Highlights

Quantum Planning for Swarm Robotics

Antonio Chella, Salvatore Gaglio, Maria Mannone, Giovanni Pilato, Valeria Seidita, Filippo Vella, Salvatore Zammuto

- Definition of a new algorithm, joining a Grover-based individual path planning with a quantum decision-making system for robotic swarms;
- Application of the new method to ant-foraging behavior, and comparison against classical models;
- Verification of the advantages of the methodology both in terms of time required and effectiveness of the approach.

Quantum Planning for Swarm Robotics

Antonio Chella[a,b], Salvatore Gaglio[a,b], Maria Mannone[a,c], Giovanni Pilato[b], Valeria Seidita[a], Filippo Vella[b], Salvatore Zammuto[a,b]

[a]*Department of Engineering, University of Palermo, Palermo, Italy*
[b]*Institute for High-Performance Computing and Networking (ICAR), National Research Council (CNR), Palermo, Italy*
[c]*ECLT and DAIS, Ca' Foscari University of Venice, Venezia, Italy*

Abstract

Computational resources of quantum computing can enhance robotic motion, decision making, and path planning. While the quantum paradigm is being applied to individual robots, its approach to swarms of simple and interacting robots remains largely unexplored. In this paper, we attempt to bridge the gap between swarm robotics and quantum computing, in the framework of a search and rescue mission. We focus on a decision-making and path-planning collective task. Thus, we present a quantum-based path-planning algorithm for a swarm of robots. Quantization enters position and reward information (measured as a robot's proximity to the target) and path-planning decisions. Pairwise information-exchange is modeled through a logic gate, implemented with a quantum circuit. Path planning draws upon Grover's search algorithm, implemented with another quantum circuit. Our case study involves a search and rescue scenario, inspired by ant-foraging behavior in nature, as an example of swarm intelligence. We show that our method outperforms two ant-behavior simulations, in NetLogo and Java, respectively, presenting a faster convergence to the target, represented here by the source of food. This study can shed light on future applications of quantum computing to swarm robotics.

Keywords: Grover search, quantum decision-making, foraging-ant behavior

Email addresses: antonio.chella@unipa.it (Antonio Chella), salvatore.gaglio@unipa.it (Salvatore Gaglio), mariacaterina.mannone@unipa.it (Maria Mannone), giovanni.pilato@icar.cnr.it (Giovanni Pilato), valeria.seidita@unipa.it (Valeria Seidita), filippo.vella@icar.cnr.it (Filippo Vella), salvatore.zammuto@community.unipa.it (Salvatore Zammuto)

1. Introduction

Be it an expedition in the wilderness, an overseas conference trip, or a nearby weekend vacation, planning a journey requires finding a *place* where to go and a *way* to reach it. Place location implies the use of geography and cartography. A way to get to that place needs the knowledge of feasible paths and transportation. Put it simply, we need a *point* in space and a *path* toward it. If multiple people are traveling independently heading toward a location but continuously exchanging information to reach it, we will have different paths converging—hopefully—to the goal.

If we have a group of interacting, simple robots, path planning can be faced computationally through pairwise information exchange and local decision-making. In the literature, there are approaches to multiple-robot groups [57, 53] path planning [39]. Other studies also discuss work frames exploiting traditional combinatorial logic optimization to improve the efficiency of quantum circuits, such as [24]. This is a key aspect also for quantum Boolean networks implementing the path-planning logic in [10]. In this article, we include the approach developed in [10] for a single robot, recurrently calling this methodology at each step to solve a path-planning problem for a swarm of robots.

Recently, the problem of path planning for a single robot has been faced with quantum algorithms [39]. In particular, a version of the Grover search algorithm has been proposed for robotic path-planning [10].

In fact, quantum computing [61, 54, 59] is more and more exploited in robotics [51, 19, 39, 13, 5, 59, 3, 37, 4, 29]. The use of quantum computing is mainly justified by its efficiency and computational power. Examples of quantum computing applied to robots include algorithms for flying robots and implementation for games [42], and vehicles and machines whose motion rules are based on entanglement, using IBM computational tools [48] and in particular Qiskit [46], which we used for our tests. Qiskit has been exploited in recent quantum-computing applications for decision-making systems [41, 64].

In this research, we focus on path planning for a multi-robot system. In particular, we consider a *swarm* of robots, where a complex behavior is emerging from simple, individual behaviors and decisions [56, 23, 15, 8, 66]. Robotic swarms can also help understand morphogenesis patterns [58] and precision

drug-delivery in human healthcare [14]. Artificial swarms are inspired by natural swarms. Flocking birds [27], schooling fish [12, 70], working termites [50] and foraging ants [6] are all examples of swarm intelligence [65, 16]. Ant behavior, including foraging and food transporting, has inspired studies on swarm-robotic modeling [35]. A wise use of constrains about environment and on-site individual behavior allows one to broaden artificial models of swarm foraging [26]. In addition, the quantum approach can be seen as a particular case of probability-based model, which had already been proved to be a valid alternative to precise single-robot formalization for swarm robotics [33].

A robotic swarm is characterized by redundancy (a loss of one robot does not affect the whole behavior), flexibility (no specialization required), and scalability (algorithm efficacy is size-invariant) [23]. Recent quantum approaches to robotic swarms include the definition of logic gates [34] to model robots' decision-making [44]. So far, there is still little work on quantum formalization of robotic swarms. While there are some pioneering quantum-approaches to robotics, the field is yet to be explored. Here, we try to bridge studies on robotic swarms, decision-making systems, and quantum-based theoretical and computational approaches. As described in [64], we can use tree diagrams to formalize robotic decision making, and apply quantum computing over these trees. However, this approach becomes quite expensive when it comes to robotic swarms. Also, an over-detailed individual approach may hide emerging swarm behavior, hindering its understanding.

In this paper, we design a quantum-based, swarm-robotic path-planning method. We propose a new algorithm, obtained as the union of logic-gate suggested positions of robots in a swarm at each time point, and the Grover approach to path planning. Thus, we consider *where* robots have to go, and *how*, which paths should they follow, also considering the presence of obstacles.

The paper is organized as follows. In Section 2, we present our background. We first summarize the logic-gate approach [44] in a two-dimensional case [45] (Subsection 2.1) and then the Grover-search path-planning approach [10] (Subsection 2.2). In Section 3, we present our new, mixed code. In Section 4, we propose an implementation with IBM quantum simulators. We present some outputs and we compare our method with foraging-ant simulations implemented in NetLogo [62, 63] and Java [36]. Actually, ant colonies inspired several studies on robotic path planning [11, 67]. For our comparisons, we take into account the algorithm complexity, the Fréchet distance

between trajectory polylines and the number of direction changes. As a matter of fact, there are studies on the preference for straight path in ant-motion trajectories [7].

2. Background: *Where?* and *How?*

In this section, we summarize two recent algorithms, before proposing a new one, built up as the union of them (Section 3), which constitutes a promising generalization of the path-planning approach with the quantum resources intervening in several steps. First, we describe a quantum gate-driven algorithm to find positions to be reached at each time point (the *Where?*) in an ant-foraging scenario. Second, we describe a Grover-inspired path planning, to find the path between a starting and a final position (the *How?*). In Section 3, we join these two approaches, inserting the path-planning between the steps of the swarm-searching code. The result is a new algorithm which can be implemented to drive real swarms of robots in a search and rescue mission.

2.1. Where? *Quantum gate-driven search*

Let us consider a search and rescue task to be accomplished by a swarm of robots. This is typically related to an exploration task [32]. For a biological comparison, we can think of a foraging-ant scenario. We can imagine a cluster of robo-ants starting, initially, from close positions, as exiting from their nest, and looking for the food source. Then, they shuffle their positions, exchanging messages about their perception of target proximity—e.g., through their distance-assessment as smell of the food source. We can assume a broadcast communication, where each robot of the swarm sends messages to all the other robots. Each robo-ant will thus compare the information coming from their peers, and follow the most successful robot—the robot assessing to be closer to the food source, the target.

We can model a pairwise interaction, where each robot is getting information and pondering to which point should it be heading according to position precision and *reward* assessment of their peers and of itself. We call *reward* here the perception of target proximity for each robot. If a robot declares "unsuccess" in target search, then the other robots should examine different parts of the space. If a robot declares "success" with some degree of precision, but with a great incertitude on position location, then the other robots cannot follow its suggestion—at least, they can, with some incertitude. If a

robot declares "success" with a good degree of certitude, and position with some precision as well, then that position should be reached by its robotic peers.

