

# Modeling and Designing a Robotic Swarm: a Quantum Computing Approach

Maria Mannone<sup>1,2,\*</sup>, Valeria Seidita<sup>1</sup>, Antonio Chella<sup>1,3</sup>

---

## Abstract

Nature is a neverending source of inspiration for technology. Quantum physics suggests applications toward quantum computing. Swarms' self-organization leads to robotic swarm developments. Here, quantum computing **is applied** to swarm robotics. We model local interactions with a quantum circuit, testing it on simulators and quantum computers. To relate local with global behavior, we develop a block matrix-based model. Diagonal sub-matrices contain information on single robots; off-diagonal sub-matrices are the pairwise interaction terms. Comparing different swarms means comparing different block matrices. Choosing initial values and computation rules for off-diagonal blocks (with a particular logic gate), model different behaviors **can be modeled**. To show the global-behavior emergence, we propose a specific pairwise-interaction logic gate, embedding the corresponding quantum circuit in an ant-foraging-inspired algorithm. To implement a first application, we choose the foraging-ant behavior for its clarity and importance in nature, running experiments with toy swarms (3 and 10 robots). We consider ants' individual and collective back-and-forth movements between the nest and the food source, analyzing the effect of entanglement. Our research can help shed light on quantum potentialities for swarms. The implications of our findings and results concern the future development of

---

\*Corresponding author

*Email addresses:* [mariacaterina.mannone@unipa.it](mailto:mariacaterina.mannone@unipa.it) (Maria Mannone),  
[valeria.seidita@unipa.it](mailto:valeria.seidita@unipa.it) (Valeria Seidita), [antonio.chella@unipa.it](mailto:antonio.chella@unipa.it) (Antonio Chella)

<sup>1</sup>Department of Engineering, University of Palermo, Italy

<sup>2</sup>ECLT and DAIS, Ca' Foscari University of Venice, Italy

<sup>3</sup>ICAR-CNR National Research Council, Italy

a decision-making system, based on the advantages of swarms and quantum computing. While an ant-foraging scenario [is chosen](#) as an example of application, our study is not focused on optimization. We present a new methodology, open to non-optimal solutions. Future developments can concern improvements toward optimization.

*Keywords:* quantum computing, swarm robotics, search & rescue, logic gates

*2010 MSC:* 03G12, 81P68, 15A99, 93C85

---

## 1. Introduction

Antoni Gaudí, architect and saint, considered Nature as his supreme teacher and source of inspiration. Science and beauty in nature can also inspire the development of refined and autonomous human artifacts such as robots. Robotics  
5 tries to catch structures and mechanisms, including examples of self-organized collective behavior in nature, results of the so-called *swarm intelligence* [1].

The twirling elegance of flocking birds and schooling fish, the architectural ingenuity of termites [2], locusts grouping [3], the organization of ants are the input for mathematical modeling [4, 5] and robotic developments [6, 7]. Some  
10 characteristics of natural swarms are caught in human-made artificial swarms, such as self-organization, scalability, and decentralization. Computational techniques mimicking behavioral and social patterns in nature can help solve complex tasks [8].

Another source of natural inspiration is given by quantum computing, an  
15 extension of computer science derived from the principles of quantum physics [9], of growing importance for artificial intelligence [10]. The main reason is the extension of quantum computational efficiency to enhance a robotic set-up.

In this article, we develop a mathematical description of the swarm with a model of nested matrices, and pairwise interactions represented by reversible  
20 logic gates. In particular, we consider an ant-foraging scenario, for its impor-

tance in nature and recent robotic applications.<sup>4</sup>

In particular, we try to connect local and global behavior of a swarm, letting the global behavior emerge from simple local pairwise interactions. These local interactions are modeled via a quantum circuit. Then, to test the idea, the circuit in a nest-food-nest ant scenario is exploited.<sup>5</sup> The algorithm can lead to a novel decision-making system based on quantum computing. The novelty of our work is two-fold: first, we define general matrices which can be adapted to whatever swarm with whichever interaction laws; second, we shape the interaction terms starting from probability-based quantum concepts. We develop and extend dimensionally the ideas *in nuce* in [11]. The key technical difficulty to overcome is the definition from scratch of a formalism describing both local and global aspects of the robotic swarm, including a quantum-based approach to connect the world of robots with the flourishing field of quantum computing.

The article is organized as follows. A brief literature survey is presented in Section 2, providing the motivations for our research in Section 3. Then, all information to reproduce our research are presented in Section 4. Our theoretical approach is presented in Section 5, with nested matrices (subsection 5.1) and quantum computing (subsection 5.2). We briefly describe our results in Section 6, presenting our case study, with toy swarms of 3 and 10 robots on the plane simulating the case of nest-food-nest ant path. We also present a comparison between our method and two optimization approaches, the PSO and NL-SHADE-RSP algorithms. Section 7 contains a discussion of strengths and limitations of our strategy and possible developments of this research. In Section 8, our findings are summarized. The Appendix contains the search and rescue pseudocodes, the quantum codes for pairwise interactions, and an example of the nested matrices for a 3-robot swarm.

---

<sup>4</sup>See the ant-inspired codes for NASA: <https://www.nasa.gov/feature/students-develop-robotic-code-in-first-swarmathon-challenge>

<sup>5</sup>There are the following steps: starting from the nest, random reshuffle, quantum gate information exchange, and final points achievement.

## 2. Literature survey

### 2.1. Swarms of robots

Swarms of robots are an example of artificial swarm intelligence. A robotic  
50 swarm is constituted by multiple autonomous and simple robots, collaborating  
to achieve a task that is impossible for single units. Each individual robot in a  
swarm is a simple element that can perform only a few simple tasks (in tradi-  
tional swarms, which are biologically inspired, there is only one task). Moreover,  
the single element does not know the global goal of the swarm but it contributes  
55 to its achievement thanks to self-adaptation, self-regulation, communication,  
and cooperation with the other elements of the swarm [12, 13, 14]. The swarm  
behavior is in fact an *emerging* effect. If the study of nature can be the input  
for robotic applications, it can also constitute its target. In fact, robots can be  
a benchmark to model and investigate complex phenomena, such as morpho-  
60 genesis in multi-cellular creatures [15]. Medical applications include miniature  
robotic swarms to deliver medications inside the human body [16].

In robotic swarms, as well as in natural swarms, individual behavior is gov-  
erned by simple rules allowing a few simple actions. The individual element  
of the robot must interact with the environment and with its conspecifics to  
65 allow complex behavior to emerge. Communication and interaction with other  
elements of the swarm are key to coordinate and collectively perform a complex  
goal. Natural examples include heavy prey transporting, foraging, or massive  
and complex structures building.

Intuitively, in swarm robotics a large number of simple embodied agents and  
70 their actions are designed to let a complex collective behavior emerge. Robots  
have to interact between them and with their environment [17].

In robotic swarms, [one can](#) distinguish between a micro-level, with individual  
behaviors, controls, and pairwise interactions, a macro-level with the global  
swarm action [18]. Properties of a swarm include scalability, robustness (the  
75 lack of an individual does not affect the global behavior), self-regulation [13, 12]  
(no external commands are required), and self-awareness [12] (each individual

knows its own position, speed, and action) [18].

As described by [19], examples of self-organized approaches for swarms of robots concern spatial organization, navigation, decision making, and miscellaneous task. Spatial organization are aggregation, pattern formation, self-assembly, object clustering and assembling. Navigation tasks include collective exploration, coordinated motion, collective transport, collective localization. Decision-making tasks are about consensus, task allocation, collective fault detection, collective perception, synchronization, and group-size regulation. Finally, miscellaneous tasks concern self-healing, self-reproduction, human-swarm interaction. Research on swarm robotics range from the development of cyber-swarms [20] and swarm specialization into specific tasks [21].

Concerning the software improvements, there are several optimization approaches to robotic swarms. A notorious example is the application of particle swarm optimization (PSO) [22] to refine search and rescue robotic missions [23, 24], where each particle represents a robot, also in maritime scenarios [25]. Another source of natural inspiration for swarm robotics is evolution. Evolutionary algorithms, and in particular the successful genetic algorithm (GA), are based on candidate solutions of a problem, and a problem-dependent objective function [26]. Also differential evolution-based algorithms [27] can be considered. The applications of these concepts to robotics leads to evolutionary robotics [28]. Evolutionary approaches are used to optimize problem-solving techniques. Other approaches are the aforementioned ant-colony optimization (ACO) [29, 27], inspired by ants' collective behavior, and particle swarm optimization (PSO), suggested by birds' collective flight [22]. According to [30, 31], the philosophy of PSO lead to several other bio-inspired optimization methods, such as GA and ACO. In [32, 33], evolutionary and genetic processes are also exploited. However, our research does not focus on optimization. We rather investigate if quantum computing can be applied to model simple, local pairwise robot interactions, and if a global behavior of swarm can emerge from them. Nevertheless, an example of PSO is adapted to our scenario, quantitatively comparing its outcomes with ours (Table 10), and then of NL-SHADE-RSP

algorithm [34]. A basic example of PSO is considered for its conceptual importance. Further comparisons, with the aid of machine learning applications, will be considered for future research, including more recent developments of PSO itself [35], and enhancements of ant-inspired algorithms with evolutionary approaches [36, 37]. However, while a great part of recent research on swarm robotics deals with optimization techniques for specific goals and scenarios [20], there is another side of the research to be explored, that is, the definition of a more general approach. As noticed by Nedja et al. [38], several solutions are too problem-specific, and thus a general methodology which could be adapted to different problems, algorithms, and devices is yet to be found. In this regard, the matrix-based methodology that is proposed in Section 5 might help fill this gap. Our approach tries to model the emerging swarm behavior from local interactions. Indeed, global-local connections and swarm-emergence from simple rules are not trivial. Analogies with physics have been made, concerning the micro/macrosopic behavior descriptions through Langevin and Fokker-Planck equations, respectively [18]. However, the increasing complexity of robotic swarms requires a more complex and general treatment [39]. For all these reasons, robotic swarms are amongst the most challenging topics in robotics [40].

In a robotic swarm, the individual does not have a complex objective in its own. Instead, it acts to reach a higher-level objective. Therefore, one can talk of force multiplication. The individual in fact feels the environment and its peers, and communicates with them, while acting individually and autonomously.

A standard approach to engineering a robot swarm is unthinkable. A swarm is a complex system that is more than the sum of its components. Thus, to describe a swarm we should remind that it is a complex system whose parts interact with each other and their environment [41]. In addition, swarms are characterized by consensus formation [42] in networks with neighbor-dependent synergy and observer effect.<sup>6</sup> Therefore, a top-down engineering approach is

---

<sup>6</sup>In our approach, we consider information exchange between all elements of the swarm, as

not feasible. A design methodology suitable for a swarm must analyze the system and implement the model at the micro-level: the goal of each element, its capabilities, and the exchange of messages. On the other hand, at the macro  
140 level, one should analyze and implement the “how” producing the behavior needed for the swarm’s goal achievement. From a modeling point of view, the single individual task requires an interaction with the environment, while the interaction with other elements of the swarm requires a peer communication. *Collective behavior* is thus a key word in this research. Finally, concerning  
145 the hardware side, there are examples of research and commercially-available robotic swarms suitable for terrestrial [43, 44, 45], aerial [46], aquatic [47, 48], and outer space [49] motion.

## 2.2. Quantum computing and its application to biology

To model tasks and communications, we choose to exploit *quantum computing* [9, 50]. Quantum computing is a branch of computer science based on  
150 the principles of quantum mechanics. In a nutshell, it is an approach to computer science based on quantum probability amplitudes and reversible gates. Reversible gates are used in analogy with invertible operators ruling quantum mechanics.

In classic computer science, the units of memory are 0 and 1, that is, the  
155 possible values of the so-called *bit*. However, in quantum computing [there is](#) the *qubit*, the quantum bit, that can assume all values between 0 and 1. This is a consequence of the principle of state superposition in quantum mechanics. In addition, to measure the value of a qubit and store it in a classic bit, it is  
160 necessary to perform a measurement operation which is a *destructive* one: the wavefunction, representing the state, collapses to one of its values. All further measures will then give the same result. Destructive measure occurs in quantum

---

all robots were the neighbors of each other. Consensus is implicit in reaching the robot with the highest “reward,” [a concept that is explained](#) later. In the case of a larger swarm where [one can](#) consider subsets of neighbors, the consensus formation can deal with the comparison of max rewards in each subset.

physics, and it was the first operation measure where the observer influenced the observed system.

