Quantum Set Intersection with Application to Associative Memory

T. Salman,1,* K. Even,1 and Y. Baram1

1Computer Science Department, The Technion, Israel Institute of Technology, Haifa 32000, Israel

Abstract

We present a quantum algorithm for computing the intersection between two subsets of $n$-bit patterns using Grover’s quantum search algorithm. We apply this algorithm to implement a model of associative memory with pattern completion and error correction abilities. We introduce the notion of memory as a quantum operator and prove that the storage capacity is exponential in $n$, while the time complexity of operations is sub-exponential.

*Electronic address: tamer@cs.technion.ac.il
I. INTRODUCTION

Associative memory stores and retrieves patterns with error correction or pattern completion on the input. The task can be defined as memorizing \( m \) pairs of data \((x^i, y^i)\), where \( x^i \) is an \( n \) dimensional vector and \( y^i \) is a \( q \) dimensional vector, and outputting \( y^i \) when presented with \( \tilde{x} \), which is a faulty or a partial version of \( x^i \). A specific case of associative memory is the auto-associative memory, in which \( y^i \equiv x^i \), \( \forall i \in \{1, \ldots, m\} \). Associative memory can be defined over a continuum, where \((x^i, y^i) \in \mathbb{R}^n \times \mathbb{R}^q\), or over a binary space, where \((x^i, y^i) \in \{0, 1\}^n \times \{0, 1\}^q\). In this paper, we concentrate on the binary model.

Algorithms for implementation of associative memory have been extensively studied in the neural networks literature \[\text{[Hop82, MPRV87, Bar89, Bar91, Bar94, Kan93, Has95, Hay99, WB00, IWAH01]}\]. They can be divided into to categories: Recurrent models, such as synchronous and asynchronous Hopfield networks \[\text{[Hop82, MPRV87]}\], and feed-forward models, such as Hamming networks \[\text{[WB00, IWAH01]}\] and sparse distributed memory \[\text{[Kan93]}\]. All models have obvious shortcomings in capacity and time consumption. Recurrent networks, such as Hopfield network, have a capacity of \( m \approx O(n/\log(n)) \) that can mostly be retrieved with high probability, where \( n \) is the pattern size. The model requires, however, \( n \) neurons.\[\text{[MPRV87, Hay99]}\]. Similarly, the capacity of recurrent networks with \( n \) neurons is bounded linearly \[\text{[Hay99]}\]. Furthermore, these networks are designed to cope only with the auto-associative model.

On the other hand, feed forward models achieved exponential capacity, but in return paid heavily in space complexity and required an exponential number of neurons \[\text{[WB00, IWAH01]}\]. In addition, adding a new memory pattern to a feed-forward model requires the addition of a neuron in the hidden layer, which changes the structure and connectivity of the network.

In this paper we present a model of associative memory which, using quantum computation, is able to store, correct or complete an exponential number of pairs. In addition, there is no requirement for a minimal distance between stored patterns, since the quantum model chooses any of a number of closest memory patterns unlike classical models that might produce spurious memories. All stored patterns are equilibrium points of the model and can be retrieved. Spurious memories arise with very low probability, which asymptotically vanishes as the number of bits of a pattern grows.

The main algorithm in our model consists of a new quantum operator that approximates
the computation of an intersection between two subsets without the need to compute the intersection itself. It is based on a modification to Grover’s quantum search algorithm [Gro, BBHT96] and includes an alternative iterator.

A. Quantum Computation

Quantum computation is a new emerging area of research that is believed to be superior to classical computation in the complexity of computing time and space. Since Shor introduced his algorithm for factoring numbers in polynomial time in 1994 [Sho94] it is believed that quantum computation has the ability to solve problems more efficiently than classical computation. Two years later, Grover introduced a sub-exponential algorithm for searching an unsorted database of size \( N \) in \( O\left(\sqrt{N}\right) \) time [Gro, BBHT96]. These two algorithms were the most influential on quantum computation advancement for different reasons. Shor’s algorithm gave a polynomial time algorithm for a problem with an unknown classical polynomial algorithm. On the other hand, the importance of Grover’s algorithm lies in the fact that it can be easily proved that the best classical search of a database of size \( N \) consumes \( O(N) \) time.

In classical computers, information is stored in binary data units called bits that are allowed to have two values, 0 and 1. The basic quantum unit is called a quantum-bit (qubit) and, due to its quantum properties, is allowed to have any superposition of the values 0 and 1, which are denoted by the Dirac notation by \( |0\rangle \) and \( |1\rangle \). More specifically, a qubit is allowed to have any value \( \alpha |0\rangle + \beta |1\rangle \), such that \( |\alpha|^2 + |\beta|^2 = 1 \), where \( \alpha \) and \( \beta \) are complex amplitudes of the states \( |0\rangle \) and \( |1\rangle \) respectively. A unary representation of a qubit can be given as a vector of two complex values as follows

\[
|\alpha |0\rangle + \beta |1\rangle \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{1}
\]

A state can also be represented by more than one qubit. An \( n \) qubit state can be represented as a concatenation of the \( n \) qubits, \( |x_{n-1} \ldots x_1 x_0\rangle \equiv |x_{n-1}\rangle \otimes \ldots \otimes |x_1\rangle \otimes |x_0\rangle \), where \( \otimes \) denotes the tensor product. A tensor product of two qubits \( |a\rangle = \alpha |0\rangle + \beta |1\rangle \) and \( |b\rangle = \gamma |0\rangle + \delta |1\rangle \) is
\[ |a⟩ \otimes |b⟩ = (\alpha |0⟩ + \beta |1⟩) \otimes (\gamma |0⟩ + \delta |1⟩) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \gamma \\ \alpha \delta \\ \beta \gamma \\ \beta \delta \end{pmatrix} = (\alpha \gamma |00⟩ + \alpha \delta |01⟩ + \beta \gamma |10⟩ + \beta \delta |11⟩) \] (2)

The computational basis states \(|00⟩, |01⟩, |10⟩, \) and \(|11⟩\) are also referred to as \(|0⟩, |1⟩, |2⟩, \) and \(|3⟩\) respectively.

Operations applied to a system of qubits must be unitary operators and can describe either evolvement in time or change of the representation basis. An operator \(U\) is said to be unitary if \(U^\dagger U = I\), where \(U^\dagger\) is the conjugate-transpose of \(U\). For further details refer to [NC00]. Measurement operators, on the other hand, have to be only Hermitian [NC00]. The standard logic operations are not all unitary. In classical terms, a unitary operation is reversible. For example, the operation \(AND\) that has two inputs and one output is not reversible, because knowing the output does not lead to knowing the input. However, such operations can be replaced with reversible universal equivalents. The \(Controlled- NOT\) and the \(Toffoli\) operations that are depicted in Fig. 1 are reversible operations. The \(Toffoli\) operation is a universal reversible gate that can implement any classical logical operation [NC00].

FIG. 1: (a) \(Controlled- NOT (C- NOT)\). (b) \(Toffoli (C- C- NOT)\).

All the above mentioned operations are classical. However, some quantum operators have no classical equivalent. A well known quantum operator on a single qubit is the Walsh-Hadamard (Hadamrad) operator

\[ H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \] (3)
Applying Hadamard operator on an $n$-qubit system is equivalent to applying the single-qubit Hadamard to each qubit separately, where

$$H^{\otimes n} = \frac{1}{\sqrt{2}} \begin{pmatrix} H^{\otimes n-1} & H^{\otimes n-1} \\ H^{\otimes n-1} & -H^{\otimes n-1} \end{pmatrix}$$  \hspace{1cm} (4)$$

The states $\{\lvert + \rangle, \lvert - \rangle \}$ is called the Hadamard basis, where $\lvert + \rangle \equiv H \lvert 0 \rangle$ and $\lvert - \rangle \equiv H \lvert 1 \rangle$. We note that applying an $n$-qubit Hadamard operator to an $n$-qubit register set to zeros yields the superposition of all possible basis states of the $n$ qubits:

$$H^{\otimes n} \lvert 0 \rangle^{\otimes n} = \lvert + \rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \lvert i \rangle$$  \hspace{1cm} (5)$$

where $N \equiv 2^n$ is the number of all basis states and $\lvert i \rangle$ is the $n$-qubit basis state represented by the decimal number $i$.

A quantum state can retain superposition as long as no external interference is made (e.g. inspection, measurement, or observation). Once the system is measured, the quantum superposition decoheres and a classical state is revealed with some probability. The probability of measuring a state of the superposition with amplitude $\alpha$ is $|\alpha|^2$. In general, quantum algorithms aim to process a superposition of possible solutions so that upon measurement the solution is revealed with high probability. A comprehensive introduction of quantum computation can be found in [NC00].

