Quantum Algorithms for the Leader Election Problem in Anonymous Networks

Part 1: Project

Candidate: Leonardo Oleynik
Advisor: Prof. Dr. Luiz A. de P. Lima Jr

April 2020

Group: Distributed Quantum Computing

Abstract

Given the classical inexistence of an exact Leader Election algorithm (without errors and that ends in a limited time) for an anonymous network, this project aims to study and test the respective quantum solution. In this way, we will research and implement the algorithm presented in the literature, using the algebra of Quantum Mechanics. Moreover, we will develop a quantum circuit capable of testing it, in which we will use Python language and Qiskit library.

Outline

Problem to be Solved

Leader Election

Leader Election is the procedure that elects a single party (or process) as the organizer of some task.

- Election in a network with distinguishable parts;
- Distinguishability as the network grows;
- Anonymous network.

To guarantee the *anonymity* of the network, it is necessary that, in addition to each identifier being the same, the local protocol is also the same.

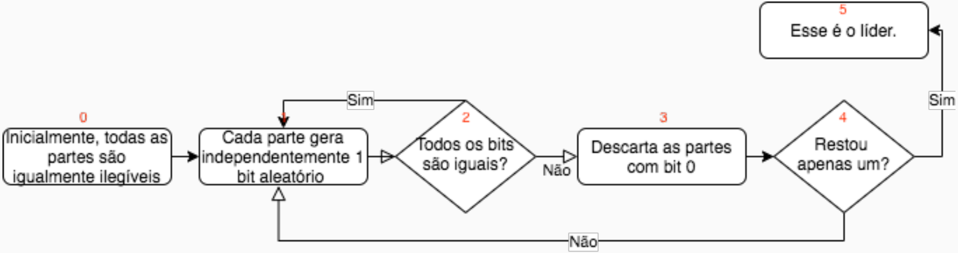

Figure 1: Classic Leader Election Algorithm

Problem: Non-zero probability that the algorithm will never (within a limited time) leave stage 2.

Reason: the random bit generation of each part is *independent* of the others.

Therefore, there is classically no so-called *exact* solution for the election of a leader in an anonymous network.

Solution: Sharing entangled (correlated) states.

Work to be Developed

Block Diagram: General

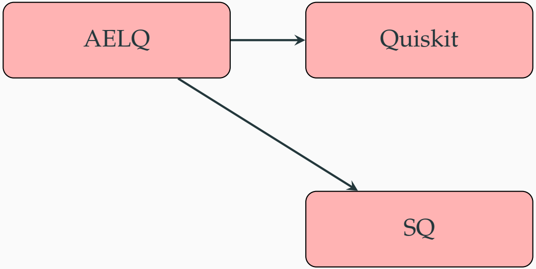

Figure 2: General functional diagram

AELQ Quantum Leader Election Algorithm

SQ Quantum (Computer) Simulator

Quiskit IBM Quantum Circuit Simulator

Language used: Circuits

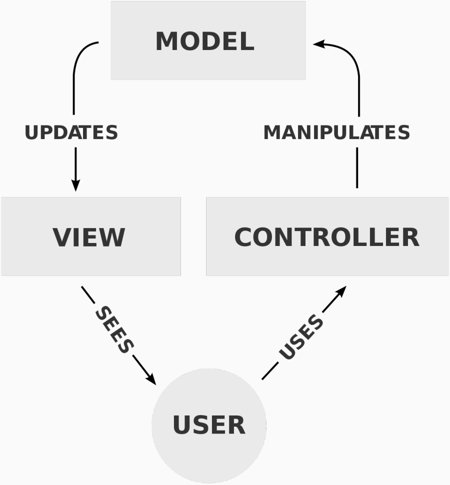

Figure 3: SQ functional diagram

Language used: Python. aided by

- Qutip;
- Tkinter.

Technologies that will be used in the project

Chosen Technologies

I Programming Language

- Python: high-level, interpreted and object-oriented.
- Pros: Highly readable, well documented and widespread, *Host* of the main simulators.

II Library

- Qutip: Quantum operations library written in Python.
- Pros: *well* documented and widespread, focused on quantum circuits, excellent graphics packages and etc.

