Quantum Algorithms for the Leader Election Problem in Anonymous Networks

Part 2: Prototype

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

September 2020

Group: Distributed Quantum Computing

Abstract

The problem to be addressed in this work is that of *leader election in an anonymous network* (ELA). As TANI states, The Leader election problem is a central issue in distributed computing because solving it would allow us to elucidate many others. Among these, we can mention the *Spanning Tree* protocol (capable of generating *loops*-free network topologies) and the maximum value problem (fundamental in the construction of more complex algorithms).

Thus, given the difficult access to quantum computers and the irreducibly quantum aspect of the solution to the problem proposed above. This work has two main objectives:

1. the use of a tool capable of filling these gaps, *the quantum computer simulator*;
2. and, through this, the implementation, as found in the literature, of a quantum algorithm that unanimously elects a leader in an anonymous network in a limited time.

Outline

1. Problem to be Solved
2. Summary of Work to be Developed
3. Test Procedure and Evaluation Criteria
4. Risk Analysis
5. Schedule
6. Conclusion

Problem to be Solved

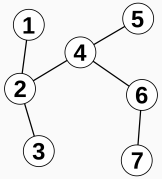


Figure 1: Distinguishable network

A	B
0	0
0	1
1	0
1	1

Table 1: Possibilities 1 =heads, 0 =tails

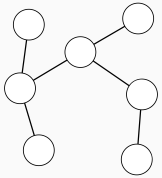


Figure 2: Anonymous network

Conclusion:

- 1/2 election is unanimous;
- 1/2 repeat the release;
- It is impossible to elect a leader unanimously and accurately.

Solution: Entangled quantum states

Summary of Work to be Developed

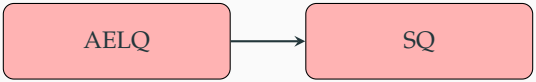


Figure 3: General functional diagram

AELQ Quantum Leader Election Algorithm

SQ Quantum (Computer) Simulator:

1. practical
2. and edit a network
3. support the AELQ

MVC chosen architecture: Interface separates user data

MVC architecture: *model-view-controller*

Model:

- Provide methods to change the core data;
- Register your collaborators;
- Notify all collaborators;

View:

- Create and initialize bound controls;
- Show information to the user;
- read model data;
- Implement the update procedure.

Control:

- Accept user input as events;
- translate events in service and view requests to the model and view;
- Implement the update procedure.

