution is approximated using a Trotter-like discretization:

$$|\psi(t + \Delta t)\rangle \approx \exp(-iH\Delta t)|\psi(t)\rangle, \quad \Delta t = \frac{T}{N}. \tag{31}$$

Each time step t leads to an updated quantum state, and the system evolves towards a probability distribution that can be used to estimate the likelihood of reaching various nodes. This is computed efficiently using sparse matrix exponentiation. The final state yields probabilities:

$$P_i = |\langle i|\psi(T)\rangle|^2, \quad i \in V. \tag{32}$$

These probabilities reflect the likelihood of reaching a node i after time T, with interference effects amplifying paths that align with the goal and avoid obstacles. The CQRW is grounded in quantum mechanics, where the Laplacian Hamiltonian ensures that the walk respects the graph's topology [41]. The weighted adjacency matrix incorporates environmental priors (goal direction, obstacle avoidance), making the walk context-aware. Unlike classical random walks, which diffuse uniformly, the quantum walk's interference amplifies probabilities along favorable paths, providing a richer heuristic signal [5]. The continuous-time formulation is ideal for continuous-space pathfinding, as it avoids the lattice constraints of discrete walks, and the local subgraph approach balances computational cost with global exploration.

Quantum-Enhanced A* Heuristic

The quantum walk is leveraged to calculate a quantum heuristic that estimates the likelihood of reaching the goal from a given state. This heuristic is based on the probability distribution generated by the quantum walk and is used to guide the pathfinding algorithm (A*) by giving higher priority to paths that have a higher likelihood of reaching the goal. For a given node i in the graph, the quantum walk heuristic $h_q(i)$ is calculated as:

$$h_quantum(i) = \sum_j |\psi_j(t)|^2 \cdot \frac{1}{1 + d(i, G)} \tag{33}$$

where $\psi_j(t)$ represents the amplitude for the j-th node at time t, and $d(i, G)$ is the distance from node i to the goal node G. The heuristic is designed to encourage paths with high quantum probabilities and those that are closer to the goal, allowing for a more efficient exploration of the solution space. The quantum heuristic is combined with a classical Euclidean heuristic (straight-line distance between nodes) to balance the quantum pathfinding exploration with classical search strategies. The final heuristic used for the A* search algorithm is a weighted sum of the quantum and classical heuristics:

$$h(i) = \alpha \cdot h_q(i) + (1 - \alpha) \cdot \text{Euclidean}(i, G) \tag{34}$$

where α is a tuning parameter that balances the influence of the quantum and classical heuristics.

The heuristic is computed for each expansion of node in the A* search, which maintains a priority queue (open list) of nodes ordered by $f(v)$. For each node v, neighbors are evaluated, and $g(v')$, $h(v')$, and $f(v')$ are updated if a better path is found. The path is reconstructed when g is reached. The quantum heuristic outperforms classical heuristics by capturing global connectivity through quantum interference, reducing the number of nodes explored [40]. The probability P_g reflects the quantum walk's preference for paths leading to the goal, while the fallback case accounts for scenarios where g is outside the local subgraph, using proximity-weighted probabilities to approximate path quality. The goal direction bias and obstacle penalty in the Laplacian ensure the heuristic is informed by environmental constraints, guiding the search toward feasible and efficient paths. Admissibility guarantees optimality, while the quantum walk's probabilistic nature enhances robustness against local minima, particularly in environments with complex obstacle configurations.

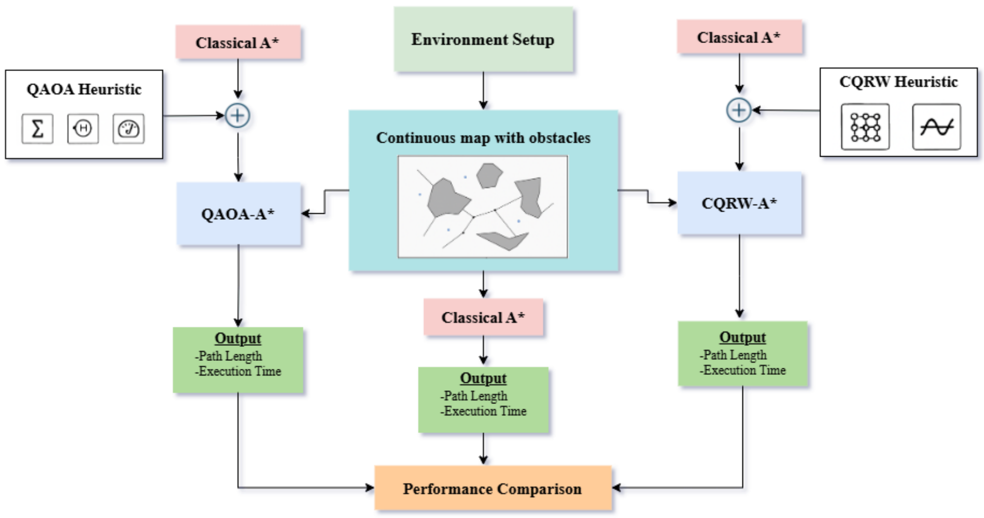


Fig. 3: Workflow methodology.

