String Matching Using Quantum Fourier Transform

Jeffrey A. Aborot*
Advanced Science and Technology Institute
Department of Science and Technology
ASTI Bldg., U.P. Techno Park, C.P. Garcia Ave.
Diliman, Quezon City
jep@asti.dost.gov.ph

Henry N. Adorna
Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines Diliman
Diliman, Quezon City
hnadorna@dcs.upd.edu.ph

ABSTRACT

Pattern matching has always been applicable to many areas of interest. Particular to the string data structure, this problem abstracts computational tasks found in areas such as computational biology, signal processing and network security. Due to its high applicability several approaches has already been devised for solving the problem using its various models. Main classical approaches developed are dynamic programming, automata-based algorithms, bit-parallelism techniques and filtering.

One particular approach to string matching problem is the semi-numerical approach of viewing pattern matching as the process of multiplying polynomials. A classical technique used for speeding up multiplication is the convolution method which uses Discrete Fourier Transform and its inverse. The Fast Fourier Transform algorithm can be used to speed up computation of the Discrete Fourier Transform by a large factor. In this paper we propose a quantum algorithm which applies the classical convolution method to pattern matching but uses Quantum Fourier Transform for computation of the Discrete Fourier Transform instead of the classical Fast Fourier Transform. Its complexity is analysed and compared to that of the current classical convolution method.

1. INTRODUCTION

In this section we briefly introduce the focus problem of this paper and present current solutions addressing the problem. We also present a summary of our proposed solution and its complexity in comparison to that of the current solutions.

*On-going graduate student under Algorithms and Complexity Laboratory, Department of Computer Science, University of the Philippines Diliman, jaaborot@up.edu.ph

1.1 Patterns on Strings

The string data structure is widely used in modelling computational problems in different areas of application. In music informatics, a sequence of notes and other musical symbols can be represented as a string of alphabet symbols which correspond to specific symbols in music notation. In bioinformatics, a DNA sequence can be represented as a string of alphabet symbols A, C, T and G which correspond to the oligonucleutides adenine, cytosine, thiamine and guanine. For an RNA sequence, the alphabet will be of size 21. Strings are also used to model contents of packets in network intrusion detection system in which the alphabet may contain alphanumeric and special characters.

Common to the aforementioned areas of application is the computational problem of finding patterns. In music informatics a common pattern in various musical compositions may represent a defining style or characterization unique to the samples. Such characterization may be the genre of music or specific author of musical pieces. Patterns may also represent key terms when retrieving musical compositions from a repository [5][3]. In bioinformatics, patterns may correspond to sections in a DNA or RNA sequence which may hold biological significance [1][7]. Searching for these patterns aids in gaining deeper understanding of biological functions of different sections of biological sequences. In network intrusion detection, patterns correspond to signatures of attacks maintained in a repository for reference. Packets of data coming into the protected network are scanned for detecting malicious contents and are treated according to defined rules [19][2][10]. The data in these computational problems of pattern matching can be modelled using string data structure. Another model used for this kind of computational problems is the tree data structure and its variants. In this paper we focus on the string data structure and on the String Matching model of the computational problem of pattern matching.

1.2 String Matching

The string matching model abstracts pattern matching problems as computational problems in which data is modelled simply as strings defined over a finite alphabet of symbols. There are also several variants of the string matching model which vary on accuracy of representing the actual computational problems. Among these are the exact and approximate string matching model.

1.2.1 Exact String Matching

The exact string matching model represents the computa-

tional problem of pattern matching in which an exact match of a sequence of symbols, called the *pattern*, is sought from a longer sequence of symbols called the *text*. Exact match means a perfect symbol to corresponding symbol match between the pattern and any same length subsequence of the text. An illustration of an exact match between a pattern and a subsequence of a text is shown in Figure . We can formally define the exact string matching problem as follows:

Exact String Matching Problem

Input:

- alphabet Σ where $|\Sigma| = \sigma$
- text T where $|T| = N$ and $T \in \Sigma^N$
- pattern P where $|P| = M$ and $P \in \Sigma^M$

Output: all indices i such that $T[i, i + M - 1] = P$ where $0 \leq i \leq N - M + 1$

The exact string matching model is a very simple representation of a pattern matching problem that it lacks accuracy in representing most of the real application problems. A much more accurate model for most of real application pattern matching problems is the approximate string matching model.

1.2.2 Approximate String Matching

Matching between a pattern and a subsequence of a text can be interpreted as the number of steps necessary to transform the subsequence into an exact copy of the pattern. The transformation steps may include substitution, deletion and insertion of symbols. In contrast to exact string matching model, a match between the pattern and any subsequence of the text in the approximate string matching model is defined using a so called *distance metric*. One specific distance metric is the *Hamming distance*. In the Hamming distance metric, the distance between the pattern and the subsequence is defined as the total number of substitutions in the subsequence necessary to transform it into an exact copy of the pattern. Given a defined maximum Hamming distance between the pattern and any same length subsequence of the text a match occurs when the required number of substitutions to transform a subsequence into an exact copy of the pattern is at most the defined Hamming distance. We denote the Hamming distance between a pattern P and a subsequence $T[i, i+M-1]$ of the text as $d(P, T[i, i+M-1])$.

