A QUANTUM ALGORITHM FOR VITERBI DECODING OF CLASSICAL CONVOLUTIONAL CODES

JON R. GRICE AND DAVID A. MEYER

ABSTRACT. We present a quantum Viterbi algorithm with better than classical performance under certain conditions (for decoding convolutional codes, for instance; large constraint length $Q$ and short decode frames $N$). The algorithm exploits the fact that the decoding trellis is similar to the butterfly diagram of the fast Fourier transform, with its corresponding fast quantum algorithm.

1. INTRODUCTION

In his 1971 paper on the Viterbi Algorithm (VA), Forney [6] noted that “Many readers will have noticed that the trellis reminds them of the computational flow diagram of the fast Fourier transform (FFT). In fact, it is identical, except for length, and indeed the FFT is also ordinarily organized cellwise. While the add-and-compare computations of the VA are unlike those involved in the FFT, some of the memory-organization tricks developed for the FFT may be expected to be equally useful here.”

The celebrated quantum Fourier transform [5], [4] uses tensor-product structure to achieve its increase in efficiency over its classical counterpart, and so in this paper we propose a quantum Viterbi algorithm (QVA) taking advantage of the tensor-product structure of the trellis that in certain applications may outperform the (classical) Viterbi algorithm.

Amongst many other applications, the Viterbi algorithm [11] is useful in the decoding of convolutional codes. Thus, in this paper we will demonstrate the application of a quantum algorithm to decoding classical convolutional codes. Quantum algorithms for decoding simplex codes [1], more generally Reed-Muller codes [8] have been proposed and show improvements over the classical algorithms. Protecting quantum information with convolutional encoding has been discussed in [3, 10].

For a finite alphabet $Q$, the Viterbi algorithm [6] is a way to find the most likely sequence of states $\pi^* \in \mathbb{Q}^N$ that a hidden Markov model (HMM) transitions through given a set of observations $Z$, called emissions. The class of HMMs $\mathcal{C}$ which can be efficiently approximated with the QVA described herein has not yet been fully elucidated. Due to the unique constraints imposed by quantum reversibility (namely unitarity), $\mathcal{C}$ may be significantly smaller than the class of all useful HMMs. Namely, the Viterbi algorithm for

Date: May 30, 2014.
convolutional decoding is not directly possible with the QVA, yet in sections 3 and 4 we derive ‘virtual probabilities’ (VPs), which are transition probabilities for a HMM that when its lattice is constructed with the QVA, efficiently decodes a convolutional code. In particular, lemma 2 demonstrates that the maximum free distance binary convolutional codes tabulated in [2] are in $C$.

The single trial performance of the QVA is $O(N|Q|F(\log F)^2)$ ($F$ is a fixed number, the ‘fanout’ defined later), compared to the classical Viterbi algorithm, which has $O(N|Q|^2)$ performance. This means that the QVA can be used with advantage when $F$ is smaller than $|Q|$ and when the number of repeat trials required is low. In particular, the convolutional code lattice has a low ‘fanout’ $F$, and for reasonably small $N$, the number of repeat trials required can be kept small. After one trial, the output of the QVA before measurement is a superposition of possible paths through the lattice with probability amplitudes monotonic with the probability of the lattice path. Taking measurements and repeating the trial a certain number of times allows one to choose the most probable solution. The trials can be performed in parallel with a classical mode finding step at the end.

The probability of selecting the wrong answer (not finding the mode) falls exponentially in the number of trials, discussed in section 3.1, yet may increase exponentially in $N$, number of timesteps for a decoding algorithm, suggesting that the QVA may be most appropriately used for short time windows. Furthermore, there are applications of HMMs where $Q$ becomes very large as to model a continuous state space [11] and when for $k$-order HMMs, which can be recast as a first order HMM with large $Q$. In those cases the QVA has the potential for a considerable speedup over the classical Viterbi algorithm.

The key idea of the QVA is to represent a particular path through the lattice, say $\pi = \pi_1 \pi_2 \cdots \pi_N$ as the quantum state $|\pi_1 \pi_2 \cdots \pi_N\rangle$, where each $|\pi_i\rangle \in \mathbb{C}^Q$. The quantum algorithm is a parallel algorithm; all possible paths exist in superposition with amplitudes according to their probabilities as the algorithm builds the lattice. So the following lattice (here $Q = \{0, 1, 2\}$)

is represented by the quantum state

$$|\psi_2\rangle = \sqrt{p_{0,0}p_{0,0}}|000\rangle + \sqrt{p_{0,0}p_{0,2}}|002\rangle + \sqrt{p_{0,2}p_{2,1}}|021\rangle + \sqrt{p_{0,2}p_{2,2}}|022\rangle.$$\n
This state would be the result of two steps of the algorithm starting with initial state $|0\rangle$. 