1. Change propagation
2. Initialization: Association

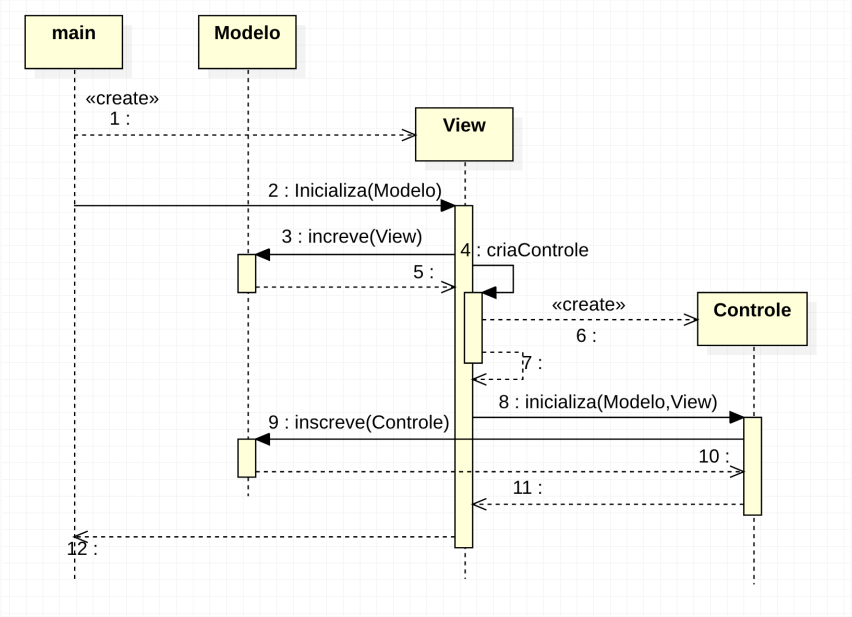


Figure 4: Initialization engine sequence diagram

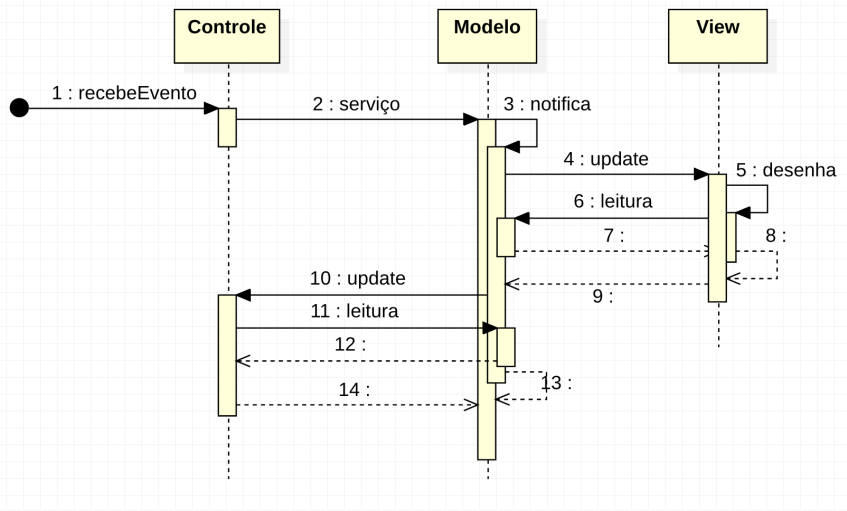
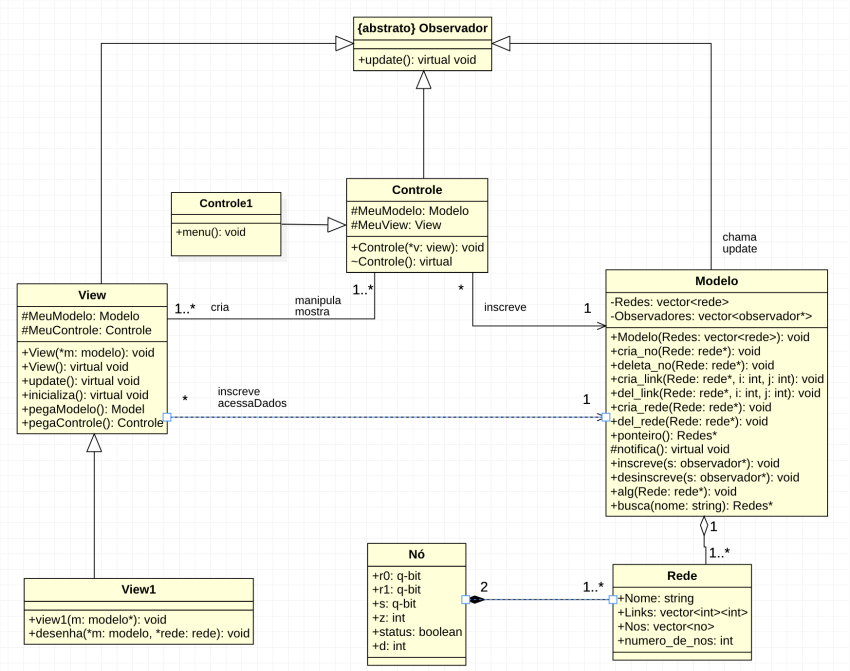


Figure 5: Change propagation mechanism sequence diagram

Class Diagram



Algorithm

Input:Variables *status*, n and d **Output**:Variable *status*

- Prepares three quantum registers r_0, r_1 and s from a q-bit.
- For $k = n$ up to 2:
  1. Prepare a consistent state in s.
  2. If *status* = 1, prepare the cat state in r_0, otherwise prepare the classic zero state.
  3. Run the A subroutine with R_0, S, *status*, n and d.
  4. The state is measured globally in S on a consistent basis. If the result is consistent and $status = 1$, prepare the classic zero state in r_1 and run subroutine B with R_0, R_1 and k; Otherwise, if the result is inconsistent, just prepare the classic zero state in r_1.
  5. If *status*=1, measure the state in r_0 and r_1 in the computational base and store the result (the two bits) in z; otherwise set $z = -1$.
  6. Perform a maximization over all z, getting the largest value z_m over all parts. If $z \neq z_m$, then *status*$= 0$.
- returns *status*.

repair

- $S_{i+1} \subseteq S_i$
- Two parts: Prepare state *Inconsistent* and Measurement/Check.
- Why $|10\rangle$, $|01\rangle$? They break symmetry.