We can then formally define the approximate string matching model of the pattern matching problem as follows:

Approximate String Matching Problem

Input:

- alphabet Σ where $|\Sigma| = \sigma$
- maximum Hamming distance D
- text T where $|T| = N$ and $T \in \Sigma^N$
- pattern P where $|P| = M$ and $P \in \Sigma^M$

Output: all indices i such that $d(P, T[i, i + M - 1]) \leq D$ where $0 \leq i \leq N - M + 1$

The approximate string matching model provides a much more accurate representation of much of the pattern matching problems in various applications since it allows errors or mutations in data. Mutations are very common in real application data such as biological sequences and musical compositions. Communication channels may also introduce noise into data being transferred in the context of digital signal processing. Also, it is easy to see that the exact model is just an instance of the approximate model in which the defined Hamming distance is 0.

1.3 Complexity of String Matching

In the exact string matching model a match between the pattern and a subsequence of the text is said to occur if their corresponding characters exactly match. Given the worst case that there does not exists a matching subsequence in the text the total number of comparison will be at most $M(N - M + 1)$. The upper bound complexity of the exact string matching model will thus be $O(MN)$.

On the other hand, in the approximate string matching model a match is said to occur it the number of mismatches between the sequence and the pattern is less or equal to a defined distance metric. In the worst case that there doest not exists a match between the pattern and the text the total number of comparison will be at most $M(N-M+1)$, which is the same with that of the exact model. Thus the upper bound complexity for the approximate model is also $O(MN)$.

1.4 Approaches to String Matching

Since the string matching model is widely applicable there has already been several developed approaches to solving it. These approaches can be divided into classical and unconventional approaches. In this paper we identify classical solutions as those which are intended to execute on our current computing machines which makes use of classical bits. Also, we identify unconventional solutions as those which are designed to execute on machines which make use of unconventional hardware for information storage and physical effects for computation.

Our proposed algorithm presented in this paper is an unconventional solution to the string matching problem since it make use of quantum bits and some quantum physical phenomena for computation.

1.4.1 Classical Solutions

Several classes of classical algorithms has been developed for the string matching problem. Some of these are dynamic programming algorithms, automata-based algorithms and filtering algorithms. Among the early dynamic programming algorithms are those by Sellers , Masek and Paterson and Ukkonen . These algorithms have the flexibility of being adaptable to different distance metrics *i.e. Hamming distance, edit distance, Episode distance, etc.* but are heavy on the time complexity requirement since dynamic programming algorithms make use of a matrix to represent the text and pattern. Out of this approach also came forth the shortest path problem reformulation on a graph built on the text and pattern.

Some early automata-based algorithms are those by Ukkonen [17], Wu and Manber, and Kurtz.

Filtering algorithms on the other hand are two-part algorithms such that they always include algorithms of another

class as sub-routine for the actual string matching. These algorithms perform initial computation on the solution space to trim it down in preparation for the string matching step. The initial time complexity allotted for filtering is compensated by a fast string matching algorithm. Examples of early filtering algorithms for string matching are those of Ukkonen and Tarhio, Baeza-Yates and Navarro and Navarro and Raffinot .

1.4.2 Convolution Algorithm

One particular approach to string matching is the use of a class of algorithms called *semi-numerical* algorithms as defined by D. Knuth [11]. Semi-numerical algorithms are those which perform matching of symbols by using arithmetic operations on the input data. One such algorithm is the *discrete convolution algorithm*. This was first proposed by Fischer and Paterson in [8]. In their paper they presented a general formula for linear product of vectors of symbols which results to a vector in which the components are interpreted as the components of the product of the two vectors. The product of the two vectors is their *convolution*. The convolution of two sequences of N complex numbers $S_1 = (\alpha_0, \ldots, \alpha_{N-1})$ and $S_2 = (\beta_0, \ldots, \beta_{N-1})$ is defined to be the sequence $S_3 = (\gamma_0, \ldots, \gamma_{N-1})$ given by

$$\gamma_k = \sum_{j=0}^{N-1} \alpha_j \beta_{k-j} \ for\ k = 0, \ldots, N-1$$

where subscripts are taken *(mod N)*. They showed that this general concept of linear product of vectors can be applied to the problems of polynomial multiplication and string matching. Specific to the string matching problem, given two strings T and P such that $|T| = N, |P| = M$ and $M < N$, string matching can be be performed by representing both strings as vectors of integers and taking their convolution. The components of the resulting vector correspond to the score of matching P with the subsequences of T. Computing the convolution directly will require $O(N^2)$ time steps since comparison of each component of the first vector to each component of the second vector will result to N^2 number of comparisons. One approach to performing the linear product multiplication of two vectors of size N and M is to first take the *Discrete Fourier Transform* (DFT) of both vectors using $N + M - 1$ samples and then performing a component-wise multiplication of the components of the transformed vectors. *Inverse* DFT (IDFT) is then applied to the the resulting vector which will result to a vector of size $N + M - 1$ with components corresponding to match score of comparing P with the subsequences of T. The only valid elements of the resulting vector are the middle $N - M + 1$ elements only and this is due to the required padding of both text. This method will require $O(N^2)$ time steps for applying DFT to both vectors, $O(N)$ for applying the component-wise multiplication step and another $O(N^2)$ for applying the IDFT step. Thus this method will require an $O(N^2)$ steps overall. Improvement can be introduced by using the *Fast Fourier Transform* (FFT) algorithm introduced by Cooley and Tukey in [4]. Using FFT to compute for DFT and IFFT to compute for IDFT, the first and third steps of the convolution algorithm will require only $O(N \log N)$ time steps. In total, the convolution algorithm will require only $O(N \log N)$ time steps which is an improvement over the direct method. Identifying a match between pattern P and subsequences of text T will translate to finding all components of output vector which are less or equal to the defined Hamming distance D in approximate string matching model and those which are equal to M in the exact string matching model. Searching for these components in the unstructured output vector will require $O(N)$ time steps. Thus, using the classical discrete convolution method for string matching will require $O(N \log N) + O(N)$ time complexity. The DFT-based method for computing the convolution of two vectors is outlined as follows:

Classical Discrete Convolution Algorithm

1. Apply DFT into input vector α.
$$\alpha' = DFT(\alpha)$$
2. Apply DFT into input vector β.
$$\beta' = DFT(\beta)$$
3. Perform component-wise multiplication between the two Fourier transformed vectors.
$$\begin{aligned}\delta &= \alpha' \cdot \beta' \\ &= DFT(\alpha) \cdot DFT(\beta)\end{aligned}$$
4. Apply IDFT into resulting vector δ.
$$\begin{aligned}\gamma &= IDFT(\delta) \\ &= IDFT(DFT(\alpha) \cdot DFT(\beta))\end{aligned}$$

The resulting vector γ is defined such that

$$\gamma_k = \sum_{j=0}^{N-1} \alpha_j \beta_{k-j} \ for\ k = 0, \ldots, N-1$$

Using DFT will require $O(N^2)$ time steps but using FFT to compute for the DFT will require $O(N \log N)$ time steps only. Thus, using this method as a sub-routine for a string matching algorithm will require $O(N \log N)$ time steps for computing the convolution of the vectors representing the input sequences and $O(N)$ time steps for determining the component i of the resulting vector with the least value. This least value represents the distance between the subsequence of text which starts at index i and the pattern. If the value of the i-th component is less than or equal to the input distance threshold then the subsequence of the text starting at index i matches the input pattern. A string matching algorithm which uses the discrete classical convolution algorithm as a sub-routine is outlined as follows:

Classical Convolution-based String Matching Algorithm

1. Perform convolution algorithm on input sequences T and P.
2. Find component values γ_k from the resulting vector in Step 1 such that
$$\gamma_k \leq D$$
where D is the input maximum Hamming distance and $0 \leq k \leq N - 1$.
3. Return each index k of satisfying component values.

1.5 Proposal: A Quantum Convolution Algorithm for String Matching

The presented convolution method is designed to execute on a classical device and thus is classical computation. In classical computation and specifically in the aforementioned classical algorithms, we represent data using the state of a two-level system called *classical bit* or *cbit*. In the conventional implementation which uses electric pulses, a cbit could either be in a ground state or an excited state and each can be denoted as 0 and 1 respectively. Given the input data represented using states of several cbits, computation is performed on the input data by applying a sequence of transformations on the state of each cbit. The resulting state of the cbits represent the output data. Computation on devices which use cbits are governed by classical physics. Thus the same goes for the previously discussed algorithms which are designed to execute on classical machines. There are some computational devices though which utilize a different kind of physical system for representing data. Data in these kind of devices are represented using the state of a two-level (or more) physical system. These are called *quantum bits* or *qubits*. Unlike cbits which could be in a definite state and thus can be determined deterministically, a qubit's state is not predefined and thus cannot be determined unless a perturbation is done on the system, which in turn will bring the system to a definite classical state. Transformations on qubits' state are also governed by quantum mechanical laws of physics in which some physical phenomena are defined that are not observable in classical devices. Some of these quantum physical phenomena are used as resource for improvement in efficiency of computation. Among these are linear superposition of states, quantum entanglement and quantum interference.

In this paper we look into a quantum version of the convolution algorithm and propose an application to the string matching problem. The proposed algorithm will require $O(\log^2 N) + O(\sqrt{N})$ quantum time complexity which is an improvement over the classical algorithm's $O(N \log N) + O(N)$ time complexity. We discuss a quantum implementation of the DFT and its inverse and assess its complexity. We then discuss its application to the convolution algorithm and assess the resulting complexity. We also compare it to the complexity of the classical convolution algorithm.

2. A QUANTUM ALGORITHM FOR STRING MATCHING

In this section we explore the possibility of a quantum version of the convolution method and propose a string matching algorithm which uses this quantum version. First we present some fundamental ideas about quantum computation so the reader may understand well the details of the proposed quantum algorithm. We then present a quantum version of the DFT and its inverse which are vital operations in the classical convolution algorithm. We then present a proposed quantum algorithm for string matching which makes use of the convolution method.

2.1 Quantum Computation

Physical devices which are composed of just a few atomic-scale components are observed to exhibit some phenomena which are not observable on their larger counterparts. These phenomena affects computation such that computing using such kind of devices may seem impractical as compared to how our regular computers perform computation. However, there has been efforts made on realizing ways on how to take advantage of these so called quantum phenomena for more efficient computation. As mentioned in the previous section, among these are linear superposition of quantum states, entanglement of quantum states and quantum interference.

