Quantum Algorithm for Multiplicative Linear Logic

Lorenzo Saraiva[1], **Edward Hermann Haeusler**[1], **Vaston Costa** [2]

[1]Pontificia Universidade Católica do Rio de Janeiro.
R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, Brazil

[2]Universidade Federal de Catalão.
Av. Dr. Lamartine Pinto de Avelar 1120. Catalão - GO, Brazil

lorenzo.saraiva@hotmail.com, hermann@inf.puc-rio.br, vaston@ufcat.edu.br

Abstract. *This paper describes a quantum algorithm for proof search in sequent calculus of a subset of Linear Logic using the Grover Search Algorithm. We briefly overview the Grover Search Algorithm and Linear Logic, show the detailed steps of the algorithm and then present the results obtained on quantum simulators.*

1. Introduction

Quantum computing has provided us with algorithms that have a better time complexity than any classical counterpart, one of those being the Grover's Search Algorithm(GSA)[Grover 1997] for searching an element in an unordered database. The GSA is used in several contexts, including SAT, kmeans, genetic algorithms and pixel identification. In this work, we use the GSA to help in searching proofs of a subset of multiplicative linear logic to improve complexity compared to classic algorithms. We show the construction of the quantum circuit from the linear logic sequent to the end result and present our conclusions.

2. Background

The GSA is one of the most famous quantum algorithms, and its goal is to search for an element in an unordered database. Assuming a database with n qubits that contains $N = 2^n$ elements in the superposition, it has time complexity of $\sqrt{N}$, which outperforms any classical algorithm. The general steps of the Grover algorithm main iteration on n qubits are as follows:

- Database initialization - In this step an operator A is applied to the database qubits to bring them from the initial state $|0\rangle^{\otimes n}$ to the desired state $|\Psi\rangle$. This state is usually the equal superposition state, and $A = H^{\otimes n}$.

- Oracle call - In this step an oracle O is applied to the prepared state $|\Psi\rangle$. The oracle will flip the phase of the searched value x_t so that:

  $O\,|x_s \neq x_t\rangle = |x_s\rangle$
  $O\,|x_s = x_t\rangle = |-x_s\rangle$

- Amplitude amplification - In this step an operator D is used to amplify the amplitude of the state marked by the oracle. For such, an inversion about the mean (IAM) is performed.

Generalizing, the Grover iteration can be described as:

$$G = AOA^TD$$

The Grover iteration has to be repeated $\lfloor \pi\sqrt{N}/4 \rfloor$ times in order to maximize the probability of measuring the desired state. Our work follows Alsing's entangled database search [Alsing and McDonald 2011] using the GSA. The main feature of Alsing's algorithm is that, instead of using $A = H^{\otimes n}$ to prepare the equal superposition state, it chooses A in order to encode an arbitrary list of pairs $\{s, t\}$. Thus, the algorithm's input is an entangled database with two sides, each side having one part of the pair. Every entry on the left side is entangled to an entry on the right side. In the GSA, it is necessary to know the searched value to construct the oracle. On Alsing's, on the other hand, one can construct the oracle based on a known entry s_1 on the left side, apply the GSA, and then measure the right side, recovering the unknown value t_1 entangled with s_1.

3. Problem Description

Linear logic is an extension of classical and intuitionistic logic that emphasizes the role of formulas as resources. For that reason, it does not allow the rules of contraction and weakening to apply to all formulas but only those formulas marked with special marks[Di Cosmo and Miller 2019]. Due to the (formal) similarity between the logical rules that deal with these marks and the modalities in systems like S4, these marks might be considered as modalities. The absence of contraction and weakening allows Linear Logic to have two different versions of conjunction and disjunction: an additive and a multiplicative. The classical $\wedge$ (and), for example, is divided between the additive version, $\&$ (with), and the multiplicative version, $\otimes$ (tensor). Linear logic also has a sequent calculus proof system. In this context, the algorithm for finding a cut-free proof in the multiplicative only version of Linear Logic has a worst-case time complexity of 2^k, where k is the number of atomic formulas. The subset of intuitionistic linear logic that deals only with the multiplicative connectives is called (intuitionistic) multiplicative linear logic(IMLL). In this work, we will be using a subset of IMLL, IMLL-$\otimes$ using only the tensor connector. The proof finding algorithm for the additive part, on the other hand, has a linear time complexity. For this reason we chose to focus solely on the multiplicative.