165 In general, quantum computing enhances speed and efficiency of classical algorithms, and nowadays quantum computers and simulators can be accessed remotely (e.g., IBM, Amazon, and so on). The main cost of these gains is the access itself to quantum computers or simulators, and thus, the need for an internet access. One of the major motivations for the use of quantum computing  
170 is its power of calculation.

The advantages of quantum speed-up come with some drawbacks: it is the case of *decoherence*, that is, a loss of information of the quantum system into the environments for effect of its interaction with it. The collapse of the wavefunction is necessary to perform measurements, but such a phenomenon should  
175 be controlled. Constructors of quantum computers have to take into account the risk of errors given to decoherence. Decoherence does not appear in simulations because there are no interaction with the environment—unless they are modeled in purpose.

Starting from initial insights in physics [51] and computer sciences [52], quantum computing has recently been applied in robotics [53, 54] and artificial intelligence [10]. Quantum particle swarm behavior inspires algorithm improvements [55] and swarm optimization [56, 19]. Examples of quantum-inspired algorithms include quantum harmonic oscillator algorithm as an heuristic optimization algorithm [57]. The quantum paradigm helped enhance techniques to solve multiobjective large-scale optimization problems [58]. Other applications involve  
185 improvements to quantum-inspired evolutionary algorithms [59], to solve the positioning-antenna problems in networks [60], enhancement of grey levels in images [61], face multi-objective large-scale problems [62]. In particular, Cao et al. [considered position](#) uncertainty, that is, as a quantum reference, to formalize  
190 their approach [62]. In fact, based on quantum mechanics postulates, quantum computing is essentially probabilistic [63]. Probability has already been used for threshold-based robotic swarm behavior [64]. One of the most singular phenomena of quantum mechanics is entanglement [65]. Two entangled particles

are parts of the same system, in the sense that a measure on one of them affects  
195 the other one. With quantum computing, [one](#) can build up circuits to make  
entangled states [50]. The entanglement has an essential role for pure-state (not  
mixed) quantum algorithms, while it does not necessarily lead in general to  
computational speed-up [66]. The idea of entanglement has been proposed to  
entirely model a swarm of robots, as a theoretical simplification to group and  
200 connect robots' behavior [67], or to enable a different communication strategy  
between two complex robots [68]. However, the application of the quantum  
paradigm to swarm autonomous devices is, to the best of our knowledge, still a  
largely unexplored field [69, 11].

In a recent study, simulated underwater swarm localization made use of fuzzy  
205 logic [70]. Because quantum logic can be seen as a particular example of fuzzy  
logic [71], we choose the quantum paradigm to investigate and model the swarm  
behavior. Also, a quantum algorithm has been developed for the path planning  
of a single robot [72].

Pioneering applications of quantum computing in biology deal with simu-  
210 lation technologies [73]: imaging, spectroscopy, microscopy, molecule dynamics  
[74], and protein structure prediction [75]. Quantum neural networks are con-  
sidered key tools for neuroscience: from genes, to molecules, to cells, to neural  
structures, up to the human behavior—seen as an emerging effect [76]. In Table  
1, a synthetic overview of the literature framework [is proposed](#).

Table 1: A synthetic overview of the literature framework.

area	topic	authors	year	ref.	
quantum mechanics	basics	Feynman et al.	1965	[77]	
		Einstein et al.	1935	[78]	
	entanglement	Bell	1964	[65]	
		Greenberger et al.	1989	[79]	
	probabilities	Hemmo and Shenker	2020	[63]	
quantum computing	general	Stolze and Suter	2004	[50]	
	entanglement and speed-up	Josza and Linden	2003	[66]	
		for AI	Wichert	2020	[10]
	AI, learning	Kwak et al.	2008	[80]	
	for robots		Dong et al.	2006	[52]
			Benioff	1998	[51]
			Dong et al.	2008	[81]
			Zhu et al.	2010	[6]
			Lamata et al.	2021	[68]
			Atchade-Adelomou et al.	2021	[54]
			Ivancevic	2016	[67]
		Chella et al.	2022	[72]	
	Koukam et al.	2021	[69]		
	Mannone et al.	2022	[11]		
swarms of robots	general	Hamann	2018	[18]	
	overview	Shranz et al.	2020	[19]	
	terrestrial (ants)	Berman et al.	2011	[7]	
	terrestrial (kilobots)	Rubenstein et al.	2014	[43]	
	terrestrial (e-pucks)	Alkilabi et al.	2017	[44]	
	terrestrial	Groß et al.	2006	[45]	
	aerial	Oung	2013	[46]	
	aquatic	Schmickl et al.	2015	[48]	
	outer space	Kang	2018	[49]	
	general	Zambonelli et al.	2011	[12]	
	response probability	Wu et al.	2011	[64]	
	hormone-inspired	Shen et al.	2020	[13]	
	foraging	Pitonakova et al.	2020	[14]	
	general	Sahin	2004	[17]	
	micro, health	Dong and Sitti	2020	[16]	
	future	Dorigo et al.	2020	[39]	
	fuzzy, underwater	Sabra and Fung	2020	[70]	
	consensus formation	Mañas-Álvarez et al.	2023	[42]	
	natural swarms	general	Eberhart	2001	[1]
		flocking birds	Hemelrijk and Hildenbrandt	2012	[4]
termites		Noirot	2000	[2]	
foraging ants		Plowes et al.	2013	[82]	

### 215 3. Motivations

We described above the motivations for quantum computing. In this Section, we enunciate the details of our research questions motivating our work.

Our first research question is: how can **one** connect local and global behavior to model the swarm behavior emergence from local behavior of the single and  
220 pairs of robots?

The second question is: is it possible to model a local decision system through quantum computing, to be the core part of a whole swarm approach?

Swarms of animals make use of a mixture of instinct, individual intelligence, environment and peer observations for individual decision-making. Individual  
225 decisions are the pieces in the puzzle of the emerging swarm intelligence.

Robotic swarms rely on simple individual behavior and decision-making which can be modeled first and coded then. The local behavior, with single-agent decisions and pairwise interactions, needs to be connected with the global behavior, that is, the emerging swarm behavior. We need a theoretical frame-  
230 work to connect the local with the global behavior, and a logic model to build up decision-making structures.

Our quantum circuit models pairwise interactions, and, to be tested, it is included inside an original algorithm (codes 1, 2) for the robotic swarm. For our case study, we consider the biological inspiration of ants moving between the  
235 nest and the food location. Group food retrieval in *Aphaenogaster cockerelli*, an ant species, is in fact the inspiration source for swarm robotic studies [7]. We focus on ant colonies because of their simplicity of modeling, importance as a classic biological model, and easy generalizability. While our research may appear as an optimization study, we are developing some initial elements for a  
240 decisional system. In fact, even though we will be considering and comparing trajectories of robots, we are not developing a system to reach a target with the fastest trajectory. We are not looking at the best possible solution of a problem, **but instead**, at applications of quantum computing. The quantum enhancement of an algorithm usually yields faster solutions than the classical

245 counterparts. In our research, we are open to sub-optimal solutions. Thus, we preferred to not analyze our results with statistical approaches, but rather with qualitative observations. Nevertheless, some quantitative comparisons (between our method and the results of a particle swarm optimization adapted to our scenario) are proposed in Table 10.

250 Quantum computing is used here to model probabilistic information sent by robots and probabilistic trajectory outcomes. Here, the quantum paradigm involves:

- definitions of rewards (as probability amplitudes of ‘target yes’ and ‘target no’ as 0/1) and positions (left/right as 0/1 along x, up/down as 0/1 along 255 y);
- use of quantum circuits, with quantum logic gates, for robots’ probabilistic decision-making;
- definition of an entangled GHZ state.

The Greenberger–Horne–Zeilinger state (GHZ state for short) is an entangled 260 state which, for three qubits, takes the form:  $|\psi_{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$  [79]. We will consider a GHZ for five qubits. In an enhanced and shortened version of our code discussed later in the article, we omit the GHZ state, because convergence is faster.

A closely-related research has been developed by Koukam and others [69]. 265 The authors, focusing on agents, considered a quantum circuit, relating robot’s perceptions with robot’s actions, picking up an action from a list, as activating an item. The idea has been implemented with IBM quantum simulators. In our study, we also build up quantum circuits, [embedding](#) them into a theoretical and nested approach to swarms, [to model](#) a search and rescue behavior. Our 270 research can inherit expertise in agent systems and multi-robots, adding the condition of swarm as a relationship between local and global behavior. In [69], the authors used entangled W states, that is, states of the form  $|\psi_W\rangle = \frac{1}{\sqrt{3}}(|100\rangle + |010\rangle + |001\rangle)$ . In our study, we choose entangled GHZ state, because

we focus on space positions elements and we are interested in bringing together  
275 all robots in a position (indicated by qubit 0) or in an opposite one (indicated  
by 1).

We aim to model the connection between local and global behavior through  
nested matrices. A block matrix represents the swarm, while each sub-matrix  
represents a single robot’s behavior or a pairwise interaction term. Changing  
280 the structure of matrices, it is possible to model swarm features. We present the  
[general theoretical idea](#) and an experiment with a toy 3- and 10-robot swarm  
implementation, confirming our expectations. [In addition, even though our  
approach is open to suboptimal solutions, we present a comparison between our  
results and the results obtained with two optimization approaches, the PSO  
285 and NL-SHADE-RSP algorithms, assessing the degree of suboptimality, and  
verifying the validity of our proposal.](#)

#### 4. Experimental

In our research, we propose a matrix-based model of a robotic swarm (the  
details [are](#) provided in Section 5). We focus on off-diagonal sub-matrices, rep-  
290 resenting interaction terms. We design a quantum-based local-interaction ap-  
proach, modeling relationships between input information (position and reward  
as food proximity) and behavioral outcome. We implement a simulation of toy  
3- and 10-robot swarms inspired by ant foraging in nature. The robots search  
for the food and then get back to the nest. The food location and the nest are  
295 targets to reach. Regarding communication strategies, broadcast communica-  
tion [14] [are considered here](#). In the Appendix, all the necessary information to  
replicate our experiment [are provided](#).

Our proposed quantum circuit is included in two Python codes, created  
in Jupyter Notebook environment and corresponding to Algorithm 1 and Al-  
300 gorithm 2, accessible online, jointly with screenshots and a video simulation.  
Qiskit QASM simulator is called locally from the Notebook. An alternative  
code could involve the loop repetition of the solely quantum gate, until the

target is reached. The additional materials can be found in the GitHub folders <https://github.com/medusamedusa/3-robot> and [https://github.com/medusamedusa/10\\_little\\_ants](https://github.com/medusamedusa/10_little_ants).  
305

## 5. Theory and Calculation

In this section, we describe the theoretical background of our method, with nested matrices (subsection 5.1) and quantum computing for decision-making in search and rescue (subsection 5.2). Then, in Section 6, a toy model with a robotic swarm is implemented.  
310

### 5.1. The overall matrix

In our former research [11], we presented a category-theoretic framework to connect specific, existing swarms of robots with their typologies, going upward in abstraction toward main classes of swarms (for underwater, flying, walking robots), up to conceptual “ideal” swarms. Thus, we can make vertical comparisons, between swarms of different ontology, and horizontal comparisons, between swarms of the same level of reality [11]. A comparison between different swarms becomes in this way a comparison between block matrices.  
315

In our approach, a swarm can be described by a block matrix at each time point, where diagonal sub-matrices represent individual information of each robot, and off-diagonal sub-matrices contain information on pairwise interaction. Equation (1) shows the matrix  $S_n$ . The dimensions of the matrix are computed as  $(nm) \times (nm)$ , where  $n$  is the number of robots and  $m$  is the size of the matrix blocks, depending upon the degree of freedom and detail of information exchange. Thus, if  $n = 3$  and  $m = 4$  (as the  $(4 \times 4)$  blocks in Paragraph 5.2),

$S_3$  has dimensions  $12 \times 12$  (see Figure 1 in the separated file of the Appendix).

$$S_n^{(t)} = \begin{pmatrix} R_1 & R_1 * R_2 & \dots & R_1 * R_{n-1} & R_1 * R_n \\ R_2 * R_1 & R_2 & \dots & R_2 * R_{n-1} & R_2 * R_n \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ R_{n-1} * R_1 & R_{n-1} * R_2 & \dots & R_{n-1} & R_{n-1} \\ R_n * R_1 & R_n * R_2 & \dots & R_n * R_{n-1} & R_n \end{pmatrix} \quad (1)$$

The case with  $n = 2$  robots is described in [11]. In the case of a 3-robot swarm, the matrix  $S_3$  is constituted by three single-robot terms (diagonal sub-matrices) and six pairwise interaction terms (off-diagonal sub-matrices).

$$S_3^{(t)} = \begin{pmatrix} R_1 & (R_1 * R_2) & (R_1 * R_3) \\ (R_2 * R_1) & R_2 & (R_2 * R_3) \\ (R_3 * R_1) & (R_3 * R_2) & R_3 \end{pmatrix} \quad (2)$$

In eq. 2, off-diagonal sub-matrices are not symmetric: e.g.,  $R_1$  might be sending  
 320 a message to  $R_2$ , while  $R_2$  might not be sending any message in turn. Even if the  
 two robots are sending messages to each other, their content will be different,  
 so the terms are different. In the case study of Section 6, we consider broadcast  
 communication: each robot sends a message to all the other robots. Inside the  
 matrix of eq. (2), one can define the structure of single-robot terms (eq. 4)  
 325 and pairwise interaction terms (eq. 5). In each single-robot sub-matrix (eq.  
 4), diagonal blocks indicate robot's own 'perception' and off-diagonal blocks  
 correspond to robot's communication tools.

