QUANTUM ALGORITHM FOR ORACLE SUBSET PRODUCT

TREY LI

ABSTRACT. In 1993 Bernstein and Vazirani proposed a quantum algorithm for the Bernstein-Vazirani problem, which is given oracle access to the function \( f(a_1, \ldots, a_n) = a_1 x_1 + \cdots + a_n x_n \) (mod 2) with respect to a secret string \( x = x_1 \cdots x_n \in \{0, 1\}^n \), where \( a_1, \ldots, a_n \in \{0, 1\} \), find \( x \). We give a quantum algorithm for a new problem called the oracle subset product problem, which is given oracle access to the function \( f(a_1, \ldots, a_n) = a_1^{x_1} \cdots a_n^{x_n} \) with respect to a secret string \( x = x_1 \cdots x_n \in \{0, 1\}^n \), where \( a_1, \ldots, a_n \in \mathbb{Z} \), find \( x \). Similar to the Bernstein-Vazirani algorithm, it is a quantum algorithm for a problem that is originally polynomial time solvable by classical algorithms; and that the advantage of the algorithm over classical algorithms is that it only makes one call to the function instead of \( n \) calls.

1. INTRODUCTION

The earliest quantum algorithms are Deutsch's algorithm [Deu85], the Deutsch-Jozsa algorithm [DJ92], the Bernstein-Vazirani algorithm [BV93], Simon's algorithm [Sim97], Shor's algorithms [Sho94; Sho99] and Grover's algorithm [Gro96; Gro97]. It is generally hard to find quantum algorithms for problems that are classically hard. Therefore in [Sho03] Shor suggests to find faster quantum algorithms for problems already known to be classically solvable in polynomial time, and aim to provide polynomial factor speedups. This is a less exciting goal but an enlightening strategy. A success example is the HHL algorithm proposed by Harrow, Hassidim and Lloyd for solving linear systems of equations [HHL09], where the authors initially aimed to achieve polynomial speedup but the resulting algorithm turns out to have exponential speedup over the best classical algorithm. In this paper we give quantum algorithms for new problems that are classically solvable in polynomial time.

In the series of nine papers [Li22a; Li22b; Li22c; Li22d; Li22e; Li22f; Li22g; Li22h; Li22i] Li considered a wide range of problems that are conjecturally post-quantum hard, where in the eighth paper [Li22h] of the series the problems are raised to a theory of discrete exponential equations and noisy (discrete exponential equation) systems, which capture some famous problems such as integer factorization [Gal12], ideal factorization [HM89], isogeny factorization [CLG09], learning parity with noise (LPN) [BMT78; BFKL94], learning with errors (LWE) [Reg09] and learning with rounding (LWR) [BPR12].

A discrete exponential equation solving problem asks to solve an equation of the form

\[
a_1^{x_1} \cdots a_n^{x_n} = b
\]

for a binary string \( x = x_1 \cdots x_n \in \{0, 1\}^n \), where the bases \( a_1, \ldots, a_n \) are from a land \( L \), which is a monoid without the axiom of associativity [Li22h]. A typical example for \( L \) is the ring of integers \( \mathbb{Z} \), over which the equation solving problem is the classical subset product problem [GJ79, p. 224].

Date: November 28, 2022.
Email: treyquantum@gmail.com
A noisy system solving problem is an oracle problem that asks to solve for the secret string \( x = x_1 \ldots x_n \in \{0, 1\}^n \) given oracle access to samples of noisy discrete exponential equations of the form
\[
a_1^{x_1} \cdots a_n^{x_n} \cdot e = b,
\]
where the bases \( a_1, \ldots, a_n \) and the noise \( e \) are sampled from two (typically high entropy) distributions over \( L \) respectively. Note that the motivation of introducing invisible noises into discrete exponential equation systems was to make the system hard to solve.

The problem we consider in this paper is slightly different from the above two, it is an oracle problem but without noises, hence it is not classically hard. Also the oracle is a “best-case oracle” rather than an “average-case oracle”, namely it is a “function oracle” which responses with the evaluation \( b \) of the function to any requested bases \( a_1, \ldots, a_n \), rather than a “sampling oracle” which outputs random subset product samples \((a_1, \ldots, a_n, b)\).