![Diagram of lattice](attachment:image.png)
2. The Algorithm

As mentioned in the introduction, the algorithm builds a quantum state, each step adding a tensor factor sequentially in the forward direction.

2.1. The Blocks. We start with a hidden Markov model of $Q$ states, where the state at time step $1 \leq n \leq N$ is written $x_n$, but is hidden; instead an emission $y_n \in Z$ is made visible by the HMM at the time the transition from $x_n$ to $x_{n+1}$ is made. The probability of the transition is written
\[ P(i, j|y) = \Pr(x_{n+1} = j|x_n = i, y_n = y), \]
with emission probabilities
\[ P(y|i, j) = \Pr(y_n = y|x_{n+1} = j, x_n = i). \]

We also let
\[ P_{i,j}(y) = P(i, j|y)P(y|i, j), \]
and clearly we have
\[ Q \sum_{j=1}^{Q} P(i, j|y) = 1 \text{ for all } y. \]  

The additional condition
\[ \sum_j P_{i,j}(y) = 1 \text{ for all } i, y, \]  

is imposed so that the representation as a quantum state of the transition from state $i$ to state $j$, given $y$,
\[ |\psi_{i,y}\rangle = \sum_j \sqrt{P_{i,j}(y)}|j\rangle, \]
is normalized. Condition (3) deserves some explanation. Suppose we have an HMM representing a convolutional encoder-channel system (CCS) that can be in state $x_a$ or state $x_b$. Supposing also that emission $y$ is received and the HMM stipulates that a system in state $x_b$ is not likely to emit $y$ for any of the transitions. Then we would expect $\sum_j P(y|x_b, j)$ to be small, and also each $P_{x_a, j}(y)$ would be small. This is indeed the case for the CCS with encoder (9) below. In that example, having $y = 01$ would give posterior evidence that the system was not in state $x = 00$, since the system should emit $y = 00$ or $y = 11$ when transitioning from that state. In fact, $P(y|00, j)$ (with $j$ being one of two valid targets) should be the probability that there was a bit-flip error occurring in the channel and the $y$ coming directly from the encoder was really $00$ or $11$. So if the probability of a channel bit-flip is $p$, we have $\sum_j P(y|00, j) = 2p < 1$. Then $\sum_j P_{00, j}(y) = pP(00, 00|y) + pP(00, 11|y) = 2p \sum_j P_{00, j}(y) = 2p < 1$, which violates (3). The trick to decoding convolutional codes with the QVA, which requires (3) to be satisfied, is, while fixing $i$ and $y$, to change the values in $\{P(i, j|y)\}_j$, preserving the relative ordering of the set with respect to $j,$
and adjusting the magnitudes of the corresponding $P(i, j|y)$ in a way that preserves (2) and (3). This gives us a new HMM which “approximates” the original CCS. The method for choosing these $P_{i,j}(y)$ “virtual probabilities” is explained in section 4.

For a state $|\psi\rangle$, let $U_\psi$ be some unitary operation such that $U_\psi : |0\rangle \rightarrow |\psi\rangle$. There are many possible operators which satisfy that requirement, with a canonical (but generally nonoptimal) choice being given in section 2.3. In particular, (3) ensures that $U_\psi |\psi_{i,y}\rangle$ is unitary.

The controlled operation which is active on the input pattern being $|\psi_c\rangle = |k\rangle$, implementing some unitary operation $U : \mathbb{C}^Q \rightarrow \mathbb{C}^Q$ will be denoted

$$
\begin{array}{c}
|\psi_c\rangle \\
|\psi_t\rangle
\end{array}
\rightarrow
\begin{array}{c}
k \\
U
\end{array}
$$

That is, the above controlled operation takes $|k\rangle|\psi_t\rangle$ to $|k\rangle U|\psi_t\rangle$, extended by linearity, and acts as the identity on all $|j\rangle|\psi\rangle$ for $j \neq k$.

For example, if $Q = 2$ then the gate

$$
\begin{array}{c}
1 \\
U_{|1\rangle}
\end{array}
$$

is the standard two-qubit controlled not gate $\text{NOT}$, up to relative phase, since for a $2 \times 2$ unitary $U_{|1\rangle} : |0\rangle \rightarrow |1\rangle$ forces $U_{|1\rangle} : |1\rangle \rightarrow \alpha |0\rangle$.