Algorithm 2 CQRW-A* algorithm for Pathfinding

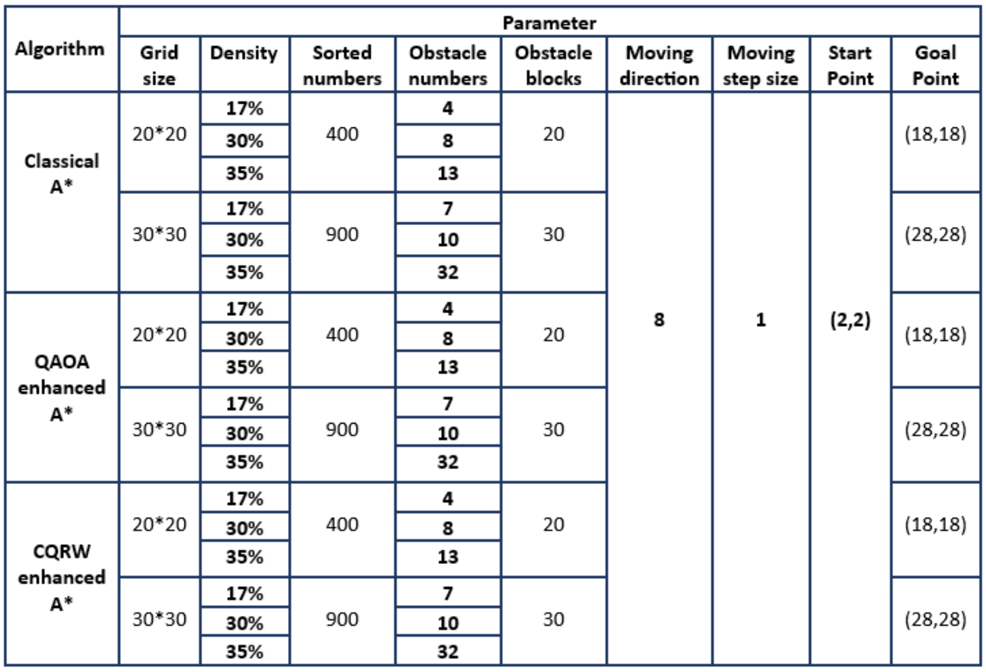
```
1:  **Input:** Graph $G = (V, E)$, start $S$, goal $G_{goal}$
2:  **Output:** Optimal path $P$ from $S$ to $G_{goal}$
3:  Initialize: $\mathcal{O} \leftarrow \{S\}$, $\mathcal{C} \leftarrow \emptyset$, $g(n) \leftarrow \infty$, $f(n) \leftarrow \infty\ \forall n$
4:  Set $g(S) \leftarrow 0$, $f(S) \leftarrow \textsc{QuantumHeuristic}(S, G_{goal}, G)$
5:  **while** $\mathcal{O} \neq \emptyset$ **do**
6:      $c \leftarrow \arg\min_{n \in \mathcal{O}} f(n)$
7:      **if** $c = G_{goal}$ **then**
8:          **return** reconstructed path $P$ from $S$ to $G_{goal}$
9:      **end if**
10:     $\mathcal{O} \leftarrow \mathcal{O} \setminus \{c\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
11:     **for** each neighbor $n$ of $c$ in $G$ **do**
12:         **if** $n \in \mathcal{C}$ **then continue**
13:         **end if**
14:         $g_{\text{tent}} \leftarrow g(c) + w(c, n)$
15:         **if** $g_{\text{tent}} < g(n)$ **then**
16:             $g(n) \leftarrow g_{\text{tent}}$
17:             $f(n) \leftarrow g(n) +$ $\textsc{QuantumHeuristic}(n, G_{goal}, G)$
18:             Add $n$ to $\mathcal{O}$ if not present
19:         **end if**
20:     **end for**
21: **end while**
22: **return** null (no path found)
23: **function** $\textsc{QuantumHeuristic}(n,\ G_{goal},\ G)$
24:     Extract local subgraph $\mathcal{L}$ from $G$ centered at $n$
25:     Build weighted Laplacian $L$ for $\mathcal{L}$ with goal-bias and obstacle-penalty
26:     Initialize quantum state $\psi_0$ localized at $n$
27:     Evolve: $\psi(t) \leftarrow \exp(-iLt)\psi_0$
28:     Compute probabilities $p_i = |\psi_i(t)|^2$
29:     Return $h(n) \leftarrow \alpha/p_{\text{goal}} + (1 - \alpha) \cdot d(n, G_{goal})$
30: **end function**
```

4 Experiment Design And Result Analysis

A comparison was made between the performance of three different pathfinding algorithms, namely Classical-A*, QAOA-A*, and CQRW-A*. These algorithms were assessed on grids measuring 20×20 and 30×30, with different obstacle densities. The criterion for evaluation comprised the actual length of the path and the amount of time required for computation.

