Quantum Algorithm for Multiplicative Linear Logic

Lorenzo Saraiva

January 2022

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)[3] for searching an element in an unordered database. GSA has been used in several different contexts, including SAT, kmeans, genetics algorithm and pixel identification. In this work we use Grover search algorithm to help in searching proofs of a subset of multiplicative linear logic in order to improve complexity when 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

GSA is one of the most famous quantum algorithms, and its goal is to search for an element in an unordered database. It has time complexity of $\sqrt{N}$, which outperforms any classical algorithm. This are the general steps of Grover algorithm main iteration n qubits:

- 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 the an operator D is used in order the amplify the amplitude of the state marked by the oracle. For such, an inversion about the mean (IAM) is performed. First, we bring the state $|\Psi\rangle$ back to $|0\rangle^{\otimes n}$ applying A^T and apply D.

Generalizing, the Grover iteration can be described as:

$G = AOA^T D$

This iteration has to be repeated $\lfloor \pi\sqrt{N}/4 \rfloor$ times in order to maximize the probability of measuring the desired state, where $N = 2^n$ is the database search space.

Our work is based on Alsing's entangled database search [1] using GSA. The main feature of Alsing's algorithm is that, instead of using $A = H^{\otimes n}$ to prepare the equal superposition state, A is chosen in order to encode an arbitrary list of pairs $\{s, t\}$. Thus, the input is an entangled database, with two sides. Each side has one part of the pair. Every entry on the left side is entangled to an entry on the right side. In GSA, it is necessary to know the searched value so that the oracle can be constructed. 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, thus 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 in the role of formulas as resources. For that reason, linear logic does not allow the rules of contraction and weakening to apply to all formulas but only those formulas marked with certain modals[2]. Linear logic has two different versions of conjunction and disjunction: an additive and a multiplicative. The classical $\wedge$, for example, is divided between the additive version, & (with), and the multiplicative version, $\otimes$ (tensor). Linear logic has also a sequent calculus proof system. The algorithm for finding a proof in this context has time complexity of N^2, where N is the number of atomic formulas (confirmar isso). The subpart of linear logic that deals only with the multiplicative connectors is called multiplicative linear logic(MLL). In this work, we will be using a subset of MLL, using only the tensor connector. The rules for the tesnor are shown in figure 2. Considering a linear logic sequent with $N = 4$ atomic clauses

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

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

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

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

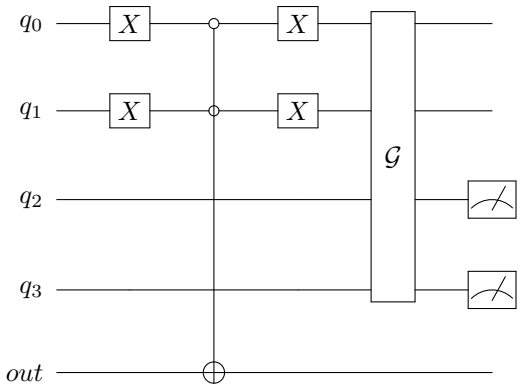

Figure 1: Circuit for GSA iteration with the Alsing entangled database for finding the value associated with 00

Algorithm 1 General Description

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

Algorithm 2 Function buildOracle() for creating the Oracle

Require: n, target
  while $i < n$ **do**
    if $target[i] == 1$ **then**
      XGate(circuit[i])
    end if
  end while
  MCTGate(register, pk)
  while $i < n$ **do**
    if $target[i] == 1$ **then**
      XGate(circuit[i])
    end if
  end while