We will let the gate

$$
\begin{array}{c}
V_y
\end{array}
$$

stand for the string of controlled operations:

$$
\begin{array}{c}
0 \\
U_{|\psi_{0,y}\rangle}
\end{array}
\rightarrow
\begin{array}{c}
1 \\
U_{|\psi_{1,y}\rangle}
\end{array}
\rightarrow
\begin{array}{c}
2 \\
U_{|\psi_{2,y}\rangle}
\end{array}
\rightarrow \cdots
\begin{array}{c}
Q \\
U_{|\psi_{Q,y}\rangle}
\end{array}
$$

One can see that the matrix representation of the unitary operator $V_y : \mathbb{C}^Q \otimes \mathbb{C}^Q \rightarrow \mathbb{C}^Q \otimes \mathbb{C}^Q$ in the computational basis is block diagonal with the $U_{|\psi_{i,y}\rangle}$ for the blocks.
2.2. The Algorithm. Given an initial distribution $|\psi_0\rangle$ and sequence of emissions $\{y_i\}_{i=1}^N$ perform
\[
|\psi_0\rangle V_{y_1} |0\rangle V_{y_2} |0\rangle V_{y_3} \cdots \cdots |0\rangle V_{y_{N-1}} |0\rangle V_{y_N}
\]
where the slashes denoting bundles of quantum wires are omitted for clarity. The algorithm implements $N$ of the $V_y$ blocks, each of which are composed of $Q$ controlled operations, for a gate complexity of $O(NQF(\log F)^2)$ where $F$ is the fanout of the HMM, explained later. The gate complexity is determined in section 2.3. Notice that the measurement operation at the end commutes with the control section of each $V$ box. That is, each controlled $V$ can be fed by classical data from the measurement of the previous bit. This suggests that the power of the quantum algorithm over a probabilistic one lies in the state preparation step. Hence for good performance in general we need $F \ll Q$, which for convolutional coders $F$ is the number of possible states that an input message may cause the encoder to transition to, $F = 2$ in the examples in this paper, the rate $1/b$ codes, which is fixed relative to $Q$, which will increase with constraint length. Thus for short $N$, the QVA is $O(NQ)$.

Each of the $V_y$ are implicitly classically controlled, conditional on the emissions which is the code as received. The meter step would be omitted if the algorithm is to be included as a sub-block in a larger quantum algorithm.

**Proposition 1.** Fixing a set of emissions $\{y_i\}$, the output of the algorithm right before the measurement step is
\[
|\psi_f\rangle = \sum_{\text{Possible paths } \pi} \sqrt{p_{\pi}} |\pi\rangle,
\]
where $p_{\pi}$ is the probability of the path $\pi \in Q^N$ being taken.

**Proof.** Assume the assertion is true for $N - 1$ steps, so
\[
|\psi_{N-1}\rangle = \sum_{\pi} \sqrt{p_{\pi}} |\pi_1\pi_2 \cdots \pi_{N-1}\rangle,
\]
with the sum over all possible paths \( \pi \) of length \( N - 1 \), and then apply the gate

\[
\begin{array}{c}
|x\rangle \\
|0\rangle
\end{array} \quad \begin{array}{c}
\text{V}_{y_N}
\end{array}
\]

with the last qubit of \( |\psi_{N-1}\rangle \) being fed into the wire marked with \( |x\rangle \). The output of that operation will be

\[
|\psi_N\rangle = \sum_{i \text{ s.t. } \pi_{N-1} = i} \sqrt{p_{i\pi'}} |\pi_1\pi_2 \cdots \pi_{N-2}\rangle |\psi_{i,y}\rangle
\]

\[
= \sum_{i \text{ s.t. } \pi_{N-1} = i} \sum_{j \text{ s.t. } \pi_N = j} \sqrt{p_{i\pi'}} |\pi_1\pi_2 \cdots \pi_{N-2}i\rangle |j\rangle \sqrt{P_{i,j}(y_N)}
\]

\[
= |\psi_f\rangle.
\]

The base case follows from the definition: If \( \pi_1 \) is the path from \( i \) to \( j \) (given \( y_1 \)), then \( \pi_1 = P_{i,j}(y_1) \). \( \square \)

2.3. Implementation of a canonical unitary \( U_\psi \). In this section we construct the sub-block \( U_\psi \) from basic gates, in particular

\[
R_y(\theta) = \begin{pmatrix}
\cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\
\sin \frac{\theta}{2} & \cos \frac{\theta}{2}
\end{pmatrix},
\]

and then use generalized Toffoli gates to construct the controlled operations in \( V_y \).

Let \( q = \log |Q| \) and if \( |\psi\rangle = \sum_{i=1}^{2^q} p_i |i\rangle \), with \( K \) of the \( p_i \) nonzero, we call \( K \) the fanout of the state, and hence the maximum \( K \) for all the \( \psi \) derived from the HMM transitions will be denoted \( F \), the fanout of the HMM. For binary convolutional codes defined in section 3, with \( k \) the number of bits in the message block, \( F = 2^k \), the number of possible states that the encoder can transition to given a new message block entering the encoder. For the codes in that section, \( Q \) can be made arbitrarily large (while \( F \) remains fixed) by increasing the constraint length. We need tighter bounds on gate complexity than what is given for general gates in [9], but the method is similar. If \( R_{y,a,b} \) is the two-level unitary acting nontrivially on the subspace spanned by \( |a\rangle, |b\rangle \) then constructing a unitary matrix \( R \) with specified first column \( |\phi\rangle \) will give \( R: |0\rangle \rightarrow |\phi\rangle \).

