Appendix of the Article: Modeling and Designing a Robotic Swarm: a Quantum Computing Approach

Maria Mannone[1,2,*], Valeria Seidita[1], Antonio Chella[1,3]

Abstract

We present codes and technical details of the article *Modeling and Designing a Robotic Swarm: a Quantum Computing Approach.*

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

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

1. Data availability

We implemented these instructions in Jupyter Notebook, calling IBM QASM simulator. Our version of Python is 3.9.7, and Jupyter 7.29.0. The algorithms are included in the main article. We used both Python and Qiskit code lines. Algorithm 1 is notably shorter and scalable: in fact, the user can choose the number of robots the swarm is made of. For this second algorithm, it is considered only the path nest→food, and, because of its faster convergence, the GHZ step was removed. The code corresponding to the enhanced algorithm is provided in the publicly-shared Git folder `https://github.com/medusamedusa/10_little_ants` as *short_2D_quantum_only_Z.ipynb*.

The file *short_2D_quantum_only_Z_matrix.ipynb*, available from the same folder, contains the code to compute matrices. For the ant lines comparison we used the NetLogo example from `www.netlogoweb.org`; for the 3-object bounded

*Corresponding author

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

[1]Department of Engineering, University of Palermo, Italy

[2]ECLT and DAIS, Ca' Foscari University of Venice, Italy

[3]ICAR-CNR National Research Council, Italy

random walk, we adapted the Python code from https://stackoverflow.com/questions/46954510. In the Net Logo Ant Lines simulation, we chose number of ants = 3, leader wiggle angle = 90 (all directions are possible) or 38, delay = 0 (we want that all ants start their research simultaneously, without a true leader). The code for the PSO example has been adapted from an example[4] [1], and it is available in the folder https://github.com/medusamedusa/10_little_ants/blob/main/pySwarm.ipynb. References and parameters for the particle swarm optimization example are provided within the main text. Finally, we show one example of the matrix output. The code for NL-SHADE-RSP with midpoint [2] can be downloaded online from https://github.com/P-N-Suganthan/2022-SO-BO.

2. Qiskit codes

Let us present the QASM code for the proposed x-position circuit:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
creg c[1];
ry(pi/2) q[0];
ry(1.2309594) q[1];
barrier q[0], q[1], q[2];
ccx q[0], q[1], q[2];
x q[0];
x q[1];
ccx q[0], q[1],q[2];
x q[0];
x q[1];
barrier q[0], q[1], q[2];
measure q[2] -> c[0];
```

[4] https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

And here, the code for the proposed xy-position circuit (Figure **??**):

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[5];
creg mq2[1];
creg mq3[1];
creg mq4[1];

x q[1];
x q[2];

barrier q[0],q[1],q[2],q[3],q[4];

ccx q[0],q[2],q[3];
ccx q[1],q[2],q[4];
x q[2];
ch q[2],q[3];
ch q[2],q[4];
x q[2];
barrier q[0],q[1],q[2],q[3],q[4];
measure q[2] -> mq2[0];
measure q[3] -> mq3[0];
measure q[4] -> mq4[0];
```

Finally, the code to obtain a GHZ state is the following:

```
from qiskit import QuantumRegister,
ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(9, 'q')
creg_c = ClassicalRegister(9, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
```

```
    circuit.h(qreg_q[0])
    circuit.cx(qreg_q[0], qreg_q[1])
    circuit.cx(qreg_q[0], qreg_q[2])
    circuit.cx(qreg_q[0], qreg_q[3])
    circuit.cx(qreg_q[0], qreg_q[4])
    circuit.cx(qreg_q[0], qreg_q[5])
    circuit.cx(qreg_q[0], qreg_q[6])
    circuit.cx(qreg_q[0], qreg_q[7])
    circuit.cx(qreg_q[0], qreg_q[8])
    circuit.measure(qreg_q[0], creg_c[0])
    circuit.measure(qreg_q[1], creg_c[1])
    circuit.measure(qreg_q[2], creg_c[2])
    circuit.measure(qreg_q[3], creg_c[3])
    circuit.measure(qreg_q[4], creg_c[4])
    circuit.measure(qreg_q[5], creg_c[5])
    circuit.measure(qreg_q[6], creg_c[6])
    circuit.measure(qreg_q[7], creg_c[7])
    circuit.measure(qreg_q[8], creg_c[8])
```

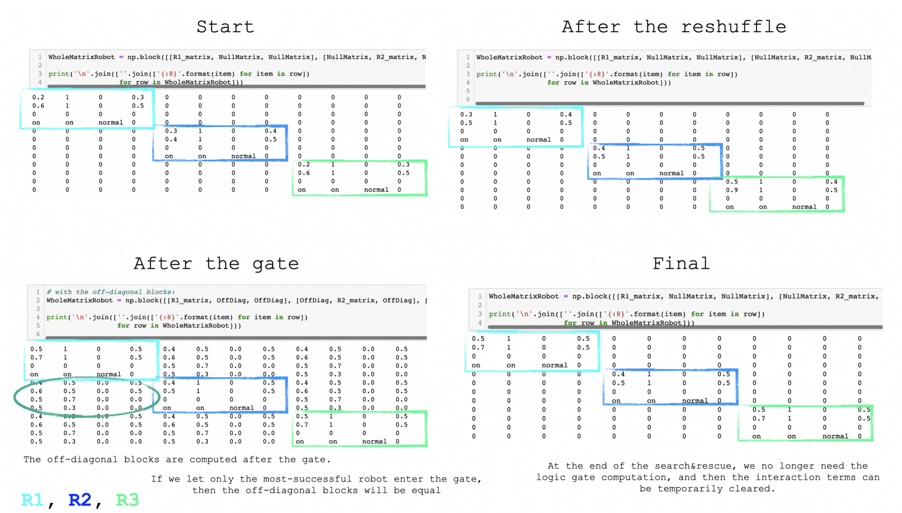

Figure 1: Matrices computed with our code for a 3-robot swarm.

In Figure 1 we can observe an example of the output of the overall block matrix for a 3-robot swarm, for the four main steps of the process. The matrix output can be found at `https://github.com/medusamedusa/10_little_ants/blob/main/short_2D_quantum_only_Z_matrix.ipynb`, in the shared repository.

References

[1] J. Brownlee, Optimization for Machine Learning, Machine Learning Mastery, 2022. URL: `https://machinelearningmastery.com/optimization-for-machine-learning/`.

[2] 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.