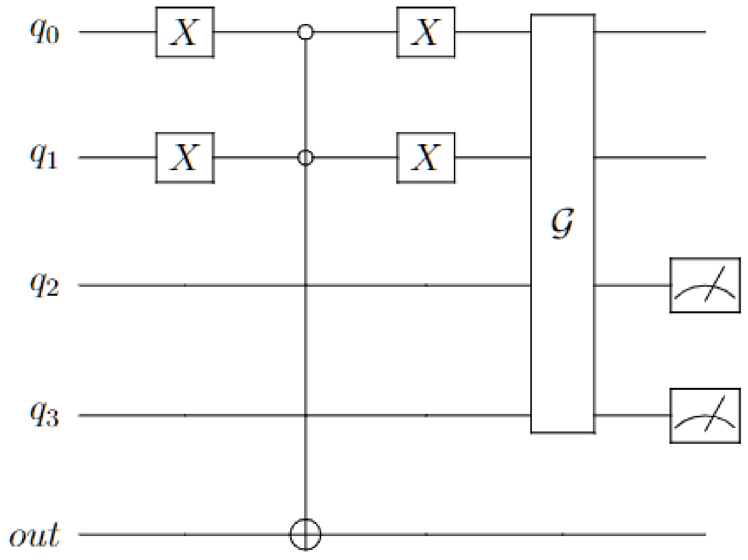

Figura 1. Oracle circuit for k=2 and searching for the value associated with 0

Considering a linear logic sequent with $k = 4$ atomic formulas

$$A \otimes (B \otimes (C \otimes D)) \vdash D \otimes (B \otimes (A \otimes C)).$$

We want to find the successive splits that verify that this is a valid proof. We have two rules that can be applied, $\otimes$-Left and $\otimes$-Right. In the classical algorithm, we apply the successive splits until we reach a valid axiom.

- Apply one of the possible rules until there are only left axioms,
- If the axioms are all valid, the sequent is valid; if not, restart.

Since there are 2^k possible splits, the algorithm has time complexity of $O(2^k)$, where k is the number of atomic formulas.

Algorithm 1 General Description

```
Require: $k$ copies of the database of $2n$ qubits and $k$ pairs, where $n = \lceil \log k \rceil$
  $numIterations \leftarrow \lfloor (\pi \times \sqrt{N}/4) \rfloor$
  for $i < N$ do
      for $j < numIterations$ do
          buildOracle(n, target)
          appendDiffuser($A$)
          measure()
      end for
  end for
```

4. Solution Steps

Our quantum algorithm input is an entangled database with two sides, each part of a pair. Every entry on the left side is entangled to an entry on the right side, and they are both unordered. We will call left side of each pair as the *search* part, or s, and the right side the *target*, or t. To be able to perform the algorithm in $\sqrt{k}$ steps, it is necessary to have k copies of the paired database, where k is the number of unique atomic formulas. The complexity of building this database is not taken into account. Our algorithm also shows an explicit dynamic construction for the Grover oracle depending on the searched value.

4.1. Preparing the entangled database

Before starting the algorithm, one must construct an entangled database that accurately represents the sequent. In order to do so, we will need $k \times 2n$ qubits. Then, the pairs $|a\rangle\,|b\rangle$ will be encoded as $|ka+b\rangle$, where a and b is the position of the formula in each side of the sequent. Assuming we have $k = 8$ and $n = 3$, where n is the number of qubits necessary to represent a solution space of k values. Thus our entangled database with 8 solutions will have 2 groups of n qubits, each representing $8 = k = 2^n$ values. We will treat both groups of 3 qubits as a single array and prepare the resulting encoding in the superposition, using k of the k^2 total possibilities that can be stored in $2n$ qubits. Then, we will need k copies of the register, one for each formula. The construction of the database is not explicitly shown but its complexity is $O(n)$ or $O(\log k)$[Alsing and McDonald 2011], taking $O(k \log k)$ in total. This process is not strictly part of the algorithm, which only receives a pre-constructed entangled database. [1]