Specifically, let \( x = x_1 \ldots x_n \in \{0, 1\}^n \) be a secret string. The subset product function with respect to \( x \) is the function \( f : \mathbb{Z}^n \rightarrow \mathbb{Z} \) defined as
\[
f(a_1, \ldots, a_n) = a_1^{x_1} \cdots a_n^{x_n}.
\]
The oracle subset product problem is given oracle access to \( f \), find \( x \).

A classical algorithm to solve this problem is to learn each bit \( x_i \) by querying \( f \) with \((a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) = (1, \ldots, 1, 2, 1 \ldots, 1)\), namely only \( a_i \) is set to be \( \neq 0 \) and \( \neq 1 \), and the rest are all set to be 1. Then one learns that \( x_i = 0 \) if \( f(a_1, \ldots, a_n) = 1 \), or \( x_i = 1 \) if \( f(a_1, \ldots, a_n) = 2 \). However this algorithm calls the oracle of \( f \) for \( n \) times.

We give a quantum algorithm that makes only one call to \( f \). The idea is to use Legendre symbols to reduce \( f : \mathbb{Z}^n \rightarrow \mathbb{Z} \) to a Boolean function \( h : \{0, 1\}^n \rightarrow \{0, 1\} \) and handle \( h \) in a similar way to the Bernstein-Vazirani algorithm.

## 2. Preliminaries

We give minimum background knowledge needed to understand our algorithm.

We denote strings as \( s = s_1 \ldots s_n \). We denote vectors using Dirac’s notation \(|v\rangle\). We denote the dual vector of \(|v\rangle\) by \langle v|\).

Let \( a \) be a nonnegative integer and \( a_1 \ldots a_n \in \{0, 1\}^n \) be its binary representation. Define
\[
|\alpha\rangle = |a_1 \ldots a_n\rangle = |a_1\rangle \cdots |a_n\rangle := |a_1\rangle \otimes \cdots \otimes |a_n\rangle,
\]
where \( \otimes \) is the Kronecker product.\(^1\) For example, when \( n = 1 \) we define
\[
|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\]
When \( n = 2 \) we define
\[
|0\rangle = |00\rangle = |0\rangle|0\rangle := |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\]
\(^1\)Note that we typically write a vector as \(|v\rangle = (v_1, \ldots, v_n)\). But if \( v = v_1 \ldots v_n \) is a binary string, then \(|v\rangle \neq (v_1, \ldots, v_n)\) because \(|v_1 \ldots v_n\rangle \neq (v_1, \ldots, v_n)\).
Let $A, B, C, D$ be matrices. The Kronecker product satisfies $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$.

In contrast to a bit in a classical computer, the basic unit of a quantum computer is a qubit. A qubit is either 0 or 1 after measurement. However, it can be both 0 and 1 before measurement. The “value” of a qubit before measurement is called its state, typically represented by a norm-1 vector $|\psi\rangle = a|0\rangle + \beta|1\rangle$, where $a$ and $\beta$ are complex numbers such that $|a|^2 + |\beta|^2 = 1$, and the squares $|a|^2$ and $|\beta|^2$ represent the probabilities of the qubit to be 0 and 1 respectively.

A unitary matrix is a matrix whose conjugate transpose is its inverse. A quantum gate is a two dimensional unitary matrix. A frequently used single-qubit gate is the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

which satisfies $H^2 = I$ (identity matrix). When act on $|0\rangle$ and $|1\rangle$ it gives

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle;$$

$$H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

We denote $|+\rangle := H|0\rangle$ and $|-\rangle := H|1\rangle$.

Let $a$ be an integer and $p$ be an odd prime. The Legendre symbol of $a$ above $p$ is $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod p \in \{-1, 0, 1\}$. Legendre symbols are multiplicative, namely $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.

### 3. Algorithm

The Bernstein-Vazirani algorithm [BV93] deals with linear functions. Our algorithm can be seen as an extension of the Bernstein-Vazirani algorithm to deal with nonlinear functions. The key question is how to transform a nonlinear function into a linear function. In our case, it is about transforming a subset product function $f(a_1, \ldots, a_n) = a_1 x_1 \cdots a_n x_n$ into a linear function $h(a_1, \ldots, a_n) = a_1 x_1 + \cdots + a_n x_n \pmod 2$ with respect to the same secret string $x = x_1 \cdots x_n$.