### B. Grover’s Quantum Search Algorithm

Given a database of $N \equiv 2^n$ unsorted elements of $n$ bits each, any classical search would require $O(N)$ queries to find a desired element. In 1996, Grover presented a quantum computational algorithm that searches an unsorted database with $O(\sqrt{N})$ operations [Gro]. The algorithm performs a series of $O(\sqrt{N})$ unitary operations on the superposition of all basis states, in which, using a phase oracle that rotates the phase of some marked state by $\pi$, then amplifies their amplitude by rotating all the states around the average of all states. A phase oracle of a basis state $\lvert \tau \rangle$ is an operator $I_\tau = I - 2 \lvert \tau \rangle \langle \tau \rvert$, where $\langle \tau \rvert$ is the complex conjugate of $\lvert \tau \rangle$, therefore, $I_\tau$ is similar to the identity matrix $I$ except it has $-1$ on the $\tau$th element of the diagonal. An oracle is a function $f_\tau$ with input $x \in \{0,1\}^n$ that outputs 1 if $x = \tau$ and 0 otherwise:
A quantum oracle realizing $f_{\tau}$ is depicted in Fig. 2 where $b = 0$ yields $|f_{\tau}(x)\rangle$ in the output and $b = 1$ yields $|\bar{f}_{\tau}(x)\rangle$ in the output, where $\bar{f}_{\tau}$ is the complementary function of $f_{\tau}$. The phase oracle $I_\tau$ is achieved when the oracle is presented with $b = |\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, since $|b \otimes f(x)\rangle = \frac{1}{\sqrt{2}}[|f_{\tau}(x)\rangle - |\bar{f}_{\tau}(x)\rangle] = (-1)^{f_{\tau}(x)}|\rangle$.

The rotation of all amplitudes around the average amplitude is done by the operator $HI_0H$, where $I_0$ rotates the phase of the state $|0\rangle^\otimes n$. More elaborate discussions of the used operators and their properties can be found in [BBHT96, BBB+99].

Consider a quantum oracle $B$ that reveals a value $|y_i\rangle$ and a key $|x_i\rangle$ when presented with a key $|x_i\rangle$ and $|0\rangle^\otimes q$ ($q$ zero qubits) as follows

$$B(|x_i\rangle \otimes |0\rangle^\otimes q) = |x_i\rangle \otimes |y_i\rangle \quad (7)$$

for each basis state $|x_i\rangle$. Assuming all possible $x_i$s exist in the database and are associated with values $y_i$s, one can apply the quantum oracle $B$ to the superposition of all possible basis states $|x_i\rangle$ simply by producing $H^\otimes n |0\rangle^\otimes n$. Non existing patterns $x_i$ can be associated with a unique identifiable $y_i$ value(e.g. $|0\rangle^\otimes q$). As a result, the superposition of all pairs $|x_i\rangle \otimes |y_i\rangle$ is created as follows:

$$B (H^\otimes n \otimes I) (|0\rangle^\otimes n \otimes |0\rangle^\otimes q) = B (|+\rangle^\otimes n \otimes |0\rangle^\otimes q) = \frac{1}{\sqrt{N}} \sum_{x_i=0}^{N-1} |x_i\rangle \otimes |y_i\rangle \quad (8)$$

In general, the oracle $B$ is applied at the beginning of Grover’s quantum search, however, in the case of an auto-associative memory, this step can be skipped and instead one starts with a superposition on all possible values given by $H |0\rangle^\otimes n$.  

6
Define Grover’s operator as

$$Q = -HI_0HI_\tau$$  \hspace{1cm} (9)$$

where the sign “−” stands for the global phase flip that has no physical meaning and is performed only for analysis convenience.

Grover’s quantum search algorithm consists of applying $Q$ to the input $H|0\rangle^{\otimes n}$ for $T \equiv \frac{\pi}{4}\sqrt{N}$ times, then measuring the output [Gro]. We refer to the measuring operation as $Measure()$. A result of [BBHT96] states that the outcome of

$$Measure (Q^T H^{\otimes n} |0\rangle^{\otimes n})$$  \hspace{1cm} (10)$$
yields $|\tau\rangle$ with almost certainty.

Grover’s algorithm was improved several times to cope with more general settings. [BBHT96] showed that the algorithm can be applied with multiple marked states. The phase oracle is then set to $I_X = I - 2\sum_{\tau \in X} |\tau\rangle \langle \tau|$, where $X$ is the set of marked states. The algorithm runs $O \left( \sqrt{\frac{N}{r}} \right)$, where $r = |X|$ is the number of marked states. Additional improvements were made by [BBHT96] and [BHT98] that coped with an unknown number of marked states in complexity $O \left( \sqrt{\frac{N}{r}} \right)$. A third improvement is an algorithm that outputs a marked state when initiated with a state of an arbitrary amplitude distribution [BBB+99]. More details concerning Grover’s algorithm and its generalizations can be found in [Ken01].

In the next sections we present our algorithm for set intersection and its use in a quantum model of associative memory. We focus our analysis on the auto-associative memory model. However, presenting the algorithm with the quantum superposition of pairs $x_i$ and $y_i$ created by the use of the oracle $B$ makes it a model for general associative memory with no additional cost, as a quantum search can yield $y_i$ upon measurement of $x_i$.

However, before doing so, we feel the need to further analyze Grover’s algorithm and determine a lower bound on the maximal probability of measuring a marked state. The same analysis will be used to determine the success probability of our algorithm.
II. ANALYZING GROVER’S ALGORITHM WITH MULTIPLE MARKED STATES

Grover’s algorithm performs a rotation in the space defined by the set of marked states and the set of unmarked states. Define the following basis \( \{ |l_1 \rangle, |l_2 \rangle, \ldots \} \), where

\[
|l_1 \rangle = \frac{1}{\sqrt{r}} \sum_{i \in X} |i \rangle
\]

and

\[
|l_2 \rangle = \frac{1}{\sqrt{N-r}} \sum_{i \notin X} |i \rangle
\]

and the rest of the basis states are any orthonormal extension. Grover’s operator can be represented in the space defined by \( (|l_1 \rangle, |l_2 \rangle) \) as follows:

\[
\begin{pmatrix}
1 - \frac{2r}{N} & 2\sqrt{r} \sqrt{N-r} \\
-2\sqrt{r} \sqrt{N-r} & 1 - \frac{2r}{N}
\end{pmatrix}
\]

Each iteration performs a rotation of angle \( w \), such that \( \cos w = 1 - \frac{2r}{N} \) and the initial angle between the initial state and \( |l_1 \rangle \) is \( \Phi \), such that \( \tan \Phi = \sqrt{\frac{r}{N-r}} \). Therefore, the state after \( t \) iterations is

\[
|\Psi(t)\rangle = \sin (wt + \Phi) |l_1 \rangle + \cos (wt + \Phi) |l_2 \rangle
\]

The probability of measuring a marked state from \( \Psi(t) \) is

\[
Pr(Succ) = \sin^2 (wt + \Phi)
\]

and the optimal \( t \) for measuring is, hence, \( t_{opt} \), such that

\[
\sin^2 (wt_{opt} + \Phi) = \frac{1}{2} - \frac{1}{2} \cos (2wt_{opt} + 2\Phi) = 1
\]

which means that

\[
2wt_{opt} + 2\Phi = \pi
\]

resulting in

\[
t_{opt} = \frac{\pi - 2\Phi}{2w} = \frac{\pi - 2 \arctan \left( \frac{r}{N-r} \right)}{2 \arccos \left( 1 - \frac{2r}{N} \right)}
\]

But since we have to perform a complete number of iterations, then the algorithm measures the state after

\[
T = \left\lfloor \frac{\pi - 2 \arctan \left( \frac{r}{N-r} \right)}{2 \arccos \left( 1 - \frac{2r}{N} \right)} \right\rfloor
\]
iterations.

Grover’s algorithm assures that for a large enough $N$ the probability of measuring a marked state is asymptotically close to 1 [Gro, BBHT96]. However, for a small number of qubits, this assumption does not hold, and we are interested in bounding the success probability from below. The reason the probability does not reach 1 in such cases is that the optimal time to measure the state is in between the two best iterations, as demonstrated in Fig. 3. Suppose that the algorithm stopped with the state in angle $\delta$ from the marked state after $T$ iterations, then the probability of measuring the marked state is

$$Pr(T) = \sin^2 (wT + \Phi) = \sin^2 (w (t_{opt} - \Delta T) + \Phi) = \sin^2 (wt_{opt} + \Phi - w\Delta T)$$ (20)

where

$$\Delta T = t_{opt} - T < 1$$ (21)

Using trigonometric identities and Eq. 21, we obtain

$$Pr(T) = [\sin (wt_{opt} + \Phi) \cos (w\Delta T) - \cos (wt_{opt} + \Phi) \sin (w\Delta T)]^2$$

$$= \sin^2 (wt_{opt} + \Phi) \cos^2 (w\Delta T)$$

$$- 2 \sin (wt_{opt} + \Phi) \cos (w\Delta T) \cos (wt_{opt} + \Phi) \sin (w\Delta T)$$

$$+ \cos^2 (wt_{opt} + \Phi) \sin^2 (w\Delta T)$$

$$= \cos^2 (w\Delta T)$$

$$> \cos^2 (w) = \left(1 - \frac{2r}{N}\right)^2 = 1 - \frac{4r}{N} + \frac{4r^2}{N^2}$$ (22)

For above lower bound on the probability of success is valid for any number of qubits. For instance, for a 4-qubits register and 1 marked state, Grover’s algorithm finds the marked state.
state with probability higher than \((1 - \frac{2}{16})^2 = \frac{49}{64} \approx 77\%\) and for a 7-qubits register with 1 marked state, the probability is higher than \((1 - \frac{2}{16})^2 = 92\%\).