First, we may model the state of a qubit to be a linear combination of states with corresponding *complex amplitudes* as follows,

$$|\delta\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

These amplitudes α_i represent the probability of a classical value represented by a corresponding quantum state $|i\rangle$ occurring as a result of disturbance of the quantum system. The probability of a classical value occurring as a result of disturbance of a quantum system is defined as the modulo square of its corresponding amplitude.

$$P_i = |\alpha_i|^2$$

The state of a quantum system is required to be normalized to model the reversibility of a quantum operation as defined in quantum mechanics. Thus, the total sum of the probabilities of occurrence of each possible classical value of a property of a quantum system must always equate to 1.

$$\sum_i P_i = 1$$

Operations on a quantum system which will result to a non-normalized quantum state will require a succeeding normalization step.

Given a qubit q_0 initially set to be in state $|0\rangle$, we may put it into a linear superposition of quantum states $|0\rangle$ and $|1\rangle$ by applying to it a *Walsh-Hadamard* operator which we denote as H. Doing so will result to the quantum state

$$\begin{aligned} |\delta\rangle &= H|0\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \end{aligned}$$

in which the classical states 0 and 1 have equal probabilities of occurring ($|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$) once the state of q_0 is measured. We borrow notation from quantum mechanics for our computation, called *Dirac* notation. We call the notation $|\ \rangle$ as *ket* and $\langle\ |$ as *bra*. We read $|0\rangle$ as "ket 0", $|1\rangle$ as "ket 1", $\langle 0|$ as "bra 0" and $\langle 1|$ as "bra 1". Also, due to linearity of quantum operations we can evaluate a function F on all values represented by all of the possible quantum states of a qubit. Again, given the quantum operator H as the function and the quantum state $|\delta\rangle$ of qubit q_0, applying H to q_0 will

result to the state

$$\begin{aligned}
|\delta'\rangle &= H|\delta\rangle \\
&= H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= H\left(\frac{1}{\sqrt{2}}|0\rangle\right) + H\left(\frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}H|0\rangle + \frac{1}{\sqrt{2}}H|1\rangle \\
&= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|0\rangle - \frac{1}{2}|1\rangle \\
&= |0\rangle
\end{aligned}$$

Notice how the operator H is distributed to the linear superpositioned states $|0\rangle$ and $|1\rangle$. This is one of the most used technique for quantum computation. It is somehow related to the classical concept of parallelizing the task of evaluating a function against all possible values. The correspondence is not exact though. In classical computation we get all the result of all the function evaluation at the end of the computation. In quantum computation, we will only get a single classical value upon intentional disturbance of the quantum system.

Quantum interference is also one phenomena which distinguishes quantum computation from classical computation. It may seem that quantum computation is just plain classical probabilistic computation. The distinguishing factor is the positive or negative interference of the linear superpositioned states of a quantum system. Note in the above quantum computation of the state $|\delta'\rangle$. The state $|0\rangle$ was subjected to positive interference which resulted to a probability of 1 of occurrence. On the other hand, the state $|1\rangle$ has undergone negative interference which resulted to a 0 probability of its occurrence. Quantum interference is used in quantum computation for eliminating non-solutions to a given computational problem so that measurement of state of the quantum system will result to the desired solution.

Quantum entanglement in a quantum system is another phenomena which is not observable in classical systems. Einstein even described it as a spooky action at a distance. When two qubits in a quantum system is entangled with each other they are considered as linked such that operations on one of them has an effect on the value of the same property of the other. This applies even when the two qubits are far apart from each other. The state of entangled qubits is defined such that their composite state cannot be described as just a product of each qubit's state. The description of each component quantum system should be made with reference to the state of the other system. Quantum entanglement is found to be useful in applications such as secure data exchange and super dense coding. An example of an entangled state of a composite quantum system is the *Bell state* $|\beta_{01}\rangle$ defined as

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

A measurement on the state of the first qubit which will result to a classical value of 0 (1) will have its effect on the state of the second qubit such that upon measurement we will get a classical value of 1 (0).

2.2 Quantum States and Operators in Matrix Representation

Quantum states are modelled in quantum mechanics as vectors in a unit complex vector space of some dimension n in which scalar multiplication and vector addition operations are defined. Given the quantum state of a single qubit denoted as $|0\rangle$ we may represent such state as the vector $\begin{pmatrix}1\\0\end{pmatrix}$ and the state $|1\rangle$ as the vector $\begin{pmatrix}0\\1\end{pmatrix}$. These vectors are orthogonal with each other and are unitary. These vectors are usually referred to as the *computational basis* for the 2-dimensional complex vector space in which we perform computation. Using these vectors we can represent all the other vectors in the 2-dimensional complex vector space. A linear superposition quantum state can be represented in column vector notation as

$$\begin{aligned}
|\delta\rangle &= \alpha_0|0\rangle + \alpha_1|1\rangle \\
&= \alpha_0\begin{pmatrix}1\\0\end{pmatrix} + \alpha_1\begin{pmatrix}0\\1\end{pmatrix} \\
&= \begin{pmatrix}\alpha_0\\0\end{pmatrix} + \begin{pmatrix}0\\\alpha_1\end{pmatrix} \\
&= \begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix}
\end{aligned}$$