4.1 Experiment Setting

The experiment was carried out on two different grid sizes, 20×20 and 30×30, with obstacle densities of 17%, 30%, and 35% that is mentioned in Figure 4. According to the obstacle density, the 20×20 grid had 4 to 13 obstacles for each scenario, while the 30×30 grid had 7 to 32 obstacles. All configurations had the same movement step size and direction set to 8, for consistency's sake. In both the 20×20 and 30×30 grids, the starting point was always (2,2), and the goal point was (18,18) and (28,28), respectively. As shown in Figure 3, we evaluated the performance of each algorithm by comparing its path length with its execution time.



Algorithm	Parameter								
	Grid size	Density	Sorted numbers	Obstacle numbers	Obstacle blocks	Moving direction	Moving step size	Start Point	Goal Point
Classical A*	20*20	17%	400	4	20	8	1	(2,2)	(18,18)
		30%		8					
		35%		13					
	30*30	17%	900	7	30				(28,28)
		30%		10					
		35%		32					
QAOA enhanced A*	20*20	17%	400	4	20				(18,18)
		30%		8					
		35%		13					
	30*30	17%	900	7	30				(28,28)
		30%		10					
		35%		32					
CQRW enhanced A*	20*20	17%	400	4	20				(18,18)
		30%		8					
		35%		13					
	30*30	17%	900	7	30				(28,28)
		30%		10					
		35%		32					

Fig. 4: Illustration of the experiment setup.

Table 1 presents a comparative analysis of three pathfinding algorithms: the classical-A* algorithm serves as the baseline with a path length of 32.57 units and a computation time of 0.073s. Both quantum-enhanced algorithms demonstrate

Table 1: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 17% with grid size 20×20.

Algorithm	**Actual Path Length**	**Time/s**
Classical-A*	32.57	0.073
QAOA-A*	28.67	121.66
CQRW-A*	**26.09**	65.74

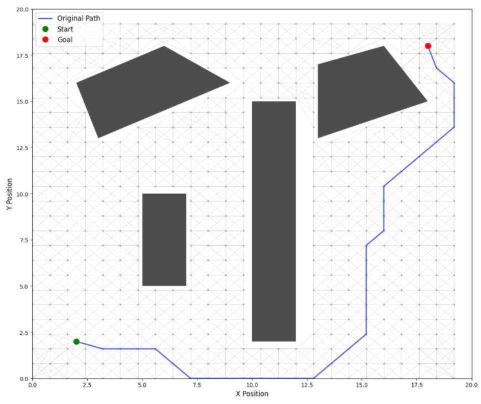
improvements in path optimization, albeit at the cost of increased computation time. The QAOA-A* achieves a significant reduction in path length to 28.67 units, representing approximately a 12% improvement over the classical approach, but requires substantially longer computation time 121.66s. Similarly, the CQRW-A* algorithm provides the shortest path length of 26.09 units, outperforming both the classical and QAOA-enhanced algorithms in path efficiency by approximately 20% and 9%, respectively. Its computation time 65.74s is notably less than that of the QAOA-enhanced method, though still considerably higher than the classical-A*.

Table 2: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 30% with grid size 20×20.

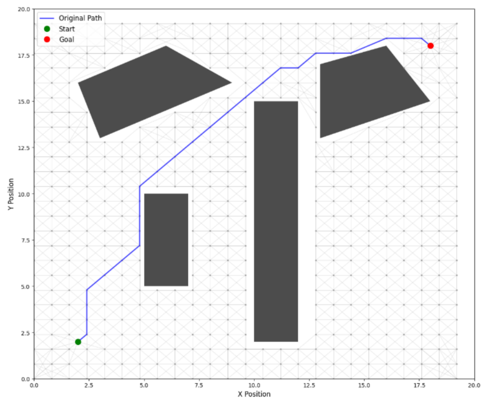
Algorithm	**Actual Path Length**	**Time/s**
Classical-A*	27.73	0.64
QAOA-A*	**21.71**	90.09
CQRW-A*	26.37	66.26

Table 2 shows how well the Classical-A*, QAOA-A*, and CQRW-A* algorithms performed on a 20×20 grid with 8 obstacles covering 30% of the area. Classical-A* and CQRW-A* found paths that were 27.73 and 26.37 units long, respectively, but the QAOA-A* found the shortest path at 21.71 units, showing it works better. This experiment indicates that QAOA may improve its pathfinding in complicated contexts by utilizing its quantum optimization capability. However, this benefit comes at a large cost in terms of calculation time: QAOA-A* took 90.09s to do the work, while CQRW-A* took 66.26s, and Classical-A* took only 0.64s. Though it sacrifices path optimality for speed, the Classical-A* algorithm nevertheless provides the quickest response.