III. INTERSECTION VIA QUANTUM COMPUTATION

Grover’s quantum search algorithm for multiple marked states [BBHT96] produces any member of a set of marked states \(K\) when given a phase version \(I_K\) of an oracle \(f_K\) of the form

\[
f_K(x) = \begin{cases} 
1, & x \in K \\
0, & x \notin K
\end{cases}
\]  

Consequently, given a phase oracle \(I_K\), Grover’s algorithm chooses a member of the subset \(K\) of basis states out of the set of all \(n\)-qubit basis states \(A = \{|i\rangle | 0 \leq i \leq 2^n\}\).

Suppose that in addition to the oracle \(f_K\), we have an oracle \(f_M\), such that

\[
f_M(x) = \begin{cases} 
1, & x \in M \\
0, & x \notin M
\end{cases}
\]  

where \(K, M \subseteq A\). We are interested in outputting any member of the intersection set \(K \cap M\).

We present two algorithms for finding a member of the intersection between two subsets \(K\) and \(M\) based on the oracles \(f_K\) and \(f_M\). The first relies on the computation of the phase version of the intersection oracle and uses it in Grover’s quantum search to mark the wanted states. The second uses a compound operator for quantum search that approximates the intersection operator without the need to compute the intersection itself. The latter is better suited for cases where the two subsets differ in size and changing rate. Suppose one of the subsets changes more often than the other, then, the algorithm requires computing only the phase oracle of the changing subset. When the changing subset is the smaller one, the advantage of the second algorithm in time consumption becomes bigger. The two algorithms create a quantum superposition, in which the members of the intersection subset have high amplitudes and the rest of the states have low amplitudes. The last step of both algorithms is measuring the system, which leads to revealing a member of the intersection with high probability.
1. Using an Apriori Intersection Oracle

In Grover’s algorithm, an operator $I_\tau$ that flips the phase of a state is a phase version of a given oracle $f_\tau$ that outputs 1 if the input is $\tau$ and 0 otherwise. The case is similar for a set of marked states $X$, where $f_X$ outputs 1 if and only if the input is a member of the set $X$ and 0 otherwise. Consequently, choosing a member of an intersection between two subsets $K$ and $M$ is simply using Grover’s quantum search with a phase oracle that marks the states in $K \cap M$. Define the intersection oracle as

$$f_{K\cap M}(x) = f_K(x)f_M(x) = \begin{cases} 1, & x \in K \cap M \\ 0, & x \notin K \cap M \end{cases}$$

(25)

which can be implemented using the oracles in Eq. 23, Eq. 24, and a Toffoli gate as depicted in Fig. 4.

![Fig. 4: The intersection oracle $F_{K\cap M}$ created using $f_M$, $f_K$, and Toffoli.](image)

When the input to the oracle is $|x\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$ then the output is $|x\rangle \otimes |f_K(x)\rangle \otimes |f_M(x)\rangle \otimes |f_{K\cap M}(x)\rangle$. The intersection phase oracle is realized when $|b\rangle = |-\rangle$, then, the input $|x\rangle \otimes |0\rangle \otimes |0\rangle \otimes |-\rangle$ will cause the output to be $|x\rangle \otimes |f_K(x)\rangle \otimes |f_M(x)\rangle \otimes \left[(-1)^{f_{K\cap M}(x)} |-\rangle\right]$.

Retrieving a pattern in the intersection between the two subsets is then accomplished by using Grover’s quantum search algorithm with the phase version $I_{K\cap M}$ of the oracle $f_{M\cap K}$. If the size of the intersection set is unknown, one may use a generalization of the quantum search for an unknown number of marked states [BBHT96, BHT98].

However, in certain applications, such as the one presented in the following sections, one subset might be much larger and less often changed than the other. In such cases, the
computation of the intersection oracle requires unnecessary heavy computations. Denote 
k = |K| and \(m = |M|\), and assume \(k << m\). We offer a modified quantum search algorithm that does not require the computation of the intersection oracle, but instead approximates it using two different oracles, \(I_M\), which we previously own, and \(I_K\), which we prepare for each query. Consequently, we only have to create the smaller oracle upon each query. The algorithm is presented in the next subsection.

2. Using a Modified Iterator for Quantum Search

Suppose that we own a phase oracle \(I_M\) and we are presented with a subset \(K\) of size \(k << m\). We are interested in retrieving a pattern \(x \in K \cap M\). We create the completion operator \(I_K\) in \(O(k)\) by marking all states that are possible completions, independent of the memory size \(m\). Retrieving a pattern \(x \in K \cap M\) using the two operators \(I_K\) and \(I_M\) is possible through a modification of Grover’s quantum search algorithm. At first, assume that the intersection set is not empty and its size \(|K \cap M|\) is known. These assumptions can be omitted by using the generalizations of Grover’s algorithm as presented in [BBHT96] and [BHT98]. The suggested modification simply applies Grover’s iterator in an alternating fashion, marking a different subset each time. In the odd iterations we mark the states in \(K\) and in the even iterations we mark the states in \(M\). The number of iterations remains the same as in the original algorithm. Accordingly, we present the following algorithm for observing a member of the intersection set:

**Algorithm 1 Quantum Subset Intersection**

*Given: Phase oracles \(I_M\) and \(I_K\)*

*Denote: \(Q_M \equiv -HI_M^0H\), \(Q_K \equiv -HI_K^0H\), and \(Q \equiv Q_MQ_K\)*

1. Let \(|\Psi\rangle = H|0\rangle^\otimes n\).
2. Repeat \(T = \left\lfloor \frac{\pi}{8} \sqrt{N/|K \cap M|} \right\rfloor\) times
   \(|\Psi\rangle = Q|\psi\rangle\)
3. Measure \(|\psi\rangle\).

As shown below, in Trm. 1, Alg. 1 produces a state in the intersection between \(K\) and \(M\) with probability close to 1.
We validated this algorithm also by simulations for all subset sizes with small intersection sets up to 100-qubits, and found that a state in the intersection can be observed with probability very close to 1. We also traced the amplitudes of the states in Eq. 30-Eq. 33 by simulation. In Fig. 5 we plot 3-dimensional graphs showing the amplitude of the state $|l_1\rangle$ vs. the combinations of couples of the states $|l_2\rangle$, $|l_3\rangle$, and $|l_4\rangle$. The graphs trace the amplitudes for three iterations of the algorithm, in which the state changes in each iteration by the operator $Q$ from Eq. 26. The graphs show that the amplitude of $|l_1\rangle$ increases with each iteration.

![Graph]

FIG. 5: The amplitudes of the different basis states $|l_1\rangle$, $|l_2\rangle$, $|l_3\rangle$, and $|l_4\rangle$. The plots show 3 iterations of Alg. 1. (a) $|l_1\rangle$ vs. $|l_2\rangle$ and $|l_3\rangle$. (b) $|l_1\rangle$ vs. $|l_2\rangle$ and $|l_4\rangle$. (c) $|l_1\rangle$ vs. $|l_3\rangle$ and $|l_4\rangle$.

A more general look at the simulations can be seen in Fig. 6, where we traced the amplitudes for a complete circle in the 4-dimensional state space. We gathered the plotted graphs by sampling the amplitudes every 100 iterations of the algorithm. Fig. 6 shows that the amplitude of $|l_1\rangle$ reaches a value very close to 1.

However, Alg. 1 assumes that the size of the intersection subset $|K \cap M|$ is known to determine the number of iterations. In the more general case, were $|K \cap M|$ is unknown we apply the same modification for an unknown number of marked states presented in [BBHT96]. Alternatively, we can use the quantum counting algorithm presented in [BHT98] for the apriori estimation of the number of marked states. In both cases the time complexity of the subset intersection algorithm is still sub-exponential, and both methods do not impose any computational overhead.
FIG. 6: The amplitudes of the different basis states $|l_1⟩$, $|l_2⟩$, $|l_3⟩$, and $|l_4⟩$. The plots show a complete 4 dimensional circle created by repeated applications of $Q$ in Eq. 26. The plots are created by sampling each 100 iterations of Alg. 1. (a) $|l_1⟩$ vs. $|l_2⟩$ and $|l_3⟩$. (b) $|l_1⟩$ vs. $|l_2⟩$ and $|l_4⟩$. (c) $|l_1⟩$ vs. $|l_3⟩$ and $|l_4⟩$.

IV. ANALYSIS OF THE QUANTUM INTERSECTION ALGORITHM

In the following theorem we prove that Alg. 1 measures a member of the intersection with high probability.