A specific form that can be taken by diagonal and off-diagonal sub-matrices is  
 proposed at the end of Section 5.2. Let us now introduce the quantum formalism  
 330 needed to develop our modeling.

## 5.2. The quantum machinery

We formalize pairwise robotic interaction terms as quantum gates. To this  
 aim, we quantize robots positions and target positions. That is, for each robot,

we model the position along the  $x$ -axis, the position along  $y$ , and the reward as  
 335 quantum superposition of  $|0\rangle$  and  $|1\rangle$ . In our study, we measure the individual  
 reward in terms of target proximity.

In the case of one-dimensional movement, the  $x$ -position of the  $i$ -th robot is  
 given by:

$$|q_0\rangle = \alpha_i^x|0\rangle + \beta_i^x|1\rangle,$$

where  $\alpha_i^x$  is the probability amplitude for outcome  $|0\rangle$  along  $x$  (left) and  $\beta_i^x$  is  
 the amplitude for outcome  $|1\rangle$  (right), and the reward is given by:

$$|q_1\rangle = \gamma_i|0\rangle + \delta_i|1\rangle,$$

where  $\gamma_i$  is the probability amplitude to obtain  $|0\rangle$ , *failure*, and  $\delta_i$  to obtain  $|1\rangle$ ,  
*success*. In the case of a 2-robot toy swarm and motion along one dimension  
 only, [one](#) can thus define a reversible logic gate as the one presented in Table 2.  
 340 If a robot had a successful reward (1) in position 1 at time 0, the other robot  
 reaches it at time 1. Otherwise, the second robot explores around position 0.  
 The so-obtained decision system can be described through a reversible gate. Actual  
 configurations are quantum superposition of states. For a detailed account  
 of states and some simulations, see [11]. The code to implement this gate can  
 345 be found in the Appendix.

While considering a motion along the plane, we have:

$$|q_0\rangle = \alpha_i^x|0\rangle + \beta_i^x|1\rangle, \quad |q_1\rangle = \alpha_i^y|0\rangle + \beta_i^y|1\rangle, \quad |q_2\rangle = \gamma_i|0\rangle + \delta_i|1\rangle, \quad (3)$$

where  $|q_0\rangle$ ,  $|q_1\rangle$  are the positions along  $x$  and  $y$ , respectively, and  $|q_2\rangle$  is the  
 reward. In particular,  $\alpha_i^x$  is the probability amplitude for outcome  $|0\rangle$  along  $x$   
 (left),  $\beta_i^x$  for  $|1\rangle$  along  $x$  (right),  $\alpha_i^y$  for  $|0\rangle$  along  $y$  (up), and  $\beta_i^y$  for  $|1\rangle$  along  $y$   
 (down). The pairwise interaction of two robots on the  $x$ - $y$  plane can be described  
 350 through Table 3, implemented through the circuit of Figure 1 obtained with IBM  
 Quantum Composer. On the plane, in case of failure (reward 0) of robot 1, then  
 robot 2 has more than one option, leading to outcome indeterminacy—and the  
 gate is no longer reversible.

Table 2: Truth tables (reversible equivalents of XNOR gates), representing the interaction between robot 1,  $R_1$  ( $q_0$ : position,  $q_1$ : reward) and robot 2,  $R_2$  ( $q_2$ : position,  $q_3$ : reward). At time  $t_0$ ,  $R_1$  sends to  $R_2$  a message with its position and reward. According to this information,  $R_2$  can choose to reach the first robot or not at  $t_1$ . The reward of  $R_1$  is copied in the output to guarantee the same number of inputs and outputs, making the gate reversible. After having sent the message,  $R_1$  stops and waits for the motion of  $R_2$ . Once the  $R_2$  reaches the new position at  $t_2$ , it can send to  $R_1$ , in turn, its position and obtained reward. And, similarly,  $R_1$  can decide to reach  $R_2$  or not at  $t_2$ . The first table represents the situation  $t_0 \rightarrow t_1$ , and the second table,  $t_1 \rightarrow t_2$ .

$q_0$	$q_1$	$q_0$	$q_2$	$q_2$	$q_3$	$q_2$	$q_0$
0	0	0	1	0	0	0	1
0	1	0	0	0	1	0	0
1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1

Table 3: The idea of Table 2 is now extended to two dimensions in space, with the truth table for two robots  $R_i, R_j$  on the plane, no longer reversible because of the indeterminacy on x, y in the case of 0 reward. In this application, we do not consider any waiting time. All robots are exchanging information and moving only if its reward is lower than their own.

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

In a 3-robot swarm, such an indeterminacy is solved because each robot is receiving information from two robots, not only one, and it can let only the most successful robot (reward = 1 or just higher than the other) enter the table. However, if the most successful robot presents a  $H$  (Hadamard) gate for

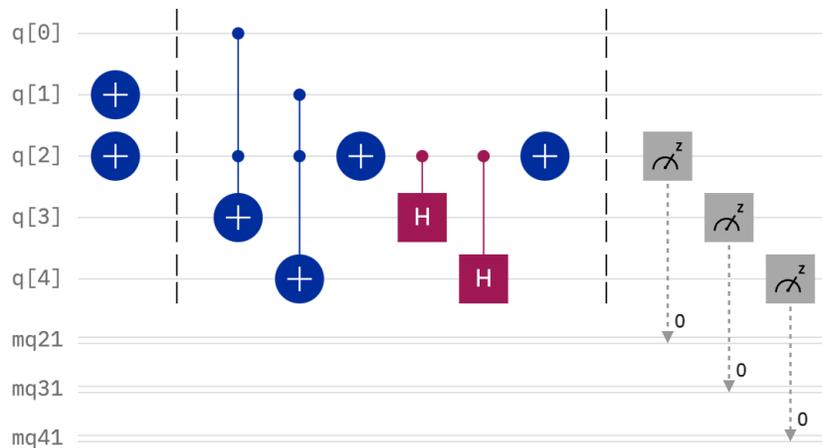


Figure 1: Quantum circuit realizing the truth table of Table 3. The circuit is made of NOT, Toffoli, and Hadamard gates. The white “plus” inside a blue circle indicates the NOT gate; the same symbol connected with two other smaller blue circles indicates the Toffoli gate. The symbol containing the red square with the white H indicates the Hadamard gate. The gray boxes with the Z letter characterize the measurement operation. These are standard symbols used in quantum computing, to indicate logic gates. Each line indicate a qubit ( $q[0], \dots, q[4]$ ) and a classical bit ( $mq21, mq31, mq41$ ), where the results of the measurements are stored. At the end of the circuit, there are measurements for each qubit. As an example, the initial configuration with  $|q_0\rangle = 0, |q_1\rangle = 1, |q_2\rangle = 1$  is shown.

the reward state, then **one can** also get a superposition of possibilities as the output (Tables 4, 5, 6).

Table 4: Theoretical expectations for inputs and outputs of the proposed quantum circuit for a selection of eigenstates and state superpositions. The input is given by the values of  $(x, y)$ -position and reward values of robot  $R_i$ , that is, the message sent by  $R_i$  to  $R_j$ , the second robot. The output is thus the expected  $(x, y)$ -position to be reached by  $R_j$ . The reward of the first robot is just copied into the output, to have the same number of inputs and outputs, necessary for reversible gates in quantum computing. In fact, in the output, the reward is the one of  $R_i$ . The reward of  $R_j$  will be computed once the second robot actually will have reached the expected position.

label	$R_i$			(exp.) $R_j$		
	x	y	rew.	y	x	(rew.)
$A$	$R_y(1.9)$	1	1	1	$R_y(1.9)$	1
$B$	$R_y(1.9)$	0	1	0	$R_y(1.9)$	1
$C$	0	0	1	0	0	1
$D$	0	0	0	H	H	0
$E$	1	1	0	H	H	0
$F$	1	0	1	0	1	1
$G$	H	1	1	1	H	1
$H$	1	1	H	H (more 1)	H (1)	H
$I$	H	H	H	H	H	H
$L$	1	1	$R_y(1.9)$	1	1	$R_y(1.9)$
$M$	0	1	1	1	0	1
$N$	1	1	1	1	1	1

Table 5: Comparison and agreement between theoretical expectations (Table 4) and measured outcomes, obtained with the QASM simulator. Empty cells indicate 0 measurements. Screenshots of measurements  $A_1, \dots, N_1$  can be retrieved at [https://github.com/medusamedusa/3-robot/blob/main/QASM\\_simulator.zip](https://github.com/medusamedusa/3-robot/blob/main/QASM_simulator.zip). Here and in Table 6, the states indicate x, y, and reward, respectively.

label	states								agr.
	000	001	010	011	100	101	110	111	
$A_1$						345		679	✓
$B_1$		345		653					✓
$C_1$		1024							✓
$D_1$	245		265		254		260		✓
$E_1$	254		232		258		280		✓
$F_1$				1024					✓
$G_1$						540		484	✓
$H_1$	152		124		146		128	474	✓
$I_1$	247		259		270		248		✓
$L_1$	87		84		100		75	678	✓
$M_1$						1024			✓
$N_1$								1024	✓

Table 6: Comparison between the outcomes obtained with three IBM quantum computers (C), located in Bogotá (B), Lima (L), and Manila (M), for states  $A, \dots, N$  and agreement with respect to theoretical expectations (Table 5). Some relevant results in agreement with the expectations are highlighted in bold. Screenshots of  $A_2, \dots, N_2$  can be retrieved at <https://github.com/medusamedusa/>, in the zip files *quantum\_computer\_Bogota.zip*, *quantum\_computer\_Lima.zip*, and *quantum\_computer\_Manila.zip*, respectively.

label	C	states							agr.	
		000	001	010	011	100	101	110		111
$A_2$	B	177	197	99	136	129	<b>139</b>	74	82	~
	L	144	163	108	117	152	<b>160</b>	63	<b>117</b>	~
	M	88	78	83	119	70	<b>234</b>	88	<b>264</b>	✓
$B_2$	B	212	<b>181</b>	127	<b>146</b>	120	94	73	71	~
	L	240	<b>196</b>	136	156	126	75	53	42	~
	M	99	<b>283</b>	86	<b>302</b>	66	67	44	77	✓
$C_2$	B	144	<b>251</b>	88	128	111	132	78	92	✓
	L	199	<b>318</b>	100	130	122	168	65	22	✓
	M	114	<b>384</b>	70	96	69	129	84	78	✓
$D_2$	B	<b>368</b>	59	<b>141</b>	21	<b>294</b>	25	<b>106</b>	10	✓
	M	<b>248</b>	106	<b>174</b>	44	<b>225</b>	33	<b>173</b>	21	✓
	L	<b>260</b>	44	<b>250</b>	33	<b>208</b>	33	<b>179</b>	17	✓
$E_2$	B	<b>203</b>	66	<b>125</b>	59	<b>256</b>	90	<b>154</b>	71	✓
	L	304	51	218	41	216	26	131	37	✓
	M	259	41	184	34	229	51	174	52	✓
$F_2$	B	98	234	106	<b>209</b>	77	115	74	111	~
	L	165	257	111	<b>230</b>	106	49	74	32	~
	M	69	324	72	<b>280</b>	62	91	49	77	~
$G_2$	B	135	206	71	125	138	<b>188</b>	74	87	~
	L	120	137	106	73	109	<b>222</b>	114	<b>143</b>	✓
	M	55	177	45	129	75	<b>293</b>	59	<b>191</b>	✓
$H_2$	B	161	109	141	55	202	113	152	91	✓
	L	185	61	204	72	156	64	205	77	✓
	M	184	77	162	112	113	95	113	168	✓
$I_2$	B	198	140	137	121	155	89	110	74	✓
	L	231	104	143	94	216	62	106	68	~
	M	188	128	128	83	190	109	112	86	✓
$L_2$	B	224	85	192	100	122	93	119	89	x
	L	142	156	101	154	138	124	108	101	x
	M	128	107	148	123	102	110	77	<b>229</b>	✓
$M_2$	B	134	115	108	<b>192</b>	160	68	119	128	x
	L	146	173	127	199	149	118	106	86	x
	M	68	198	65	101	81	<b>339</b>	76	96	✓
$N_2$	B	147	154	112	253	58	108	65	127	x
	L	150	190	99	139	103	130	111	102	x
	M	67	98	81	304	78	104	76	<b>216</b>	~

360 Table 4 presents, for a selection of states, input values and expected values. Table 5 shows the agreement between these expectations and the output obtained with the QASM simulator. Table 6 shows the comparison between the outcomes obtained through three different IBM quantum computers, presenting a larger quantum noise.