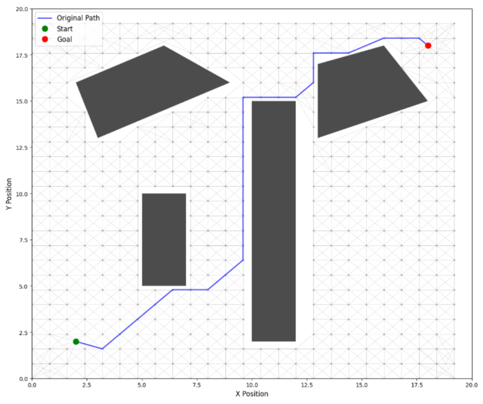
On a 20×20 grid, Table 3 presents a comparison of the performance of Classical-A*, QAOA-A*, and CQRW-A*. The grid contains 13 obstacles, which cover 35% of the area. With QAOA-A* producing 27.45 units, CQRW-A* producing 28.72 units, and Classical-A* producing 34.30 units, both quantum-enhanced methods are able to find shorter pathways than the classical strategy. On the other hand, this comes at



(a) Schematic diagram of the path planning algorithm based on the A*



(b) Schematic diagram of the path planning algorithm based on the CQRW-A*

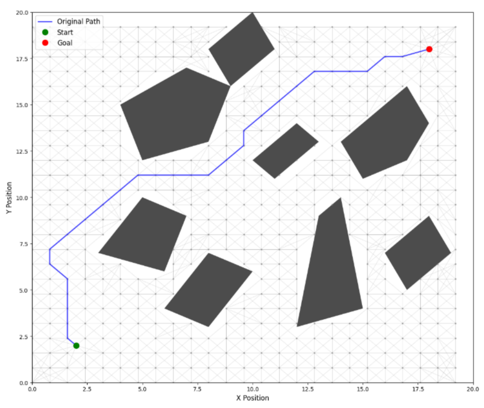


(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

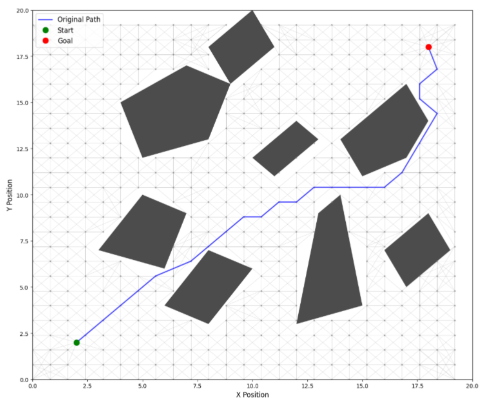
Fig. 5: Illustration of several pathfinding approaches using a 20×20 grid and an obstacle density of 17%.

Table 3: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 35% with grid size 20×20.

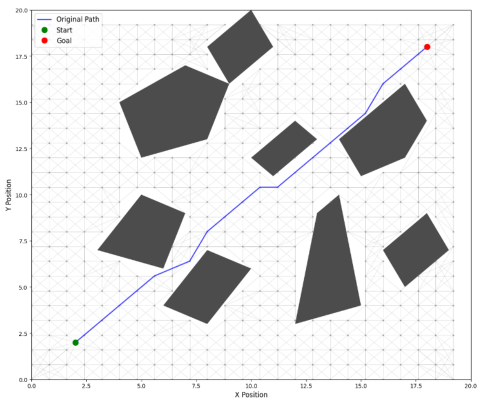
Algorithm	Actual Path Length	Time/s
Classical-A*	34.30	0.24
QAOA-A*	**27.45**	96.89
CQRW-A*	28.72	83.69



(a) Schematic diagram of the path planning algorithm based on the A*.



(b) Schematic diagram of the path planning algorithm based on the CQRW-A*

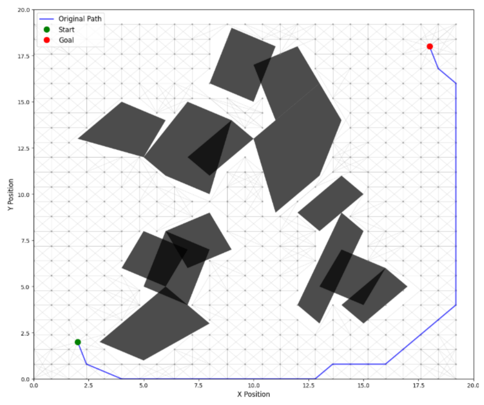


(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

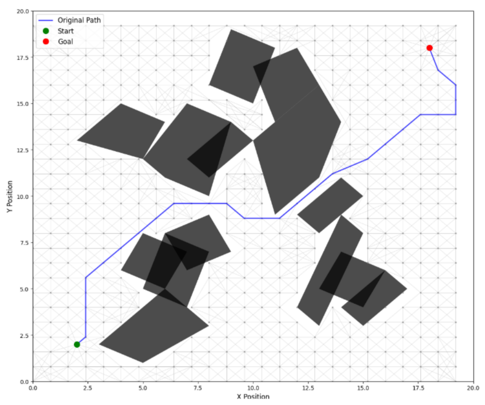
Fig. 6: Illustration of several pathfinding approaches using a 20×20 grid and an obstacle density of 30%.

the expense of somewhat longer computation times: 96.89s for QAOA-A*, 83.69s for CQRW-A*, and as little as 0.24s for classical-A*.