Let $p$ be a prime number and $c_0, c_1$ be positive integers such that the Legendre symbols satisfy $\left(\frac{c_0}{p}\right) = 1$ and $\left(\frac{c_1}{p}\right) = -1$. Define a Boolean function $h : \{0,1\}^n \rightarrow \{0,1\}$ as the following.
It takes as input a binary string \( \alpha = \alpha_1 \ldots \alpha_n \in \{0, 1\}^n \), calls \( f \) for \( f(c_{\alpha_1}, \ldots, c_{\alpha_n}) \in \mathbb{Z} \), computes the Legendre symbol \( \ell = \left( \frac{f(c_{\alpha_1}, \ldots, c_{\alpha_n})}{p} \right) \in \{-1, 1\} \), and outputs the bit \( \beta = \frac{1-\ell}{2} \in \{0, 1\} \).

Define a unitary operator (i.e. unitary matrix) as
\[
U_h = \sum_{\alpha \in \{0, 1\}^n} \sum_{j=0}^1 |\alpha, j \oplus h(\alpha)\rangle \langle \alpha, j|,
\]
where \( \oplus \) is the XOR operation.

Our algorithm is the following. It looks like the Berstein-Vazirani algorithm but the difference is the core function \( h \).

**Algorithm 1 Quantum Algorithm For Oracle Subset Product**

**Input:** The oracle \( O_f \) of a subset product function \( f \).
**Output:** The secret string \( x = x_1 \ldots x_n \) of \( f \) (with probability 1).

1. Initialize the input qubits to the \( |0\rangle^{\otimes n} |1\rangle \) state;
2. Apply Hadamard gates \( H^{\otimes (n+1)} \) to the state;
3. Apply \( U_h \) to the current state;
4. Apply Hadamard gates \( H^{\otimes (n+1)} \) to the current state;
5. Measure the first register \( |x_1\rangle \otimes \cdots \otimes |x_n\rangle \) and output the string \( x_1 \ldots x_n \).

The advantage of our algorithm over classical algorithms is similar to the advantage of the Berstein-Vazirani algorithm. That is, by applying \( U_h \) we make only one call to the function \( f \) for \( 2^n \) input vectors \( (c_1, \ldots, c_n) \in \mathbb{C}_0, c_1 \}^n \) simultaneously, which exhausts all \( 2^n \) possible inputs \( (\alpha_1, \ldots, \alpha_n) \in \{0, 1\}^n \) to \( h \). We will show in Theorem 1 that \( h(\alpha_1, \ldots, \alpha_n) = \alpha_1 x_1 + \cdots + \alpha_n x_n \mod 2 \) and thus the \( 2^n \) inputs \( (\alpha_1, \ldots, \alpha_n) \in \{0, 1\}^n \) fix the solution \( x = x_1 \ldots x_n \).

### 4. ANALYSIS

To prove the correctness of the algorithm, we need the following two well-known lemmas, whose proofs can be found in \([Por22]\).

**Lemma 1.** Let \( g : \{0, 1\}^n \to \{0, 1\} \) be any Boolean function and
\[
U_g = \sum_{\alpha \in \{0, 1\}^n} \sum_{j=0}^1 |\alpha, j \oplus g(\alpha)\rangle \langle \alpha, j|.
\]
Then
\[
U_g(|\alpha\rangle \otimes |\rangle) = (-1)^{g(\alpha)} |\alpha\rangle \otimes |\rangle.
\]

**Lemma 2.** Let \( x \in \{0, 1\}^n \) be an \( n \)-bit string \( x_0 \ldots x_{n-1} \). Then
\[
H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{\alpha=0}^{2^n-1} (-1)^{\alpha \cdot x} |\alpha\rangle,
\]
where \( \alpha \cdot x = \alpha_1 x_1 + \cdots + \alpha_1 x_1 \mod 2 \).

Now we are ready to prove the correctness of the algorithm.