where α_0 and α_1 are complex values. On the other hand, the states of qubits of a composite quantum system composed of an n-dimensional and m-dimensional component systems is defined on an mn-dimensional complex vector space. Given a 2-qubit quantum system, it is defined on a 4-dimensional complex vector space in which its element vectors are made up of four complex valued elements. The quantum state $|00\rangle$ of a composite quantum system made up of two qubits may thus be defined by a complex vector

$$\begin{aligned}
|00\rangle &= |0\rangle \otimes |0\rangle \\
&= \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \\
&= \begin{pmatrix}1\\0\\0\\0\end{pmatrix}
\end{aligned}$$

Quantum operators on the other hand can be represented as matrices and are required to be unitary (and therefore invertible) to model the reversibility of a quantum transformation in a closed quantum system. For example, given the Pauli-X operator for a single qubit which is defined by the transformation on the states $|0\rangle$ and $|1\rangle$

$$X|0\rangle = |1\rangle \text{ and } X|1\rangle = |0\rangle$$

its counterpart matrix operator is defined as the 2×2 matrix

$$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$$

Applying this matrix to the vector representations of the

states $|0\rangle$ and $|1\rangle$ will result to

$$\begin{aligned} X|0\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= |1\rangle \end{aligned}$$

and

$$\begin{aligned} X|1\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= |0\rangle \end{aligned}$$

Thus, a quantum operator for an n-dimensional quantum system can be represented by an $n \times n$ dimensional matrix which carries out the transformation on the state of the system.

There are much more basic concepts in quantum computation but we will not discuss them furthermore in this paper to keep us focused on the topic. The reader is encouraged to read more on these concepts to have a deeper appreciation of the computing model. Valuable exposition of these topics can be found in [14], [15] and [18].

2.3 Quantum Fourier Transform

One important operation in quantum computation is the Quantum Fourier Transform (QFT). This is the quantum version of the classical DFT for Fourier analysis of any data which are temporal in nature. Given a set of sample points in the time domain, applying DFT will represent the samples in the frequency domain. If we represent some N sample points as elements $f(i)$ of a N-dimensional vector and DFT as an $N \times N$ matrix operator acting on the vector, application of the DFT is defined as

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{N-1} \\ 1 & w^2 & w^4 & \dots & w^{2(N-1)} \\ & & \vdots & & \\ 1 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ \vdots \\ f(N-1) \end{pmatrix}$$

where $w = e^{\frac{2\pi i}{N}}$ and will result to another N-dimensional vector with elements defined as

$$F(i) = \sum_{j=0}^{N-1} f(j) w^{ij}$$

for $0 \leq i \leq N-1$.

QFT also has the same effect as DFT. Only, it includes a normalizing scalar factor to keep the resulting quantum state normalized. Given a computational base state $|i\rangle$, application of QFT to it has the effect

$$QFT|i\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} w^{ij} |j\rangle$$

Thus, given a quantum system in a linear superposition state $|\delta\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$, applying QFT to it will result to

$$\begin{aligned} QFT|\delta\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \alpha_i \left(\sum_{j=0}^{N-1} w^{ij} |j\rangle \right) \\ &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \left(\sum_{i=0}^{N-1} \alpha_i w^{ij} \right) |j\rangle \end{aligned}$$

A fast classical algorithm for computing DFT on N elements is the Fast Fourier Transform (FFT), which computes DFT using $\Theta(N \log N)$ classical logic gates. On the other hand, QFT will only require $\Theta(\log^2 N)$ quantum logic gates for the same number of elements. The QFT can also be represented as a matrix operator defined as

$$\frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{N-1} \\ 1 & w^2 & w^4 & \dots & w^{2(N-1)} \\ & & \vdots & & \\ 1 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)^2} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N-1} \end{pmatrix}$$

where $w = e^{\frac{2\pi i}{N}}$.

One non-trivial computational problem to which QFT finds good application is the *phase estimation* problem. Given a unitary operator U which has an eigenvector $|u\rangle$ with corresponding eigenvalue $e^{2\pi i \gamma}$ where γ is unknown, the phase estimation problem requires an estimate of γ as an output. QFT can be used to get a good estimate of γ with a high probability. This *quantum phase algorithm* can then be used to solve computational problems which are reducible to phase estimation problem. Peter Shor used the quantum phase estimation algorithm for the order-finding sub-routine of his well known quantum algorithm for prime factorization which will have a huge impact to cryptography once a quantum computer is realized and his algorithm be programmed to execute on it.

As we saw in QFT it is so much the same as the classical DFT with just some subtle differences. The distinguishing factor of QFT is its use of qubits instead of cbits and with it comes properties not present in cbits. As mentioned in the previous sections, qubits behave differently as compared to cbits such that they can be put into a linear superposition of states. The number of qubits needed to represent N sample points is $\lceil \log N \rceil$ and we can put these qubits into a linear superposition of all the N sample points. Also, there is a normalizing factor in QFT to keep the transform unitary. The encoding of sample points is not as direct though as we thought it to be just like in the classical DFT. In classical DFT, the transformation is directly applied to the sample points represented by the cbit states. In QFT, transformation is applied to the amplitudes of the quantum states. There is no way of accessing the transformed amplitudes of the states through measurement thus we cannot determine the transformed value represented in the amplitude of the states. There is a method though called as *phase kickback* which we could use to encode the sample points into the amplitudes and later get an estimate of the resulting transformed value.