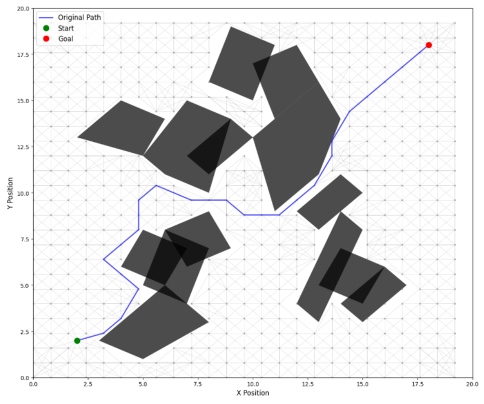
In Table 4, a 30×30 grid with 13 obstacles, which covers 17% of the grid, the quantum-enhanced methods outperform the classical one when it comes to path length. The QAOA-A* reaches the second-shortest path at 41.78 units, after the CQRW-A* at 40.70 units, and Classical-A* at 43.66 units. Even in less highly blocked situations, these results show that techniques influenced by quantum mechanics provide better path optimization. Nevertheless, there is a substantial processing expense associated



(a) Schematic diagram of the path planning algorithm based on the A*



(b) Schematic diagram of the path planning algorithm based on the CQRW-A*



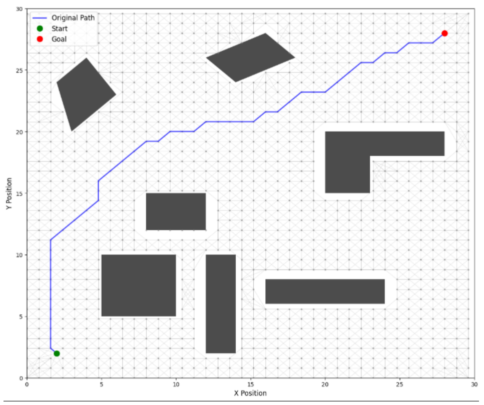
(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

Fig. 7: Illustration of several pathfinding approaches using a 20×20 grid and an obstacle density of 35%.

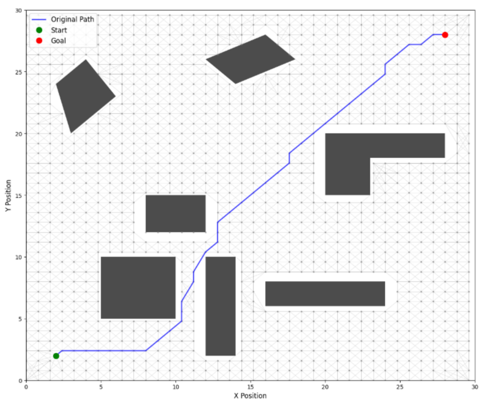
Table 4: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 17% with grid size 30×30.

Algorithm	**Actual Path Length**	**Time/s**
Classical-A*	43.66	0.28
QAOA-A*	41.78	178.02
CQRW-A*	**40.70**	197.60

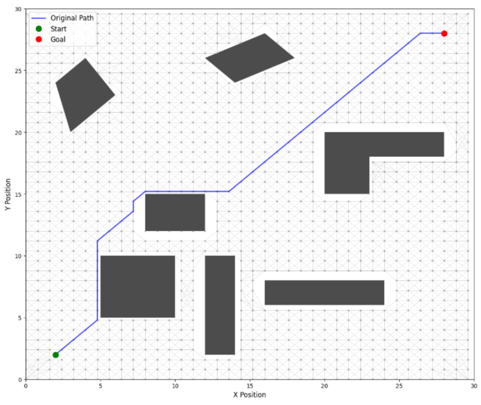
with this enhancement of path quality. Classical-A* finishes in 0.28s, whereas QAOA-A* takes 178.02s and CQRW-A* takes 197.60s.



(a) Schematic diagram of the path planning algorithm based on the A*



(b) Schematic diagram of the path planning algorithm based on the CQRW-A*



(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

Fig. 8: Illustration of several pathfinding approaches using a 30×30 grid and an obstacle density of 17%.

According to Table 5, the quantum-enhanced algorithms provide better results than the classical-A* algorithm when it comes to path length on a 30×30 grid with 30% obstacle density. In contrast to the QAOA and CQRW-A* algorithms, which produced shorter paths of 48.17 and 46.29 units, respectively, the conventional A* algorithm generated paths that averaged 50.00 units. However, these enhancements in path optimality are accompanied by significantly longer computational durations.

Table 5: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 30% with grid size 30×30.

Algorithm	**Actual Path Length**	**Time/s**
Classical-A*	50.00	1.91
QAOA-A*	48.17	186.86
CQRW-A*	**46.29**	211.80

The classical-A* requires 1.91s, while the QAOA and CQRW-A* approaches require 186.86s and 211.80s, respectively.