For some \( 2 \times 2 \) unitary matrix \( W \), we write \( W_{\{|a\rangle,|b\rangle\}} \) for the two-level unitary operator which acts as \( W \) on the subspace spanned by \( \{|a\rangle,|b\rangle\} \) and as the identity elsewhere. That is,

\[
W_{\{|a\rangle,|b\rangle\}} |a\rangle = (W)_{1,1} |a\rangle + (W)_{1,2} |b\rangle,
\]

\[
W_{\{|a\rangle,|b\rangle\}} |b\rangle = (W)_{2,1} |a\rangle + (W)_{2,2} |b\rangle \text{ and}
\]

\[
W_{\{|a\rangle,|b\rangle\}} |\psi\rangle = |\psi\rangle.
\]
A QUANTUM ALGORITHM FOR VITERBI DECODING OF CLASSICAL CONVOLUTIONAL CODES

for $|\psi\rangle$ not in the span of $\{|a\rangle, |b\rangle\}$. Let

$$R_y(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

be the y-rotation operator, or the exponentiated $Y$ gate. If the context is clear we will also write

$$R_y(t) = \begin{pmatrix} t & -\sqrt{1-t^2} \\ \sqrt{1-t^2} & t \end{pmatrix}.$$  

We claim that the first column of

$$V := R_y(\theta_1)\{|0\rangle, |1\rangle\} R_y(\theta_2)\{|0\rangle, |2\rangle\} \cdots R_y(\theta_N)\{|0\rangle, |N\rangle\}$$

in the computational basis is the vector

$$\begin{pmatrix} \cos(\theta_1) \cos(\theta_2) \cdots \cos(\theta_K) \\ \cos(\theta_2) \cos(\theta_3) \cdots \cos(\theta_K) \sin(\theta_1) \\ \cos(\theta_3) \cos(\theta_4) \cdots \cos(\theta_K) \sin(\theta_2) \\ \vdots \\ \cos(\theta_{K-1}) \cos(\theta_{K-2}) \sin(\theta_{K-1}) \\ \cos(\theta_K) \sin(\theta_K) \end{pmatrix},$$

which is equivalent to spherical coordinates in $\mathbb{R}^{K+1}$. Thus we have constructed a gate

$$V : |0\rangle \rightarrow |\psi\rangle$$

for any $|\psi\rangle \in \mathbb{R}^{K+1}$, which is sufficient for our purposes. To see that this is true, assume that a product of $k$ $R_y$ matrices has the block form

$$\prod_{j=1}^k R_y(\theta_j)\{|0\rangle, |j\rangle\} = \begin{pmatrix} \tilde{a} & A \\ 0 & I_{K-1-(k+1)} \end{pmatrix} \oplus I_{K-1-(k+1)}.$$  

Since $R_y(\theta_{k+1})\{|0\rangle, |k+1\rangle\}$ has the matrix form

$$\begin{pmatrix} \cos(\theta_{k+1}) & 0 & \cdots & 0 & -\sin(\theta_{k+1}) \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sin(\theta_{k+1}) & 0 & 0 & \cdots & \cos(\theta_{k+1}) \end{pmatrix} \oplus I_{K-1-(k+1)},$$

we find the product

$$\prod_{j=1}^k R_y(\theta_j)\{|0\rangle, |j\rangle\} = \begin{pmatrix} \cos(\theta_{j+1}) \tilde{a} & A' \\ \sin(\theta_{j+1}) & 0 \cdots & 0 \end{pmatrix} \oplus A'' \oplus I_{K-1-(k+1)}.$$  

Thus using the base case: $\tilde{a} = (\cos \theta_1, \sin \theta_1)$ and the recursion for $\tilde{a}$ we get $\mathcal{S}$. Therefore it requires $K$ ($F$) rotation operators $R_y$ to implement $V$ in addition to the logic to change the subspaces that the $R_y$ act on. Using the
Gray coding technique for the universal construction of quantum gates [9], the subspace changing logic requires \( f^2 2^f \), (where \( F = 2^f \)) operations per \( V \), or \( F(\log F)^2 \). One can obtain even better results in the combinatorial control logic by exploiting the special structure of \( V \), but we will not need them here.