365 Once our quantum circuit [is defined and tested](#) on simulators and quantum computers, [it can be embedded](#) into a code for a robotic swarm, to verify the impact of quantum-modeled local interactions on the swarm global behavior. Thus, we can now more precisely shape our matrices. In fact, in practical implementations with robots, we can either use the circuit of Figure 1 in a loop  
 370 until robots reach the target, or we can embed the circuit in a more complex code with several steps. Each step corresponds to a shot of the matrix, with broadcast information and consequent action of each robot, activating the sub-matrices corresponding to pairwise interaction with the most successful robot. This strategy [has been](#) chosen for the case study in Section 6. In the Appendix,  
 375 the matrices for each step of a run [are presented](#).

$$R_i(t) = \begin{pmatrix} x_i(t) \text{ (Where I am)} & \dot{x}_i(t) \text{ (Where I go)} & \text{send (A) on} & \text{send (B) on} \\ y_i(t) \text{ (Where I am)} & \dot{y}_i(t) \text{ (Where I go)} & \text{send (C) on} & \text{send (D) off} \\ \text{receive (A) on} & \text{receive (B) on} & \text{type e-puck} & \text{wheels 2} \\ \text{receive (D) on} & \text{receive (D) off} & \text{camera on} & \text{motor 1} \end{pmatrix} \quad (4)$$

Let us focus on a 3-robot toy swarm moving on the plane, analyzing in more detail the structure of matrices. In (4), positions  $x, y$  are considered as the quantum superposition of states, see Section 5.2 for details. Letters A, ..., D indicate different communication channels.

In each pairwise-interaction sub-matrix (eq. 5), the first diagonal block contains the information, sent by the  $i$ -th robot ( $R_i$ ), about its (x,y)-position probability amplitudes and reward. The second diagonal block contains the possible behavioral response of the  $j$ -th robot ( $R_j$ ), according to the reward of  $R_i$ . If the reward is high,  $R_j$  follows  $R_i$ ; otherwise, the position of the  $R_j$  robot

at the subsequent time interval is assumed as a quantum superposition of the possible outcome of x- and y-positions.

$$(R_i * R_j)(t) = \begin{pmatrix} \alpha_i^x(t) & \beta_i^x(t) & \gamma_i(t) & 0 \\ \alpha_i^y(t) & \beta_i^y(t) & 0 & \delta_i(t) \\ 0 & 0 & \alpha_j^x(t+1) & \beta_j^x(t+1) \\ 0 & 0 & \alpha_j^y(t+1) & \beta_j^y(t+1) \end{pmatrix} \quad (5)$$

380 In principle, [one](#) should evaluate six interaction terms. However, we can compute only three of them, halving computational times. In fact, the  $R_j$  gets signals from  $R_i$  and  $R_k$ , comparing this information with its own information on position and reward.  $R_j$  chooses which robot to follow (or, more precisely, which robot should enter the quantum circuit to decide the position) according to its highest reward.  $R_j$  remains stationary if it already has the highest  
 385 reward. In the presented pseudocodes (see the Appendix) and in the corresponding Jupyter/Python codes we created, it has been necessary to compute just one interaction term, having as inputs positions' and rewards' probability amplitudes of the robot with the highest reward, and as output, positions  
 390 probability amplitudes assigned to the other two robots. Small fluctuations are added to avoid a superposition of these two robots. An example of the matrix output, when the proposed quantum circuit is included in our ant-inspired code for three robots, is proposed at the end of the Appendix.

### 5.3. Pseudocodes

395 Before moving to the results, let us describe the pseudocodes used in our [method](#). We present here our original and improved algorithms, Algorithms 1 and 2, respectively.

---

**Algorithm 1** original quantum-gate driven search

---

```
1: Class Target and instances ( $T_1$ : Food,  $T_2$ : Nest)
2: Robots  $R_1 R_2, R_3$  as classes, values as attributes
3: if All robots have a high but not max reward: then
4:   little random variation of  $R_1$ 
5: end if
6: Inputs: initial robots' positions
7: Reward evaluation Robots  $i, j, k$  (as Euclidean distance from the target)
8: if  $R_{1,2,3}$  have a reward lower than a threshold: then
9:   Re-initialize randomly their positions and rewards
10: else Do not re-evaluate positions and rewards
11: end if
12: for  $i, j, k = 1, 2, 3$  do
13:    $R_i$  gets information from robots  $j, k$ 
14:   Evaluation of the highest probability of 'Yes' as reward
15:   if If the  $i$ -th robot had already the highest reward: then
16:     the other robots reach it
17:   end if
18: end for
19: Evaluation of the highest probability of 'Yes' as reward
20: if If the  $i$ -th robot has the highest reward: then
21:   it does not enter the circuit of Figure 1 for probabilistic decision-making
22:   it does not move
23: else if robot  $j$  has the highest reward: then
24:   x-y positions and reward of  $R_j$  at time  $t$  are the circuit inputs
25:   circuit output: most likely positions at  $t + 1$  for both  $R_i$  and  $R_k$ 
26:   the 2 most frequent configurations: arrays; occurrences: weights
27: end if
28: update positions, reward, graph
29: repeat lines 3, 8, 15
30: if  $R_{1,2,3}$  have a reward difference lower than a threshold: then
31:   lock robots in a GHZ. 1(0): higher-reward robot position +(-) fluctuation;
   measure of the qubits
32: end if
33: overall fluctuation or overall flip
34: Outputs: final robots' positions and final rewards
35: repeat from Line 3 to Line 34 to get back to the nest, calculating rewards with
   respect to the nest ( $T_2$ ).
```

---

---

**Algorithm 2** enhanced quantum-gate driven search

---

```
1: choose the number of robots
2: initialize robots' positions as state superpositions
3: if all rewards are below a certain threshold then
4:   for each robot do
5:     randomly reshuffle positions
6:     if a position hits the obstacle then
7:       reshuffle the position for that robot
8:     end if
9:   end for
10: end if
11: find the robot with the highest reward and let it enter the circuit
12: find the new suggested position through the circuit
13: for all robots do
14:   evaluate the new rewards
15: end for
```

---

Regarding time complexity, conditional statements have a complexity of  $O(n \log n)$ , and time loops of  $O(n)$ ; because our algorithms contain nested *for* and *if*, the complexity is of  $O(n)$ . Space complexity is also  $O(n)$ , because of the sorting function: *What is the most successful robot?*. The sorting function adopted for our examples is classical. The quantum part regards the computation of the logic gate  $R_i \rightarrow R_j$ . In next research, the sorting operation could be made quantum as well, using the Grover search, with  $O(\sqrt{n})$ .

## 405 6. Results: A case study

At the end of Section 5, the pseudocodes used for our method have been presented. In this Section, we analyze the effectiveness of our quantum circuit inside the algorithm, and then we compare it with other search methods. First, setup and conducted experiments based on your model are presented (subsection 410 6.1); then, a quantitative comparison between our method and the PSO variant

(subsection 6.2) and the NL-SHADE-RSP with midpoint algorithm (subsection 6.3) are discussed.

### 6.1. Setup and experiments with our model

Before presenting the implementation results, we recall that robot positions  
415 are defined here as quantum state superpositions: for  $R_i$ , there is a probability  
amplitude  $\alpha_i^x$  to stay in the point 0 of the x-axis,  $\beta_i^x$  to stay in the point 1 of  
the x-axis, and corresponding  $\alpha_i^y, \beta_i^y$  for the y-axis, respectively. If  $(\beta_i^x, \beta_i^y) =$   
 $(0.99, 0.99)$  is almost sure to find the robot in (1,1): visually, wave functions have  
a peak in (1,1). So, **one** can reasonably (within quantum indeterminacy) localize  
420  $R_i$  in (1,1). If  $(\beta_i^x, \beta_i^y) = (0.5, 0.5)$ , however, there is the same probability  
amplitude to find the robot in any point of the  $xy$ -plane. With a little abuse of  
notation, in this situation we indicate the robot in (0.5, 0.5), that is, considering  
the positions as the peaks of the wavefunctions describing the states: half-way  
between the two extremes of the  $[0, 1]$  segment along  $x$  and  $y$ . This is why  $\beta_i^x, \beta_i^y$   
425 **are considered** to build our (approximate) visual representations. The target **is**  
**treated** the same way, making possible the comparisons between target's and  
robots' positions.

The reward **is evaluated** as the distance from the target. It is the measure of  
 $\delta$ , the amplitude probability to get 1, that is, 'success.' This is a conceptual sim-  
430 plification: the robots cannot truly know their distance from the target. They  
can assess their approximate positions and distances interpreting their vision (if  
we are considering robot vision, or smell for ants), as 'yes,' 'no,' 'maybe.' In  
principle, robots can rotate to improve their camera vision, and they should also  
avoid eventual obstacles in their path. In future research, **one can model** robot  
435 vision, leaving the evaluation of rewards to the 'precision' of target proximity  
assessment. E.g., if only the 30% of target is visible on a camera, then that  
robot will have, as a reward, 30% or 'yes' and 70% of 'no.' The overall code  
would remain the same, except for the reward's evaluation.

Let us now describe an implementation of our method for a small-sized

440 robotic swarm. We start with the comparison of convergence efficiency<sup>7</sup> of our  
 decision-making system for swarms of different sizes. In Table 7, we consider  
 the enhanced and shortened code, allowing the customization of the number  
 of elements of the swarm. **Once can** notice that the scalability starts being  
 visible for  $N \geq 5$ . A more robust convergence is presented for 10 robots. Trials  
 445 performed with swarms of 15 units, not reported here, show results similar to  
 the ones obtained for 10 robots. In analogy with the biological model of foraging  
 ants [82], in this analysis we consider the motion from the nest to the source  
 of food (the target). In the table, the closer the average reward to 1, the more  
 successful the swarm.

Table 7: We present a comparison between the rewards for swarms of different sizes, with  
 $N = 2, 3, 5, 10$  robots, respectively. For these simulations, we used the second (and shorter)  
 algorithm, and we placed the nest (start) in  $(0.2, 0.9)$ , and the food (target) in  $(0.9, 0.2)$ .

		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

<sup>7</sup>While talking about our algorithms, we use the term “efficiency” to indicate the reduction  
 of code lines and the improved accuracy in target reaching, shown by the second code.

450 Let us now observe in detail the performance with the longer algorithm,  
complete with GHZ, for a 3-robot swarm. Following the biologic example of  
ants, we consider a motion back-and-forth between the nest ( $T_2$ ) and the food  
location ( $T_1$ ), evaluating robots' behavior in terms of their rewards (Table 8).  
When the convergence is not optimal, additional run of the code allow to reach  
455 a more precise convergence. In the majority of our tests, the robots converge to  
the targets, showing a concordance with the model. In the lower part of Table  
8, the food position is [changed](#).