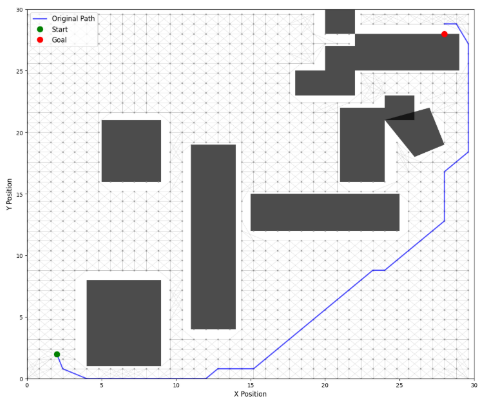
Table 6: Comparison of Classical-A*, QAOA-A*, and CQRW-A* for obstacle density of 35% with grid size 30×30.

Algorithm	**Actual Path Length**	**Time/s**
Classical-A*	48.18	0.92
QAOA-A*	42.10	169.69
CQRW-A*	**39.30**	515.52

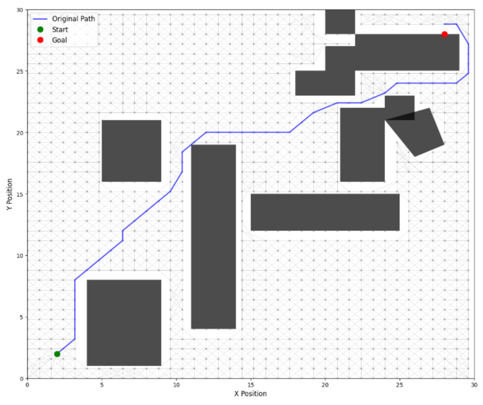
In Table 6, the results of comparing the performance of Classical-A*, QAOA-A*, and CQRW-A* algorithms on a 30×30 grid with a 35% obstacle density are shown. The Classical-A* algorithm produced a path length of 48.18 units and a computation time of 0.92s. The path length was lowered to 42.10s by the QAOA-A*, but the computation time increased to 169.69s. The CQRW-A* had the longest computation time at 515.52s, but the shortest path length at 39.30s.

5 Conclusion and Feature Work

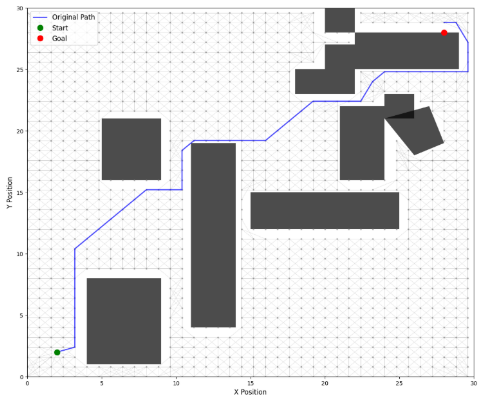
This paper presents a comprehensive evaluation of hybrid quantum-classical algorithms for path planning, analyzing the performance of Classical-A*, QAOA-A*, and CQRW-A* algorithms. Experiments were systematically designed on grids of dimensions 20x20 and 30x30, with obstacle densities varied at 17%, 30%, and 35%. The results indicate that quantum-inspired approaches consistently surpass Classical-A* in terms of path optimization. In a 20x20 grid with 17% obstacle density, Classical-A* produced a path length of 32.57 units in 0.073s. In contrast, QAOA-A* reduced the path length to 28.67 units but took 121.66s, while CQRW-A* achieved a further reduction to 26.09 units in 65.74s. In a more complex scenario involving a 30x30 grid with a 35% obstacle density, Classical-A* produced a path length of 48.18 units in



(a) Schematic diagram of the path planning algorithm based on the A*



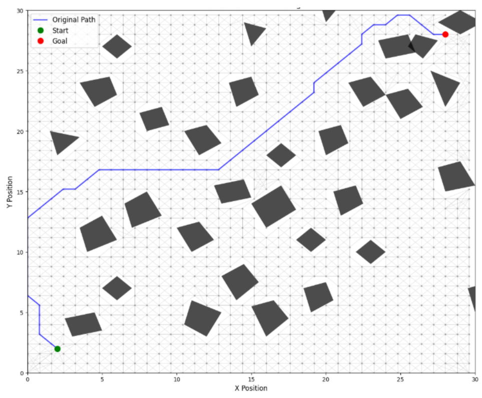
(b) Schematic diagram of the path planning algorithm based on the CQRW-A*



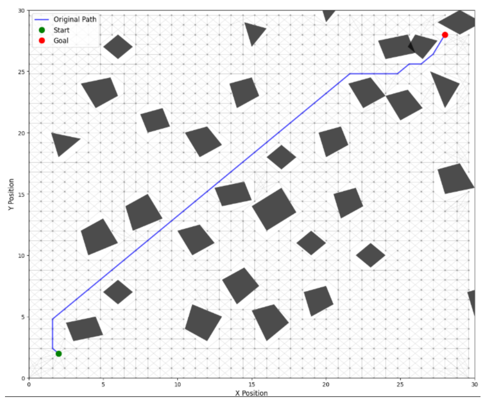
(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

Fig. 9: Illustration of several pathfinding approaches using a 30×30 grid and an obstacle density of 30%.