2.4 Phase Kickback

As mentioned in the previous section QFT affects the amplitude of states of a quantum system and not the states

themselves. The phase kickback method encodes the value of sample points in the amplitude of the states so that QFT effectively transforms the sample points just like DFT does. Given a quantum system prepared in the state

$$|x\rangle|y\rangle$$

the phase kickback method effects the following transformation

$$|x\rangle|y\rangle \rightarrow |x\rangle|y \ \oplus f(x)\rangle$$

from the initial composite state, where $\oplus$ is the exclusive-OR operation. This transformation can be interpreted as the transformation

$$|x\rangle|y\rangle \rightarrow (-1)^{f(x)}|x\rangle|y\rangle$$

For this simple case in which $f(x) = 0$ or $f(x) = 1$, the value of $f(x)$ is encoded into the phase factor of the state. Thus the phase kickback method can be viewed as moving the values $f(x)$ from the kets into the *phase factors* of the quantum system's linear superposition state. Given such a linear superposition state with encoded phase factors, application of QFT will effectively transform the sample points $f(x)$.

In the succeeding sections we discuss the application of QFT and phase kickback method as sub-routines for the convolution algorithm. We then discuss how we can use this quantum convolution algorithm for performing string matching.

2.5 Quantum Convolution Algorithm

Not all classical algorithms have a direct quantum version. This can be attributed to the unique properties of quantum systems and quantum operators which are not present in classical systems. It would then be just right to ask first if a quantum convolution algorithm is even feasible.

2.5.1 Is an Exact Quantum Convolution Algorithm Even Possible?

In [13] it was shown that an *exact* quantum analogue of the convolution algorithm is not physically realizable in a quantum computer. Specifically, Lomont was able to show that in the quantum analogue of the convolution problem, given two quantum states

$$|\alpha\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$$

and

$$|\beta\rangle = \sum_{j=0}^{N-1} \beta_j |j\rangle$$

representing the input sequences there is no sequence of unitary transformations and measurement operations applied to the input quantum states and an ancilla quantm state $|\rho\rangle$ that will lead to computation of the state

$$|\gamma\rangle = \sum_{k=0}^{N-1} \gamma_k |k\rangle$$

where $|\gamma\rangle$ represents the normalized sequence defined as

$$c_k = \sum_{j=0}^{N-1} \alpha_j \beta_{k-j} \ for \ k = 0, \ldots, N-1$$

and where subscripts are taken ($mod\ N$).

In their paper it was proven that the step in the classical convolution algorithm called *component-wise multiplication* has no quantum version. There does not exist a unitary transformation which can perform component-wise multiplication of two vectors. They have also shown the same result for the quantum correlation algorithm which make use of the same component-wise multiplication step.

2.5.2 How About We Approximate?

Though the exact quantum version of the convolution and correlation algorithm are physically impossible, an *approximate component-wise multiplication* step was suggested by Curtis and Meyer in [6]. In their paper they investigated the problem of finding from a larger graphical image the coordinates of a smaller sub-image whose pixel values matches that of a so called template image.

A classical solution to the problem which they considered is the classical *correlation algorithm*. In application to their problem, the correlation between the pixel values of sub-images and those of the template are computed in which a maxima is identified with the offset a being searched for. To improve the efficiency of the correlation algorithm for image processing they intended to come up with a quantum analogue of the algorithm. Their hope for a more efficient quantum image processing algorithm is based on the exponential speed-up introduced by the use of QFT in Shor's quantum algorithm for factoring large integers.

As expected and in agreement with the result in [12], they recognized the impossibility of performing the component-wise multiplication step of the classical correlation algorithm in the quantum setting. Thus, they suggested an approximation of the quantum analogue of the component-wise multiplication step. As they have shown in their paper, the only difference between the exact and the approximate quantum version of the multiplication step is the *normalizing multiplicative factor* for the Fourier transform of one of the sequences. This factor is necessary to qualify the approximate component-wise multiplication operation as a unitary transformation.

Given quantum state $|\beta\rangle = \sum_{j=0}^{N-1} \beta_j |j\rangle$, we denote its Fourier transform in the quantum setting as $|\beta'\rangle$ and is defined as follows

$$\begin{aligned}
|\beta'\rangle &= QFT|\beta\rangle \\
&= QFT\left(\sum_{j=0}^{N-1} \beta_j |j\rangle\right) \\
&= \sum_{j=0}^{N-1} \beta_j \ QFT|j\rangle \\
&= \sum_{j=0}^{N-1} \beta_j \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk}\right) |k\rangle \\
&= \sum_{j=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \beta_j w^{jk}\right) |k\rangle
\end{aligned}$$

where $w = e^{\frac{2\pi i}{N}}$. Likewise, the quantum Fourier transform

for a quantum state $|\alpha\rangle = \sum_{j=0}^{N-1} \alpha_j |j\rangle$ will be

$$|\alpha'\rangle = \sum_{j=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_j w^{jk} \right) |k\rangle$$