The information on position and reward can be coded through quantum state superposition. We can set "failure" as 0 and "success" as 1, with respective probability amplitudes, for the i-th robot R_i, γ_i and δ_i. Similarly, in a squared and unitary arena, we can indicate as 0 and 1 the left and right extremes of a segment x. Numbers α_i^x, β_i^x will thus describe the probability amplitudes to be in the left or in the right point, respectively, for the i-th robot. Same idea for the y-axis, see eq. (1) and (2).

$$\begin{aligned} |q_0\rangle &= \alpha_1^x|0\rangle + \beta_1^x|1\rangle, \\ |q_1\rangle &= \alpha_1^y|0\rangle + \beta_1^y|1\rangle, \\ |q_2\rangle &= \gamma|0\rangle + \delta|1\rangle \end{aligned} \tag{1}$$

$$\begin{aligned} &|\alpha_i^j|^2 + |\beta_i^j|^2 = 1, \\ &|\gamma_i|^2 + |\delta_i|^2 = 1, \\ &(i = 0, ..., N-1;\ j = x, y) \end{aligned} \tag{2}$$

Pairwise interaction terms can be modeled upon a logic gate. In the 1-dimensional case, with movements along either the x- or y-axis, we can build up a reversible logic gate [44], a quantum version of an XNOR. In fact, a "failure" of a robot along one extreme of the $[0, 1]$ segment would suggest a search of the robot in the direction of the other extreme. In a 2-dimensional arena, a failure in a quadrant requires a search through the three other quadrants. Therefore, for the same output of the gate, there exists more than one input [45]. Because of this indeterminacy, the gate is non-reversible, see Table 1. The Table contains the following information: $|q_0\rangle$ is the x-position of R_i at t_1; $|q_1\rangle$ is the y-position of R_i at t_1; $|q_2\rangle$ is the reward of R_i at t_1; $|q_3\rangle$ is the suggested x-position of R_j at t_2, and $|q_4\rangle$ is the suggested y-position of R_j at t_2. The information concerning the reward of R_i, that is, $|q_2\rangle$, is copy-pasted to the right side of the table to have the same number of inputs and outputs. The proposed logic gate can be implemented through the quantum circuit shown in Figure 1, where the input state used as example is $|q_0\rangle = 0$, $|q_1\rangle = 1$, and $|q_2\rangle = 1$. If R_i is at 1 ($|q_0\rangle$, $|q_1\rangle$=1) and it is successful ($|q_2\rangle = 1$), then the suggested positions $|q_3\rangle$, $|q_4\rangle$ for R_j (the probability amplitudes to be in 1 on x and y) are set to 1 by the Toffoli gates. If R_i is at 0 along x, y and it is successful, the Toffoli gates are not

activated, and R_j will explore around $0, 0$. The Toffoli gates are not activated also if R_i is not successful (reward 0). If this is the case, R_j has to explore other regions of space; thus we have an indeterminacy of the positions along x and y to be reached, and the peak of the wavefunction is on the center of the $[0, 1] \times [0, 1]$ square where the robots move. For this reason, we use two Hadamard gates, one for each dimension. In fact, if $|q_2\rangle = 0$, the NOT gate switches 0 to 1, activating the Hadamard gates. Conversely, if $|q_2\rangle = 1$, the NOT gate switches it to 0, and the Hadamard gates are not activated. The second NOT gate sets back $|q_2\rangle$ to the input state.

Table 1: Truth table for two robots R_i, R_j on the xy-plane

q_0	q_1	q_2	q_3	q_4	q_2
x-pos R_i	y-pos R_i	reward R_i	x-pos R_j	y-pos R_j	reward R_i
0	0	0	0/1	0/1	0
0	0	1	0	0	1
0	1	0	0/1	0/1	0
0	1	1	0	1	1
1	1	1	1	1	1
1	0	0	0/1	0/1	0
1	1	0	0/1	0/1	0
1	0	1	1	0	1

Algorithm 1 quantum-gate driven search

```
initialize robots' positions as state superpositions
if all rewards are below a certain threshold then
    for each robot do
        randomly reshuffle positions
    end for
end if
find the robot with the highest reward and let it enter the circuit
find the new suggested position through the circuit
for all robots do
    evaluate the new rewards
end for
```

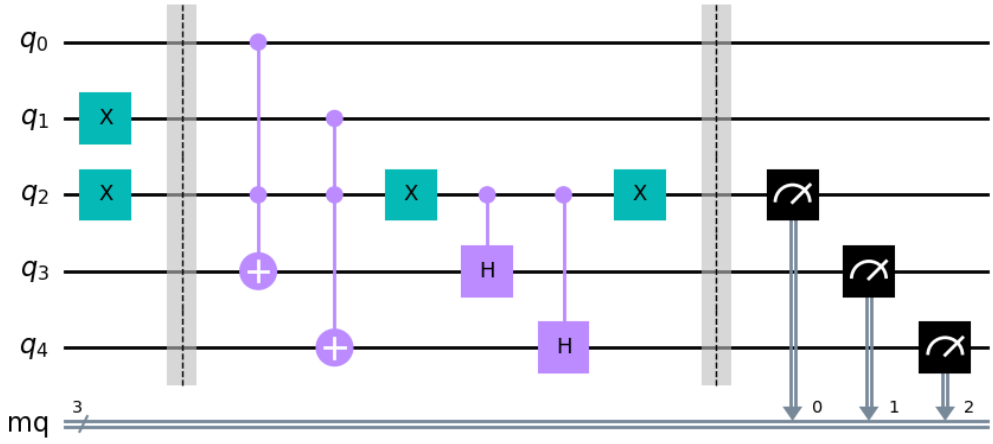


Figure 1: Quantum circuit implementing the logic gate of Table 1.

The circuit can be embedded in a search and rescue algorithm for a robotic swarm, modeling pairwise interactions between robots of the swarm. Given N robots, there are $N!$ interaction terms, so the circuit should be ran $N!$ times. To reduce computational time and increase efficiency, instead of computing all terms, we can just select only the robot with the highest reward and let it enter the gate. The circuit will thus be computed just once. The output of the gate will suggest the x, y-positions for all the robots at the following time point. The pseudocode of this approach is proposed in Algorithm 1 [45], whose lines 7 and 8 concern the discussed quantum circuit.

2.2. How? *Grover-search path planning*

In both ideal and practical scenarios, the swarm behavior is also affected by the environment the robots exist in. Being aware of the surroundings of the swarm entities means determining the optimal shortest path from a starting point to the evaluated destination, while at the same time taking into account the presence of impediments or forbidden areas of the map. Finding such optimal path is historically a burdensome endeavor and therefore an ideal candidate for quantum computation.

Here, we approach quantum path planning with the methodology described in [10], where this task is interpreted as a search problem and a Grover procedure is applied to solve it. With E and S being respectively the

number of elements of the search space and the number of solutions to the problem, the quantum search algorithm is able to achieve a $O(\sqrt{E/S})$ time complexity [49], a quadratic speed-up over the classical approach. Depending on the structure of the tree that is searched on, the improvement over traditional search may be slightly less than quadratic, and if certain conditions are met, it may even perform worse than randomly guessing elements from the search space. The study of [10] also discusses the compatibility between the Grover procedure and the path-planning problem, showing that a speed-up over classical techniques is always present, and that also, as the map instances get larger, the quantum algorithm asymptotically behaves as one with the maximum possible speed-up to achieve with respect to the linear complexity of the classical unstructured search counterpart.

Setting up a Grover procedure means developing a suitable encoding of the problem and individually constructing the components of the Grover operator, namely the *oracle* U_ω and the *diffuser* U_ψ, to implement the amplitude amplification technique that evolves the initial uniform superposition of the search space $|\psi\rangle$ into a solution state $|\omega\rangle$, where any of the solutions can be measured with near-certainty.

The encoding stage is arguably the most crucial one in terms of design choices, since both the complexity of the system and its use of resources strictly depend from it. In this context, the path-planning problem is formulated as finding the sequence of actions (i.e., the *moves*) the robot should carry out to reach the goal. Particularly, the path that we are trying to determine is that of a robot within a $n \times n$ grid map with obstacles, from the starting position $\mathbf{r}_{start}$ to a goal cell $\mathbf{g}$. This type of environment can be interpreted as a taxicab geometry [47], with the single grid cells corresponding to the taxicab points and their adjacency being modeled by the connections between neighbor nodes. In this world, the taxicab metric to quantify the distance between two points, which in our case are vectors in a 2-dimensional vector space with fixed Cartesian coordinate system, is the L^1 Manhattan distance, which can be directly used to evaluate the overall number of moves that need to be carried out in order to trace the desired path.

The starting environment is therefore masked with a square lattice, which is interpreted as we would do for a $n \times n$ matrix, where each cell is labeled the same way as a matrix element m_{ij} with $0 \leq i, j \leq n-1$; in this regard, m_{00} and $m_{n-1\,n-1}$ are associated respectively with the top-leftmost and bottom-rightmost cells. These positions are encoded with $\eta_{MAP}(n) = \lceil \log_2 n^2 \rceil$ qubits. Similarly, in regard to movement encoding, the taxicab interpre-

tation of the grid maps allows only for vertical and horizontal movements, and not diagonal ones; it follows that, in its most general form, the information about the 4 possible movement directions, up, down, left, and right can be encoded with a superposition of a total of 2 qubits for each action. In the sample case of a 2×2 map, movement can also be viewed as either a clockwise or an anti-clockwise rotation, requiring just 1 qubit to encode the binary choice of rotation direction.

One such encoding crucially affects the composition of the corresponding quantum circuit that solves the path-planning problem in the sense that most of the operators that implement the algorithm, while having an overall defined behavior shared across all of the differently-coded instances, have an inner gates' structure that is highly depending on the way the system's state is fed to them. This also goes for the final measurement, where the resulting classical bitstring, in this case, needs to be interpreted as a sequence of pairs corresponding to the sequence of moves that achieve the desired planned path.

That being said, the quantum circuit that achieves a Grover search is constructed by preparing the inputs, applying the Grover operator $G = U_\psi U_\omega$ for $O(\sqrt{N/S})$ times, and then measuring the designated result registers.

On one hand, the diffusion operator U_ψ always takes the form $U_\psi = 2\left|\psi\right\rangle\left\langle\psi\right| - I$, which can be implemented on a quantum computer considering that

$$U_\psi = H^{\otimes k} X^{\otimes k} (MCZ) X^{\otimes k} H^{\otimes k} \tag{3}$$

where H is the Hadamard gate, X the Pauli-X gate and MCZ corresponds to the multi-controlled-Z operation; k is the number of qubits that encode the search space. This is convenient for us, since this definition is problem-independent, so that the quantum circuit realization of the diffuser U_ψ is always that of expression (3).