3. Decoding a rate 1/2 convolutional code

An \((n, k)\)-convolutional code [2] is a trellis code, which divides a datastream into message blocks of length \( k \) and encodes them into code blocks of length \( n \). We will limit our discussion to binary convolutional codes, in which the datastream is a string of bits. An encoder can be defined by its generator matrix \( G(x) \), a \( k \) by \( n \) matrix of polynomials in \( \mathbb{Z}_2[x] \). If the encoder has a memory of the previous \( m \) blocks, the generator polynomials have degree \( m \) at most. The following diagram represents an encoder for a (2, 1)-convolutional code with \( m = 2 \).

\[
(9)
\]

\[
\begin{array}{c}
\downarrow \quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\end{array}
\]

The datastream is split into single bits and enters the encoder at the symbol \( d \). At each discrete timestep the bit at \( d \) is shifted right to the memory cell labeled \( 1 \), and the previous contents of \( 1 \) are shifted into \( 2 \). The contents of \( 2 \) are then discarded. The outputs \( O_1 \) and \( O_2 \) are formed by the various sums (in the field \( \mathbb{Z}_2 \)) of memory contents and \( d \). The generator matrix for encoder \((9)\) is

\[
G(x) = [1 + x^2 \quad 1 + x + x^2],
\]

where the power of \( x \) represents the time delay on the bit. Presumably the code words pass through a noisy channel and are received as receive blocks by the decoder. Since the sequence of states (shift register contents) in the encoder can be modeled by a hidden Markov model, the classical Viterbi algorithm can be used to implement the decoder module by tracing the most probable sequence of states for the encoder given the sequence of receive blocks.
The state diagram for the code generated by (9) is

where the states are the boxed memory contents of the shift registers in the encoder. The transition to the next state is an arrow labeled by \(i/o_0o_1\), where \(i\) is the message bit and \(o_0o_1\) are the code bits.

To implement the algorithm, we consider a lattice segment representing the reception of the codeword 00:

![Lattice Segment](image)

with arrows representing

- 0 errors \(\rightarrow\), with probability \(E_0\),
- 1 error \(\longrightarrow\), with probability \(E_1\),
- 2 errors \(\Longrightarrow\), with probability \(E_2\),

occurring in the channel. The values \(E_i\), which we will call virtual probabilities (VPs), are only true probabilities in the approximating HMM. Hence they are not directly related to the probability that \(i\) errors have occurred, but need to be chosen so that heuristics such as

\[
E_{i_1} \cdots E_{i_M} \geq E_{j_1} \cdots E_{j_M} \text{ whenever } \sum i_k \leq \sum j_k,
\]

hold as often as possible. The number of situations that (11) hold determines the quality of the approximating HMM, for instance, the number of burst errors that can be correctly decoded. For instance, for a burst length \(M = 2\), the conditions (11) amount to

\[
E_0^2 \geq E_0E_1 \geq E_1^2,
\]

and also

\[
E_0E_1 \geq E_0E_2.
\]

By condition (3) (and see lemma 2), we must have

\[
E_0 + E_2 = 1,
\]
and $E_1 = 1/2$. By (11) we have also $E_0 > 1/2$ and $E_2 < 1/2$. To determine a reasonable value of $E_0$, we consider an $M$-length sub-factor of $p_\pi$ in (6), the output of the above algorithm. In the event (called an event window) that two errors occurred ($M = 2$), the sub-factor will be $E_0E_2$, $E_2E_0$ or $E_1^2$. Notice that once the $E_i$ are chosen for some $M$ and satisfy (11), any $N$-length event of $M$ or fewer errors will also satisfy (11), since if

$$E_{i_1} \cdots E_{i_M} \geq E_{j_1} \cdots E_{j_M} \text{ whenever } \sum i_k \leq \sum j_k,$$

then

$$E_0^{N-M}E_{i_1} \cdots E_{i_M} \geq E_0^{N-M}E_{j_1} \cdots E_{j_M} \text{ whenever } \sum i_k \leq \sum j_k.$$

It would be nice to have $E_1^2 = E_0E_2$, which would give all events of two errors equal measure, but that would force $1/4 = E_0(1 - E_0)$ or $E_0 = 1/2$ which is undesirable.

A property of these approximating HMMs is that burst errors are slightly disfavored, which in this case, it must be that

$$E_1^2 > E_0E_2,$$

where $E_0E_2$ represents a ‘burst’ event of two errors on the same message word. More generally, the tradeoffs between the VPs given to equal-error events is explored in the next section. Since convolutional codes tend to have weak performance subject to close proximity errors, this restriction may not be as inconvenient as it seems. When few errors are anticipated, it is optimal to let $E_0$ approach 1. However, if one were to set $E_0 = 1$, then that would force $E_2 = 0$ and then no paths having error events where 2 errors occurred on a single message block could be returned by the decoding algorithm. The optimal value of $E_0$ will depend on the error source statistics. We will use $E_0 = 0.8$ for the analysis of the algorithm in this section.