A quantum operator V_α can be defined to approximate $QFT|\alpha\rangle$ instead. V_α can be defined as the unitary and diagonal $N \times N$ matrix whose diagonal elements $V_{i,i}$ is defined as

$$V_{i,i} = \frac{\langle i|QFT|\alpha\rangle}{|\langle i|QFT|\alpha\rangle|}$$

for $0 \leq i \leq N-1$. Denoting $V_\alpha|\beta'\rangle$ as the quantum superposition state $|\gamma\rangle$ and $|\gamma_i\rangle$ as the i-th basis state in $|\gamma\rangle$, applying V_α to $|\beta'\rangle$ will have the following effect

$$|\gamma_i\rangle = \frac{\langle i|QFT|\alpha\rangle}{|\langle i|QFT|\alpha\rangle|} \langle i|QFT|\beta\rangle$$

for $0 \leq i \leq N-1$. The effect is an approximation of the component-wise multiplication step

$$QFT|\alpha\rangle \cdot QFT|\beta\rangle$$

The only difference between the approximate quantum version and the impossible exact quantum version are the additional normalizing multiplicative factor of $|\langle i|QFT|\alpha\rangle|$ in the approximate version.

Given the above approximate quantum version of the component wise multiplication step, we can define an approximation of the quantum analogue of the classical convolution algorithm as follows:

Approximate Quantum Convolution Algorithm

1. Apply QFT to input state $|\beta\rangle$.
$$|\beta'\rangle = QFT|\beta\rangle$$
2. Perform approximate component-wise multiplication step on $|\beta'\rangle$.
$$\begin{aligned} |\gamma\rangle &= V_\alpha|\beta'\rangle \\ &\approx QFT|\alpha\rangle \cdot QFT|\beta\rangle \end{aligned}$$
3. Apply IQFT to state $|\gamma\rangle$.
$$\begin{aligned} |\gamma'\rangle &= IQFT|\gamma\rangle \\ &\approx IQFT\left(QFT|\alpha\rangle \cdot QFT|\beta\rangle\right) \end{aligned}$$

Both the QFT and IQFT can be implemented in a quantum circuit using $O(\log^2 N)$ gates while the approximate component-wise multiplication step can be executed using the V_α operator in $O(1)$ time step by linearity. In total, the approximate quantum version of the convolution algorithm will require $O(\log^2 N)$ quantum time complexity.

2.6 String Matching Using Approximate Quantum Convolution Algorithm

The classical convolution algorithm is able to improve the performance of the naive algorithm for pattern matching up to a logarithmic factor by using FFT for computing the DFT of the two input sequences. From the $O(MN)$ time complexity of the naive algorithm the classical convolution algorithm is able to improve it to $O(N \log N) + O(N)$ whenever $\log N < M$. In this section we present a proposed quantum algorithm for pattern matching on strings using the approximation of the quantum analogue of the classical convolution algorithm.

We define the proposed quantum algorithm as follows:

A Quantum Algorithm for String Matching

1. Prepare a quantum state vector $|\beta\rangle$ to represent text T using phase kickback.
2. Apply QFT to state $|\beta\rangle$ to obtain state
$$|\beta'\rangle = QFT|\beta\rangle$$
3. Perform approximate quantum component-wise multiplication by applying unitary transform V_α to $|\beta'\rangle$ to obtain state
$$|\delta\rangle = V_\alpha|\beta'\rangle$$
4. Apply IQFT to state $|\delta\rangle$ to obtain state
$$|\gamma\rangle = IQFT\left(|\delta\rangle\right)$$
5. Perform measurement on the linear superposition state $|\gamma\rangle$ to obtain a classical value i corresponding to an index in text T.
$$i = M|\gamma\rangle$$
6. Verify correctness of resulting classical value i by determining corresponding index j in text T. If
$$d(P, T[j, j+M-1]) \leq D$$
return j. Else, re-run the algorithm.

In a binary alphabet setting of $\Sigma = \{0, 1\}$ and single existence of an exact match of P in T, the author's initial investigation of the need for an amplitude amplification step prior to measurement of the state of the qubits resulted to the knowledge that state $|\gamma\rangle$ is the basis state $|i\rangle$ which corresponds to the matching index in T. The amplitude of the other bases states in the linear superposition state $|\delta\rangle$ collapses to 0 upon application of $IQFT$ while that of the solution basis state $|i\rangle$ results to 1. Thus, measurement of the state $|\gamma\rangle$ will result to the classical value i. This implies that in the investigated alphabet and solution existence setting there is no need for an amplitude amplification step.

The initial result has a positive implication of saving the quantum algorithm $O(\sqrt{N})$ time steps for the amplitude amplification. On the other hand, it also has the negative implication on what variants of the string matching problem the algorithm can solve. In the initially investigated setting, the algorithm tells us about the index in T in which an *exact* match of P occurs. It does not give us data on indices in T in which P *approximately* occurs though. Also, the design of the algorithm implies that a unique unitary operator V_α and unique unitary operators for encoding T and P during initialization phase has to be realized. This requirement is inherent in the convolution algorithm.