4.2. $\otimes$-Left

The first step of the algorithm itself is to apply the $\otimes$-Left rule until it cannot be applied anymore, so we have a sequent of the form:

$$A^1, A^2, ..., A^N \vdash B \otimes \Delta$$

Where $\Delta = A \otimes (\Delta)$ or $\Delta = A$ Now, we can use our entangled database to find out the correct split for the leftmost atomic formulas of the right side.

4.3. Grover Search

Now that we have the entangled database of k copies of $2n$ qubits, we can perform the GSA. We start by picking the leftmost atomic formula of the right side. The first step is to construct the oracle dynamically for the chosen element on the left side. The construction of this Oracle takes into account the binary representation of the chosen formula position. We apply the necessary X-Gates to leave all the search space qubits in $|1\rangle$ and apply a multi-controlled Toffoli Gate with a prepared qubit as a target to perform the phase kickback, as can be seen in 1. This oracle is applied only on the search, that is, the first n qubits of the first copy of the $2n$ qubits. Then, the Grover operator for amplitude amplification is applied to all $2n$ qubits $\sqrt{k}$ times, and the measurement to the right side of the $2n$ qubits.

An example of this circuit for $n = 2$ qubits is shown in 1. It is important to note that in the IAM step of the GSA, it is necessary to apply the A operator, which takes $\log k$ steps, making the overall complexity of the GSA step $O(\sqrt{k} \log k)$. This process finds

[1] In our case, the entangled state, it is necessary to store the gate sequence A used to encode a copy of the entangled database state so it can be used later in the IAM step of the GSA.

$$\frac{\Delta, B_0, B_1 \vdash \gamma}{\Delta, B_1 \otimes B_2 \vdash \gamma}\otimes\text{-Left} \qquad \frac{\Delta_0 \vdash A_0 \qquad \Delta_1 \vdash B_1}{\Delta_0, \Delta_1 \vdash A_0 \otimes A_1}\otimes\text{-Right} \qquad \frac{}{B \vdash B}$$

Figura 2. Rules of $\otimes$-only Linear Logic

the corresponding entry of a pair, but we need to find the k corresponding pairs. The issue is that measuring the qubits destroys the prepared superposition corresponding to the pairs. Therefore, we need the k copies of the prepared $2n$ qubit entangled database - so we perform the GSA for each pair on a different copy of the database, in $k^{1.5}$ steps in total - k times $\sqrt{k}$ steps.

5. Example

We want to find out if

$$A \otimes (B \otimes (C \otimes D)) \vdash D \otimes (B \otimes (A \otimes C)))$$

is a valid sequent in linear logic. We have $k = 4$ and consequently $n = \log_2^4 = 2$, thus 2 qubits are used for each side, and $2n$ for each copy in total. We will construct of the entangled representation of this sequent.

A	0	2
B	1	1
C	2	3
D	3	0

Using the formula $|ka + b\rangle$, with $k = 4$, we have

A	k0 + 2 = 2
B	k1 + 1 = 5
C	k2 + 3 = 11
D	k3 + 0 = 12

Thus we have the state of the quantum database as

$$\sqrt{\tfrac{1}{4}}(|2\rangle + |5\rangle + |11\rangle + |12\rangle)$$

or

$$\sqrt{\tfrac{1}{4}}(|0010\rangle + |0101\rangle + |1011\rangle + |1100\rangle)$$

We perform the Grover search on any of the sides and are able to recover the value on the other side. But first, let's go back to the sequent. The rules for $\otimes$ are shown in 2.