On the contrary, considering that the oracle U_ω tests solution states on the result of a processing of the input register (i.e., we search on the moves, but check if the target has been reached on the positions those moves result into) some additional considerations must be embedded, and designing a suitable Grover oracle becomes the main concern of solving the path-planning problem.

Our U_ω operator is broken down into submodules, the M and T blocks: the former performs a *quantum move*, that is, it takes as input the super-

position of the movement and current position registers, and outputs the combination of all of the valid[1] positions that can be reached from the starting point. The latter tests if the output state of the M operator contains the target cell and marks the solution by inverting the phase of the oracle qubit. The presence of obstacles is wired directly into each of the M operators, where a quantum move toward an obstacle results in the output of the same position it was fed as input. As an example, the Grover oracle for a 2×2 map instance, where the starting and target positions are encoded respectively in the $[0, 0]$ and $[1, 1]$ cells, has the structure of Figure 2.

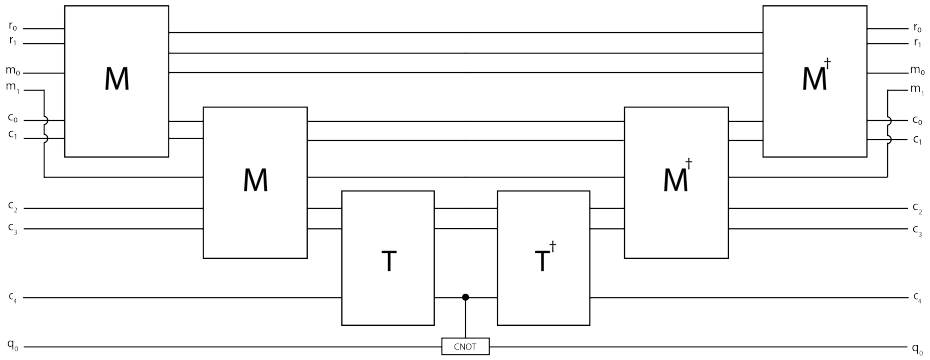


Figure 2: Block scheme of the oracle for the 2×2 case where start and target cells lie on the opposite corners of the map, as depicted in [10].

The qubits of the structure are organized as follows:

- the $|r\rangle$ register has $\eta_{MAP}(2) = \lceil \log_2 2^2 \rceil = 2$ qubits that encode the position of the robot;
- the $|m\rangle$ register is that of moves, with size $m = d_1(\mathbf{r}_{start}, \mathbf{g}) = 2$; m also corresponds to the number of cascaded M operators in the circuit before the test on solutions is performed with a T block;
- $|c\rangle$ is the ensemble of auxiliary qubits of the oracle workspace, needed to achieve reversibility of the movement computations;
- $|q\rangle$ is the oracle qubit used to mark the solutions of the problem.

[1] A non-valid move is one that either brings the robot into an obstacle or makes it jump from an edge of the map to another.

After the required M and T operators have been applied, the information about the solution states is kicked back to the qubits that encode the search space through an uncomputing stage. Indeed, the elaborations of the M and T operators are mirrored with respect to the controlled operation on the oracle qubit, in the sense that each of the individual operators is inverted (which is possible to do since they have been constructed with reversibility in mind in the first place) and applied in reverse order. This oracle structure, together with the diffuser U_ψ, which is appended to the $|m\rangle$ register, completes the Grover operator G that performs one iteration of the quantum search algorithm.

The following Section describes the newly proposed quantum swarm planning procedure, where the single-robot motion planning methodology we just outlined (summarized in the pseudocode of Algorithm 2) is embedded within the more complex scenario of a searching and rescuing robotic swarm: for each single robot in the swarm a path is quantumly planned on its (current) starting position and the evaluated target cell computed by the swarm algorithm. As a result, when dealing with swarms, the quantum circuit to plan the path of each entity is dynamically constructed according to the destination, the presence of obstacles and the number of moves of the desired minimum-length path.

Algorithm 2 quantum path planning

Inputs: map dimension n, initial and target positions of one robot, obstacles' positions
Preprocessing: compute the number of iterations of the procedure and the number of moves m
Procedure:

initialize $|r\rangle$ and $|m\rangle$ in a uniform superposition with $H^{\otimes(\eta_{MAP}(n)+m)}$ and $|q\rangle$ as $|-\rangle$
for each iteration **do**
    for move **in** m **do**
        cascade an M operator to the circuit
    end for
    apply a T operator on the output of the final M
    perform the controlled operation on $|q\rangle$
    apply the inverse $T^{\dagger}$
    for move **in** m **do**
        cascade an $M^{\dagger}$
    end for
    append the diffuser U_{ψ}
end for
measure the output registers

3. *Where* + *How*: The new, mixed code

Having introduced the individual components of the approaches of [44, 45, 10], we now describe how the methodology for modeling the complete behavior of the robotic swarm was assembled. As previously mentioned, we take advantage of the swarm algorithm to compute the locations of the robots according to their global target and the interactions with each other, subsequently evaluating, for each of them, the optimal trajectory to reach such destinations, according to the path-planning algorithm. The full procedure is able to exhaustively model any complex system with swarm agents that operates within a 2-dimensional world at the significantly improved speeds of quantum computation.

In the interest of merging the discrete taxicab world described in Section 2.2 with the continuous world the swarm algorithm of 2.1 is designed for, a discretization phase of the frame of work is required. Masking the initial map with a square mesh means both interpreting the environment as a grid and locating the position of the robots, target, and obstacles with the corresponding cells they fit into. Grid encoding has been recently used to model robotic-swarm behavior in probabilistic terms [32].

While an arbitrarily-sized single-layer lattice is sufficient to achieve such a discrete representation of the environment, increasing the n dimension of the grid can appreciably grow the overall spatial complexity and qubit utilization of the quantum path-planning algorithm. To contrast this resource expenditure, we considered grids whose individual cells are hierarchically partitioned (with at most the same dimension as the outermost grid) and interpreted as smaller map portions themselves, with their own sub-cells that can in turn be further decomposed, and so on. The main advantage of this strategy consists in the fact that there is not a single, large quantum planning circuit anymore, rather, a number (depending on the amount of subdivision layers) of smaller circuits now cooperate for the determination of a useful path of the robots. These reduced circuits are easier to run as they both take up fewer qubits and have a shallower time evolution, compensating for the intermediate size of quantum processors and lack of stable error correcting codes of the state-of-the-art current technologies of the NISQ era. Generally, the space partitioning can be varied in accordance with the desired resolution and precision that a specific problem is required to have. In this sense, both the maps at each layer can be square grids of any dimension, and they also need not be the same; however, in the case that two or more subdivision

layers shared the same partitioning dimension, the same circuit structure can be applied for all such cases, sparing the effort of redefining multiple different sub-routines for each of the hierarchies. The hierarchical nature of the strategy also allows for a depth of micro-subdivision that is also arbitrary, and is again left as a design choice. Practically, all of such parameters should be selected in regard to considering the smallest cells (those at the deepest layer) as units of space, that is, of the same dimension order of the static entities of the problem, namely the target and the obstacles. An obstacle that spans over more than one such cells is translated to a set of unit obstacles on the respective cells it covers.

In this work, we developed our approach on the assumption that the meshed representation of the map is known. However, it should be noted that, in the case that this information was not given beforehand, getting to it can be done classically and in a very efficient way through the use of quadtrees [20] and the related algorithms for fast single-point location on planar maps [18, 25].

We can contextualize this approach with a use case that takes into consideration a tessellation of the robots' environment with a 2×2 grid, where each of the cells is referred to as a *macro-cell* (Figure 3a). Each macro-cell is consequently split into a smaller 2×2 map, whose squares are *micro-cells*, as Figure 3b shows.

With the grid being laid out, the elements of the environment are placed inside it, with their locations, continuous, being associated with the discrete centroids of the micro- and macro-cells they belong to. These centroids correspond to the binary matrix indexing referenced in Section 2.2: for each cell, a tuple containing the indices of the micro- and macro-cells is considered; as an example, a robot located in the top-leftmost corner of the map of Figure 3b is associated with the 2-tuple $([0, 0], [0, 0])$, where the indices respectively refer to those of the micro- and macro-cell of the robot. This allows us to both deal with the swarm behavior in a continuous way, while at the same time having a discrete representation that can be directly exploited by the quantum path-planning algorithm.