Algorithm for $n = 2$

1.
2.
3. Run the A subroutine with R_0, S, *status*, n and d.
4. The state is measured globally in S on a consistent basis. If the result is consistent and *status* $= 1$, prepare the classic zero state in r_1 and run subroutine B with R_0, R_1 and k; Otherwise, if the result is inconsistent, just prepare the classic zero state in r_1.
5. If *status*=1, measure the state in r_0 and r_1 in the computational base and store the result (the two bits) in z; otherwise set $z = -1$.
6. Perform a maximization over all z, getting the largest value z_m over all parts. If $z \neq z_m$, then *status*$= 0$.

repair:

- A prepares: $|\psi\rangle = \alpha_0(|00\rangle \otimes |C\rangle) + \alpha_1(|01\rangle \otimes |I\rangle) + \alpha_2(|10\rangle \otimes |I\rangle) + \alpha_3(|11\rangle \otimes |C\rangle)$, such that the second is consistent if the first is
- Consistent: $|\alpha_0|^2 + |\alpha_3|^2$, B fixes R_0
- inconsistent: $|\alpha_1|^2 + |\alpha_2|^2$
- We always have $|I\rangle$ in 6.

Test Procedure and Evaluation Criteria

Test Procedure and Assessment Criteria: Algorithm

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

Test Procedure and Evaluation Criteria: MVC

1. Test each class individually
2. Test architecture interdependence, verifying the main mechanisms.
3. Debug or small programs

Risk Analysis

Risk Analysis: Original

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

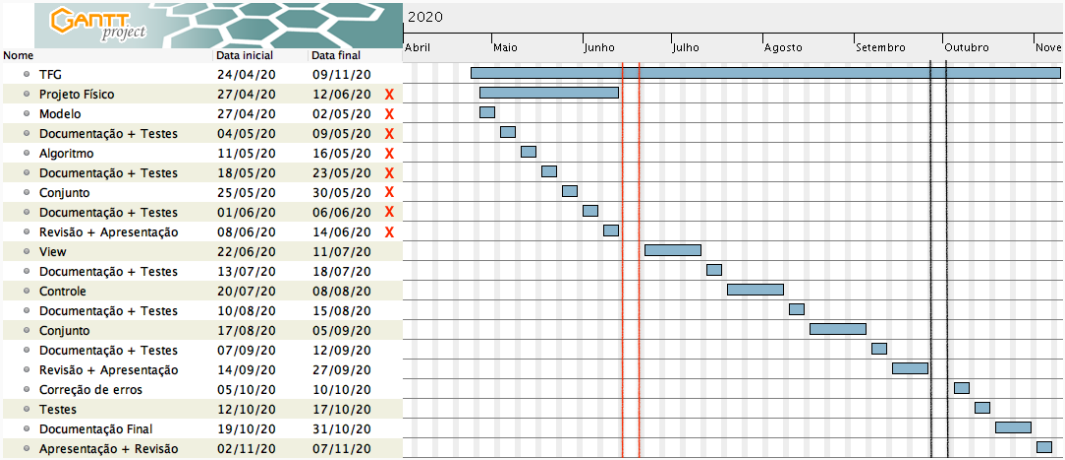
Table 2: Risk analysis summary table

Risk	Nature	Probability	Impact	Justification
Simulator Complexity*	Design	Low	Low	Simpler
Interface Engineering	Design	Low	Medium	Great bibliography
Language Unknown	Technical	Low	Medium	We Learned,

Table 3: Summary table of the new risk analysis

- Changes in impact: 1
- Changes in probability: 2 and 3
- It is feasible.

Schedule



Nome	Data inicial	Data final	
TFG	24/04/20	09/11/20	
Projeto Físico	27/04/20	12/06/20	X
Modelo	27/04/20	02/05/20	X
Documentação + Testes	04/05/20	09/05/20	X
Algoritmo	11/05/20	16/05/20	X
Documentação + Testes	18/05/20	23/05/20	X
Conjunto	25/05/20	30/05/20	X
Documentação + Testes	01/06/20	06/06/20	X
Revisão + Apresentação	08/06/20	14/06/20	X
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 7: Schedule

Conclusion

Conclusion

- Objectives
  - I Implement AELQ and
  - II Develop a quantum circuit simulator.
- Solutions
  - I Understand and design TANI algorithm; MQ
  - II Three-part construction (MVC); Python (or C) and Qutip
- Validation
  - I Unanimity of the election and *average time* of execution.
  - II MVC: Debugging the methods of each class