We can convert (10) to a unitary operation $V_{00}$ (5) using the representation as was obtained in (11). The operation $V_{00}$ then must map the following:

$$|00\rangle|00\rangle \longrightarrow |00\rangle(\sqrt{E_0}|00\rangle + \sqrt{1 - E_0}|01\rangle),$$
$$|01\rangle|00\rangle \longrightarrow |01\rangle(\sqrt{1 - E_0}|00\rangle + \sqrt{E_0}|01\rangle),$$
$$|10\rangle|00\rangle \longrightarrow |10\rangle(\sqrt{1/2}|01\rangle + \sqrt{1/2}|11\rangle) \text{ and}$$
$$|11\rangle|00\rangle \longrightarrow |11\rangle(\sqrt{1/2}|11\rangle + \sqrt{1/2}|11\rangle),$$

which can be implemented (up to an unimportant negative factor) by the following quantum circuit:

(12)
using the definition $\text{(7)}$. We rewrote the first register-controlled block

\begin{align*}
R_y(\sqrt{E_0}) & \quad R_y(\sqrt{1-E_0})
\end{align*}

as

\begin{align*}
R_y(\sqrt{E_0})
\end{align*}

for consistency with the general algorithm and also because reducing the $C^n$ gates is generally a good idea (see section 2.3). Essentially the same operations are implemented in $V_{01}$, $V_{10}$ and $V_{11}$ so we will only state them here:

\begin{align*}
V_{01} &= \quad R_y(\sqrt{1/2}) \quad R_y(\sqrt{E_0})
\end{align*}

$V_{10}$ has the same circuit as $V_{01}$ except $R_y(\sqrt{E_0})$ is replaced by $R_y(\sqrt{E_2})$. Similarly $V_{11}$ has the same circuit as $V_{00}$ except that $R_y(\sqrt{E_0})$ is replaced by $R_y(\sqrt{E_2})$.

3.1. **Number of repeat trials.** The sequence of steps composing the algorithm above is performed in parallel on an ensemble of $r$ quantum systems each containing $Q$ qubits. After performing the measurement step, we have $S$ samples of size $Q^N$. Finding the most probable event is an instance of the multinomial selection problem [12]. We use the following lemma from [7]:

**Lemma 1.** Consider a collection of numbers with sum one. Denote the largest $p$ and the next largest by $p'$. Carry out $r$ runs in the corresponding probability distribution and denote by $\kappa(r)$ the probability that the most frequent outcome is not the most probable (i.e. the $p$-outcome). Then the following limit exists:

\begin{align*}
\lim_{r \to \infty} -\frac{\log \kappa(r)}{r} & = \lambda = \frac{(p-p')^2}{2[p(1-(p-p'))^2 + p'(1+(p-p')^2)]},
\end{align*}

Hence $\kappa(r) \to 0$ as $e^{-\lambda r}$.

When no errors have occurred in the channel, $p = E_0^N$ and $p' = E_1E_0^{N-1}$, so $\lambda = \frac{1}{6-2E_0^N}$. If it is decided, for example, that a decode error probability of $e^{-2} \approx 0.13$ is acceptable, then we have

\begin{align*}
\frac{0.8^N}{6-0.8^N} & = \frac{4}{r}.
\end{align*}
or
\[ r \sim 24(1.25)^N - 4. \]
The classical processing time and space requirements therefore grow exponentially in \( N \), as each \( r \) trials are performed in parallel and then measured giving an array of length \( r \) of strings of length \( N \log Q \). The array is sorted in \( rN \log(Q) \log(rN \log Q) \) time and then the mode is extracted by scanning the array [13].

The QVA seems to be most naturally applied to problems with a large \( Q \) and small \( N \). Indeed the short coherence times for quantum computers [9] would severely limit \( N \). The next section will show how to adapt the QVA to general convolutional codes where \( Q \) can be made arbitrarily large.

3.2. Decoding example. Starting with the encoder (9) and message stream \{00101101111\}, the code stream is
\[ \vec{c}_0 = \{11, 01, 00, 10, 10, 00, 10, 01, 01\}. \]
The output of the above algorithm with input \( \vec{c}_0 \) is, right before measurement,
\[ \sqrt{E_0^5}[02123123] + \sqrt{E_0^5 E_2}[02123121] + \sqrt{E_0^5 E_1 E_2}[0123331] \]
\[ + \sqrt{E_0^5 E_1 E_2}[02123102] + \sqrt{E_0^5 E_1 E_2}[02123100] \]
\[ + \sqrt{E_0^4 E_1^2 E_2}[02123312] + \cdots \]
One can see that the sequence of states \( 0 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3 \), corresponding to the no-error event with virtual probability \( E_0^7 \), is the same sequence of states that the encoder went through. If the encoder actually went through the sequence 0, 2, 1, 2, 3, 1, 2, 1, corresponding to virtual probability \( E_0^6 E_2 \), then we would assume that the channel had two errors on a single code word and no errors elsewhere. As we chose \( E_0 = 0.8 \) above, the largest six probabilities, respectively, are
\[ 0.209, 0.052, 0.032, 0.032, 0.032, 0.020. \]