III Simulators

- Quiskit: Quantum circuit simulator integrated into the *IBM-Q experience* platform.
- Pros: Graphical and command-line manipulation, didactics, numerous internal simulators, etc.

Other Technologies

I Programming Language

- Python: chosen!
- JavaScript: No library for quantum circuits.
- Java: Incompatible with most commercial simulators.

II Library

- Qutip: chosen!
- PyQu: Outdated (10 years)
- Libquantum: focused on factoring and search algorithms, outdated.

III Simulators

- Quiskit: chosen!
- MVM: slightly less famous than the chosen one, offering fewer q-bits (simulation and testing)

Test Procedure and Evaluation Criteria

Test Procedure and Assessment Criteria

I Algorithm

- Performance: average runtime *time* curve vs. theoretical result.
- Functionality: *exact* election.

II set

- Performance and functionality: similar to the one above, but with the simulator to be developed.

Test Procedure and Assessment Criteria

I Model

- Performance 1: pairs of identical virtual machines were running different simulators, submitted to the same algorithms. Compare runtime in each scenario.
- Performance 2: same, but with algorithms in the *host* language.
- Functionality: The output on each machine must be the same.

II Interface

- Performance and functionality: Python authorized *Frameworks* (Pytest, PyUnit and Behave).

Risk Analysis

Risk Analysis

Risk	Nature	Probability	Impact	Justification
Simulator Complexity	Design	Low	Medium	Independent Implementation, Partial Solution
Interface Engineering	Design	Average	Average	Lack of punctual skill
Unknown Language	Technical	Average	Medium	Experience with more complicated languages

Table 1: Risk analysis summary table

Schedule

Schedule

Nome	Data inicial	Data final
TFG	24/04/20	09/11/20
Projeto Físico	27/04/20	12/06/20
Modelo	27/04/20	02/05/20
Documentação + Testes	04/05/20	09/05/20
Algoritmo	11/05/20	16/05/20
Documentação + Testes	18/05/20	23/05/20
Conjunto	25/05/20	30/05/20
Documentação + Testes	01/06/20	06/06/20
Revisão + Apresentação	08/06/20	14/06/20
View	22/06/20	11/07/20
Documentação + Testes	13/07/20	18/07/20
Controle	20/07/20	08/08/20
Documentação + Testes	10/08/20	15/08/20
Conjunto	17/08/20	05/09/20
Documentação + Testes	07/09/20	12/09/20
Revisão + Apresentação	14/09/20	27/09/20
Correção de erros	05/10/20	10/10/20
Testes	12/10/20	17/10/20
Documentação Final	19/10/20	31/10/20
Apresentação + Revisão	02/11/20	07/11/20

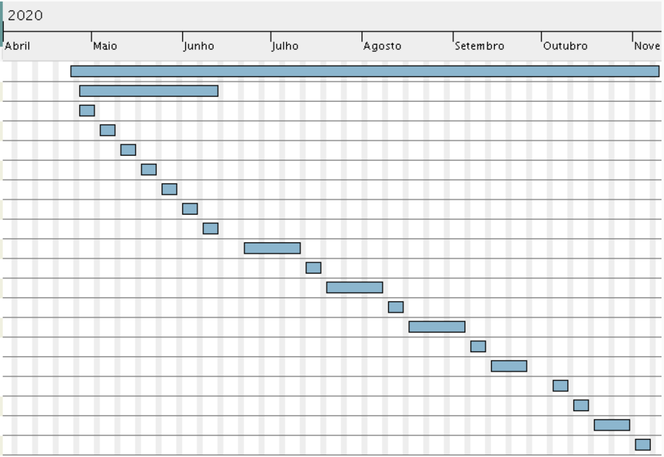

Figure 4: Schedule

Conclusion

Conclusion

- Objectives
  I Implement AELQ and
  II Develop a quantum circuit simulator.
- Solutions
  I Understand and design the algorithm of *TANI*; Quiskit
  II Three-part construction (MVC); Python and Qutip
- Validation
  I Unanimity of the election and average execution time.
  II Model: Comparison of the performance and output of our simulator with another.
  III Interface:*frameworks* authorized by Python.