Table 8: Top: Comparative table with the results of eight runs of the first code (longer),  
between two targets:  $T_2$ , the food source, and  $T_1$ , the nest, in analogy with the Ant Lines  
model. Numbers in bold highlight the most relevant contributions of the gate to improve  
rewards. The arrival position of the first path is the starting point for the second path. The  
last column shows the convergence, which is successful if the final rewards are  $\geq 0.8$ , and  
approximate if  $\geq 0.6$ ,  $< 0.8$ . Bottom: Robots' behavior in terms of reward time points for  
changing food locations. Details of the gate states outcomes and populations can be found  
at <https://github.com/medusamedusa/3-robot>; files test(number)\_Jan\_29.pdf (table above)  
and trial(number)\_different\_food\_Jan\_29.pdf (table below).

test	path	initial rewards	rewards before circuit	rewards after circuit	rewards before GHZ	final rewards
1	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	(0.68, 0.6, 0.57)	(0.68, 0.8, 0.53)	(0.8, 0.8, 0.82)	(0.85, 0.85, 0.85) ✓
	$\rightarrow T_2$	(0.15, 0.15, 0.15)	<b>(0.58, 0.56, 0.38)</b>	<b>(0.58, 0.78, 0.38)</b>	(0.78, 0.86, 0.81)	(0.92, 0.92, 0.92) ✓
2	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	(0.78, 0.84, 0.81)	(0.7, 0.84, 0.69)	(0.87, 0.84, 0.53)	(0.81, 0.81, 0.81) ✓
	$\rightarrow T_1$	(0.25, 0.25, 0.25)	(0.9, 0.93, 0.86)	(0.68, 0.93, 0.86)	(0.93, 0.93, 0.89)	(0.9, 0.9, 0.9) ✓
3	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	<b>(0.76, 0.55, 0.41)</b>	<b>(0.76, 0.6, 0.6)</b>	(0.7, 0.7, 0.7)	(0.67, 0.67, 0.67) ~
	$\rightarrow T_1$	(0.33, 0.33, 0.33)	(0.86, 0.8, 0.83)	(0.86, 0.78, 0.83)	(0.78, 0.76, 0.83)	(0.83, 0.83, 0.83) ✓
4	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	<b>(0.78, 0.56, 0.25)</b>	<b>(0.78, 0.78, 0.78)</b>	(0.78, 0.78, 0.78)	(0.67, 0.67, 0.67) ~
	$\rightarrow T_1$	(0.41, 0.41, 0.41)	<b>(0.41, 0.41, 0.41)</b>	<b>(0.7, 0.7, 0.41)</b>	(0.7, 0.7, 0.7)	(0.77, 0.77, 0.77) ~
5	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	(0.79, 0.4, 0.65)	(0.79, 0.52, 0.67)	(0.79, 0.71, 0.71)	(0.77, 0.77, 0.77) ~
	$\rightarrow T_1$	(0.53, 0.53, 0.53)	(0.53, 0.53, 0.53)	(0.46, 0.46, 0.53)	(0.89, 0.89, 0.93)	(0.85, 0.85, 0.85) ✓
6	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	<b>(0.68, 0.21, 0.79)</b>	<b>(0.8, 0.8, 0.79)</b>	(0.78, 0.8, 0.6)	(0.77, 0.77, 0.77) ~
	$\rightarrow T_1$	(0.29, 0.29, 0.29)	(0.35, 0.42, 0.53)	(0.46, 0.46, 0.53)	(0.7, 0.7, 0.72)	(0.72, 0.72, 0.72) ~
7	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	(0.92, 0.9, 0.9)	(0.92, 0.9, 0.9)	(0.9, 0.69, 0.98)	(0.9, 0.9, 0.9) ✓
	$\rightarrow T_1$	(0.25, 0.25, 0.25)	(0.63, 0.57, 0.66)	(0.46, 0.66, 0.66)	(0.75, 0.75, 0.66)	(0.66, 0.66, 0.66) ~
8	$T_2 \rightarrow T_1$	(0.3, 0.34, 0.29)	(0.64, 0.62, 0.59)	(0.64, 0.77, 0.54)	(0.77, 0.77, 0.6)	(0.81, 0.81, 0.81) ✓
	$\rightarrow T_1$	(0.38, 0.38, 0.38)	(0.86, 0.93, 0.87)	(0.59, 0.93, 0.87)	(0.93, 0.96, 0.91)	(0.92, 0.92, 0.92) ✓
food position ( $T_1'$ )	path	initial rewards	rewards before circuit	rewards after circuit	rewards before GHZ	final rewards
(0.2, 0.9)	$T_2 \rightarrow T_1'$	(0.6, 0.6, 0.64)	<b>(0.6, 0.6, 0.64)</b>	<b>(0.86, 0.86, 0.64)</b>	(0.81, 0.86, 0.82)	(0.81, 0.81, 0.81) ✓
	$\rightarrow T_2'$	(0.65, 0.65, 0.65)	(0.65, 0.65, 0.65)	(0.68, 0.64, 0.75)	(0.74, 0.84, 0.82)	(0.92, 0.92, 0.92) ✓
(0.9, 0.9)	$T_2 \rightarrow T_1'$	(0.19, 0.23, 0.2)	(0.6, 0.6, 0.64)	(0.2, 0.23, 0.6)	(0.98, 0.94, 0.98)	(0.93, 0.93, 0.93) ✓
	$\rightarrow T_2$	(0.65, 0.65, 0.65)	(0.86, 0.88, 0.91)	(0.86, 0.88, 0.91)	(0.74, 0.84, 0.82)	(0.91, 0.91, 0.91) ✓
(0.5, 0.0)	$T_2 \rightarrow T_1'$	(0.42, 0.44, 0.38)	(0.5, 0.44, 0.5)	(0.5, 0.5, 0.5)	(0.55, 0.55, 0.5)	(0.71, 0.71, 0.71) ~
	$\rightarrow T_2$	(0.69, 0.69, 0.69)	(0.69, 0.69, 0.69)	(0.78, 0.7, 0.69)	(0.76, 0.76, 0.76)	(0.81, 0.81, 0.81) ✓

From our tests, [one can see that](#) the proposed gate works better when the robots are in exploration phase, that is, in sight of the target but not very close to it. In a real application, [one](#) can argue that, if robots are all already close enough to the target, they are not required to enter the gate. In the majority of cases, the gate actually improves the robot reward statuses. Fluctuations are justified by the probabilistic nature of this approach. Our toy-swarm converges to the target (final rewards  $\geq 0.8$ ) in 14 out of 19 paths, and it reaches some closeness (final reward  $\geq 0.6$ ) in all paths.

Here, we propose the entanglement (GHZ state) to let robots act as a whole system in the latest steps of the search process. In other studies, the entanglement condition is imposed right at the beginning [67]. However, here we avoid any strong hypothesis in the first steps of the process, [preferring](#) instead to focus on the proposed circuit action.

Figure 2 shows the path simulation obtained with the shorter code and 10 robots. [One can](#) notice that, to reach the target, a smaller number of passages is required, and the convergence is high (the average reward is 0.925). Considering as metrics the number of direction changes across the different methods, we can qualitatively assess that, even in its original version, our method requires a smaller number of turns for the robots. Moreover, in the shorter version of the code, there is only one change of direction, after the computation of the gate. The number of direction changes for the bounded round walk and the NetLogo simulation is visibly higher than 1. To make the code more general, we included a known obstacle (indicated by the red pentagon), that is avoided by the swarm. If, after the random generation of positions (step 1), the obstacle position would be hit by a robot, a new cycle of random-number generation is activated.

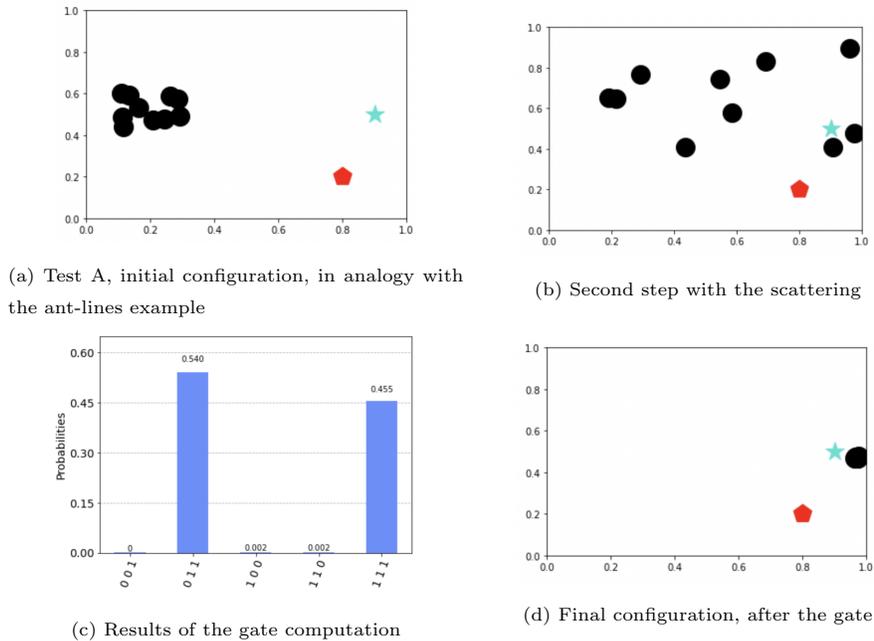


Figure 2: Same setup of Figure 3, improved code (the second and shorter one) and results of the test for 10 robots. The red pentagon indicates an obstacle which is avoided. The complete output of this test is in the PDF *testA.pdf*, available in the shared Git folder [https://github.com/medusamedusa/10\\_little\\_ants](https://github.com/medusamedusa/10_little_ants).

## 6.2. Comparison with a PSO variant

In this subsection, we first present a qualitative comparison with another ant-colony model, coded in NetLogo; then, we focus on a PSO variant. Our method is applied to a 3-robot swarm for the first case, and to a 10-robot swarm in the second case. Figure 3 also shows a qualitative visual comparison between paths back-and-forth in Test 1, a similar setup with Ant Lines in NetLogo and a Python-made 3-objects bounded random walk. A video simulation for Test 1 is available at <https://github.com/medusamedusa/3-robot>.

Let us now present a comparison between particle swarm optimization (PSO) approach [83] and our proposed method. See the Appendix for the code references. Our results are shown in Table 9 and Figure 4. For both approaches, we consider a  $1 \times 1$  square. However, PSO is conceptually different from our

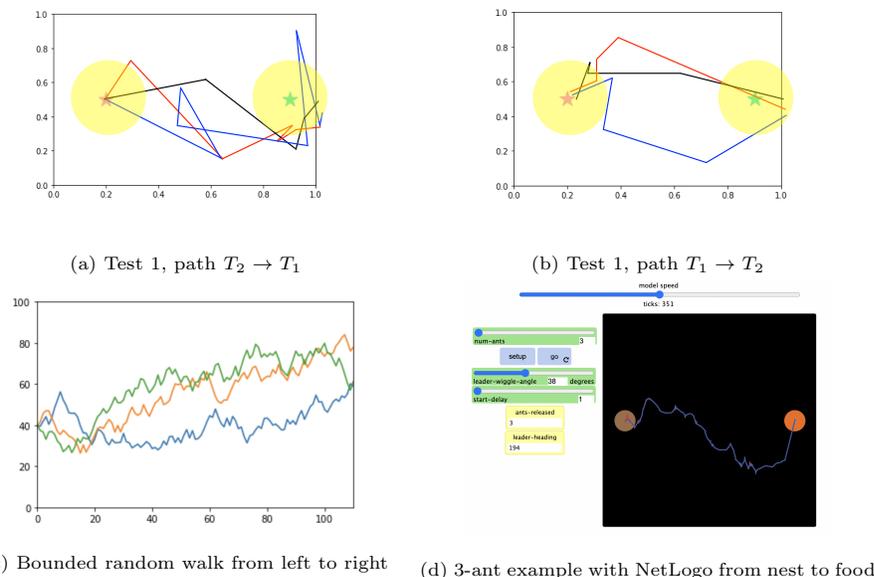


Figure 3: Visual comparison between paths for our setup, ant lines, and bounded random walks. For the simulations of our method, we used here the first (longer) code. In (a) and (b): the red star indicates the center of the nest, and the turquoise star indicates the food source. The yellow disks highlight the range of convergence. As one can see in Figure 2, with the enhanced and shortened code, the convergence is significantly improved, presenting a smaller number of steps. Computational details of the bounded random walks and the 3-ant example with NetLogo are provided in the Appendix.

approach. Our method is not an optimization technique. For our PSO example, there is a surface living in three dimensions. The equivalent of the target is the minimum of the surface. We define an object function, that is,  $f(x, y) = (x - 0.9)^2 + (y - 0.5)^2$ , having its minimum in  $(0.9, 0.5)$ , the same point where the considered target is. In Table 10, the considered target is  $(0.8, 0.9)$ , and the objective function is changed accordingly. Running the simulation, we estimate the precision of PSO particles in making it to the target. Starting with the PSO with the same initial conditions of our approach (all robots in a small cluster centered in a point of the plane), obtained with

$$X = \text{np.random.rand}(2, \text{n\_particles}) * 0.1 + 0.2,$$

$$Y = \text{np.random.rand}(2, \text{n\_particles}) * 0.1 + 0.2,$$

( $X, V$  in the code), the convergence is 10 times less precise than for our method.