4 Steps

The algorithm input is an entangled database, with two sides, each with one 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 $\sqrt{N}$ steps, it is necessary to have N copies of the paired database, where N is the number of unique atomic clauses. 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, an entangled databse that accurately represents the sequent must be constructed. In order to do so, we will need $N * 2n$ qubits, where $N = 2^n$. Then, the pairs $|a\rangle |b\rangle$ will be encoded as $|Na + b\rangle$, where a and b is the position of the clause in each side of the sequent. This encoding is proposed by Alsing[1]. Assuming we have $N = 4$ and $n = 2$, where n is the number of qubits necessary to represent a solution space of N values. n qubits can represent a solution space of 2^n, in this case 4. Thus our entangled database with 4 solutions will have 2 groups of n qubits. We'll treat the both groups of 2 qubits as a single array and prepare the 4 resulting encodings in the superposition, effectively only using N of the N^2 total possibilities in the superpostion. Then, we'll need N copies of the register, one for each clause. The construction of the database is not explicitly shown but its complexity is $O(n)$ or $O(\log N)$[1], taking $O(N \log N)$ in total. This process is not strictly part of the algorithm, which only receives a pre-constructed entangled database. With A being the operator that takes the circuit from the $|0\rangle^{\otimes n}$ state to the initial state - in our case, the entangled state, it is necessary to store the gate sequence used to encode a copy of the entangled database state so it can be used later in the IAM step of the GSA.

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:

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

Now, we can use our entangled database to find out the correct split for the leftmost atomic clauses of the right side.

4.3 Grover Search

Now that we have the entangled database of N copies of $2n$ qubits, we can perform the GSA. We start by picking the leftmost atomic clause of the right side.

The first step is to construct dynamically the oracle for the chosen element in the left side. The construction of this Oracle is based on the binary representation of the chosen clause 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 target, to perform the phase kickback, as can be seen in 2. 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 $\sqrt{N}$ times, and the measurement is applied 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 $\log N$ steps, making the overall complexity of the GSA step $O(\sqrt{N} \log N)$. This process finds the corresponding entry of a pair, but we need to find the N corresponding pairs. The issue is that the act of measuring the qubits destroys the prepared superposition corresponding to the pairs. This is the reason why we need the N copies of the prepared $2n$ qubit entangled database - so we perform the GSA for each pair on a different copy of the database in parallel. This can be done in a single circuit, in $\sqrt{N}$ 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. Thus, we will find a proof of the sequent We have $N = 4$ 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 $|Na + b\rangle$, with $N = 4$, we have

A	N0 + 2 = 2
B	N1 + 1 = 5
C	N2 + 3 = 11
D	N3 + 0 = 12

Thus we have the state of the quantum database as

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

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

Figure 2: Rules of $\otimes$-only Linear Logic

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 L$\otimes$ successive times, until every atomic clause 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 in parallel for every clause of the right side, so now we just apply the splits following the indexes $(3, 1, 0, 2)$, with the following results

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

6 Results

Given a previously constructed entangled database state, our algorithm has time complexity of $O(\sqrt{N}\log N)$ and has a space qubit complexity of $N \log N$, where N is the number of atomic clauses on the sequent. Even when taking into account the construction of the database , which takes $O(N \log N)$, steps, we're left with a time complexity of $O(\sqrt{N}\log N + N \log N) = O(N \log N)$ which outperforms the classical algorithm. Additionally, it is important to note that that when $N > 4$, we would need a controlled-NOT gate with more than 2 control qubits. For that, we need to concatenate the results of Toffoli gates, introducing additional $n - 2$ ancillary qubits[4]. We ran our circuit in the *qasm_simulator* provided by IBM. Each execution consisted in 1000 runs of the circuit. Simulations of the complete algorithm with the N clauses being searched for in the same circuit were made up to 4 atomic clauses, with a total of $N2log(N) + 1$ qubits, that is, 17, with every outcome being the correct state. An implementation of Alsing algorithm with only one copy of the entangled database was simulated up to 10 qubits each side with high probability of measuring the correct state.

7 Conclusion

References

[1] Paul M Alsing and Nathan McDonald. 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, 2011.

[2] Roberto Di Cosmo and Dale Miller. Linear Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2019 edition, 2019.

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

[4] Francesco Piro, Mehrnoosh Askarpour, Elisabetta Di Nitto, et al. 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, 2020.