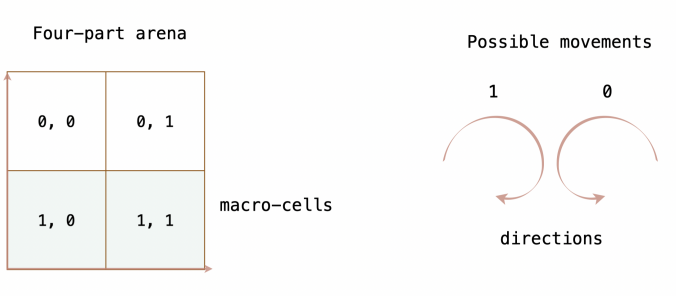
A path within such partitioned map is therefore interpreted as a combination of micro- and macro-paths, each computed through the micro- and macro-displacements between the starting cells and the new evaluated positions of the swarm algorithm. As a consequence, four distinct path types arise from the problem instance of Figure 3b:

a. macro: the robot changes macro-cell but stays in the same micro-cell;

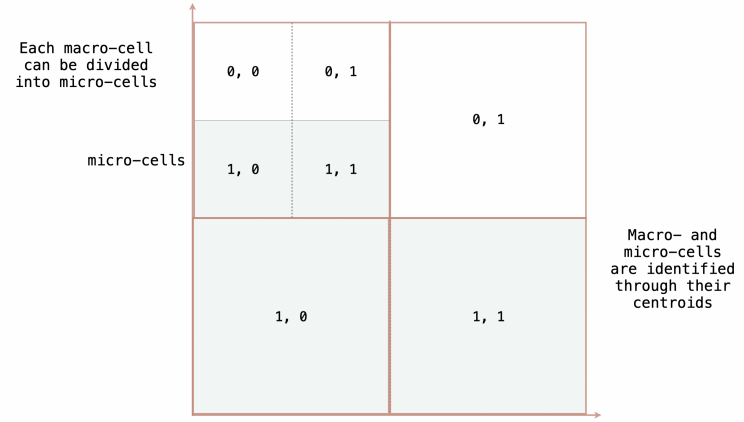
b. micro: the robot changes micro-cell but stays in the same macro-cell;

c. macro and micro: the robot changes both macro and micro cells;

d. if the new position is in the same micro- and macro-cell, nothing happens.



(a) Binary encoding of macro-cells and movement directions.



(b) Organization of micro- and macro-cells.

Figure 3: Schematic representation of cells and robotic movements.

Figure 4 shows a displacement of type *a* where the computed location ends up in a cell that has the same micro-indices of the starting one, but different macro ones. We apply the quantum path-planning algorithm for each micro- and/or macro-paths; in the *c* case, where both classes of trajectories need to be evaluated, the global macro-path is computed first, and then the route is narrowed down with the appropriate micro-path.

That being said, the complete joint algorithm is composed by taking the swarm logic of Algorithm 1 and inserting the path planning stage each time a robot changes position, whether it happens through a randomized reshuffling, or through the evaluation of the quantum circuit of 2.1.

It should also be noted that a real, practical scenario would also need to take into account the possibility for dynamic obstacles. The methodology presented in this article is necessarily a quantum-classical hybrid one in nature, i.e., we make quantum computation do the heavy lifting, while keeping the sensor processing and teleo-reactivity classical. Specifically, the physical robots read data from the environment, process it, and subsequently build the appropriate quantum circuits for the swarm behavior and path-planning tasks. That being said, non-static obstacles can be essentially dealt with in two ways, corresponding to the two main possible situations that may occur:

- a fixed-position obstacle suddenly appears in the path of a robot as it is traveling it: here, teleo-reactivity is achieved by simply halting the course of action, stopping the robot in order to avoid collision, and constructing a new procedure initialized with the current system's state and a map representation with the updated impediments;

- a moving obstacle is detected: in this case, the classical intra-quantum sensory reading and processing will also be required to compute an estimate of the future positions of the robot, and accordingly build the swarm planning quantum circuits by considering an obstacle in each of the cells that the expected trajectory it computed to be in.

The full procedure starts by preparing the grid mesh over the environment and locating, through the corresponding centroids, the starting position of the swarm of N robots R_i with $i = 0, ..., N-1$ (initially, the robots take off from the same micro-cell, that is, $\mathbf{r}_{start}$ is equal for all R_i), together with that of obstacles O_j with $j = 0, ..., o-1$ (where o is the total number of micro-obstacles in the map) and target T, associated with the goal cell $\mathbf{g}$. Then, we initialize the robots' rewards δ_i, checking if any of them is above the $\theta = 0.8$ threshold, which assimilates to a "good enough" proximity to the goal; generally, at time t_0, these rewards should be approximately all equal to each other, and below the threshold.

Once the initialization stage is complete, on the assumption that none of the rewards is above θ, a randomized reshuffling scatters the robots over the map; to improve the convergence of the algorithm, the reshuffling is

performed in a goal-oriented manner, in the sense that it is evaluated over the sub-rectangle pinpointed by the micro-cells of the target $\mathbf{g}$ and the robot with lowest reward $\mathbf{r}_{farthest}$ (i.e., the farthest from the goal, with the idea of maximum coverage of the working map to restrict the occurrence of local minima). For each of the new evaluated positions, a path-planning procedure is applied, with the exception of those that result into an obstacle micro-cell, which are discarded and recomputed; this is done to improve efficiency and reduce resources utilization by avoiding the construction and execution of the quantum circuits that plan a path toward a forbidden location.

The rewards get updated, and the α, β, γ, and δ parameters of the $R_{closest}$ robot, the one that is the closest to the target, are extracted and fed to the quantum circuit of Figure 1. The output of the circuit is filtered on a subset of the states that have been measured with the highest probabilities and is used to weigh the outcome positions of the robots with non-maximum rewards. Those robots reach the new location following the path evaluated by the procedure of 2.2.

Ultimately, the δ_i are updated once again and the swarm is routed toward the position of the robot with the new highest reward, the new $R_{closest}$. In the case that $\delta_i > \theta$ for none of the robots, another informed reshuffling is executed, and the algorithm undergoes another iteration. These considerations are summarized by Algorithm 3.

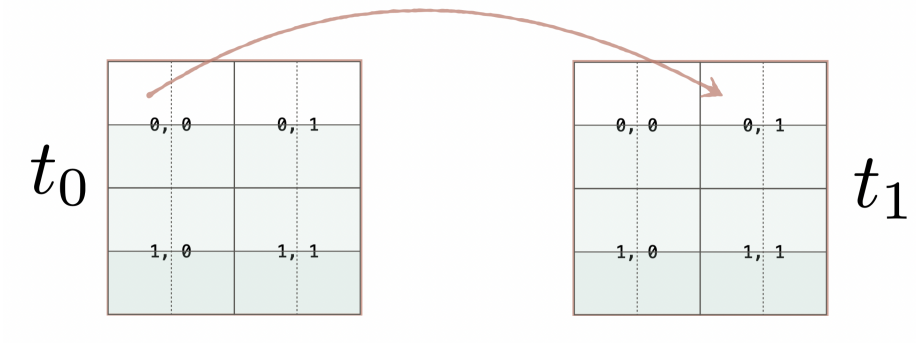


Figure 4: Example of macro-only displacement.

Algorithm 3 quantum swarm planning

Preprocessing: arrange the space partitioning of the map and locate the problem's entities with the centroids of the cells they belong to
Procedure:

initialize the rewards δ_i for each robot R_i with $i = 1, ..., N$
while $\delta_i < \theta$ for every robot **do**
    do
        randomly reshuffle each robot's position on the sub-rectangle spanned by $\mathbf{r}_{farthest}$ and $\mathbf{g}$
    while every position does not end up in a cell associated with any of the obstacles O_j
    for each R_i and its new position **do**
        apply the path-planning procedure from the current position to the evaluated destination
        update δ_i
    end for
    $R_{closest} \longleftarrow \max_{\delta} R_i$
    extract the parameters α, β, γ, and δ from $R_{closest}$ and input them to the swarm algorithm
    filter the p most probable states from the swarm algorithm output
    compute the weighted position outcome on the p states
    for each $R_i \neq R_{closest}$ **do**
        travel to the evaluated position outcome through the appropriate planned path
        update δ_i
    end for
    parse the new $R_{closest}$
    for each $R_i \neq R_{closest}$ **do**
        reach $R_{closest}$ through the appropriate planned path
        update δ_i
    end for
end while

4. Results

In this Section, we first evaluate our algorithm, investigating its scalability for swarms of different sizes (paragraph 4.1). Then, we compare our method against two foraging-ant simulations (paragraph 4.2), quantitatively analyzing the findings. In our study, we focus on a theoretical approach and on its simulated implementation. An experimental validation with physical robots, at this stage, would not add any information to the core of our study. A physical implementation will be the object of future research, and it will involve conditions such as hardware choice, message exchanging protocols, communication strategies. A simulation in Webots, with the e-pucks, provides an idea of how an experimental setup could look like.