**Theorem 1** Let $A$ denote the set of $n$-qubit basis states, $|A| = N = 2^n$, and let $I_K$ and $I_M$ be phase oracles that mark a number of $n$-qubit states $|K⟩$ and $|M⟩$ respectively, where $K, M \subset A$, and

$$Q \equiv Q_MQ_K = (H_{l_0}H_{l_M}H_{l_0}H_{l_K})$$

then the following hold true:

$$T = \arg \max_t \Pr (\text{Measure} (Q^tH |0⟩) \in K \cap M)$$

$$= \frac{\pi/2 - \arctan \left(\sqrt{\frac{|K \cap M|}{N - |K \cap M|}}\right)}{\arccos \left(1 - \frac{8|K \cap M|(N - |M|)}{N^2}\right)}$$

$$\Pr (\text{Measure} (Q^T H |0⟩) \in K \cap M) \geq \left(1 - \frac{4|K \cap M|(N - |M|)}{N^2}\right)^2$$

$$\Pr (\text{Measure} (Q^T H |0⟩) \in K \cap M) \xrightarrow{N \gg |K \cap M| \geq 1} 1$$

**Proof of Theorem 1** The Hilbert space of the states of the $n$-qubit register can be divided into four different subspaces; the subspace spanned by the states in $K \cap M$, the subspace
spanned by the states in $K \setminus M$, the subspace spanned by the states in $M \setminus K$, and the subspace spanned by the states in $A \setminus (K \cup M)$. Accordingly, we select an orthonormal basis for representing the operator. The first four basis states are:

\[ |l_1\rangle \equiv \frac{1}{\sqrt{|K \cap M|}} \sum_{i \in K \cap M} |i\rangle \quad (30) \]
\[ |l_2\rangle \equiv \frac{1}{\sqrt{|K \setminus M|}} \sum_{i \in K \setminus M} |i\rangle \quad (31) \]
\[ |l_3\rangle \equiv \frac{1}{\sqrt{|M \setminus K|}} \sum_{i \in M \setminus K} |i\rangle \quad (32) \]
\[ |l_4\rangle \equiv \frac{1}{\sqrt{|A \setminus (K \cup M)|}} \sum_{i \in A \setminus (K \cup M)} |i\rangle \quad (33) \]

The rest of the basis consists of orthonormal extensions of these states. Denote $k = |K|$, $m = |M|$, and $r = |K \cap M|$, then

\[ H \langle 0 | = \sqrt{\frac{r}{N}} |l_1\rangle + \sqrt{\frac{k-r}{N}} |l_2\rangle + \sqrt{\frac{m-r}{N}} |l_3\rangle + \sqrt{\frac{N-m-k+r}{N}} |l_4\rangle \quad (34) \]

The operator $Q_K$ affects only the four states $(|l_1\rangle, |l_2\rangle, |l_3\rangle, |l_4\rangle)$ as follows:

\[ Q_K |l_1\rangle = -H |l_0\rangle H |l_1\rangle = -H (I - 2 |0\rangle \langle 0|) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_1\rangle = \quad (35) \]
\[ \left( 1 - \frac{2r}{N} \right) |l_1\rangle + \left( -\frac{2\sqrt{r(k-r)}}{N} \right) |l_2\rangle + \left( -\frac{2\sqrt{r(m-r)}}{N} \right) |l_3\rangle + \left( -\frac{2\sqrt{r(N-m-k+r)}}{N} \right) |l_4\rangle \]

\[ Q_K |l_2\rangle = -H |l_0\rangle H |l_2\rangle = -H (I - 2 |0\rangle \langle 0|) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_2\rangle = \quad (36) \]
\[ \left( -\frac{2\sqrt{r(k-r)}}{N} \right) |l_1\rangle + \left( 1 - \frac{2(k-r)}{N} \right) |l_2\rangle + \left( -\frac{2\sqrt{(k-r)(m-r)}}{N} \right) |l_3\rangle + \left( -\frac{2\sqrt{(k-r)(N-m-k+r)}}{N} \right) |l_4\rangle \]
Similarly, we obtain the matrix form of $Q_K$:

$$Q_K |l_3\rangle = -HI_0H_I K |l_3\rangle = -H (I - 2 |0\rangle \langle 0|) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_3\rangle =$$

$$\left( \frac{2 \sqrt{r(m-r)}}{N} \right) |l_1\rangle + \left( \frac{2 \sqrt{(k-r)(m-r)}}{N} \right) |l_2\rangle + \left( \frac{2(m-r)}{N} - 1 \right) |l_3\rangle + \left( \frac{2 \sqrt{(m-r)(N-m-k+r)}}{N} \right) |l_4\rangle$$

$$Q_K |l_4\rangle = -HI_0H_I K |l_4\rangle = -H (I - 2 |0\rangle \langle 0|) H \left( I - 2 \sum_{i \in K} |i\rangle \langle i| \right) |l_4\rangle =$$

$$\left( \frac{2 \sqrt{(N-m-k+r)}}{N} \right) |l_1\rangle + \left( \frac{2 \sqrt{(k-r)(N-m-k+r)}}{N} \right) |l_2\rangle + \left( \frac{2 \sqrt{(m-r)(N-m-k+r)}}{N} \right) |l_3\rangle + \left( \frac{2(N-m-k+r)}{N} - 1 \right) |l_4\rangle$$

yielding $Q_K$ in matrix form:

$$Q_K = \begin{pmatrix}
1 - \frac{2r}{N} & -\frac{2 \sqrt{r(k-r)}}{N} & \frac{2 \sqrt{r(m-r)}}{N} & \frac{2 \sqrt{r(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(k-r)}}{N} & 1 - \frac{2(k-r)}{N} & \frac{2 \sqrt{k-r(m-r)}}{N} & \frac{2 \sqrt{(k-r)(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(m-r)}}{N} & -\frac{2 \sqrt{k-r(m-r)}}{N} & 2(m-r) - 1 & \frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(N-k-m+r)}}{N} & -\frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} & \frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} & 2(N-k-m+r) - 1
\end{pmatrix}$$

(39)

Similarly, we obtain the matrix form of $Q_M$:

$$Q_M = \begin{pmatrix}
1 - \frac{2r}{N} & \frac{2 \sqrt{r(k-r)}}{N} & -\frac{2 \sqrt{r(m-r)}}{N} & \frac{2 \sqrt{r(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(k-r)}}{N} & \frac{2(k-r)}{N} - 1 & -\frac{2 \sqrt{k-r(m-r)}}{N} & \frac{2 \sqrt{(k-r)(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(m-r)}}{N} & \frac{2 \sqrt{k-r(m-r)}}{N} & 1 - 2(m-r) & \frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} \\
-\frac{2 \sqrt{r(N-k-m+r)}}{N} & \frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} & \frac{2 \sqrt{(m-r)(N-k-m+r)}}{N} & 2(N-k-m+r) - 1
\end{pmatrix}$$

(40)

Substituting Eq. 39 and Eq. 40 in Eq. 26 we obtain:

$$Q = \begin{pmatrix}
1 - \frac{8r(N-m)}{N^2} & \frac{4 \sqrt{r(k-r)(N-2m)}}{N^2} & \frac{8 \sqrt{r(m-r)(N-m)}}{N^2} & \frac{4 \sqrt{r(N-k-m+r)(N-2m)}}{N^2} \\
\frac{4 \sqrt{r(k-r)(N-2m)}}{N^2} & \frac{8m(k-r)}{N^2} - 1 & \frac{4 \sqrt{k-r(m-r)(N-2m)}}{N^2} & \frac{8 \sqrt{(k-r)(N-k-m+r)(N-2m)}}{N^2} \\
\frac{8 \sqrt{r(m-r)(N-m)}}{N^2} & \frac{4 \sqrt{k-r(m-r)(N-2m)}}{N^2} & \frac{8(N-m)(m-r)}{N^2} - 1 & \frac{4 \sqrt{(m-r)(N-k-m+r)(N-2m)}}{N^2} \\
\frac{4 \sqrt{r(N-k-m+r)(N-2m)}}{N^2} & \frac{8 \sqrt{(m-r)(N-k-m+r)(N-2m)}}{N^2} & \frac{4 \sqrt{(m-r)(N-k-m+r)(N-2m)}}{N^2} & 1 - \frac{8m(N-k-m+r)}{N^2}
\end{pmatrix}$$

(41)
where the compound operator $Q$ is a rotation in the 4-dimensional space spanned by Eq. 30 - Eq. 33. Finding the amplitude of the state $|l_1\rangle$ after $t$ iterations of the operator $Q$ can be done by the diagonalization method as in [BBB+99]. Let the amplitudes of the states Eq. 30 - Eq. 33 be $u(t) = (\alpha(t), \beta(t), \gamma(t), \delta(t))$ respectively, then $u(t + 1) = Q \cdot u(t)$. Diagonalizing $Q$ yields a solution for $u(t)$. Let $Q = VDV^{-1}$, where $V$ is a matrix whose columns are the eigenvectors of $Q$ and $D$ is a diagonal matrix whose diagonal is the eigenvalues of $Q$. Since, $Q$ is a rotation, then

$$D = \begin{pmatrix} \lambda_{1+} & 0 & 0 & 0 \\ 0 & \lambda_{1-} & 0 & 0 \\ 0 & 0 & \lambda_{2+} & 0 \\ 0 & 0 & 0 & \lambda_{2-} \end{pmatrix}$$

(42)

where $\lambda_{1\pm} = e^{\pm iw_1}$, $\lambda_{2\pm} = e^{\pm iw_2}$, $w_1 = \arccos\left(1 - \frac{8r(N-m)}{N^2}\right)$, and $w_2 = \arccos\left(\frac{8m(k-r)}{N^2} - 1\right)$.