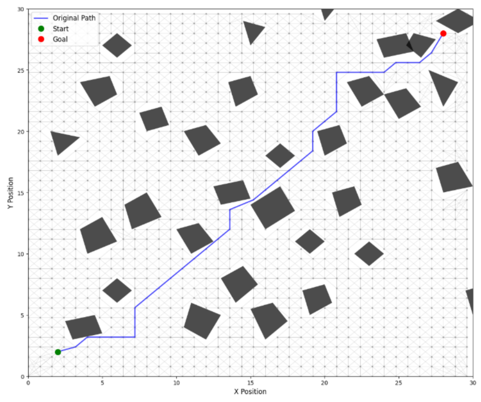
0.92s. In contrast, QAOA-A* achieved a reduced path length of 42.10 units, requiring 169.69s. CQRW-A* yielded the shortest path of 39.30 units, but necessitated the longest computation time of 515.52s. The findings demonstrate the capability of quantum-enhanced algorithms to achieve improved pathfinding performance in complex, obstacle-laden environments, though this comes with increased computational requirements. This suggests their appropriateness in scenarios where optimal pathfinding is valued more than computational efficiency.



(a) Schematic diagram of the path planning algorithm based on the A*



(b) Schematic diagram of the path planning algorithm based on the CQRW-A*



(c) Schematic diagram of the path planning algorithm based on the QAOA-A*

Fig. 10: Illustration of several pathfinding approaches using a 30×30 grid and an obstacle density of 35%.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
- Ethics approval and consent to participate

- Consent for publication
- Data availability
- Materials availability
- Code availability
- Author contribution

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

[1] Xu, L., Huang, K., Liu, J., Li, D., Chen, Y.F.: Intelligent planning of fire evacuation routes using an improved ant colony optimization algorithm. Journal of Building Engineering **61**, 105208 (2022)

[2] Costa, M.M., Silva, M.F.: A survey on path planning algorithms for mobile robots. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1–7 (2019). IEEE

[3] Karur, K., Sharma, N., Dharmatti, C., Siegel, J.E.: A survey of path planning algorithms for mobile robots. Vehicles **3**(3), 448–468 (2021)

[4] Mac, T.T., Copot, C., Tran, D.T., De Keyser, R.: Heuristic approaches in robot path planning: A survey. Robotics and Autonomous Systems **86**, 13–28 (2016)

[5] Childs, A.M.: Universal computation by quantum walk. Physical review letters **102**(18), 180501 (2009)

[6] Luo, Q., Wang, H., Zheng, Y., He, J.: Research on path planning of mobile robot based on improved ant colony algorithm. Neural Computing and Applications **32**, 1555–1566 (2020)

[7] Xu, X., Yu, X., Zhao, Y., Liu, C., Wu, X.: Global path planning of mobile robot based on improved genetic algorithm. Comput. Integr. Manuf. Syst **28**, 1659–1672 (2022)

[8] Li, C., Huang, X., Ding, J., Song, K., Lu, S.: Global path planning based on a bidirectional alternating search a* algorithm for mobile robots. Computers & Industrial Engineering **168**, 108123 (2022)

[9] Weng, L., Ji, Z., Xia, M., Wang, A.: Robot path planning based on improved multi-objective particle swarm. Journal of System Simulation **26**(12), 2892–2898 (2014)

[10] Zha, M., Wang, Z., Feng, J., Cao, X.: Unmanned vehicle route planning based on improved artificial potential field method. In: Journal of Physics: Conference Series, vol. 1453, p. 012059 (2020). IOP Publishing

[11] Gul, F., Mir, I., Abualigah, L., Sumari, P., Forestiero, A.: A consolidated review of path planning and optimization techniques: Technical perspectives and future directions. Electronics **10**(18), 2250 (2021)

[12] Chen, R., Hu, J., Xu, W.: An rrt-dijkstra-based path planning strategy for autonomous vehicles. Applied Sciences **12**(23), 11982 (2022)

[13] Roy, S., Zhang, Z.: Route planning for automatic indoor driving of smart cars. In: 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), pp. 743–750 (2020). IEEE

[14] Liu, Y., Vogiatzis, C., Yoshida, R., Morman, E.: Solving reward-collecting problems with uavs: A comparison of online optimization and q-learning. Journal of Intelligent & Robotic Systems **104**(2), 35 (2022)

[15] Hu, C., Xia, Y., Zhang, J.: Adaptive operator quantum-behaved pigeon-inspired optimization algorithm with application to uav path planning. Algorithms **12**(1), 3 (2018)

[16] Li, J., Xu, B., Yang, Y., Wu, H.: Quantum ant colony optimization algorithm for agvs path planning based on bloch coordinates of pheromones. Natural Computing **19**, 673–682 (2020)

[17] Karalekas, P.J., Tezak, N.A., Peterson, E.C., Ryan, C.A., Da Silva, M.P., Smith, R.S.: A quantum-classical cloud platform optimized for variational hybrid algorithms. Quantum Science and Technology **5**(2), 024003 (2020)