4.1. Evaluation of the proposed algorithm

In this paragraph we present some outcomes of the Python code (available at [69]) that implements the full procedure of Algorithm 3; we build our quantum circuits with IBM's Qiskit SDK [46] and execute them through the *Aer simulator*. First, we visualize the different stages of the procedure and the scheme of movement across cells for a swarm composed of 2 robots. Then, a 10-robot swarm is considered in order to show the improved convergence speed of higher-numbered swarms. Ultimately, we exhibit a situation where a single iteration of the algorithm is not sufficient to reach an adequate proximity to the target, so that one more cycle of reshuffling and gate is required. For the sake of comparison, we keep the map partitioning, obstacle, and position targets the same for each of these three scenarios, in particular:

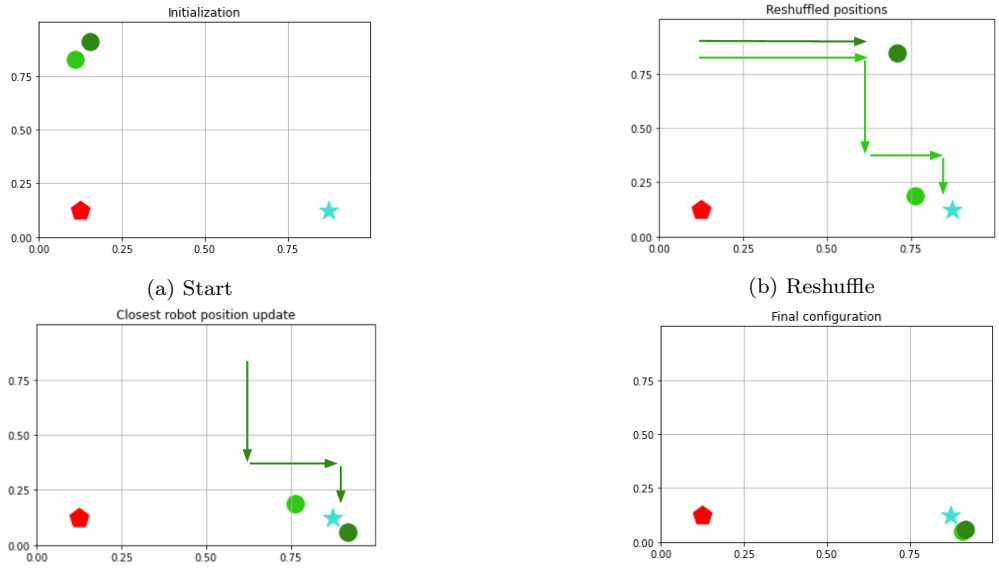
- the environment is tessellated as of Figure 3b of Section 3;
- the swarm starts in the top-leftmost cell, that associated with the centroids $([0,0],[0,0])$;
- a single obstacle is present in the cell $([1,0],[1,0])$;
- the target is located in $([1,1],[1,1])$.

Figure 5 portrays the procedure applied to a 2-robot swarm, with a focus on the evaluated paths by the quantum path-planning algorithm. Figure 5a displays the state of the system at time t_0, with the swarm agents (green circles, the higher the i index of the robot R_i, the darker the color gradient is), obstacle (red pentagon), and target (turquoise star). In Figure 5b the

positions are randomly reshuffled in the goal-oriented manner; note that here, in view of the fact that the swarm and the target have been initialized in opposite corners of the map, the sub-rectangle defined by the farthest robot and the target of the informed reshuffling described in Section 3 is indeed the entire map. The arrows show the paths evaluated by the circuits constructed with Algorithm 2; their color is of the same green gradient of the robot they refer to: R_0 (light green) reached its destination with a macro-micro path type, which the quantum path-planning algorithm measured as the $|11\rangle$ state for both the macro- and micro-paths, meaning that the robot has to perform two macro and two micro clockwise rotations to reach its destination. As for R_1 (in dark green), just one macro clockwise rotation is needed, since the corresponding displacement was of the macro kind (the same as Figure 4) and the measured output of the circuit was the $|1\rangle$ state. After updating the rewards, R_0 emerges as the closest to the target, and its attributes are fed into the swarm algorithm that computes the new position of the rest of the swarm, that is, R_1. This is depicted in Figure 5c, together with the worked out path from the previous location, which is of the macro-micro type, with the moves encoded by the $|0\rangle$ state for the macro trajectory and $|11\rangle$ for the micro one. After the gate, the closest robot to the goal is now R_1 and therefore R_0 reaches it. Since this last manoeuvre occurs within the same micro-cell (path of type d), no path planning is required, and the robot travels to the end location in a straight line. The final configuration is that of Figure 5d.

While the code output shown in Figure 5 is a good example of the overall behavior of the proposed algorithm, we should take into account the fact that this was indeed a "lucky" run, since the first randomized reshuffling got immediately the R_0 robot nearby the target. With 2 robots, this is not trivial at all, and more often than not, with the randomized positions the robots do not achieve such proximity.

When considering a bigger swarm, for example of 10 robots, the probability of getting shuffled on neighboring cells of the goal is much higher, for the reason that more robots that are uniformly distributed ensure a better coverage of the working environment. The case for 10 robots is represented in Figure 6; to avoid cluttering the plots we omit the visualization of the trajectories evaluated by the path-planning algorithm, which, as always, is executed each time a robot requires to reach a new position with a non-zero displacement for macro- and/or micro-cells. The steps of the procedure are the exact same as the previous example, with the difference that here, after



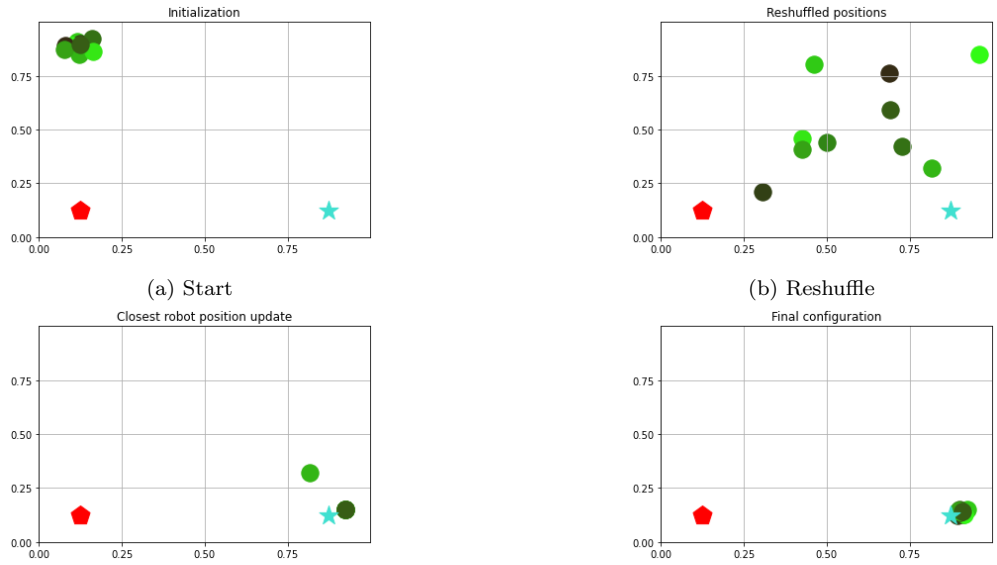
(a) Start (b) Reshuffle

(c) After the gate (d) Target reached!

Figure 5: An example of run of the code with 2 robots. The swarm is initialized within a single micro-cell (a), then, each of the robots' position is randomly reshuffled over the map (b), so that the newly computed highest-reward robot is fed to the swarm algorithm that outputs the an updated position for the rest of the swarm (c). Eventually, the robots reunite toward the new $R_{closest}$ (d). An approximate rendition of the robotic paths is shown by our simulation in Webots.

the initial reshuffling (Figure 6b), the swarm agents are more evenly spread out over the map and more than one robot happens to be in the vicinity of the target. Figure 6c highlights the fact that the swarm algorithm, executed on $R_{closest}$ (in lighter green), made the remaining robots travel toward the same computed location; as a consequence, the points that represent the peaks of probability wavefunctions to find them in the target are overlapped. The endmost stage (Figure 6d) shows the complete swarm that reached the final location, where a negligible fluctuation on the robots' position has been artificially inserted to better display that every entity has in fact gotten to its destination.

Practical cases often distinguish themselves with a limited amount of resources that can be exploited to solve that particular problem. In our context, it may not be always possible to arbitrarily increase the size of the swarm to achieve fast convergence to the target, and working with just a few robots may not get us as close to the target as we desire. This is why, in the



(a) Start

(b) Reshuffle

(c) After the gate

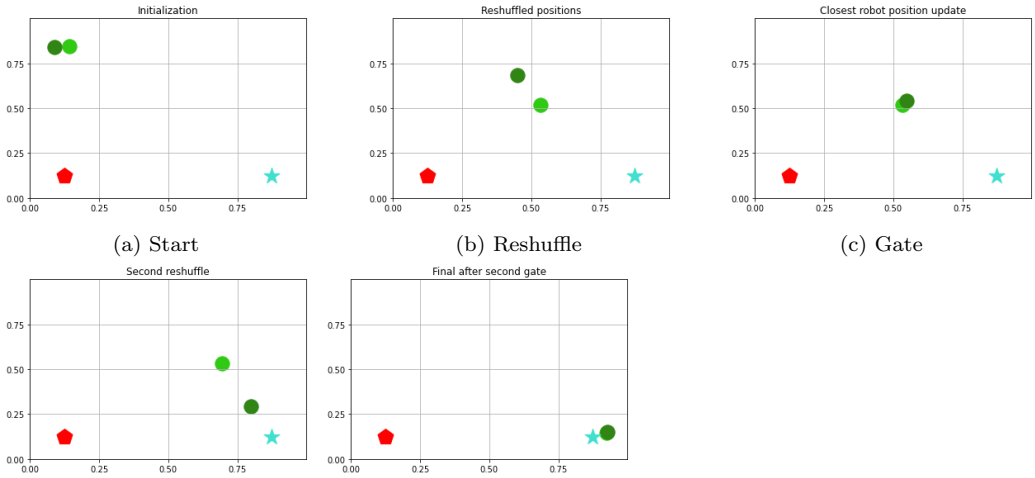
(d) Target reached!

Figure 6: An example of run of the code with 10 robots.

case that at the end of the procedure we evaluate each of the robots' rewards to be below the threshold θ, we perform another iteration of the algorithm to get as near to the goal as possible. This is repeated until the threshold value is reached by at least one of the robots.