Applying $Q$ for $t$ times yields:

$$Q^t u(0) = VDV^{-1}u(0) = V \begin{pmatrix} e^{iw_{1t}} & 0 & 0 & 0 \\ 0 & e^{-iw_{2t}} & 0 & 0 \\ 0 & 0 & e^{iw_{2t}} & 0 \\ 0 & 0 & 0 & e^{-iw_{1t}} \end{pmatrix} V^{-1}u(0)$$

(43)

Therefore, after applying $Q$ for $t$ times the amplitude of $|l_1\rangle$ becomes $\sin(w_1 t + \phi)$, where $\phi$ is the initial angle between the initial state $H|0\rangle^{\otimes n}$ and $|l_1\rangle$:

$$\phi = \arctan\sqrt{\frac{r}{N-r}}$$

(44)

Consequently, the probability of measuring $|l_1\rangle$ after $t$ applications of the operator $Q$ is

$$Pr(t) = Pr\left(\text{Measure}\left(Q^t H |0\rangle\right) \in K \cap M\right) = \sin^2(w_1 t + \phi) = \frac{1}{2} - \frac{1}{2}\cos(2w_1 t + 2\phi)$$

Let

$$T = \arg\max_t Pr(t)$$

(45)

then

$$\cos(2wT + 2\phi) = -1$$

(46)
hence

\[ 2w_1 T + 2\phi = \pi \]  \hspace{1cm} (47)

yielding

\[ T = \left[ \frac{\pi}{2} - \arctan \left( \frac{r}{\sqrt{N-r}} \right) \right] \frac{\arccos \left( 1 - \frac{8r(N-m)}{N^2} \right)}{2} \]  \hspace{1cm} (48)

proving Eq. 27.

Similarly to Sec. II we denote

\[ t_{opt} = \frac{\pi}{2} - \arctan \left( \frac{r}{\sqrt{N-r}} \right) \frac{\arccos \left( 1 - \frac{8r(N-m)}{N^2} \right)}{2} \]  \hspace{1cm} (49)

which is not an integer. It holds that \( Pr(t_{opt}) = 1 \). Define \( \Delta T = t_{opt} - T < 1 \). We devise a lower bound on the success probability of measuring a marked state both by \( K \) and \( M \) after \( T \) iterations, \( Pr(T) \), as in Eq. 22:

\[
Pr(T) = \sin^2 (w_1 T + \Phi) = \sin^2 (w_1(t_{opt} - \Delta T) + \Phi) \\
= \sin^2 (w_1 t_{opt} + \Phi - w_1 \Delta T) \\
= [\sin (w_1 t_{opt} + \Phi) \cos (w_1 \Delta T) - \cos (w_1 t_{opt} + \Phi) \sin (w_1 \Delta T)]^2 \\
= \cos^2 (w_1 \Delta T) \\
> \cos^2 (w_1) = \left( 1 - \frac{8r(N-m)}{N^2} \right)^2
\]

proving Eq. 28.

When \( 1 \leq r \ll N \) the right hand side of Eq. 50 approaches 1, proving Eq. 29 and proving the theorem.

An important conclusion concerning the time complexity of the algorithm when \( r \ll N \) and \( |M| < N/2 \). In such case, the angle of the rotation can be approximated by the second order Taylor series approximation:

\[ w_1 \approx \frac{4\sqrt{r}\sqrt{N-m}}{N} \]  \hspace{1cm} (51)

and the number of iterations can be approximated by

\[
T \approx \left[ \frac{\pi/2 - \sqrt{\frac{r}{N-r}}}{4\sqrt{r}\sqrt{N-m}} \right] = \left[ \frac{\pi}{8\sqrt{r}\sqrt{N-m}} - \frac{N}{4\sqrt{N-r}\sqrt{N-m}} \right]
\]  \hspace{1cm} (52)

and the second term in the right hand side of the equation can then be bounded by

\[
\frac{1}{4} = \frac{N}{4\sqrt{N}\sqrt{N}} \leq \frac{N}{4\sqrt{N-r}\sqrt{N-m}} \leq \frac{N}{4\sqrt{N}\sqrt{N/2}} = \frac{1}{2\sqrt{2}}
\]  \hspace{1cm} (53)
which yields

\[ T \approx \left\lfloor \frac{\pi}{8} \frac{N}{\sqrt{r} \sqrt{N - m}} \right\rfloor = O \left( \sqrt{\frac{N}{r}} \right) \] (54)

The following observations can be directly derived from Trm. 1 and they sum up the most important properties of Alg. 1 both in the case of a single state intersection and multiple states intersection. They address the probability of measuring a member of the intersection subset in the cases of a single marked state and multiple marked states, respectively.

**Observation 1** Let \( I_K \) and \( I_M \) be phase oracles that mark the states of subsets \( K \) and \( M \) respectively, where \(|M|, |K| < N/2\), \( Q = (HI_0HI_MHI_0HI_K) \), and \( T = \left\lfloor \frac{\pi}{8} \frac{N}{\sqrt{N - |M|}} \right\rfloor \), and let \( \{s\} = K \cap M \), then

\[ Pr(\text{Observe}(Q^T H |0\rangle^\otimes n) = s) > \left( 1 - \frac{4}{N} \right)^2 \xrightarrow{N \gg 1} 1 \] (55)

More generally,

**Observation 2** Let \( I_K \) and \( I_M \) be phase oracles that mark the states of subsets \( K \) and \( M \) respectively, where \(|M|, |K| < N/2\), \( Q = (HI_0HI_MHI_0HI_K) \), and \( T = \left\lfloor \frac{\pi}{8} \frac{N}{\sqrt{|K \cap M| \sqrt{N - |M|}}} \right\rfloor \), then

\[ \forall s \in K \cap M : Pr(\text{Observe}(Q^T H |0\rangle^\otimes n) = s) > \frac{1}{|K \cap M|} \left( 1 - \frac{4}{N} \right)^2 \xrightarrow{N \gg 1} \frac{1}{|K \cap M|} \] (56)

V. QUANTUM ASSOCIATIVE MEMORY

In this section we introduce our associative memory model and the operations possible on it, storing, completing, and correcting a pattern. We first note that our goal is to store classical patterns that are binary strings and not quantum states.

In the model we present, the concept of memory is a quantum operator rather than a state of superposition of the memory patterns. The memory operator is simply a phase flip operator that flips the phase of the memory patterns, which is also called "marking" states. This allows the initiation of our algorithm for completion and correction queries to be independent of the memory set. The input of our query algorithms is simply an \( n \)-qubit register that contains the superposition of all basis states. This is acquired by a Hadamard operation on an \( n \)-qubit register set to zeros, \( H^\otimes n |0\rangle^\otimes n \).
In the rest of this section we describe the algorithms for the various operations possible on the memory. Storing new patterns, completing a pattern with missing bits, and correcting a pattern with faulty bits.

A. Storing New Patterns

For an empty memory set the operator is simply the identity operator, \( I_M = I \). Given a memory set \( M \) of size \( m \) stored in a memory operator \( I_M \), storing a new pattern \( x^{m+1} \) is creating the operator \( I_{M'} \) that stores the set \( M' = M \cup \{x^{m+1}\} \). The memory operator \( I_M \) flips the phase of the \( m \) stored patterns and does not affect any other pattern. Creating the updated memory operator that stores the set \( M \) can be expressed by the following operation:

\[
I_{M'} = I_M - 2|x^{m+1}\rangle\langle x^{m+1}|
\]

This operation can be performed using only the given operator \( I_M \) and the new pattern to be stored, and not requiring the storage operation to be applied to the whole set \( M \) from scratch. Generally, we are able to store up to \( N \) memory patterns using the operator \( I_M \), however, as we show in the next sections, there is a limitation on the size of the memory that depends on the number of mistakes we are interested in correcting in a noisy input pattern. Furthermore, we show that the quantum search amplifies the desired memories only when the number of patterns is less than \( 2^{n-1} \). This will be discussed in later sections.

Additionally, one can apply a deletion operation on the memory, which removes a stored pattern via the same phase flipping of the desired pattern. Let \( x^j \) be a memory pattern, then the operator \( I_{M'} = I_M + 2|x^j\rangle\langle x^j| \) stores the memory set \( M' = M \setminus \{x^j\} \).

B. Pattern Completion

Let \( I_M \) be a phase oracle of a memory set \( M \) of size \( m \) and let \( x' \) be a version of a memory pattern \( x \in M \) with \( d \) missing bits. We are required to output the pattern \( x \) based on \( I_M \) and \( x' \). The partial pattern is given as a string of binary values 0 and 1 and some unknown bits marked with ‘?’ . Denote the set of possible completions of the partial pattern by \( K \) and its size by \( k \). The completion problem then can be reduced to the problem of retrieving a member \( x \) in the intersection between two subsets \( K \) and \( M \), \( x \in K \cap M \). For example,
let \( M = \{0101010, 0110100, 1001001, 1111000, 0010001, 1101100, 1010101, 0000111, 0010010\} \) be a 7-bit memory, ")0110?0?" a partial pattern with 2 missing bits, and the completion set is \( K = \{0110000, 0110001, 0110100, 0110101\} \). Pattern completion is simply the computation of the intersection between \( K \) and \( M \), which is the memory pattern 0110100.

Pattern completion can use either one of the algorithms presented in Sec. III. It can be solved using the intersection oracle, or alternatively by the quantum intersection algorithm. For this purpose we need to create the completion oracle \( f_K \) or the completion operator \( I_K \) from the set \( K \). Both can be accomplished using \( k = 2^d \) operations, where \( d \) is the number of missing bits. The algorithm for pattern completion through the quantum intersection algorithm is given in the following:

**Algorithm 2: Quantum Pattern Completion**

*Given:*

- A memory operator \( I_M \)
- A pattern \( x' \in \{0,1\}^n \), which is a partial version of some memory pattern with up to \( d \) missing bits

1. Create a completion query \( K \) from \( x' \).
2. Create the completion oracle \( I_K \).
3. Apply Alg. 1 with \( I_M \) and \( I_K \)

A generalization for the case of unknown number of possible completions is straightforward using quantum search for multiple marked states [BBHT96] or quantum counting [BHT98].