Starting with the PSO with all-scattered robots, that is, with

$$X = \text{np.random.rand}(2, \text{n\_particles}) * 0.9,$$

$$Y = \text{np.random.rand}(2, \text{n\_particles}) * 0.01,$$

then the target is more precisely reached. In both tests with PSO, 49 iterations are considered. However, with our method, only four passages are needed. To quantitatively compare our results against the PSO ones, we compute distances (Euclidean, Manhattan, cosine dissimilarity) between the swarm barycenter and the target coordinates. Results are presented in Table 10. We consider PSO taking into account the constraints of our scenario. When the initial parameters are the same as the ones we used, the results obtained with our approach appear as more precise. In fact, the distance values between the barycenter of the swarm and the target are smaller, see Table 10. If the PSO initial condition is the scattered search rather than all particles in a “nest,” then in one case PSO outperforms our method (test f), and in another one, our method outperforms PSO (test n). However, these are only initial considerations. In future research, one can exploit machine learning to run further tests, and statistical techniques to analyze the results, also taking into account recent developments of PSO [35], and enhancements of ant-colony approaches with the genetic algorithm (GA) [36, 37]. More details about future improvements, also inspired by GA-derived and differential evolution-based competition-winner algorithms [34, 84], are discussed in Section 7.

Table 9: Comparison between our method (second code) and a PSO test (see Figure 4). The PDF with the complete outputs can be found in the folder [https://github.com/medusamedusa/10\\_little\\_ants/tree/main/test\\_PSO](https://github.com/medusamedusa/10_little_ants/tree/main/test_PSO).

	nest (starting points)	food (target)	error
expected	(0.6, 0.6)	(0.9, 0.5)	0.
our method	centered in (0.6, 0.6)	(0.89, 0.65/0.66)	0.01
PSO - test 1	centered in (0.6, 0.6)	(0.79..., 0.59,...)	0.1
PSO - test 2	scattered	(0.90, 0.49)	0., 0.01

Table 10: Distance comparison between the position of the robotic-swarm barycenter and the target coordinates, at the end of the search process. In tests (a)-(e) and (g)-(m), the starting point of robots are centered in the nest at (0.6,0.6), while in tests (f) and (n) they are scattered through the arena (that is, with an initial condition that differs from ours). The pairs of tests (d),(e) and (l),(m) lead to the same numerical results, respectively. The complete outputs can be found in the folder [https://github.com/medusamedusa/10\\_little\\_ants/tree/main/test\\_distance\\_comparison\\_PSO](https://github.com/medusamedusa/10_little_ants/tree/main/test_distance_comparison_PSO).

test	method	start	target	barycenter/target distance		
				Euclidean	Manhattan	Cosine
a	our	centered	(0.9, 0.5)	0.047	0.066	0.001
b	our	centered	”	0.124	0.131	0.006
c	our	centered	”	0.019	0.025	$2.95 * 10^{-6}$
d	PSO	centered	”	0.137	0.193	0.010
e	PSO	centered	”	0.137	0.193	0.010
f	PSO	scattered	”	$7.65 * 10^{-5}$	$7.70 * 10^{-5}$	$2.90 * 10^{-9}$
g	our	centered	(0.8, 0.9)	0.0196	0.025	0.0001
h	our	centered	”	0.071	0.099	0.0018
i	our	centered	”	0.126	0.155	0.0014
l	PSO	centered	”	0.418	0.506	0.057
m	PSO	centered	”	0.418	0.506	0.057
n	PSO	scattered	”	0.386	0.462	0.0497

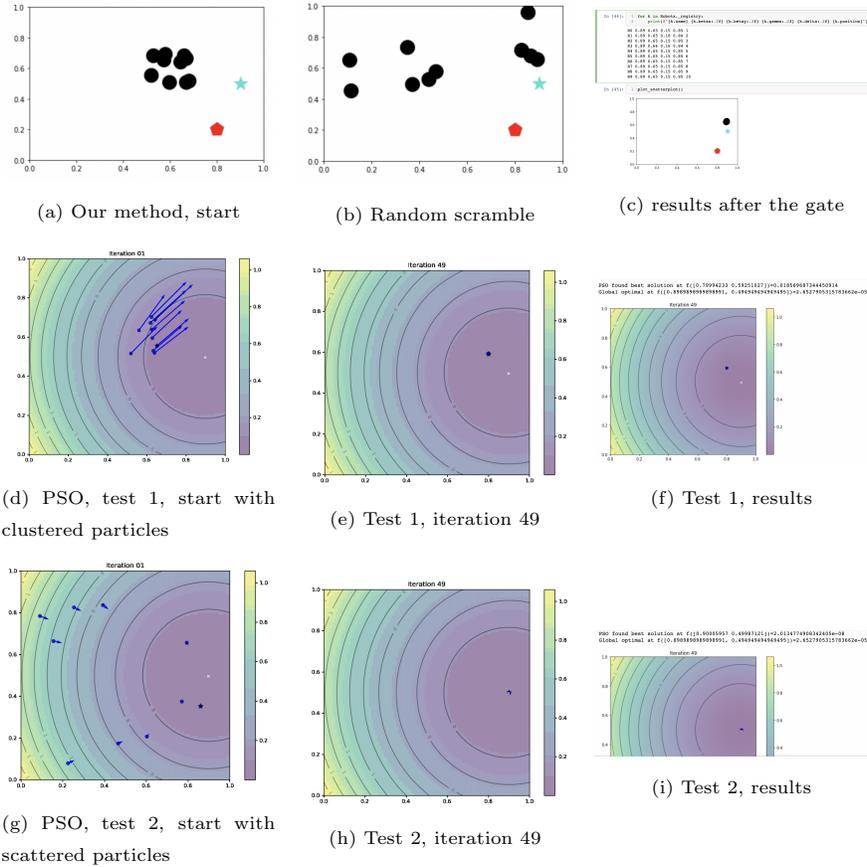


Figure 4: A comparison between our method (a-c), applied to 10 robots, for a target in (0.9, 0.5), clustered robots around (0.6, 0.6), and an obstacle, with an example of PSO (d-i), where the target is represented by the minimum of an objective function.

### 6.3. Comparison against an NL-SHADE-RSP algorithm

510 Finally, we performed a comparison between the precision of the results achieved with our method, and with a winner of a CEC-winner in 2022. In particular, we focused on a differential evolution-based algorithm, NL-SHADE-RSP with midpoint [34]. During the competition, the code has been used to optimize objective functions. Here, we wrote two new objective functions, having global  
515 minima in correspondence of two target locations of our scenario, respectively. In [34], the conditions of our scenario are reproduced, with a population of ten units initially starting in a disk centered on  $(0.6, 0.6)$  with a radius of 0.1. The exploration space is a square of side  $[0, 1]$ . A total of 25 runs for each one of the two objective functions have been completed. Table 11 shows the data of first  
520 evaluations of the two functions in ten different runs for each objective function. The objective function is evaluated 300 or 400 times to reach the target with an error of  $10^{-9}$ . After one or three evaluations, the results have an error of the same order of magnitude of the results attained with our methodology: see test a-c and test g-h from Table 10. After four evaluations, the error is of  
525 the same order of magnitude, or one order less. The advantage shown by our methodology (which leads to suboptimal solutions) is the simplicity of passages and the precision corresponding to the first evaluations of an advanced and way more complex optimization algorithm. It should be precised that the method NL-SHADE-RSP [34] has not been devised for small functions, and that our  
530 methodology is not an optimization technique. Thus, such a comparison gives an idea of the orders of magnitude of the error a target is reached with, in our 4-step code, and according to the first evaluations of a recent and efficient algorithm.

Table 11: Outputs of a recent version of NL-SHADE-RSP with midpoint [34], tested on two different objective functions (f. obj. 1 and f. obj. 2), having as minima the targets in our scenario: (0.9, 0.5) and (0.8, 0.9), respectively. Population is constituted by 10 units, as in our test of Table 10. The complete data can be retrieved at [https://github.com/medusamedusa/10\\_little\\_ants/tree/main/SHADE](https://github.com/medusamedusa/10_little_ants/tree/main/SHADE).

run	no. of evaluations	distance from the target	
		f. obj. 1	f. obj. 2
1	1	0.0564498	0.129196
	4	0.00699566	0.121177
	50	0.00428091	0.0590199
2	1	0.052212	0.0804435
	4	0.052212	0.0804435
	50	0.00898058	0.0434829
3	1	0.0368419	0.0804435
	4	0.0135726	0.0804435
	50	0.000914063	0.0434829
4	1	0.0135726	0.126022
	4	0.0135726	0.0887293
	50	0.000914063	0.026067
5	1	0.0961737	0.218962
	4	0.0785412	0.109809
	50	0.035633	0.0689708
6	1	0.158216	0.167935
	4	0.0593486	0.0561922
	50	0.0136765	0.0326968
7	1	0.0320644	0.170921
	4	0.0320644	0.147435
	50	0.000156165	0.0736671
8	1	0.0583604	0.0661496
	4	0.0583604	0.0661496
	50	0.0140189	0.0661496
9	1	0.0961737	0.218962
	4	0.0785412	0.109809
	50	0.035633	0.0689708
10	1	0.0583604	0.0804435
	4	0.0583604	0.0804435
	50	0.0140189	0.0434829

## 7. Discussion

535 Addressing the problematics of a formal description of robotic swarms, we developed a matrix-based approach distinguishing between single robots' information and pairwise interactions. Then, we focused on interactions, developing a quantum-based decision model to connect information exchange and local decisions with the overall swarm behavior. We devised a strategy and implemented  
540 it for 3-robot and 10-robot toy swarms. The proposed strategy could constitute a new software design paradigm, with communications included in the feedback loop. The core idea of our study can in the future be enriched making the whole system scalable, finding patterns of behavior, adding one more spatial dimension, and introducing learning.

545 While we worked with three and ten robots only, our algorithm can be extended to a generic number of robots. The structure of the matrix would remain the same. Scalability is thus possible. From 3 to  $N$  robots, **one** would have no longer  $3! = 6$  but  $N!$  interaction sub-matrices. However, given the described patterns of behavior, with only the more successful robot entering  
550 the circuit, also in the case of  $N$  robots, the evaluation of only one interaction term could just be required. At time  $t$ , all robots broadcast information about their reward. The position of the only robot with the highest reward enters the decision system of each robot, allowing them to move accordingly at  $t + 1$ . Other steps of reshuffle (see the Pseudocode in Materials and Methods) allow  
555 robots to improve their exploration also in cases of low reward for all of them.

Here, the reward has been evaluated as the distance from the target. In the future, this information might be recovered through camera observations, sound recognition, or even odor recognition, as it happens for ants with pheromones. In any case, the main structure of the algorithm would remain unchanged.

560 Our study involved robots in the plane. This is an extension of our first approach with movements along a line [11]. **One** can extend the present study by adding one more spatial dimension, as sketched in [85]. The circuit does only require one more qubit. This change does not alter the core idea. A

tridimensional motion would make the modeling of swarm robotic motion in  
565 the air or underwater possible. In our study, we first considered the ideal model  
of motion in an obstacle-less two-dimensional space, with battery-less robots.  
Then, in our enhanced code, we included a known obstacle. The passage from  
simulation to real robots would of course require some battery information. The  
precision has been greatly improved through the enhanced and shortened code.  
570 It includes the nest→food path, allowing robots to quickly reach the target  
without the need for the GHZ passage. We also quantitatively measured the  
different precision of target reaching achieved with our method and with an  
application of particle swarm optimization. While we are not considering an  
optimization approach here, we set up the parameters of a PSO example [and of](#)  
575 [a NL-SHADE-RSP algorithm \[34\]](#) to match the characteristics of our scenario.  
When we have the same starting parameters, the results obtained with our  
approach appear as being more precise. Refined approaches of machine learning  
will be exploited to investigate in more detail the advantages of our code with  
respect to existing classic approaches, and ultimately to refine our method. We  
580 considered a basic example of PSO for its conceptual importance, being it at  
the base of recent and valuable developments [35]. Regarding the quantum  
improvement of classical ant-foraging approaches, we qualitatively considered  
the number of steps and direction changes in the NetLogo application and in  
our simulation. Our robots reach the target with less changes of directions, that  
585 is, with different steps. However, this article presents the methodology. More  
detailed evaluations of performance time will be considered in future research.  
In our current research, we considered sub-optimal solutions. However, in our  
case, the focus was on the methodology definition, rather than on the search of  
an optimal solution. In addition, our methodology is general. It is specialized  
590 into a specific task according to the choices made for the logic gate. Thus, the  
comparison we have run are merely examples of the results we can obtain with  
respect to other methods.