Figure 7 illustrates the stages of the algorithm for a problem instance where a second iteration is required. The Figures 7a, 7b, and 7c are related to the first run of the algorithm: as it can be easily seen, after the gate of the swarm algorithm, the robots did get closer to the target with respect to their starting positions, but they stopped roughly halfway through the actual destination, meaning that the robots' rewards ended up way lower than the θ threshold. The condition of the outermost *while* loop of Algorithm 3 therefore triggers another iteration, starting from the reshuffling stage, as shown in Figure 7d. This particular plot also allows one to visualize a non-trivial reshuffling (i.e., one that is not performed over the entire map), since the sub-rectangle it is computed over is that of the cells located by the centroids $([1, 0], [0, 1])$ —that of the farthest robot, which in this case is the same as the other one, having the two been reunited at the end of the first iteration—and $([1, 1], [1, 1])$, that of the target. Later on, the swarm algorithm is invoked

once again, this time with $R_{closest} \longleftarrow R_1$, and the rest of the procedure is carried out, with the final configuration being that of Figure 7e, where the desired proximity to the target has been achieved.



(a) Start (b) Reshuffle (c) Gate

(d) Second reshuffle (e) Gate

Figure 7: An example of code run with a pair of robots and one additional cycle of reshuffling and logic gate. In the second cycle, the reshuffle is restricted to high-reward regions of the plane.

To quantitatively assess the reliability of our proposed methodology, we compare the average rewards of robots at different steps of the process while varying the size of the swarm.

First, we test our code for a 10-robot swarm, comparing the target-reaching accuracy in 5 runs (Table 2). Then, we investigate target convergence for robotic swarms of different sizes (Table 3).

Table 2 shows the evaluated average rewards of a swarm of 10 robots at each of the four steps of the procedure. The gathered data exhibit the aforementioned enhanced convergence of a bigger-sized swarm, which is translated to a result, at time t_3, which is above the $\theta = 0.8$ threshold for each run.

We notice that our algorithm is scalable for a swarm of ≥ 5 robots: in fact, all trials with this minimum size show an average reward higher than the threshold. Scalability is an essential element of swarm behavior. Table 3 presents the average rewards obtained at each step for swarms of 2, 3, 5, and 10 robots, respectively. In a few trials, the 2- and 3-robot swarms do not reach the proposed convergence threshold. In those cases, an additional

		average reward			
N	trial	t_0	t_1	t_2	t_3
	1	0.245	0.669	0.765	0.898
	2	0.252	0.703	0.687	0.858
10	3	0.248	0.774	0.756	0.883
	4	0.261	0.701	0.888	0.910
	5	0.247	0.673	0.802	0.921

Table 2: Mean values of rewards for a 10-robot swarm the beginning of the process (t_0), after the reshuffle (t_1), after the logic gate (t_2), and at the end (t_3). The values have been computed during five trials, with fixed swarm and target positions.

cycle of the whole code would be necessary to reach the target.

		average reward			
N	trial	t_0	t_1	t_2	t_3
	1	0.249	0.689	0.682	0.760
2	2	0.217	0.637	0.777	0.781
	3	0.245	0.534	0.698	0.733
	1	0.244	0.694	0.779	0.801
3	2	0.256	0.546	0.585	0.673
	3	0.271	0.756	0.836	0.920
	1	0.253	0.778	0.798	0.892
5	2	0.245	0.637	0.831	0.899
	3	0.241	0.685	0.774	0.856
	1	0.256	0.596	0.684	0.821
10	2	0.251	0.661	0.843	0.878
	3	0.255	0.601	0.833	0.924

Table 3: Comparison between average rewards for swarms of different size, with $N =$ 2, 3, 5, 10 robots. The scalability of swarm behavior starts being visible for $N \geq 5$, showing a more robust convergence (final reward ≥ 0.8 in every run of the code).

4.2. Comparison against other methods

Finally, we compare our method against two ant-behavior computational models. We choose the Ant Lines simulation coded in NetLogo [62, 63] and another simulation of ant nest-food paths in Java. To compare the three

methods, we consider computational complexity, maximum number of direction changes, minimum number of robotic ants necessary for convergence toward the food (Table 4). We consider the minimum number of ants required to the convergence, the computational time, and the approximate number of direction changes. We find that our method presents a faster convergence than the Java-based algorithm. The last one converges also with 50 ants, but with one third of the pheromonic decay constant. The duration of pheromonic tracks left by foraging ants is currently investigated by entomologists [31]. In the considered simulation, a quicker convergence is ensured by a null decay, which would however be unrealistic. We set up a decay of 0.003, half than the default value, but triple than the 0.001 required for a 50-ant swarm. Figure 9 shows the paths with NetLogo and Java simulators. The choice of pheromonic-decay rate in ant-colonization models has been studied in detail for a classically-driven robotic swarm [40]. In particular, if the pheromonic-decay rate is beyond a certain threshold, ants even could not reach the target at all. This fact justifies our choice of small decay values.

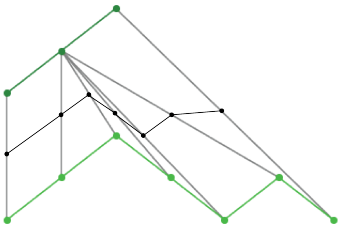
In general, the computational time in a swarm-robotic task is related to the exploration time, which has been investigated in terms of number of robots, characteristics of the environment, and complexity of the task [32]. Here, we chose a simple environment without obstacles, a variable number of robots, and a simple task, but we focused on a new quantum approach.

	our code	NetLogo	Java
min no. of ants for convergence	5	1	50
no. of ants considered in our comparison	10	10	100
time to convergence	10 sec.	14 sec.	3 min.
average no. of direction changes	3	10	6

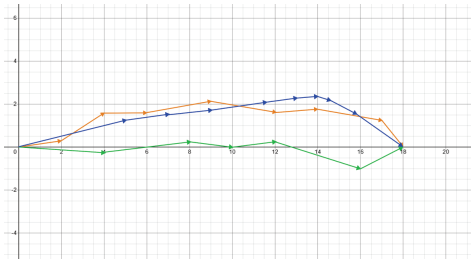
Table 4: Parameter comparison between our method, the NetLogo Ant Lines, and the Java-based ant simulation.

Let us now compare the mean paths of all robotic ants that we obtain with the considered three methods. To find the mean path with our code, we first consider a 10-robot simulation, taking into account the polylines corresponding to the individual paths for each robot. We distinguish between three chunks of the simulation: $t_0 \to t_1$, $t_1 \to t_2$, and $t_2 \to t_3$. Observing the trajectories in the first chunk, we notice that two robots present short paths, which are exactly the one the opposite of the other, giving a null contribution to the overall mean. Thus, we neglect these contributions, and we are left

with 8 paths. In the second chunk, we again neglect the contribution of two robots that were already at the target and in its closest cell, respectively. These two robots presented the null path, and thus their contribution to the overall mean is zero. In the third chunk, all robots are gathered around the target. Then, we computed the pairwise distance of the remaining polylines via the Dynamic Time Warping (DTW) algorithm, comparing corresponding pairs of coordinates. As an example, Figure 8a shows the DTW between the two robotic paths from Figure 5b. The mean of all components is the mean path for all the robots. The so-obtained mean curve is then compared with the mean paths resulting from the other methods (Figure 9). We apply DTW to the robotic paths in each chunk. Concerning NetLogo and Java, we divided the overall path according to direction changes, see Figure 9. In this way, even though we are not considering a precise speed of each robot, we can nevertheless create a comparison between the different steps of the simulations.



(a) An example of DTW comparison between the paths of the two robots from Figure 5b, obtained with our method. The paths are green; the grey segments indicate pairwise distances; the black line is the mean curve.

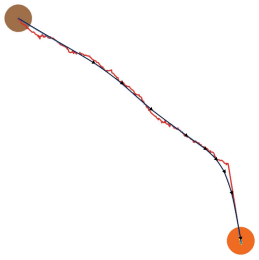


(b) Visual comparison among average paths obtained with our method (green), Java (orange), and NetLogo (blue). The paths are rotated: the closer the path to the x-axis, the closer to a nest→food diagonal path.

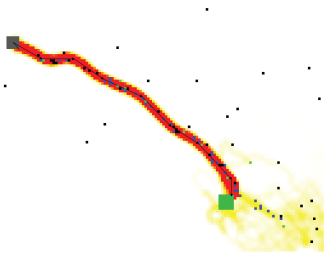
Figure 8: Strategies for robotic-path comparison (a) and average robotic-path comparison (b).

Observing the results (Table 5), we find that our quantum-based method, similarly to the considered nature-inspired pheromonic track, allows robots to reach the target from the nest in an approximately diagonal mean path, that is, the shortest path.[2] It means that our method confirms the expectations,

[2]We notice that a diagonal path is also observed in a classically-driven robotic swarm modeling the ant-foraging behavior [40]. In addition, our method leads to a good convergence already in the first run of the code, while other algorithms need a higher number of



(a) Nest-food path of a 10-ant swarm simulated with NetLogo.



(b) Pheromonic tracking and nest-food path of a 100-ant swarm simulated with Java. Yellow and red indicate weak and strong pheromonic paths, respectively.