**C. Pattern Correction**

Let \( I_M \) be a phase oracle of a memory set \( M \) of size \( m \) and let \( x' \) be a version of a memory pattern \( x \in M \) with up to \( d \) faulty bits. We are required to output the pattern \( x \) based on \( I_M \) and \( x' \). Denote the set of possible corrections of the faulty pattern by \( K \) and size by \( k \). The set \( K \) consists of all pattern in Hamming radius up to \( d \) from \( x' \). The correction problem then can be reduced to the problem of retrieving a member \( x \) in the intersection between two subsets \( K \) and \( M \), \( x \in K \cap M \). For example, let \( M = \{0101010, 0110100, 1001001, 1111000, 1101100, 1010101, 0000111, 0010010\} \) be a 7-bit
memory, "0110001" a faulty pattern with 2 faulty bits (The first and the third from the right). The correction set $K$ consists of all patterns that are in Hamming distance 2 from $x'$. Pattern correction is then the computation of the intersection between $K$ and $M$, which is the memory pattern 0110100.

Consequently, pattern correction can also be solved using the quantum intersection algorithm. However, we need to create the correction subset $K$ and the correction operator $I_K$. This can be done using

$$k = 1 + \sum_{i=0}^{d} \binom{n}{i}(2^d - 1)$$

operations, where $d$ is the number of faulty bits. The algorithm for pattern correction through the quantum intersection algorithm is given in the following:

**Algorithm 3 Quantum Pattern Correction**

*Given:*

- A memory operator $I_M$
- A pattern $x' \in \{0,1\}^n$, which is a faulty version of some memory pattern with up to $d$ faulty bits

1. Create a correction query $K$ from $x'$.
2. Create the correction oracle $I_K$.
3. Apply Alg. 1 with $I_M$ and $I_K$

A generalization for the case of unknown number of possible corrections is straightforward using quantum search for multiple marked states [BBHT96] or quantum counting [BHT98].

Alg. 3 finds a memory pattern that is in Hamming distance up to $d$ from $x'$. If the memory is within the correction capacity bounds, Alg. 3 finds the correct pattern $x$ with high probability (This will be proved in Sec. VI). However, if we are interested in ensuring that we find the closest memory pattern to $x'$, with no dependency on the capacity bound, then we can apply Alg. 3 for $i = 0$ bits and increase it up to $i = d$ bits. In that case we ensure that we find a pattern $x$, such that

$$\{ x : (x \in M) \land (\forall x'' \in M : \text{dist}(x'',x) \geq \text{dist}(x',x)) \}$$

where $\text{dist}(\cdot,\cdot)$ is the Hamming distance. The correction algorithm will have to be further changed, since we cannot assure that for each $i$ there exists a memory correction. Each
iteration should include a verification step to verify that the output is indeed a memory pattern. Furthermore, the first iteration, where \( d = 0 \), could then be a regular quantum search with no need for intersection.

VI. ANALYSIS OF THE QUANTUM ASSOCIATIVE MEMORY

In this section we present a full analysis of our associative memory algorithms. We analyze the time complexity and memory capacity. We show that the time complexity of memory operations is sub-exponential in the number of bits. Then we show that the number of memory patterns that can be stored with correction and completion abilities is exponential in the number of bits. These results are due to the properties of the quantum search and the quantum intersection algorithms.

A. Time Complexity Analysis

The maintenance of the memory operator \( I_M \) is simple. We are only required to update it when a new pattern is presented to the model, or alternatively an existing pattern is deleted. The storing, and hence, the deletion operation, can be performed in \( O(1) \) operations. It only requires the flipping of the phase for a single basis state. In general, building a memory of size \( m \) requires \( O(m) \) operations.

The time complexity of pattern completion consists of two phases. The creation of the completion query and the quantum intersection. The first depends on the size of the completion query and can therefore be accomplished in

\[
O(k) = O(2^d)
\]

and the second, according to Trm. 1, is performed in

\[
T \approx \left| \frac{\pi}{8} \frac{N}{\sqrt{|K \cap M|} \sqrt{N - |M|}} \right| = O \left( \sqrt{\frac{N}{|K \cap M|}} \right)
\]

operations. Therefore, the time complexity of pattern completion using Alg. 2 is

\[
O \left( 2^d + \sqrt{\frac{N}{|K \cap M|}} \right) = O \left( \sqrt{\frac{N}{|K \cap M|}} \right)
\]

which is sub-exponential and similar to the complexity of the quantum search under the assumption that \( d \ll n \). It has been shown in [BBHT96] that this analysis is applicable
also to the case where the number of marked states, which is the size of the intersection subset, is unknown. We note that as in quantum search, the time complexity is smaller when the size of the intersection between possible completions and memory is larger. The worst case is given when the intersection includes one pattern, in which case the time complexity of retrieving the pattern is \( O\left(\sqrt{N}\right) \).

Similarly, pattern correction for a signal to noise ratio \( d/n \) consists of the creation of the correction query and the quantum intersection. The creation of the correction query can be accomplished in

\[
O(k) = O\left(1 + \sum_{i=0}^{d} \binom{n}{i} (2^i - 1)\right) = O\left(n^d\right)
\]

Therefore, the time complexity of pattern correction using Alg. 3 is

\[
O\left(n^d + \sqrt{\frac{N}{|K \cap M|}}\right) = O\left(\sqrt{\frac{N}{|K \cap M|}}\right)
\]

which is also sub-exponential in the number of bits, assuming that \( d << n \).

B. Capacity Analysis

We divide our analysis to three different capacity measures. The first is the equilibrium capacity \( M_{Eq} \), which is the maximal memory size that ensures that all memory patterns are equilibrium points of the model. An Equilibrium point is a pattern that when presented to the model as input it is revealed also as output with high probability. The second is the pattern completion capacity \( M_{Com} \), which is the maximal memory size that enables the completion of any partial pattern with up to \( d \) missing bits with high probability. The third is the pattern correction capacity \( M_{Cor} \), which is the maximal memory size that enables the correction of any faulty pattern with up to \( d \) faulty bits with high probability.

The Equilibrium capacity is a special case of completion or correction with neither missing nor faulty bits. Therefore, it should be equal to the completion and correction capacities for \( d = 0 \).

The completion and the correction capacities that we find give another important measure. They give an upper bound on the number of missing or faulty bits a memory of size \( m \) can complete or correct without ambiguity. Let the completion or correction capacity be \( M = g_N (d) \), where \( d \) is the number of missing or faulty bits, then, a memory of size \( m \) can complete or correct up to \( d = g_{N-1}^{-1} (m) \) bits without ambiguity.
1. Equilibrium Capacity

The following theorem concerns the equilibrium capacity and states that any number of patterns can be stored and retrieved. The number of $n$-bit patterns is $N = 2^n$, therefore, $M_{Eq} = N$.

**Theorem 2** An $n$-bit associative memory $M$ of any size ($m \leq N$) can retrieve any memory pattern with probability close to 1; e.g.

\[
\forall x \in M : \ Pr(Q^T_x H |0\rangle) = x \rightarrow 1 \quad (65)
\]

where $Q_x = -(HI_0HI_x)$ and $T = \left\lfloor \frac{\pi}{4} \sqrt{N} \right\rfloor$

**Proof of Theorem 2** This is a direct result of Grover’s algorithm [Gro] and the results obtained by [BBHT96] concerning the ability to find any member of the $N$ elements database with probability close to 1. $\square$

2. Completion Capacity

Given a pattern $x'$, which is a partial copy of some memory pattern $x_c$ with $d$ missing bits, we seek the maximal memory size, for which the pattern can be completed with high probability from a random uniformly distributed memory set.

The completion capacity is bounded from above by two different bounds. The first is a result of Grover’s quantum search algorithm limitations and the second is a result of the probability of correct completion.

**A Bound on Memory Size due to Grover’s Quantum Search Limitations**

Grover’s operator flips the marked states around the zero amplitude (negating their amplitudes) then flips all amplitudes around the average of all amplitudes [BBB+99]. Amplification of the desired amplitudes occurs only when the average of all amplitudes is closer to the amplitudes of the non-marked states than to the marked states. This imposes an upper bound on the memory size that is given in the following theorem:

**Theorem 3** Grover’s operator $Q_M = -(HI_0HI_M)$ does not amplify the memory states $M$ of size $m$ if

\[
m \geq \frac{N}{2} \quad (66)
\]
Proof of Theorem 3 Observe the first iteration of the quantum search algorithm on a number of marked states. The initial amplitude of all basis states in $H^\otimes n |0\rangle^\otimes n$ is $1/\sqrt{N}$. Flipping the phase of $m$ marked states by $I_M$ to $-1/\sqrt{N}$ yields an average amplitude of

$$(N - m) * \left(1/\sqrt{N}\right) - m * \left(1/\sqrt{N}\right) = \frac{N - 2m}{\sqrt{N}}$$ \hspace{1cm} (67)$$

Flipping the phases of all basis states around this average by $HI_0H$ yields two values of amplitudes. The amplitudes of marked and un-marked states become

$$2 \left(\frac{N - 2m}{\sqrt{N}}\right) \pm \frac{1}{\sqrt{N}} = \left(\frac{2N - 4m \pm 1}{\sqrt{N}}\right)$$ \hspace{1cm} (68)$$

where $\pm$ correspond to marked and un-marked states respectively.