### 7.1. *Hints toward future research*

In next steps of the research, **one** can take into account the progressive  
595 refinement of its applications. We can find out which logic gate or which degree  
of parameters' detail may yield the more precise results. And, only in that case, a  
precise comparison with optimization approaches can make sense. At that point,  
we will dig into the most recent and effective approaches of PSO [35], GA + ant-  
colony [36], competition-winners of GA-based **and differential evolution-based**  
600 algorithms [34, 84] (CEC 2022 Bound Constrained Single Objective Numerical  
Optimization benchmark problems). These comparisons would, in turn, help  
us refine the logic gates to obtain the best results. We have already considered  
NL-SHADE-RSP with midpoint [34] for a first comparison, commenting the  
results of Table 11 with respect to the outputs of our methodology shown in  
605 Table 10. We noticed that the error in target-reaching with our 4-step method,  
leading to suboptimal solutions, are of the same order of magnitude of the **first**  
objective-function evaluations in [34]. In general, the idea of genetic selection  
could correspond, for us, to the attention devoted to the most successful robots.  
However, the other robots would not die or be substituted; instead, they would  
610 just follow the more successful ones. As an example of how these connections  
could be developed, let us consider the key ideas of the CEC-winner, **differential  
evolution-based** algorithm proposed by Biedrzycki and co-authors [34], called  
NL-SHADE-RSP. It is a GA-derived algorithm where the k-means is used to  
split into half large populations, evaluating their midpoint. In our case, we could  
615 run experiments with a large number of robots, and split the population into  
two parts, distinguishing between the robots closer and less close to the target.  
The bound constraint can be given, in our case, by the size of the arena (as  
we already did for PSO). The objective could be a function having a minimum  
or so in a neighborhood of the target (similarly to what we did for PSO **and**  
620 **NL-SHADE-RSP**). The step of population size diminution can lead, in our case,  
to the shift of attention toward only those robots that are really close to the  
target. In the following step, **one** would just give all other robots the command  
to reach their most successful peers. From such a comparison, we could refine

our search-and-rescue applications, getting to the target with a limited number  
625 of passages and with a greater precision. We could also, quantitatively, find out  
how to tune the parameters within our matrices to get scenarios and results as  
close as possible to NL-SHADE-RSPv[34] and other recent works [84].

As another future development, learning can be added by defining arrays of  
“probable target locations.” This can easily be done for  $T_2$ , with robots storing  
630 their initial position if, as in our examples, they start from the nest. It can  
also be done for food position ( $T_1$ ): once a robot obtains a reward  $\geq 0.8$ , it can  
store its (probable) position within an array, and communicate this information  
to the other robots. If, during subsequent exploration steps, the reward of a  
robot gets higher than the stored information, the memory array is updated.  
635 Otherwise, if rewards are constant or diminish, at least one robot can get back  
to the stored position. The entire algorithm can be performed as a loop, and  
the search can be stopped once the stored position corresponds to a reward of  
around 0.9. Thus, [one](#) would have a loop while the rewards are  $\leq 0.9$ .

The introduction of learning would connect our research with Quantum Rein-  
640 forcement Learning studies, with discrete successive states and decisions [81, 80].  
Our research may thus constitute a key connection between quantum comput-  
ing, swarm robotics, swarm modeling, and future developments in this research  
area. As mentioned above, we found that the proposed gate works better when  
the robots are in exploration phase, rather than already close to the target. The  
645 enhancement of the code eliminated this issue.

## 8. Conclusions

In this article, we proposed a quantum approach to swarm robotics. Our  
strategy is a two-step modeling technique, where we first model the individual  
behavior, and then we tune the swarm to observe the emerging behavior.

650 Our approach to the swarm is based on nested matrices. Diagonal sub-  
matrices represent information (position, speed, activated sensors) of each in-  
dividual robot. Off-diagonal sub-matrices are the pairwise interaction terms,

containing the information about sent and received messages, and suggestions of behavior according to the received signals. A core element of our approach is the use of an original quantum circuit. To test it and measure the effect of quantum noise, we compared theoretical expectations of our circuit with measurements obtained through IBM QASM simulator and three real IBM quantum computers. To test our approach on a swarm of robots, our circuit was then included in a Jupyter Notebook original code, written in Python. This idea can lead to a quantum-based decision-making system.

We simulated the behavior of a 3-robot swarm and then of a 10-robot swarm as case studies. We considered a search and rescue mission, inspired by the foraging ant behavior, for its simplicity and importance in nature. We obtained an overall success in target finding (food retrieval and return to the nest), observing the circuit effect on swarm behavior. We compared our method with other case studies coded in NetLogo, with bounded random walks, and with an optimization approach of particle swarm, trying to recreate conditions similar to the ones in our research. We also compared our results against two optimization algorithms, the particle swarm optimization (PSO) and the NL-SHADE-RSP with midpoint algorithms.

In our code (second version), a known obstacle to be avoided is also included. The size of the swarm can be modified by suitably setting the  $N$  number of robots. The study has been developed for a 2-dimensional scenario, but it can be easily generalized to a 3-dimensional scenario, as sketched in [85]. In our study, we chosen to focus on a simplified scenario to validate the approach. We deliberately ignored unknown obstacles, battery information, information storage, and techniques for target detection. Regarding the last point, in our current simulation, we considered the Euclidean distance as a measure of target proximity. However, in a realistic scenario, we should consider visual feedback, or infrared detection, or sonar-collected information to name but a few. A limitation on the feasibility of our approach is the access to quantum simulators and the risk of decoherence, that are common issues for all quantum applications.

Swarm robotics is one of the countless technologies inspired by nature, with

its swarms and self-organizational structures. Another natural source of inspi-  
685 ration for computer science is quantum physics. Our research aims to merge  
nature-inspired swarms and quantum computing. Our work constitutes one  
of the first steps toward the adaptation of the quantum paradigm to swarm  
robotics development. The resources of quantum computing have started being  
explored in full only in the latest years, and the extension of their computational  
690 power to artificial intelligence and robotic development is yet to come.

### **Funding**

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  
695 ARS01\_00682.

### **Author contributions**

M.M., V.S., and A.C. designed the experiments; M.M. conducted the experi-  
ments; M.M. and V.S. wrote the paper; M. M. revised the manuscript according  
to the reviewers’ and editor’s comments.

### **Acknowledgments**

The authors express their heartfelt thanks to Dr. Rafał Biedrzycki from the  
Warsaw University of Technology (Poland), first author of [34], and Salvatore  
Zammuto from the University of Palermo (Italy), for their help with understand-  
ing, adapting to our objective functions, and running the code NL-SHADE-RSP  
705 with midpoint [34].

### **Declaration of interests**

The authors declare no competing interests.

### **Data availability**

All [codes and](https://github.com/medusamedusa/) data are available at <https://github.com/medusamedusa/>  
710 [3-robot](https://github.com/medusamedusa/3-robot) and [https://github.com/medusamedusa/10\\_little\\_ants](https://github.com/medusamedusa/10_little_ants). These are  
public repositories, copyright M.M.

## References

- [1] R. Eberhart, Y. Shi, J. Kennedy, *Swarm Intelligence*, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufman, Burlington, MA, USA, 2001.
- [2] C. Noirot, J. P. E. C. Darlington, *Termite nests: Architecture, regulation and defence*, in: T. Abe, D. E. Bignell, M. Higashi (Eds.), *Termites: Evolution, Sociality, Symbioses, Ecology*, Springer Netherlands, Dordrecht, 2000, pp. 121–139. URL: [https://doi.org/10.1007/978-94-017-3223-9\\_6](https://doi.org/10.1007/978-94-017-3223-9_6).
- [3] D. Knebel, C. Sha-ked, N. Agmon, G. Ariel, A. Ayali, *Collective motion as a distinct behavioral state of the individual*, *iScience* 24 (2021) 3.
- [4] C. H. Hemelrijk, H. Hildenbrandt, *Schools of fish and flocks of birds: Their shape and internal structure by self-organization*, *Interface Focus* 2 (2012) 726–737.
- [5] J. Delcourt, N. Bode, M. Denöel, *Collective Vortex Behaviors: Diversity, Proximate, and Ultimate Causes of Circular Animal Group Movements*, *The Quarterly Review of Biology* 91 (2016) 1–24.
- [6] K. Zhu, M. Jiang, *Quantum Artificial Fish Swarm Algorithm*, in: *Proceedings of the 8th World Congress on Intelligent Control and Automation*, 2010.
- [7] S. Berman, Q. Lindsey, M. S. Sakar, V. Kumar, 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 (2011) 1470–1481.
- [8] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. Coello Coello, F. Herrera, *Bio-inspired computation: Where we stand and what’s next*, *Swarm and Evolutionary Computation* 48 (2019) 220–250.

- [9] J. Preskill, Quantum computing 40 years later, in: A. Hey (Ed.), Feynman  
740 Lectures on Computation, 2nd ed., Taylor & Francis Group, Abingdon,  
UK, 2021.
- [10] A. Wichert, Principles of Quantum Artificial Intelligence, World Scientific,  
Singapore, 2020.
- [11] M. Mannone, V. Seidita, A. Chella, Categories, Quantum Computing, and  
745 Swarm Robotics: A Case Study, *Mathematics* 10 (2022) 372.
- [12] F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, M. Puviani, On Self-  
adaptation, Self-expression, and Self-awareness in Autonomic Service Com-  
ponent Ensembles, in: 2011 Fifth IEEE Conference on Self-Adaptive and  
Self-Organizing Systems Workshops, 2011.
- 750 [13] W.-M.-. Shen, P. Will, A. Galstyan, C.-M. Chuong, Hormone-Inspired Self-  
Organization and Distributed Control of Robotic Swarms , *Autonomous  
Robots* 17 (2020) 93–105.
- [14] L. Pitonakova, R. Crowder, S. Bullock, Task Allocation in Foraging Robot  
Swarms: The Role of Information Sharing , *Artificial Life Conference  
755 Proceedings: ALIFE 2016, the Fifteenth International Conference on the  
Synthesis and Simulation of Living Systems* 17 (2020) 93–105.
- [15] I. Slakov, C. Carrillo-Zapata, D. Jansson, H. Kaandorps, J. Sharpe, Mor-  
phogenesis in robot swarms, *Science Robotics* 3 (2018).
- [16] X. Dong, M. Sitti, Controlling two-dimensional collective formation and  
760 cooperative behavior of magnetic microrobot swarms, *The International  
Journal of Robotic Research* 39 (2020) eabe4385.
- [17] E. Šahin, Swarm robotics: From sources of inspiration to domains of ap-  
plication, in: E. Šahin, W. M. Spears (Eds.), *International Workshop  
on Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*,  
765 Springer, Berlin/Heidelberg, 2004.

- [18] H. Hamann, *Swarm Robotics: A Formal Approach*, Springer, Cham, 2018.
- [19] M. Schranz, M. Umlauf, M. Sende, W. Elmenreich, *Swarm Robotic Behaviors and Current Applications*, *Frontiers in Robotics and AI* 7 (2020).
- [20] M. Schranz, G. A. Di Caro, T. Schmickl, W. Elmenreich, F. Arvin, A. Şekercioğlu, M. Sende, *Swarm intelligence and cyber-physical systems: Concepts, challenges and future trends*, *Swarm and Evolutionary Computation* 60 (2021) 100762.
- [21] P. Suárez, A. Iglesias, A. Gálvez, *Make robots be bats: specializing robotic swarms to the bat algorithm*, *Swarm and Evolutionary Computation* 44 (2019) 113–129.
- [22] J. Kennedy, R. Eberhart, *Particle swarm optimization*, in: *IEEE Proceedings, IEEE*, 1995, pp. 1942–1948.
- [23] D. Paez, J. P. Romero, B. Noriega, G. A. Cardona, J. M. Calderon, *Distributed particle swarm optimization for multi-robot system in search and rescue operations*, *Science Direct, IFAC papers online* 54 (2021) 1–6.
- [24] A. S. Kumar, G. Manikutty, B. Rao, M. S. Couceiro, *Search and rescue operations using robotic darwinian particle swarm optimization*, in: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017.
- [25] Y. Sun, X. Ling, Q. Hu, S. Biancardo, *Exploring maritime search and rescue resource allocation via an enhanced particle swarm optimization method*, *Journal of Marine Science and Engineering* 10 (2022) 9067.
- [26] J. Holland, *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Univ. Michigan Press, 1975.
- [27] D. Simon, *Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence*, John Wiley & Sons, Hoboken, New Jersey, 2013.