[18] Williams, C.P., Williams, C.P.: Quantum gates. Explorations in quantum computing, 51–122 (2011)

[19] Liu, J., Cai, Y., Cao, Y.: A robot path-planning method based on an improved genetic algorithm. Transactions of FAMENA **48**(3), 141–154 (2024)

[20] Li, Y., Liu, F.: Path planning algorithm for mobile robot based on improved ant colony algorithm. In: Journal of Physics: Conference Series, vol. 2083, p. 042033 (2021). IOP Publishing

[21] Jiachen, H., Li-hui, F.: Robot path planning based on improved dung beetle optimizer algorithm. Journal of the Brazilian Society of Mechanical Sciences and Engineering **46**(4), 235 (2024)

[22] Chen, R., Hu, J., Xu, W.: An rrt-dijkstra-based path planning strategy for autonomous vehicles. Applied Sciences **12**(23), 11982 (2022)

[23] Lim, J., Tsiotras, P.: A generalized a* algorithm for finding globally optimal paths in weighted colored graphs (2020) arXiv:2012.13057 [cs.RO]

[24] Liu, Y., Zhang, P., Ru, Y., Wu, D., Wang, S., Yin, N., Meng, F., Liu, Z.: A scheduling route planning algorithm based on the dynamic genetic algorithm with ant colony binary iterative optimization for unmanned aerial vehicle spraying in multiple tea fields. Frontiers in Plant Science **13**, 998962 (2022)

[25] Li, X., Li, Q., Zhang, J.: Research on global path planning of unmanned vehicles based on improved ant colony algorithm in the complex road environment. Measurement and Control **55**(9-10), 945–959 (2022)

[26] Khalidi, D., Gujarathi, D., Saha, I.: T: A heuristic search based path planning algorithm for temporal logic specifications. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 8476–8482 (2020). IEEE

[27] Rao, P.U., Speelman, F., Sodhi, B., Kinge, S.: A quantum computing approach for multi-robot coverage path planning. arXiv preprint arXiv:2407.08767 (2024)

[28] Chella, A., Gaglio, S., Pilato, G., Vella, F., Zammuto, S.: A quantum planner for robot motion. Mathematics **10**(14), 2475 (2022)

[29] Maity, R., Mishra, R., Pattnaik, P.K.: Analysis of path finding techniques for flying robots through intelligent decision-making algorithms in quantum inspired computing environment. Wireless Personal Communications **135**(3), 1561–1580 (2024)

[30] Fan, X., Wang, J., Wang, H., Yang, L., Xia, C.: Lqr trajectory tracking control of unmanned wheeled tractor based on improved quantum genetic algorithm. Machines **11**(1), 62 (2023)

[31] Mannone, M., Seidita, V., Chella, A.: Modeling and designing a robotic swarm: A quantum computing approach. Swarm and Evolutionary Computation **79**, 101297 (2023)

[32] Schuetz, M.J., Brubaker, J.K., Montagu, H., Dijk, Y., Klepsch, J., Ross, P., Luckow, A., Resende, M.G., Katzgraber, H.G.: Optimization of robot-trajectory planning with nature-inspired and hybrid quantum algorithms. Physical Review Applied **18**(5), 054045 (2022)

[33] Jiao, M.-h., Wei, H.-x., Zhang, B.-w., Jin, J.-q., Jia, Z.-q., Yan, J.-l.: Path planning of escort robot based on improved quantum particle swarm optimization. In: 2019 Chinese Control And Decision Conference (CCDC), pp. 3730–3735 (2019). IEEE

[34] Atchade-Adelomou, P., Alonso-Linaje, G., Albo-Canals, J., Casado-Fauli, D.: qrobot: A quantum computing approach in mobile robot order picking and batching problem solver optimization. Algorithms **14**(7), 194 (2021)

[35] Du, Z., Li, H.: Research on application of improved quantum optimization algorithm in path planning. Applied Sciences **14**(11), 4613 (2024)

[36] Lathrop, P., Boardman, B., Martínez, S.: Quantum search approaches to sampling-based motion planning. IEEE Access **11**, 89506–89519 (2023)

[37] Li, Y., Qin, S., Jing, L.: Research on flight trajectory optimization based on quantum genetic algorithm. In: Journal of Physics: Conference Series, vol. 1549, p. 022074 (2020). IOP Publishing

[38] Chella, A., Gaglio, S., Mannone, M., Pilato, G., Seidita, V., Vella, F., Zammuto, S.: Quantum planning for swarm robotics. Robotics and Autonomous Systems **161**, 104362 (2023)

[39] Lilja, J.: Quantum computing for automated vehicle trajectory planning (2023)

[40] Venegas-Andraca, S.E.: Quantum walks: a comprehensive review. Quantum Information Processing **11**(5), 1015–1106 (2012)

[41] Farhi, E., Gutmann, S.: Quantum computation and decision trees. Physical Review A **58**(2), 915 (1998)

[42] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)