A necessary condition for the amplification of marked states is that the absolute value of their amplitudes after the algorithm’s iteration, is higher than the absolute value of the amplitude of unmarked states. The condition is satisfied if and only if the two equations given in Eq. 68 satisfy

$$\left|\left(\frac{2N - 4m + 1}{\sqrt{N}}\right)\right| > \left|\left(\frac{2N - 4m - 1}{\sqrt{N}}\right)\right|$$ \hspace{1cm} (69)$$

which holds true if and only if $m < N/2$. Therefore, if $m \geq N/2$ the amplitudes of the marked states will not increase. \hfill \Box

Trm. 3 gives the following upper bound on the completion capacity:

$$M_{Com} < \frac{N}{2}$$ \hspace{1cm} (70)$$

However, this is a very loose bound and the completion success probability will give a tighter bound.

A Bound on Memory Size due to Pattern Completion

The bound on memory size that ensures a high probability of correct completion depends on the definition of pattern completion procedure. If one defines pattern completion as the process of outputting any of a number of possible memory patterns when given a partial input, then the capacity bound of our memory is the amplification bound given in Eq. 70. However, this is not always the case. Pattern completion capacity is usually defined as the maximal size of a random uniformly distributed memory set that, given a partial version $x'$
of a memory \( x_c \in M \) with \( d \) missing bits, outputs \( x_c \). In this case, the following theorem gives an upper bound on the capacity for pattern completion with high probability:

**Theorem 4** An \( n \)-bit associative memory with \( m \) random patterns can complete up to \( d \) missing bits on average when

\[
m < 2^{n-d}
\]

with probability higher than 75% as \( n \) grows infinitely.

**Proof of Theorem 4** Let \( M \) be a random uniformly distributed memory set of size \( m = v2^{n-d} \), where \( 0 < v < 1 \). Let \( x' \) be a partial pattern of \( x_c \in M \) with \( d \) missing bits. \( x' \) induces a set

\[
K = \{ x | x \text{ is a completion of } x' \}
\]

where \(|K| = 2^d\). Let \( Z_i \) be random indicator variables representing the existence of the \( i \)th element of \( M \) in \( K \). Denote \( p = Pr(Z_i = 1) = \frac{2^d}{2^n} = 2^{d-n} \), then

\[
m = \frac{v}{p}
\]

Define \( S \) as the sum of the variables \( S = \sum_{i=1}^{m} Z_i \). If there is only one memory completion then it’s \( x_c \) and if there is two then \( x_c \) is one of two possible completions and so on. Therefore, the probability of successfully outputting \( x_c \) from the partial pattern \( x' \) is the sum of the conditional probabilities that there are \( i \) memory completions divided by \( i \):

\[
Pr(\text{Obs}(Q^T H | 0)) = \sum_{i=1}^{m} \frac{Pr(S = i | S \geq 1)}{i}
\]

\[
= \sum_{i=1}^{m} \frac{Pr(S = i)}{i Pr(S \geq 1)}
\]

\[
= \sum_{i=1}^{m} \frac{\binom{m}{i} p^i (1-p)^{m-i}}{1 - (1-p)^m}
\]

**Since**

\[
\binom{m}{i} \leq \frac{m^i}{i!}
\]

it hold that

\[
\sum_{i=1}^{m} \frac{\binom{m}{i} p^i (1-p)^{m-i}}{1 - (1-p)^m} \leq Pr(\text{Obs}(Q^T H | 0)) \leq \frac{\binom{m}{i} p^i (1-p)^{m-i}}{1 - (1-p)^m}
\]

27
Substituting Eq. 73 in Eq. 78 we have

\[
\sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right) (1-p)^{v/p-i} \leq Pr\left( \text{Obs}\left( Q^T H \mid 0 \right) \right) \leq \sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right) (1-p)^{v/p-i} \tag{79}
\]

and since \( p << 1 \)

\[
\sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right)^{-v} \frac{1}{1-e^{-v}} \leq Pr\left( \text{Obs}\left( Q^T H \mid 0 \right) \right) \leq \sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right) e^{-v} \frac{1}{1-e^{-v}} \tag{80}
\]

and since \( (1-p)^i \approx 1 \)

\[
\sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right) e^{-v} \frac{1}{1-e^{-v}} \leq Pr\left( \text{Obs}\left( Q^T H \mid 0 \right) \right) \leq \sum_{i=1}^{m} \left( \frac{v_i}{\pi} \right) e^{-v} \tag{81}
\]

Substituting \( \sum_{i=1}^{m} \frac{v_i}{\pi} \approx e^v - 1 \)

\[
\left( \sum_{i=1}^{m} \frac{v_i}{\pi} \right) e^{-v} \frac{1}{1-e^{-v}} \leq Pr\left( \text{Obs}\left( Q^T H \mid 0 \right) \right) \leq 1 \tag{82}
\]

and the lower bound is 75%, hence

\[
75\% \leq Pr\left( \text{Obs}\left( Q^T H \mid 0 \right) \right) \leq 1 \tag{83}
\]

Trm. 4 yields that \( M_{\text{Com}}(d) = 2^{n-d} \), which agrees with the result of Trm. 2 concerning the equilibrium capacity, since \( M_{\text{Com}}(0) = 2^{n-0} = 2^{n} = N = M_{\text{Eq}} \).

3. Correction Capacity

Similarly to the completion capacity analysis, we state the following theorem:

**Theorem 5** Any \( n \)-bit associative memory with \( m \) random patterns can correct up to \( d \) faulty bits on average when

\[
m < 2^{n-d} \binom{n}{d} \tag{84}
\]

with probability higher than 75% as \( n \) grows infinitely.

**Proof of Theorem 5** Let \( M \) be a random uniformly distributed memory set of size

\[
m = \nu 2^{n-d} \binom{n}{d} \tag{85}
\]
where $0 < v < 1$. Let $x'$ be a pattern $x_c \in M$ with $d$ faulty bits. $x'$ induces a set
\[
D = \{ x | \text{dist}(x, x') \leq d \}
\]  
where $|D| = \binom{n}{d}2^d$. Let $Z_i$ be random indicator variables representing the existence of the $i$th element of $M$ in $D$. Denote $p = Pr(Z_i = 1) = \binom{n}{d}2^{d-n}$, then
\[
m = \frac{v}{p}
\]  
Define $S$ as the sum of the variables $S = \sum_{i=1}^{m} Z_i$. The probability of successfully retrieving $x_c$ from the pattern $x'$ is then
\[
Pr(\text{Obs}(Q^TH | 0)) = \sum_{i=1}^{m} Pr(S = i / S \geq 1)
\]  
which according to Eq. 78-Eq. 82 also satisfies
\[
75\% \leq Pr(\text{Obs}(Q^TH | 0)) \leq 1
\]
Trm. 5 yields that $M_{Cor}(d) = \binom{n}{d}2^{n-d}$, which also agrees with the result of Trm. 2 concerning the equilibrium capacity, since $M_{Cor}(0) = \binom{n}{0}2^{n-0} = 2^n = N = M_{Eq}$.

For example, let $M$ be a memory set over $\{0, 1\}^{100}$, then as long as $|M| < 2^{80}$ we can complete to to $d = 100 - \log |M| = 20$ bits and correct up to $d = 13$ bits. The latter result is due to the fact that $2^d(\binom{100}{d}) < 2^{100}/|M|$ if $d < 14$.

4. Increasing Memory Size Beyond the Capacity Bounds

The various capacities presented above are exponential in $n$ under the assumption $d << n$. However, an increase of $m$ beyond the capacity bound results in a decay of the correct completion probability as depicted in Fig. 7.

It can be seen in the graph in Fig. 7 that it is more likely to find the correct completion than not as long as $v < 2$. The pattern completion and the pattern correction bounds are not obligatory and can be treated with no additional cost. As long as the number of possible solutions are small, one can run the algorithm a constant number of times and check each answer or, alternatively, output them as a superposition of solutions. The latter case suggests to skip the last step of the quantum intersection algorithm (Alg. 1) and not observe the system. The quantum state that is kept is a superposition of all possible solutions. This,
FIG. 7: Pattern completion or correction probability vs. \( v \). For \( 0 < v < 1 \), the probability is above 75%. For \( v > 1 \) the probability drops below 75%. The probability reaches 50% only for \( v > 2 \).

of course, does not exist in most classical memory models were spurious memories arise in such cases and the output is usually not a memorized pattern, but rather some spurious combination of multiple memory patterns.

C. Storing the Memory Operator

Another aspect of associative memory we are interested in is the number of bits required to store the whole memory system. How much information about the system do we need to store in order to reuse it? In recurrent networks, such as Hopfield network we need to store all weights and thresholds of the system, this would require \( n^2 + n \) numbers, where each number should be represented by a multiple number of bits. Denote the number of bits required to store a number representing a weight or a threshold in the network by \( p \), then the number of required bits is \( (n^2 + n)p \). However, this amount depends on \( m \), since \( m \leq O(n/\log n) \). The dependency on \( m \) is higher than quadratic. In feed-forward networks, such as hamming network, we need to store \( m(n + 1) \) weights and thresholds for the input layer that require \( m(n + 1)p \) bits, and another \( mn \) bits for the retrieval phase, which sums up to \( m(np + p + n) \) bits.