Now suppose that an error occurred in the channel on the first code word giving the received stream
\[ \vec{c}_1 = \{10, 01, 00, 10, 10, 00, 10, 01, 01\}. \]
Running the algorithm on \( \vec{c}_1 \) leaves the system in the state
\[ \sqrt{E_0^6 E_1}[02123123] + \sqrt{E_0^6 E_1 E_2}[02123121] + \sqrt{E_0^6 E_1^4}[00002331] \]
\[ + \sqrt{E_0^4 E_1^2 E_2}[02123331] + \sqrt{E_0^4 E_1^2 E_2}[02123102] \]
\[ + \sqrt{E_0^4 E_1^2 E_2}[02123100] + \cdots \]
With \( E_0 = 0.8 \) as before, the six largest probabilities are
\[ 0.131, 0.0327, 0.0320, 0.0204, 0.0204, 0.0204. \]
To see how the algorithm performs with two errors on a code word, we can use
\[ \vec{c}_2 = \{00, 01, 00, 10, 10, 00, 10, 01, 01\}, \]
then the system will be in the state
\[ \sqrt{E_0^6 E_2^0 |02123123\rangle} + \sqrt{E_0^4 E_1^3 |0002331\rangle} + \cdots \]
The six largest probabilities are then
\[ 0.053, 0.051, 0.033, 0.032, 0.032, 0.032. \]
The trade-off that was made is apparent here – \( E_2^0 \) \( E_2^0 \) is just barely larger than \( E_1^3 \), and it would take many trials to find the mode in this case. If this sort of error were anticipated, one would set \( E_2 \) to be closer to 0.3 or 0.4 (making \( E_0 \) smaller). When the errors are spread out in the code words, the performance of the algorithm is better. Choose, say,
\[ \vec{c}_3 = \{10, 11, 00, 10, 10, 00, 10, 01, 01\}, \]
with output
\[ \sqrt{E_0^5 E_2^1 |02123123\rangle} + \sqrt{E_0^3 E_1^4 |00231231\rangle} + \cdots \]
The six largest probabilities are then
\[ 0.082, 0.032, 0.032, 0.032, 0.032, 0.020, \]
which will provide for much better performance than the previous scenario, as the difference \( p - p' \) is key.

4. Algorithm for rate 1/n codes

In order to apply the QVA to general convolutional codes one needs the appropriate quantum circuits, and building them is quite easy; the following algorithm details the procedure for laying out a generalized version of the circuit \([12]\).

**Algorithm 1.**

For each receive block \( r \in Q \), (classical code space requirement)

For each state \( s \in Q \),

For each message \( d \in \mathbb{Z}_2^k \) (2\(^k\) = fanout)

Make link \( l : s \rightarrow \text{rsh}(d|s) \) of strength \( E(l, r) \), see \([4]\). Where \( \text{rsh}(d|s) \) performs a right shift out \( k \) bits on \( d \) and then concatenates in the bits \( s \).

Finding \( E(l, r) \) is not straightforward, however, so for simplicity of argument, we will assume that \( k = 1 \) in this section, hence codes of rate 1/n. In this case, the transition probabilities \( E(l, r) \) are determined by the number of errors \( e \) that must have occurred for a transition from state \( l \) to \( \text{rsh}(d|s) \) given received code \( r \). Will will call these virtual probabilities (VPs), as introduced in the previous section. That is, \( E_e \) is the VP associated to a transition caused by an event with \( e \) errors. The following lemma gives a condition that the \( E_e \) must satisfy.
Lemma 2. Let a binary convolutional code have generator matrix $G(x)$ with $k = 1$. Let $E_0, \ldots, E_n$ be the VPs for the decoding algorithm. If $G_j(0) = 1$ for all $j$, i.e., all nonzero constant terms, then

$$E(i) + E(n - i) = 1,$$

for all $i = 0, \ldots, n$.

Proof. The code block is of length $n$, so the code word can have between zero and $n$ bits flipped in the channel. Presuming the decoder is in state $s$, with a fanout of $F = 2^k = 2$, the decoder expects to transition to either state $s_0$, presuming the input message was 0, with emission $c_0$, or to $s_1$, with input message 1 and emission $c_1$. If the received code is in actuality $c_r \in Q^n$, then we have $H(c_0, c_r) + H(c_1, c_r) = n$, where $H$ is the Hamming distance. This follows from the fact that the mod 2 sum of $c_0 + c_1 = 111 \ldots 1$, the string of $n$ 1-bits. This, in turn holds because

$$c_i = (G_0(s) + i, G_1(s) + i, \cdots G_n(s) + i),$$

where $i$ is the message bit. Hence the result of the lemma since the VP $E_e$ is defined on the number of errors $e$ in the transition along with application of condition (3). $\square$

The maximum free distance codes as tabulated in [2] all have the property that satisfy lemma 2.