Next in line for investigation is the setting of binary alphabet $\Sigma = \{0, 1\}$ and multiple existence of an exact match of P in T. Time complexity of the initialization phase will also be investigated to have a more detailed assessment of the complexity of the proposed algorithm. The authors also aim to attain a conclusion on whether the algorithm can indeed solve the *approximate* variant of the string matching problem.

2.7 Complexity of Proposed Solution

Computing naively the DFT with matrix dimension $N \times N$ requires $O(N^2)$ time complexity. Using the FFT algorithm by Cooley and Tukey [4] to compute for the DFT of same dimension provides improvement in complexity and requires only $O(N \log N)$ time steps. On the other hand, the quantum analogue of DFT applies the same dimension transformation by requiring $O(\log^2 N)$ quantum time complexity which is a huge improvement over the classical naive DFT and FFT algorithm. Thus, steps 1 to 4 of the algorithm will require $O(\log^2 N)$ time steps while step 5 will require $O(1)$ time steps. Step 7 will require $O(M)$ sequential time steps and can be classically parallelized to improve it to $O(1)$ time steps instead. Thus, in total, the proposed algorithm will require $O(\log^2 N) + O(1)$ time complexity. This is also an improvement over the quantum algorithm of Ramesh and Vinay [16] which aims to solve the same problem through random sampling techniques.

3. CONCLUSION

In this paper an approximation of a quantum version of the classical convolution algorithm is presented. The approximate quantum convolution algorithm is presented as a sub-routine for a proposed quantum algorithm for pattern matching on strings. The complexity of the proposed quantum string matching algorithm is $O(\log^2 N)$ quantum time and is an improvement compared to the classical convolution algorithm when used for string matching. Initial investigation on setting for binary alphabet and single existence of pattern in the text shows that the algorithm provides the index of the exact match in the text. This eliminates the need for an amplitude amplification step in the final phase of the algorithm but restricts the algorithm to the exact variant of the string matching. Further investigation is needed to understand the limits of the proposed algorithm and further results will be published in succeeding papers.

4. REFERENCES

[1] Lok-Lam Cheng, D.W. Cheung, and Siu-Ming Yiu. Approximate string matching in DNA sequences. In *Eighth Int. Conf. Database Syst. Adv. Appl. 2003. (DASFAA 2003). Proceedings.*, pages 303–310. IEEE, 2003.

[2] Yoon-ho Choi, Moon-young Jung, and Seung-woo Seo. A fast pattern matching algorithm with multi-byte search unit for high-speed network security q. *Comput. Commun.*, 34(14):1750–1763, 2011.

[3] R. Clifford and C. Iliopoulos. Approximate string matching for music analysis. *Soft Comput.*, 8(9):597–603, July 2004.

[4] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.*, 19(90):297–301, 1965.

[5] Maxime Crochemore, Costas S. Iliopoulos, Thierry Lecroq, and Y. J. Pinzon. Approximate String Matching in Musical Sequences. In Miroslva Balik and Milan Simanek, editors, *Proc. Prague Stringology Conf. 2001*, number September, pages 26–36, 2001.

[6] Daniel Curtis and David A Meyer. Towards Quantum Template Matching. In Ronald E. Meyers and Yanhua Shih, editors, *Proc. SPIE 5161, Quantum Commun. Quantum Imaging*, pages 134–141, February 2004.

[7] Carla Correa Tavares dos Reis. Approximate String Matching Algorithm Using Parallel Methods for Molecular Sequence Camparisons. In *2005 Purtuguese Conf. Artif. Intell.*, pages 140–143. IEEE, December 2005.

[8] Michael J. Fischer and Michael S. Paterson. String Matching and other Products. Technical report, Massachusetts Institute of Technology, Cambridge, 1974.

[9] LK Grover. A fast quantum mechanical algorithm for database search. *Proc. twenty-eighth Annu. ACM ...*, pages 212–219, 1996.

[10] Y U Jianming, X U E Yibo, and L I Jun. Memory Efficient String Matching Algorithm for Network Intrusion Management System *. *TSINGHUA Sci. Technol.*, 12(5):585–593, 2007.

[11] Donald E Knuth. *The Art of Computer Programming*, volume 3, chapter Sorting an, page 704. Addison-Wesley, second edition, 1981.

[12] Chris Lomont. Quantum Convolution and Quantum Correlation Algorithms are Physically Impossible. pages 1–10, 2003.

[13] Chris Lomont. Robust String Matching in O (\ sqrt {N}+ M) Quantum Queries. pages 1–10, 2003.

[14] David McMahon. *Quantum Computing Explained*. John Wiley & Sons, Inc., 2008.

[15] Michael A Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, illustrate edition, 2000.

[16] H Ramesh and V Vinay. String matching in O(sqrt(n)+sqrt(m)) quantum time. *J. Discret. Algorithms*, 1:103–110, 2003.

[17] Esko Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118, January 1985.

[18] Colin P Williams. *Explorations in Quantum Computing*. Springer-Verlag London Limited, second edition, 2011.

[19] Wu Yang, Bin-xing Fang, Bo Liu, and Hong-li Zhang. Intrusion detection system for high-speed network. *Comput. Commun.*, 27:1288–1294, 2004.