Because $\otimes$ is a binary operator, we can't search for values inside the parentheses and apply the rules, so we treat the sequent as

$A \otimes \Delta^1 \vdash D \otimes \Delta^2$

For that reason, we first apply the $\otimes$-Left successive times, until every atomic formula is alone

$$\dfrac{\dfrac{\dfrac{A, B, C, D \vdash D \otimes \Delta^2}{A, B, C \otimes D \vdash D \otimes \Delta^2}\ \otimes\text{-Left}}{A, B \otimes \Delta^3 \vdash D \otimes \Delta^2}\ \otimes\text{-Left}}{A \otimes \Delta^1 \vdash D \otimes \Delta^2}\ \otimes\text{-Left}$$

Now, we run the quantum algorithm for every entry on the right side, and apply the results to the sequent, following the order of appearance. D is encoded to the pair $(3, 0)$, but the algorithm only knows the 0, which is the position of the value we're querying, which is encoded by $\sqrt{\frac{1}{4}}\,|1100\rangle$. We'll apply the Oracle on the two last qubits, that represent the 0 part of the pair, with the shown circuit, and then measuring the first two qubits, with high probability of the result being 3. This process is done for every formula of the right side, so now we just apply the splits following the indexes $(3, 1, 0, 2)$, with the following results

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A \qquad C \vdash C}{A, C \vdash A \otimes C}\ \otimes\text{-Right} \qquad B \vdash B}{A, B, C \vdash B \otimes (A \otimes C)}\ \otimes\text{-Right} \qquad D \vdash D}{A, B, C, D \vdash D \otimes \Delta^2}\ \otimes\text{-Right}}{A, B, C \otimes D \vdash D \otimes \Delta^2}\ \otimes\text{-Left}}{A, B \otimes \Delta^3 \vdash D \otimes \Delta^2}\ \otimes\text{-Left}}{A \otimes \Delta^1 \vdash D \otimes \Delta^2}\ \otimes\text{-Left}$$

6. Results

From a given entangled database state, our algorithm has time complexity of $O(k^{1.5} \log k)$ and has a space qubit complexity of $\log k$, where k is the number of atomic formulas on the sequent. Even when taking into account the construction of the database, which takes $O(k \log k)$ steps, we're still left with a time complexity of $O(k^{1.5} \log k + k \log k) = O(k^{1.5} \log k)$ which outperforms the classical algorithm.

Additionally, it is important to note that when $k > 4$, we would need a controlled-NOT gate with more than two control qubits. For that, we need to concatenate the results of Toffoli gates, introducing additional $n - 2$ ancillary qubits[Piro et al. 2020]. We ran our circuit in the several simulators provided by IBM, such as the *qasm_simulator* and *simulator_mps*. Each execution consisted of 1000 circuit's runs. We tested the implementation of the algorithm up to 64 atomic formulas with high precision, using $2 \times \log k = 12$ qubits as search space.

7. Future Work

While this solution uses the GSA to get an advantage when searching the matching pairs, it has some weaknesses. The first is the fact that you need to prepare the quantum database for each execution, since the quantum state is destroyed in the process. Another issue is that the algorithm fully quantum: while the index matching is found with the GSA, the splits are done classically taking into account the position of each atomic formula, and one could argue that this could add an overhead of complexity. For that reason, a different quantum approach is being currently developed, where each qubit value will represent the side of an atomic formula in a specific split. Another point to be noted is that we left out the multiplicative disjunction. A quantum algorithm exclusively for the disjunction would be rather similar to the presented one. On the other hand, combining both the disjunction and the conjunction in the same algorithm would present more challenges and might be addressed in future works.

8. Full Quantum approach

This algorithm also uses the GSA to help in proof search for IMLL, but there is considerable difference between this and the first one. Now, we don't use Alsing's entangled database nor do we need to prepare a specific quantum state prior to the execution. The algorithms uses $(k-1)+\log k$ qubits, where k is the number of atomic formulas in the right side of the sequent. The first $(k-1)$ qubits represent the side picked by a formula in each of the $(k-1)$ splits and the last $\log k$ qubits act as an index for the formulas. A qubit measured 0 means a formula will go to the left in a split and 1 means it will go the right. Starting with the simplest case:

$$A, B \vdash A \otimes B$$

For $k = 2$ We will need $(k-1)+\log k = 2$ qubits. The quantum state that represents the correct splits is $|00\rangle + |11\rangle$. The $|00\rangle$ state is the A going to the right side and the $|11\rangle$ is the B going to the left side. The GSA Oracle will mark both these states as correct ones. These states are defined by the right side of the sequent. Let's go over a slightly more complicated example:

$$A, B, C, D \vdash D \otimes (B \otimes (A \otimes C)))$$

We'll look at the right side to define the states that will be marked by the Oracle. First, D will go to the left side and all everybody else to the right. D will have no future splits, and in the case we fill the rest of its correspondent state with 0s. Thus, one of the Oracle correct states is $|010|11\rangle$. Applying a similar process we can construct the other three: $|110|00\rangle$, $|100|01\rangle$ and $|111|10\rangle$, for A, B and C respectively. Now we just apply the GSA a sufficient time to measure the four possibilities and we'll have recovered the splits necessary to form a valid sequent. This has a time complexity of $\sqrt{\frac{2^{k+\log k}}{k}}$. This can be simplified:
$2^{k+\log k} = 2^k \times 2^{\log k} = 2^k \times k$

$\sqrt{\frac{2^k \times k}{k}} = \sqrt{2^k}$, which is the expected quadratic speedup from the GSA.

9. Adding Linear Implication

Following the full quantum approach, the next step is to add linear implication to the connectors accepted by the algorithm. This comes with some challenges. First, we can no longer use the right side as a fixed reference for the oracle to apply the successive splits based on the $\otimes$-Right rule - if we add linear implication, now the atomic formula can switch sides depending on the rule, and the initial sequent no longer needs to have a balanced number of atomic formula in each side. So, instead of only specifying the splits of the left side to follow a fixed order of the right side, we need to handle every atomic formula. Also, we have four options of "places to go"when applying the $\multimap$-Left rule: left side of the left sequent, right side of the left sequent, left side of the right sequent and right side of the right sequent. This is also an issue with the $\otimes$-Right, since now we have to explicitly say where each formula will go. To solve this, each step will use 2 qubits instead of one. The first qubit of the pair represents which sequent the formula will go, 0 for left, 1 for right. The second will represent which side of sequent the formula will go, again 0 for left, 1 for right. When applying the $\multimap$-Right, it will count as everybody going to the left sequent. Let's go over a simple example:

$$A^1, A^2 \multimap B^1 \vdash C^1 \multimap B^2, C^2$$

$$A^1, A^2 \multimap B^1, C^1 \vdash B^2, C^2$$

$$A^1 \vdash A^2 \qquad\qquad B^1, C^1 \vdash B^2, C^2$$

Thus, the correct states for the oracle will be: $A^1 = |0000|000\rangle$; $A^2 = |0001|001\rangle$; $B^1 = |0010|010\rangle$; $B^2 = |0111|011\rangle$; $C^1 = |0010|100\rangle$; $C^2 = |0111|101\rangle$.

There's a few interesting things to point out here. The first is the increase of qubits. The complexity of the last solution was $\sqrt{2^k}$, where k is the number of logical connectors. This solution, on the other hand, has the complexity of $\sqrt{\frac{2^{2k+\log k}}{k}}$. Simplifying: $2^{2k+\log k} = 2^{2k} \times 2^{\log k} = 2^k \times k$, gives $\sqrt{\frac{2^{2k} \times k}{k}} = \sqrt{2^{2k}} = 2^k$. So, 2^k is the final complexity.

Referências

Alsing, P. M. and McDonald, N. (2011). Grover's search algorithm with an entangled database state. In *Quantum Information and Computation IX*, volume 8057, page 80570R. International Society for Optics and Photonics.

Di Cosmo, R. and Miller, D. (2019). Linear Logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2019 edition.

Grover, L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325.

Piro, F., Askarpour, M., and Di Nitto, E. (2020). Generalizing an exactly-1 sat solver for arbitrary numbers of variables, clauses, and k. In *1st International Workshop on Software Engineering and Technology, Q-SET 2020*, volume 2705, pages 27–37. CEUR-WS.