On the other hand, in the quantum associative memory model, we only need to store the
information needed for the memory operator $I_M$, which can be stored naively in $mn$ bits, marking the basis states that need flipping.

VII. COMPARISON TO PREVIOUS ATTEMPTS

Previous attempts of applying quantum computation algorithms for associative memories were made [VM00, ENV00, HYV00]. Many other works were based on these attempts. An algorithm recently proposed by [AMSM08] is similar to our algorithm and was independently devised based on [VM00]. We analyze the two main algorithms from [VM00] and [AMSM08] and show their shortcomings and differences from our algorithm. But first, the algorithms are presented in Alg. 4 and Alg. 5 respectively.

Algorithm 4 The algorithm proposed by Ventura et al. [VM00]

Given:

Phase oracles $I_M$ and $I_K$
1. Denote $Q_M = -H_i_0 H_I_M$ and $Q_K = -H_i_0 H_I_K$
2. Let $|Ψ⟩ = \frac{1}{m} \sum_{i=1}^{m} |i⟩$.
3. Apply $Q_M Q_K$ on $|Ψ⟩$.
4. Apply $Q_K$ on $|Ψ⟩$ for $T = \frac{\pi/4\sqrt{N/|K \cap M|}}{2}$ times.
5. Observe $|Ψ⟩$.

Algorithm 5 The algorithm proposed by Arima et al. [AMSM08]

Given:

Phase oracles $I_M$ and $I_K$
1. Denote $Q_M = -H_i_0 H_I_M$ and $Q_K = -H_i_0 H_I_K$
2. Let $|Ψ⟩ = \frac{1}{m} \sum_{i=1}^{m} |i⟩$.
3. Apply $Q_M Q_K$ on $|Ψ⟩$ for $T$ times. ($T$ was not found in [AMSM08])
4. Observe $|Ψ⟩$.

Ventura’s algorithm (Alg. 4) can find only a single marked state with high probability when the memory size $m$ is close to $\frac{N}{4} - 2$ as depicted by the solid line in Fig. 8. The
Amplitude of this state drops in half when there are two marked state and only one of them is a memory pattern as depicted by the dashed line in Fig. 8, and so on.

FIG. 8: The memory size vs. the success probability in Ventura’s algorithm, Alg. 4. Optimal results are achieved only when the memory size is close to \( \frac{N}{4} \).

Arima’s algorithm gives satisfying results only when the memory size exceeds \( \frac{N}{4} \), which is exponential in the number of qubits, and therefore, is not helpful for associative memory with pattern completion and correction abilities. The success probability of Arima’s algorithm (Alg. 5) vs. the memory size is depicted in Fig. 9. The algorithm is presented only for one marked state with no completion and correction abilities. The time complexity and stopping criteria are not found and no proofs were presented.

FIG. 9: The memory size vs. the success probability in Arima’s algorithm, Alg. 5. Satisfying results are achieved only when the memory size exceeds \( \frac{N}{4} \).
Our algorithm, on the other hand achieves high success probability up to memory size $\frac{N}{4}$ as depicted in Fig. 10, which satisfies any needs of associative memory.

![Graph showing memory size vs. success probability](image)

**FIG. 10:** The memory size vs. the success probability in algorithms Alg. 2 and Alg. 3. High success probability is achieved for all memory sizes up to $\frac{N}{4}$.

Furthermore, both algorithms (Alg. 4 and Alg. 5) need to initialize the system to a superposition of the memory states:

$$|\Psi\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |i\rangle$$

(90)

which is an obvious shortcoming for two reasons. The time complexity that of initialization that becomes exponential for large memory sizes and the need for the repetition of the initialization upon every application of the memory. The latter is very important since it adds an exponential factor to the query time, whether it’s completion or correction, and since it adds exponential time also to the a single query time when amplitude amplification is needed. Amplitude amplification ensures that we pick the correct pattern with probability 1 by performing the algorithm a multiple number of times.

Our algorithm, on the other hand, gives satisfactory results for any memory size up to $\frac{N}{4}$, which is large enough for any pattern completion and correction needs.

**VIII. NUMERICAL EXAMPLES AND SIMULATIONS**

We start this section by presenting an example of an associative memory in 7 qubits. We have randomly chosen a set of 20 patterns $M$ out of possible 128 to be stored in memory.
We also chose two partial patterns each with 3 missing qubits yielding two completion sets $K_1$ and $K_2$ of 8 possible completions each. We chose $K_1$ and $K_2$ such that they have one and two completions in memory respectively. The memory patterns, the completion set and the memory completion result are depicted in Fig. 11 and Fig. 12 for both completions $K_1$ and $K_2$ respectively.

FIG. 11: A set of memory patterns $M$, a set of completions $K_1$, and the memory completion result in amplitudes.

FIG. 12: A set of memory patterns $M$, a set of completions $K_2$, and the memory completion result in amplitudes.
As can be seen, applying our algorithm on both completion sets amplified the states that are possible completions in memory. The amplitudes of the desired states reached up to $89.86\%$ in the first case and $90.54\%$ in the second case. Therefore, the probability of measuring the correct completion in the first case is $80.75\%$ and the probability of measuring one of the two correct completions in the second case is $81.97\%$.

It is clear from the figures that the results yield high amplitudes only for the states in the intersection between the completion group $K_1$ or $K_2$ and the memory group $M$.

![FIG. 13: Simulation of a series of iterations of the completion algorithm. The graph shows the different behavior of the different subgroups of basis states. The correct memory completions are amplified while the amplitudes of other subgroups are kept close to zero. $K$ is the completion set, $M$ is the memory set and $N$ is the set of all states.](image)

Another simulation was carried out on a 10 qubits associative memory with $2^7$ memory patterns and a completion query with 3 missing bits. The Amplitudes of the different subgroups of the basis states are depicted in Fig. 13 for a series of iterations with the completion operator $Q$ from Eq. 26. It clearly shows the amplification of states in the intersection group while keeping the other subgroups close to zero.

In addition, we have also tested our algorithms with a larger number of qubits to assure that the success rate of retrieval grows asymptotically to 1 as the number of qubits grows. For instance, we have tested a 30 qubit system, with $2^{25}$ memory patterns and a completion query that has 8 missing bits. We tested different completions of 8 missing bits such that...
TABLE I: Simulation properties of the four simulations depicted in Fig. 14. The varying size is
the z-axis and the rest are constants in each simulation.

| Graph       | $|N|$ | $|M|$        | $|K|$   | $|K \cap M|$ |
|-------------|-----|-------------|--------|-------------|
| Solid       | constant | varying   | constant | constant   |
|             | $2^{30}$ | $2^3 - 2^{27}$ | $2^3$  | 1          |
| Dashed      | constant | constant   | varying | constant   |
|             | $2^{30}$ | $2^{25}$   | $2^3 - 2^{25}$ | 1          |
| Dotted      | constant | constant   | constant | varying    |
|             | $2^{30}$ | $2^{25}$   | $2^{25}$ | $1 - 2^{25}$ |
| Dash-dotted | varying | constant   | constant | constant   |
|             | $2^5 - 2^{30}$ | $2^3$     | $2^3$  | 1          |

the intersection set size varied from 1 pattern to 10 patterns. Our algorithm measured one
of the desired completions with probability 96.8%. Increasing the memory size to $2^{26}$ and
$2^{27}$, and thereby taking the capacity close to its limit, resulted with desired completion
probabilities of 93.5% and 86.7% respectively. Fig. 14 depicts the success rates of pattern
completion in a 30 qubit system. A brief explanation of the different graphs can be found
in Table I. The solid line depicts the success probability vs. the logarithm of the size of
memory with completion queries set to 8 missing bits and the number of possible memory
completions set to 1. The dashed line depicts the success probability vs. the logarithm
of the completion query size when the memory size is set to $2^{25}$ patterns and the number
of possible memory completions set to 1. The dotted line depicts the success probability
vs. the logarithm of the number of possible memory completions when both the memory
size and the completion query size are set to $2^{25}$. The dash-dotted line depicts the success
probability vs. the number of qubits in the system (growing from 5 to 30 qubits) when the
memory size, the completion query size, and the completion memories are small constants.

Fig. 14 shows clearly that the deterioration of the success probability vs. the memory size
or the completion query size is very slow. For instance, the deterioration vs. memory size
starts at $2^{26}$. Furthermore, the success probability increases when the number of possible
memory completions (the intersection) grows towards the sizes of the completion query and
the memory, which indicates that choosing a correct answer becomes very easy. Finally, the
FIG. 14: Success probability of measuring a desired memory completion vs. the log of the memory size (solid), completion query size (dashed), possible memory completions (dotted), and number of qubits (dash-dotted).

The figure also shows that as the number of qubits in the system grows the success probability becomes asymptotically 1, which indicates that also practically the our algorithm computes the intersection when $n \gg 1$.

IX. CONCLUSIONS

We have presented a quantum computational algorithm that computes the intersection between two subsets of $n$-bit strings. The algorithm is based on a modification of Grover’s quantum search. Using the intersection algorithm, we have presented a set of algorithms that implement a model of associative memory via quantum computation. We introduced the notion of memory as a quantum operator and, hence, avoided the dependency of the initial state of the system on the memory. We have shown that our algorithms have both speed and capacity advantages as they work in sub-exponential time and are able to store an exponential number of memory patterns. Storage, completion, and correction operations were presented. Bounds relating memory capacity to the maximal allowed signal to noise
ratio were found.