**Theorem 1.** Algorithm 1 outputs \( x \) with probability 1.
Proof. 1. After the first step, the state of the qubits is
\[ |\psi_0\rangle = |0\rangle^\otimes n |1\rangle. \]

2. After the second step, the state of the qubits is
\[ |\psi_1\rangle = (H^\otimes n |0\rangle) \otimes (H|1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} |a\rangle \otimes |\rangle. \]

3. After the third step, the state of the qubits is
\[ |\psi_2\rangle = U_h |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} U_h |a\rangle \otimes |\rangle. \]

By Lemma 1, we can replace \( U_h \) by \((-1)^{h(a)}\) for any Boolean function \( h : \{0,1\}^n \rightarrow \{0,1\} \). We therefore have
\[ |\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} (-1)^{h(a)} |a\rangle \otimes |\rangle. \]

Now define the characteristic number of a Legendre symbol \( \ell \in \{-1,1\} \) to be \( \bar{\ell} = \frac{1-\ell}{2} \in \{0,1\} \).

Namely the “bar” notation turns 1 into 0 and \(-1\) into 1. Also denote \( \alpha_i := \frac{a_i}{p} \). Then
\[ h(a) = \frac{f(c_{a_1}, \ldots, c_{a_n})}{p} = \left( \frac{c_{a_1} \cdots c_{a_n}}{p} \right) = \left( \frac{c_{a_1}}{p} \right)^{x_1} \cdots \left( \frac{c_{a_n}}{p} \right)^{x_n} = a_1 x_1 + \cdots + a_n x_n \pmod 2, \]
where the forth line is from the fact that \( \left( \frac{a_i}{p} \right)^{x_i} \) equals 1 if there is an even number of terms \( \left( \frac{a_i}{p} \right)^{x_i} \) equals \(-1\); and \( \left( \frac{a_i}{p} \right)^{x_i} \cdot \left( \frac{a_i}{p} \right)^{x_n} = -1 \) if there is an odd number of terms \( \left( \frac{a_i}{p} \right)^{x_i} \) equals \(-1\). Denote \( \alpha \cdot x := a_1 x_1 + \cdots + a_n x_n \pmod 2 \). Then \( h(a) = \alpha \cdot x \). I.e., the functionality of \( h \) is essentially mod 2 inner product.\(^2\) It follows that
\[ |\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} (-1)^{\alpha \cdot x} |a\rangle \otimes |\rangle. \]

4. After the forth step, the state of the qubits is
\[ |\psi_3\rangle = H^\otimes (n+1) |\psi_2\rangle = H^\otimes n \left( \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} (-1)^{\alpha \cdot x} |a\rangle \right) \otimes (H|\rangle). \]

\(^2\)Note that \( h \) does not need to know \( x \) for computing \( \alpha \cdot x \). It accomplishes the task by calling \( f \).
By Lemma 2, we have that
\[ H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} (-1)^{a \cdot x} |a\rangle. \]
Plug this in the previous equation, and notice that \( H^{\otimes n} H^{\otimes n} = I \) (identity matrix) and \( H|\rangle = |1\rangle \), we have that
\[ |\psi_3\rangle = |x\rangle \otimes |1\rangle. \]
5. Since \( |x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle \), the fifth step returns \( x = x_1 \cdots x_n \) with probability 1. \( \square \)

5. **Generalization**

Note that the key to go with the Bernstein-Vazirani scenario is the transformation from the subset product function \( f \) to the Boolean function \( h \). Hence our algorithm can be generalized to work with any function \( f \) that can be efficiently transformed into a Boolean function \( h \). In particular, it works for oracle problems of more general subset product functions \( f(a_1,\ldots,a_n) = a_1^{x_1} \cdots a_n^{x_n} \) with \( a_1,\ldots,a_n \) in the order \( O_K \) of a number fields \( K \) such that the second power residue symbol is defined in \( O_K \) and that it can be efficiently computed. To achieve a quantum algorithm for oracle subset product over an order \( O_K \), all we need to do is to replace the Legendre symbols everywhere in this paper by appropriate second power residue symbols. Then all the arguments in this paper still hold.

**REFERENCES**


[Reg09] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Journal of the ACM (JACM)* 56.6 (2009), p. 34.