Figure 9: Paths obtained with NetLogo and Java simulations. The black vectors indicate the steps of average paths.

and outperforms the Java-based algorithm thanks to the smaller number of robots required for the convergence. In preliminary tests, we also found that our method outperforms NetLogo simulation because of the higher stability with respect to changes in target position. To compare these mean curves, we use the Fréchet distance [21] (a generalization of DTW [22]) in its discrete version [17] (Table 5). Rotating the nest-food diagonal path, we can compare the three polylines against the x-axis. We immediately notice that our method (green path) outperforms the other two ones, being closer to the horizontal line (Figure 8b). This qualitative information is confirmed by quantitative distance measurements, see Table 6. The complexity of the Grover-search part of our code is discussed in Section 2.2. The *for* loops are running over the number of robots, and thus they contribute as $O(n)$. In addition, we are dealing with quantum complexity, while the two considered ant simulations are completely classical. An assessment of computational complexity for the NetLogo and Java simulation is not feasible because of structural differences. For instance, the Java-app running time also depends by the arena's size, influencing the number of steps to return to the nest after an unsuccessful food search. Regarding time, we ran our code on a quantum simulator, requiring a time of the order of tens of seconds. However, running time could be even shorter if a real quantum computer were to be used.

iterations [40].

	NetLogo (10 ants)	Java (100 ants)
our method (10 ants)	2.63	2.46

Table 5: Values of the Fréchet distance between mean curves computed with the NetLogo Ant Lines, Java-based ant simulation, and our method. We consider a different number of ants, allowing for a good convergence. The lower the distance, the closer the mean paths.

	our method (10 ants)	NetLogo (10 ants)	Java (100 ants)
distance from the diagonal	1.0	2.37	2.12

Table 6: Distance from the perfect diagonal nest→food computed for each considered method; see Figure 8b. The lower the distance, the faster the average-path convergence.

5. Discussion and Conclusions

In this research, we developed a quantum-based decision-making system for the path planning of a robotic swarm. We started from two recent studies, concerning a pairwise-communication modeling in a robotic swarm though logic gates, and a Grover-based path-planning algorithm. Here, we proposed a mixed code where the robots in a swarm exchange messages and plan their path toward a target. We proved the scalability of our code for a swarm with at least five elements.

Then, we contextualized our research in a natural-inspired search and rescue scenario, considering the ant-foraging behavior. In particular, we focused on the path from the nest, in one of the micro-cells the arena has been divided into, to the source of food, in another micro-cell. We proved that our method outperforms two existing foraging-ant simulations, regarding the computational speed of convergence, and it is more stable with respect to changes of food position than a NetLogo implementation. In addition, our method requires a smaller number of robots to ensure convergence: 5 or 10 robotic ants against the 50 robotic ants of the Java-based simulation. The research we proposed is a first step toward experiments with real robots and with swarms of different size.

The methodology we proposed aims to face the problem of path-planning for a swarm of robots with the aid of quantum computing. This is not a problem of optimization, while it has been proved that quantum computing can, for instance, reduce computational times of a software. Proposing a new method, we are open also to sub-optimal solutions. We notice that

several recent optimization methods are also inspired by naturalistic models such the ant-colony and foraging-ant models. The quantum enhancement has been applied to optimization approaches; it is the case for instance of *quantum particle swarm optimization* (quantum PSO). A search and rescue approach might be optimized with a suitable objective function having, for instance, a minimum in correspondence of the target. However, optimization techniques are conceptually different from our approach. In fact, in quantum PSO the use of "swarm" is a computational metaphor to solve problems of minimum, while in our study we are interested in developing a method to let an overall behavior emerge from local interactions in a swarm of robots. For this reason, we do not set any optimization function.

In this regard, our approach differs from state-of-the-art enhancement of ant-colony optimization algorithms. In [60], the ant-colony is explored in a grid scenario with fixed obstacles. The authors find shortest path for the ants. Here, we are not necessarily interested in the shortest individual paths: we focus on the conceptually most clear approach which exploits quantum computing in different steps of the process. In fact, our approach is *foundational*, fundamental, in the sense that we are posing the bases for a quantum-based methodology.

The limitations of our study concern the presence of fixed of moving obstacles, whose position is really unknown. The issue might be solved with robotic sensing, smell, radar detection, or target-proximity information retrieval through sound amplitude. These factors, in a real-world application, should influence the individual measure of target proximity (the reward), allowing the complete application of our method.

Our research might constitute a step forward in robotic-based stigmergy studies, whose some noticeable examples are natural swarms [28]. Possible scenarios of applications involve fire prevention, people search and rescue, and water cleaning from trash [1, 2], in the field of environmental care.

The precision of the experiment can be improved with more qubits, also involving one additional dimension and depth, and through a refinement of cells, more precisely detecting spatial positions.

In our work, we ran the code on our laptop, remotely calling the IBM Quantum simulator. However, the whole idea of the swarm also includes local calculations and exchanges of information. Therefore, next steps of the work will include the definition of an MQTT (Message Queuing Telemetry Transport) message-exchanging technique. A completely decentralized robotic swarm delegates simple calculations to all elements of the swarm,

realizing a fully-distributed structure. Indeed, a quantum swarm algorithm can be considered a form of quantum distributed algorithm [9], enhanced by the quantum component [38, 30]. Thus, the multiple processors required in a distributed algorithm can be identified with the simple processors integrated on each robot of the swarm.

In addition, further research can address comparisons with Markov-based decision processes [68], comparing Markovian (memory-less) [32] with non-Markovian time evolution [43], analyzing the necessary amount of information to be retained for a successful search and rescue swarm mission.

The forthcoming application of the proposed code to MQTT will prepare the ground for tests with real robots. Regarding possible future experiments, we envisage a test with three Peppers. Pepper is a complex robot, diverse from robots usually exploited for swarm robotics. However, mechanisms of message exchanging, position set up, path planning, obstacle avoidance can be implemented as well. Thus, we can imagine a toy-swarm constituted by three Peppers. Messages can also be loudly spoken out by robots. The steps of logic gate implementation and path planning of each robot (or, by an external computer connected with robots) can be verbalized through the inner speech [52]. In such a research, we may borrow results from the quantum vocal theory of sound [55].

Our research is meant to be a further step in quantum-computing resource exploration. The application of quantum logic gates to robotic decision-making systems fosters new developments in rapidly-developing interdisciplinary fields. This line of research opens the way to new scenarios of robotics, industry, and ultimately, human knowledge.

Acknowledgments

The research leading to these results takes place within the framework of the project "ARES, Autonomous Robotics for the Extended Ship," funded by the Italian Ministry of University and Research under grant agreement ARS01_00682.

References

[1] P. Agrawal and B. Bhattacharya. Aquatic multi-robot system for lake cleaning. In K. J. Waldron and S. V. Gurvinder, editors, *Nature-Inspired Mobile Robotics. Proceedings of the 16th International Conference on*

Climbing and Walking Robots and the Support Technologies for Mobile Machines, pages 171–178, Singapore, 2013. World Scientific.

[2] A. Akib, F. Tasnim, D. Biswas, M. B. Hashem, K. Rahman, A. Bhattacharjee, and A. S. Fattah. Unmanned Floating Waste Collecting Robot. In K. J. Waldron and S. V. Gurvinder, editors, *TENCON 2019 – 2019 IEEE Region 10 Conference (TENCON)*, pages 171–178, Kochi, India, 2019. IEEE.

[3] M. Alvarez-Alvarado, F. Alban-Chacón, E. Lamilla-Rubio, C. Rodríguez-Gallegos, and W. Velásquez. Three novel quantum-inspired swarm optimization algorithms using different bounded potential fields. *Nature Scientific Reports*, 11:11655, 2021.

[4] P. Atchade-Adelomou, P. Alonso-Linaje, J. Albo-Canals, and D. Casado-Fauli. qRobot: A Quantum Computing Approach in Mobile Robot Order Picking and Batching Problem Solver Optimization. *Algorithms*, 14(194):19, 2021.

[5] P. Benioff. Quantum robots and environments. *Physical Review A*, 58:893, 1998.

[6] S. Berman, Q. Lindsey, M. S. Sakar, V. Kumar, and S. C. Pratt. Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE*, 99(9):1470–1481, 2011.

[7] O. Bles and T. Boehly. Same length, different shapes: ants collectively choose a straight foraging path over a bent one. *Biology Letters*, 14(3), 2018.

[8] E. Bonabeau, G. Theraulaz, and M. Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1977.

[9] K. Censor-Hillel, O. Fischer, F. Le Gall, D. Leitersdorf, and R. Oshman. Quantum distributed algorithms for detection of cliques. ArXiV, `https://arxiv.org/pdf/2201.03000.pdf`, 2022.

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

[11] X. Dai, S. Long, Z. Zhang, and D. Gong. Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method. *Frontiers in Neurorobotics*, 13:15, 2019.

[12] J. Delcourt, N.W. Bode, and M. Denöel. Collective Vortex Behaviors: Diversity, Proximate, and Ultimate Causes of Circular Animal Group Movements. *The Quaterly Review of Biology*, 91(1):1–24, 2016.

[13] D. Dong, C. Chen, C. Zhang, and C. Chen. Quantum robot: structure, algorithms and applications. *Robotica*, 4:513–521, 2006.

[14] X. Dong and M. Sitti. Controlling two-dimensional collective formation and cooperative behavior of magnetic microrobot swarms. *The International Journal of Robotic Research*, 39(5):eabe4385, 2020.

[15] M. Dorigo, G. Theraulaz, and V. Trianni. Reflections on the future of swarm robotics. *Science Robotics*, 5(49):eabe4385, 2020.

[16] R. Eberhart, Y. Shi, and J. Kennedy. *Swarm Intelligence*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufman, Burlington, MA, USA, 2001.

[17] T. Eiter and H. Mannila. Computing Discrete Fréchet Distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

[18] David Eppstein, Michael T. Goodrich, and Jonathan Z. Sun. The skip quadtree: A simple dynamic data structure for multidimensional data. *CoRR*, abs/cs/0507049:12, 2005.