Recalling the heuristic (11) for a sequence of transitions with specified errors,

$$E_{i_1} \cdots E_{i_M} \geq E_{j_1} \cdots E_{j_M} \text{ whenever } \sum i_k \leq \sum j_k.$$ 

To give an example with $n = 4$, we have, in particular,

$$E_0 > E_1 > E_2 > E_3 > E_4.$$ 

The following partial order must also hold:

$$E_4^3 E_0^3$$

(13)

$$E_0 > E_1$$

$$E_0 > E_2$$

$$E_0 > E_3$$

$$E_0 > E_1$$

$$E_0 > E_2$$

$$E_0 > E_3$$

$$E_0 > E_1$$

$$E_0 > E_2$$

$$E_0 > E_3$$

$$E_0 > E_4$$
where the arrow $A \rightarrow B$ denotes $A > B$. The row $i$ in the above diagram consists of elements with the $r = 4$ factors derived from the integer partitions of $i$. These rules reduce (in fact, recursively on $n$) to the partial orders:

\[
\begin{array}{c}
E_0 \\
\downarrow \\
E_1 \\
\downarrow \\
E_2 \quad E_1^2 \quad E_0^2 E_2 \\
\downarrow \quad \downarrow \quad \downarrow \\
E_3 \quad E_0 E_3 \quad E_1^3 \quad E_1 E_2 \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
E_4 \quad E_2^2 \quad E_0 E_2^3 \quad E_0^2 E_4 \quad E_0 E_4
\end{array}
\]

Choosing the $E_i$ at this point is an art and not a science. For example, an admissible choice of the $E_i$ that satisfy (14) is $e_1 = \varphi - 1 \approx 0.618$ ($\varphi$ being the golden ratio). Since by lemma (2) $E_1 + E_3 = 1$, $E_3 = 2 - \varphi \approx 0.382$ (which satisfies the optional constraint $E_1 E_2 = E_3$). It is reasonable to mandate that $E_1^2 = E_0 E_2$ and hence we get $E_0 = 2(1 - \varphi)^2 \approx 0.764$. Alternatively, one can choose $E_3 = 1/\sqrt{8}$ as a starting point. Certainly the choice of VPs will affect the 'tuning' of the algorithm to the particular error source, but it will be the topic of a future paper to further discuss the theory of choosing VPs.

5. Discussion

One might attempt to run the QVA on a classical probabilistic computer. The main difficulty here would be storing the superposition of states which represent the possible paths in an efficient way. This suggests that the power of the QVA over the classical algorithm lies in the exploitation of quantum parallelism.

The error correction behavior for the QVA differs from the classical Viterbi algorithm somewhat, especially for errors which occur on the same code-word or in bursts. However, since convolutional codes are limited in the burst length that they correct, we feel this is not a major limitation. For example, the VPs might be chosen as above so that the Hasse diagram (13) is satisfied. Events with more than 4 errors can be handled correctly, as long as there is a no-error pause in between. For example, $E_4 E_0 E_4$ is a event with 8 errors that has the correct VP.

The algorithm explained in this paper seems to be best for short code sequences (small $N$) of large state spaces (large $Q$). Other applications that lie in $C$ would be worth exploring. One might define $C$ more rigorously, with a parameter to describe the allowed error in approximation to the HMM.
REFERENCES

[1] Alexander Barg and Shiyu Zhou. A quantum decoding algorithm of the simplex code. Proceedings of the 36th Annual Allerton Conference, 1998.
[2] Richard E. Blahut. Algebraic Codes for Data Transmission. Cambridge University Press, 2003.
[3] H. F. Chau. Quantum convolutional error-correcting codes. Phys. Rev. A., 58, 1998.
[4] D. Coppersmith. An approximate fourier transform useful in quantum factoring. IBM Research Report RC 19642, 1994.
[5] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. Proc. R. Soc. Lond. A, 400(97), 1985.
[6] G.D. Jr. Forney. The viterbi algorithm. Proceedings of the IEEE, 61(3):268–278, 1973.
[7] Robert Geroch. Perspectives in Computation, chapter 13, pages 100–101. The University of Chicago press, 2009.
[8] Ashley Montanaro. The quantum query complexity of learning multilinear polynomials. Information Processing Letters, 112(11):438–442, 2012.
[9] Michael A. Nielsen and Issac I. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.
[10] H. Ollivier and J.-P. Tillich. Description of a quantum convolutional code. Phys. Rev. Lett., 91(177902), 2003.
[11] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proc. of the IEEE, 77(2), 1989.
[12] James Ramey and Khursheed Alam. A sequential procedure for selecting the most probable multinomial event. Biometrika, 66(1):171–173, 1979.
[13] Steven Skiena. The Algorithm Design Manual. Springer, 2008.
[14] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. IEEE Transactions on Information Theory, 13(2):260–269, 1967.

E-mail address: jgrice, dmeyer@math.ucsd.edu