- [28] S. Doncieux, N. Bredeche, J.-P. Mouret, A. Eiben, Evolutionary robotics: what, why, and where to, *Front. Robot. AI* 2 (2015).  
795
- [29] M. Dorigo, V. Maniezzo, A. Coloni, Ant system: optimization by a colony of cooperating agents, in: *IEEE Trans. Syst. Man Cybern. B Cybern.*, volume 26, 1996, pp. 29–41.
- [30] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. J. Nebro, D. Molina, A. La-Torre, P. N. Suganthan, C. A. Coello Coello, F. Herrera, A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems, *Swarm and Evolutionary Computation* 64 (2021) 100888.  
800
- [31] P.-J. S. J. Molina, D., et al., Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, *Critical Analysis Recommendations. Cogn Comput* 12 (2020) 897–939.  
805
- [32] N. Geng, Z. Chen, Q. A. Nguyen, D. Gong, Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints, *Complex and Intelligence Systems* 7 (2021) 873–890.
- [33] E. Ozcan, S. Bas, Y. Akman, The improved particle swarm algorithm (pso) methods for search and rescue teams, *International Journal of Advanced Computational Engineering and Networking* 4 (2016) 22–24.  
810
- [34] R. Biedrzycki, J. Arabas, E. Warchulski, A Version of NL-SHADE-RSP Algorithm with Midpoint for CEC 2022 Single Objective Bound Constrained Problems, *IEEE*, Padua, Italy, 2022.  
815
- [35] E. H. Houssein, A. G. Gad, K. Hussain, P. N. Suganthan, Major advances in particle swarm optimization: Theory, analysis, and application, *Swarm and Evolutionary Computation* 63 (2021) 100868.
- [36] Y. Liang, L. Wang, Applying genetic algorithm and ant colony optimization algorithm into marine investigation path planning model, *Soft Computing* 24 (2020) 8199–8210.  
820

- [37] U. Schroeders, O. Wilhelm, G. Olaru, Meta-heuristics in short scale construction: Ant colony optimization and genetic algorithm, *PLoS One* 11 (2016) e0167110.
- 825 [38] N. Nedjah, L. S. Junior, Review of methodologies and tasks in swarm robotics towards standardization, *Swarm and Evolutionary Computation* 50 (2019) 100565.
- [39] M. Dorigo, G. Theraulaz, V. Trianni, Reflections on the future of swarm robotics, *Science Robotics* 5 (2020) eabe4385.
- 830 [40] G.-Z. Yang, P. Bellingham, E. Dupont, et al., The grand challenges of Science Robotics, *Science Robotics* 30 (2018).
- [41] J. Ladyman, J. Lambert, W. Wiesner, What is a Complex System?, *European Journal for Philosophy of Science* 3 (2013) 33–67.
- [42] F.-J. Mañas Álvarez, M. Guinaldo, R. Dormido, R. Socas, S. Dormido, Formation by Consensus in Heterogeneous Robotic Swarms with Twins-in-the-Loop, in: *ROBOT 2022: ROBOT2022: Fifth Iberian Robotics Conference*, volume 589 of *Lecture Notes in Networks and Systems book series*, 2023, pp. 435–447.
- 835 [43] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, R. Nagpal, kilobot: a low cost robot with scalable operations designed for collective behavior, *Robotic Autonomous Systems* 62 (2014) 966–975.
- 840 [44] M. H. M. Alkilabi, A. Narayan, E. Tuci, Cooperative object transport with a swarm of e-puck robots: robustness and scalability of evolved collective strategies, *Swarm Intelligence* 11 (2017) 185–209.
- 845 [45] R. Groß, M. Dorigo, elf-assembly at the macroscopic scale, in: *Proceedings IEEE*, volume 96, 2008, p. 1490–1508.
- [46] R. Oung, R. D’Andrea, The distributed flight array, *Mechatronics* 21 (2011) 908–917.

- [47] T. Schmickl, Collective cognitive robots—the year of cocoro.,  
850 <http://zool33.uni-graz.at/artlife/cocoro>, 2015.
- [48] M. C. Schmickl, T., K. Crailsheim, Collective Perception in a Robot Swarm,  
Springer, Berlin, Heidelberg, 2007.
- [49] C.-K. Kang, Marsbee—swarm of flapping wing flyers for enhanced mars  
exploration, [https://www.nasa.gov/directorates/spacetech/niac/  
855 2018\\_Phase\\_I\\_Phase\\_II/Marsbee\\_Swarm\\_of\\_Flapping\\_Wing\\_Flyers\\_  
for\\_Enhanced\\_Mars\\_Exploration](https://www.nasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of_Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration), 2018.
- [50] J. Stolze, D. Suter, Quantum Computing: A Short Course from Theory to  
Experiment, Wiley, Weinheim, Germany, 2004.
- [51] P. Benioff, Quantum robots and environments, Physical Review A 58  
860 (1998) 893.
- [52] D. Dong, C. Chen, C. Zhang, C. Chen, Quantum robot: structure, algo-  
rithms and applications, Robotica 4 (2006) 513–521.
- [53] C. Petschnigg, M. Brandstötter, H. Pichler, Quantum Computation in  
Robotic Science and Applications, in: IEEE International Conference on  
865 Robotics and Automation (ICRA), 2019.
- [54] P. Atchade-Adelomou, P. Alonso-Linaje, J. Albo-Canals, D. Casado-Fauli,  
qRobot: A Quantum Computing Approach in Mobile Robot Order Picking  
and Batching Problem Solver Optimization, Algorithms 14 (2021).
- [55] M. Alvarez-Alvarado, F. Alban-Chacón, E. Lamilla-Rubio, C. Rodríguez-  
870 Gallegos, W. Velásquez, Three novel quantum-inspired swarm optimization  
algorithms using different bounded potential fields, Nature Scientific Re-  
ports 11 (2021) 11655.
- [56] Z. Li, W. Liu, G. Li-E, L. Li, Path Planning Method for AUV Docking  
Based on Adaptive Quantum-Behaved Particle Swarm Optimization, IEEE  
875 Access Multidisciplinary 7 (2019) 78665–78674.

- [57] J. Jin, P. Wang, Multiscale quantum harmonic oscillator algorithm with guiding information for single objective optimization, *Swarm and Evolutionary Computation* 65 (2021) 100916.
- [58] J. Jin, P. Wang, Quantum-enhanced multiobjective large-scale optimization via parallelism, Cao, B. and Fan, S. and Zhao, J. and Yang, P. and Muhammad, K. and Tanveer, M. 57 (2021) 100697.
- [59] H. Xiong, Z. Wu, H. Fan, G. Li, G. Jiang, Quantum rotation gate in quantum-inspired evolutionary algorithm: A review, analysis and comparison study, *Swarm and Evolutionary Computation* 42 (2018) 43–57.
- [60] Z. A. E. M. Dahi, C. Mezioud, A. Draa, A quantum-inspired genetic algorithm for solving the antenna positioning problem, *Swarm and Evolutionary Computation* 31 (2016) 24–63.
- [61] S. Dey, S. Bhattacharyya, U. Maulik, Quantum inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding, *Swarm and Evolutionary Computation* 15 (2014) 38–57.
- [62] B. Cao, S. Fan, J. Zhao, P. Yang, K. Muhammad, M. Tanveer, Quantum-enhanced multiobjective large-scale optimization via parallelism, *Swarm and Evolutionary Computation* 57 (2020) 100697.
- [63] M. Hemmo, O. Shenker, Quantum Mechanics as a Theory of Probability, in: M. Hemmo, O. Shenker (Eds.), *Quantum, Probability, Logic*, Jerusalem Studies in Philosophy and History of Science, Springer, Cham, 2020, pp. 337–351.
- [64] A. S. Wu, R. P. Wiegand, R. R. Pradhan, Response probability enhances robustness in decentralized threshold-based robotic swarms, *Swarm Intelligence* 14 (2020) 233–258.
- [65] J. S. Bell, On the Einstein Podolsky Rosen Paradox, *Physics* 1 (1964) 195–200.

- [66] R. Jozsa, N. Linden, On the Role of Entanglement in Quantum-  
905 Computational Speed-Up, in: Proceedings: Mathematical, Physical and  
Engineering Sciences, volume 459, 2003, pp. 2011–2032. URL: <https://www.jstor.org/stable/3560059>.
- [67] V. Ivancevic, Entangled swarm intelligence: Quantum computation for  
swarm robotics, Mathematics in Engineering, Science and Aerospace 7  
910 (2016) 441–451.
- [68] L. Lamata, M. Quadrelli, C. de Silva, P. Kumar, G. Kanter, M. Ghazinejad,  
F. Khoshnoud, Quantum Mechatronics, Electronics 10 (2021) 2483.
- [69] A. Koukam, A. Abbas-Turki, V. Hilaire, , Y. Ruichek, Towards a Quan-  
tum Modeling Approach to Reactive Agents, in: 2021 IEEE International  
915 Conference on Quantum Computing and Engineering (QCE), 2021.
- [70] A. Sabra, W. Fung, A Fuzzy Cooperative Localisation Framework for  
Underwater Robotic Swarms, Sensors 20 (2020) 5496.
- [71] S. Nădăban, From classical logic to fuzzy logic and quantum logic: a general  
view, International Journal of Computers Communications and Control 16  
920 (2021) 4125.
- [72] A. Chella, S. Gaglio, G. Pilato, F. Vella, S. Zammuto, A quantum planner  
for robot motion, Mathematics 10 (2022) 2475.
- [73] A. K. Fedorov, M. S. Gelfand, Towards practical applications in quantum  
computational biology, Nature Computational Science 1 (2021) 114–119.
- 925 [74] V. Marx, Biology begins to tangle with quantum computing, Nature  
Methods 18 (2021) 715–719.
- [75] C. Outeiral, M. Strahm, J. Shi, G. M. Morris, S. C. Benjamin, C. M. Deane,  
The prospects of quantum computing in computational molecular biology,  
Wires Computational Molecular Science 11 (2021) e1481.

- 930 [76] P. S. Emani, J. Warrell, A. Anticevic, et al., Quantum computing at the  
frontiers of biological sciences, *Nature Methods* 18 (2021) 701–709.
- [77] R. Feynman, M. A. Gottlieb, R. Pfeiffer, Quantum behavior, in: *The  
Feynman Lectures on Physics*, California Institute of Technology, Califor-  
nia, USA, 1965.
- 935 [78] A. Einstein, B. Podolsky, N. Rosen, Can Quantum-Mechanical Description  
of Physical Reality Be Considered Complete?, *Physical Review Letters* 47  
(1935) 777.
- [79] D. M. Greenberger, M. A. Horne, A. Zeilinger, Going Beyond Bell’s The-  
orem, in: M. Kafatos (Ed.), *Bell’s Theorem, Quantum Theory, and Con-  
ceptions of the Universe*, Dordrecht, Kluwer, 1989, pp. 69–72.
- 940 [80] Y. Kwak, W. J. Yun, S. Jung, J.-K. Kim, J. Kim, Introduction to Quan-  
tum Reinforcement Learning: Theory and PennyLane-based Implementa-  
tion, in: *International Conference on Information and Communication  
Technology Convergence (ICTC)*, 2021.
- [81] D. Dong, C. Chen, H. Li, T.-J. Tarn, Quantum Reinforcement Learning, in:  
945 *IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics)*,  
volume 38, 2008. URL: [https://fr.art1lib.org/book/18461932/  
6b4359](https://fr.art1lib.org/book/18461932/6b4359).
- [82] N. J. R. Plowes, R. A. Johnson, B. Hölldobler, Foraging behavior in the ant  
950 genus *Messor* (hymenoptera: Formicidae: Myrmicinae, *Myrmecological  
News* 18 (2013) 33–49.
- [83] J. Brownlee, *Optimization for Machine Learning*, *Machine Learn-  
ing Mastery*, 2022. URL: [https://machinelearningmastery.com/  
optimization-for-machine-learning/](https://machinelearningmastery.com/optimization-for-machine-learning/).
- 955 [84] V. Stavonov, S. Akhmedova, E. Semenkin, NL-SHADE-LBC algorithm  
with linear parameter adaptation bias change for CEC 2022 Numerical  
Optimization, IEEE, Padua, Italy, 2022.

- [85] M. Mannone, V. Seidita, A. Chella, Quantum RoboSound: Auditory Feedback of a Quantum-Driven Robotic Swarm, in: RO-MAN 2022 Proceedings, IEEE, Naples, Italy, 2022, pp. 287–292.

960