[19] A. K. Fedorov and M. S. Gelfand. Towards practical applications in quantum computational biology. *Nature Computational Science*, 1:114–119, 2021.

[20] Raphael Finkel and Jon Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 03 1974.

[21] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 22:1–72, 1906.

[22] C. Genolini, R. Ecochard, M. Benghezal, T. Driss, S. Andrieu, and F. Subtil. kmlShape: An Efficient Method to Cluster Longitudinal Data (Time-Series) According to Their Shapes. *PlosOne*, 11(6):e0150738, 2016.

[23] H. Hamann. *Swarm Robotics: A Formal Approach*. Springer, Cham, 2018.

[24] T. Häner and M. Soeken. Lowering the t-depth of quantum circuits via logic network optimization. *ACM Transactions on Quantum Computing*, 3(2):6:1–6:15, 2022.

[25] Sariel Har-Peled. *Geometric approximation algorithms*. American Mathematical Society, USA, 2006.

[26] J. Harwell and M. L. Gini. Broadening applicability of swarm-robotic foraging through constraint relaxation. In N. Ye, H. Kurniawati, B. MacDonald, E. Drumwright, and T. Fraichard, editors, *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR 2018.*, pages 116–122, Heidelberg, 2018. IEEE, Institute of Electrical and Electronics Engineers Inc.

[27] C. H. Hemelrijk and H. Hildenbrandt. Schools of fish and flocks of birds: Their shape and internal structure by self-organization. *Interface Focus*, 2:726–737, 2012.

[28] F. Heylighen. Stigmergy as a universal coordination mechanism I: Definition and components. *Cognitive Systems Research*, 38:4–13, 2016.

[29] V. Ivancevic. Entangled swarm intelligence: Quantum computation for swarm robotics. *Mathematics in Engineering, Science and Aerospace*, 7(3):441–451, 2016.

[30] T. Izumi and F. Le Gall. Quantum distributed algorithm for the all-pairs shortest path problem in the congest-clique model. In P. Robinson and F. Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 84–93, Toronto, Canada, 2019. ACM Digital Library.

[31] R. Jeanson, F. L. W. Ratnieks, and J.-L. Deneubourg. Pheromone trail decay rates on different substrates in the Pharaoh's ant, *Monomorium pharaonis*. *Physiological Entomology*, 28:192–198, 2003.

[32] M. Jeong, J. Harwell, and M. Gini. Analysis of exploration in swarm robotic systems. In M. H. Ang Jr, H. Asama, W. Lin, and S. Foong, editors, *Intelligent Autonomous Systems 16 - Proceedings of the 16th International Conference IAS-16. (445-457). (Lecture Notes in Networks and Systems. 412 LNNS. (2367-3370))*, Springer Science and Business Media Deutschland GmbH., pages 445–457, Heidelberg, 2022. Springer.

[33] S. Konur, C. Dixon, and M. Fischer. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2):199–213, 2012.

[34] A. Koukam, A. Abbas-Turki, V. Hilaire, and Y. Ruichek. Towards a Quantum Modeling Approach to Reactive Agents. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2021, pages 130–136, 2021.

[35] R. A. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 100:85–101, 2000.

[36] P. Kuhn, J. Luboeinski, L. Martin, and M. Schneider. Ants Simulation. `https://itp.uni-frankfurt.de/~gros/StudentProjects/Applets_2014_AntsSimulation/ants.htm`, 2020.

[37] L. Lamata, M. Quadrelli, C. de Silva, P. Kumar, G. Kanter, M. Ghazinejad, and F. Khoshnoud. Quantum Mechatronics. *Electronics*, 10:2483, 2021.

[38] F. Le Gall and F. Magniez. Sublinear-time quantum computation of the diameter in congest networks. In C. Newport and I. Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC 2018)*, pages 337–346, Egham, UK, 2018. ACM Digital Library.

[39] Z. Li, W. Liu, G. Li-E, and L. Li. Path Planning Method for AUV Docking Based on Adaptive Quantum-Behaved Particle Swarm Optimization. *IEEE Access Multidisciplinary*, 7:78665–78674, 2019.

[40] F. Ma and X. Ma. Research on path planning of mobile robot based on global view ant colony algorithm. In *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*, pages 1314–1320, Xiamen, China, 2021. ACM Digital Library.

[41] L. Madden and A. Simonetto. Best Approximate Quantum Compiling Problems. *ACM Transactions on Quantum Computing*, 2(3):7:1 – 7:15, 2022.

[42] S. Mahanti, S. Das, B. K. Behera, and P. K. Panigrahi. Quantum robots can fly; play games: an IBM quantum experience. *Quantum Information Processing*, 18(210):11, 2019.

[43] M. Mannone, Rosario Lo Franco, and Giuseppe Compagno. Comparison of non-Markovianity criteria in a qubit system under random external fields. *Physica Scripta*, 2013(T153):014047, 2013.

[44] M. Mannone, V. Seidita, and A. Chella. Categories, Quantum Computing, and Swarm Robotics: A Case Study. *Mathematics*, 10:372, 2022.

[45] M. Mannone, V. Seidita, and A. Chella. Modeling and Designing a Robotic Swarm: a Quantum Computing Approach. Submitted, 2022.

[46] Abby-Mitchell et al. MD SAJID ANIS. Qiskit: An open-source framework for quantum computing, 2021.

[47] H. (Hermann) Minkowski. *Geometrie der Zahlen*. Teubner, Leipzig, 1910.

[48] N. Mishra, R. S. Chandra, B. K. Behera, and P. K. Panigrahi. Automation of quantum Braitenberg vehicles using finite automata: Moore machines. *Quantum Information Processing*, 19(17):12, 2020.

[49] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Higher Education from Cambridge University Press, Cambridge, England, 2010.

[50] C. Noirot and J. P. E. C. Darlington. Termite nests: Architecture, regulation and defence. In Takuya Abe, David Edward Bignell, and Masahiko Higashi, editors, *Termites: Evolution, Sociality, Symbioses, Ecology*, pages 121–139. Springer Netherlands, Dordrecht, 2000.

[51] C. Petschnigg, M. Brandstötter, and H. Pichler. Quantum Computation in Robotic Science and Applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 803–810, Montreal, Canada, 2019. IEEE.

[52] A. Pipitone and A. Chella. A cognitive architecture for inner speech. *Cognitive Systems Research*, 59:287–292, 2020.

[53] L. Pitonakova, R. Crowder, and S. Bullock. Task Allocation in Foraging Robot Swarms: The Role of Information Sharing . *Autonomous Robots*, 17:93–105, 2020.

[54] J. Preskill. Quantum computing 40 years later. In A.J.G. Hey, editor, *Feynman Lectures on Computation, 2nd ed.* Taylor & Francis Group, Abingdon, UK, 2021.

[55] D. Rocchesso and M. Mannone. A Quantum Vocal Theory of Sound. *Quantum Information Processing*, 19(292):28, 2020.

[56] E. S̈ahin. Swarm robotics: From sources of inspiration to domains of application. In E. S̈ahin and W. M. Spears, editors, *International Workshop on Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, page 11, Berlin/Heidelberg, 2004. Springer.

[57] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong. Hormone-Inspired Self-Organization and Distributed Control of Robotic Swarms . *Autonomous Robots*, 17:93–105, 2020.

[58] I. Slakov, C. Carrillo-Zapata, D. Jansson, H. Kaandorps, and J. Sharpe. Morphogenesis in robot swarms. *Science Robotics*, 3(25):38, 2018.

[59] J. Stolze and D. Suter. *Quantum Computing: A Short Course from Theory to Experiment*. Wiley, Weinheim, Germany, 2004.

[60] H. Wenbin, Z. Xiong, C. Wang, and H. Chen. Enhanced ant colony algorithm with communication mechanism for mobile robot path planning. *Robotics and Autonomous Systems*, 148(103949):1–10, 2022.

[61] A. Wichert. *Principles of Quantum Artificial Intelligence*. World Scientific, Singapore, 2020.

[62] U. Wilensky. NetLogo Ant Lines model. `http://ccl.northwestern.edu/netlogo/models/AntLines`, 1997. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[63] U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo/`, 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[64] R. Wille, S. Hillmich, and L. Burgholzer. Tools for Quantum Computing Based on Decision Diagrams. *ACM Transactions on Quantum Computing*, 3(3):13:1 – 13:17, 2021.

[65] A. S. Wu, R. P. Wiegand, and R. R Pradhan. Response probability enhances robustness in decentralized threshold-based robotic swarms . *Swarm Intelligence*, 14:233–258, 2020.

[66] G.-Z. Yang, P. Bellingham, E. Dupont, and et al. The grand challenges of Science Robotics. *Science Robotics*, 30(14):14, 2018.

[67] L. Yang, L. Fu, P. Li, J. Mao, and N. Guo. An Effective Dynamic Path Planning Approach for Mobile Robots Based on Ant Colony Fusion Dynamic Windows. *Machines*, 10(1):50, 2022.

[68] S. Ying and M. Ying. Reachability analysis of quantum markov decision processes. *Information and Computation*, 263:31–51, 2018.

[69] S. Zammuto and M. Mannone. quantum-swarm-path-planning, May 2022. https://github.com/salvatorezam/quantum-swarm-path-planning.

[70] K. Zhu and M. Jiang. Quantum Artificial Fish Swarm Algorithm. In *Proceedings of the 8th World Congress on Intelligent Control and Automation*, page 5, Jinan, China, 2010. IEEE.