8

Evolutionary Computation in Coded Communications: An Implementation of Viterbi Algorithm

Jamal S. Rahhal, Dia I. Abu-Al-Nadi and Mohammed Hawa
Electrical Engineering Dept.
The University of Jordan
Amman
Jordan

1. Introduction

Quantum Computing hopefully is the future of computing systems. It still on its first steps. The development of some quantum algorithms gives the quantum computing a boost on its importance. These algorithms (such as Shor's and Grover's algorithms) proved to have superior performance over classical algorithms [1-4]. The recent findings, that quantum error correction can be used, showed that the decoherence problem can be solved and hence the quantum computers can be realized [5-7]. The quantum algorithms are based on the use of special gates applied on one, two or more qubits (quantum bits). The classical computer uses different gates (NOT, AND, NAND, OR and XOR). Quantum gates are in many aspects different from classical gates where all gates must be reversible. This makes the quantum gates act as 2^nx2^n transformation operators, where we have n input qubits and n output qubits.

To understand the quantum bits and gates we describe the group of amplitudes that describes the state of a quantum register as a vector. A qubit with state $|0\rangle$, which is guaranteed to read logic **0** when measured, is represented by the vector $\begin{pmatrix}1\\0\end{pmatrix}$, and a qubit with state $|1\rangle$ which is guaranteed to read logic **1** when measured is represented by the vector $\begin{pmatrix}0\\1\end{pmatrix}$. An arbitrary qubit state is then represented by the vector $\begin{pmatrix}\alpha\\\beta\end{pmatrix}$ as:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$.

One important quantum gate is the Hadamard gate given by:

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

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

When the input is $|0\rangle$, Hadamard gate changes the state of the qubit to:

$$|\varphi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{3}$$

that is, $|\varphi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. So when reading the qubit at the end, we have exactly 50% chance of seeing a **0**, and an equal chance of seeing a **1**. Generalizing the above example, if an *n*-qubits register originally contains the value $|0^n\rangle$, it can be transformed using the Hadamard gate to the superpositional state:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{4}$$

where we would see each of the 2^n binary numbers x with equal probability when we observe the register. Other gates operate similar to Hadmard gate with different matrices, where Pauli gates are given by:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5}$$

and phase gates:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{j} \end{bmatrix} \quad U = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{j}\theta} \end{bmatrix} \tag{6}$$

The quantum computers use quantum gates to produce results in a faster and more efficient way than the classical computers. Implementation of quantum computers is still in its very beginning state, therefore, in this chapter, we need not to worry about the implementation issues. In addition to entanglement, the strength of quantum computing comes from the parallelism feature of the quantum computers and the fact that the quantum state is a superposition state. Using classical bits an n bit vector can represent one of 2^n symbols, but in quantum bits an n qubits vector can represent the 2^n symbols simultaneously.

Quantum Algorithms are usually organized in the following three steps:

Step 1. Initialize the quantum states.

Step 2. Apply the oracle quantum core as many times as needed.

Step 3. Measure the output states (results).

In many classical algorithms especially those used for searching mechanisms, speed and complexity are the main limiting factors in their implementation. Viterbi decoding algorithm is an important algorithm that is used to decode the received data when using Convolutional or Turbo codes at the transmitter. These codes are superior in their performance over many other types of codes. In the following we devise a quantum algorithm to implement the Viterbi algorithm (VA) [8-15].

2. Coded communication and quantum computation

Coded communication uses one type of channel coding that introduces a redundancy in the transmitted data. At the receiver different techniques are used to detect the transmitted information by correcting errors if occurred. All these techniques can be replaced by exhaustive search for the maximum likely data that is assumed to be the correct one.
Several quantum algorithms have been designed to perform classical algorithms with remarkable speedups. In adiabatic algorithms, the solution of the problem is encoded to the problem Hamiltonian. Since the mechanics of the Oracle remains unknown, the encoding process of the Hamiltonian in the adiabatic algorithm is unclear. Instead, just like Grover's algorithm did, the adiabatic search algorithm forms the Hamiltonian directly from the solution state, which means we have to know the state in prior and then perform an algorithm to show it.
Like many quantum computer algorithms, Grover's algorithm is probabilistic in the sense that it gives the correct answer with high probability. The probability of failure can be decreased by repeating the algorithm. Grover's algorithm can be used for estimating the mean and median of a set of numbers. In addition, it can be used to solve NP-complete problems by performing exhaustive searches over the set of possible solutions. This, while requiring prohibitive space for large input, would result in a considerable speed-up over classical algorithms.

3. The classical Viterbi algorithm

In classical communication systems a channel error correcting codes is used to detect and/or correct errors introduced to the data while travelling in a noisy channel. One important class of these codes is the Convolutional Code, where the data is convolved with the code generating polynomials prior to transmitting such data to the receiver. At the receiver, a decoding mechanism is used to recover the transmitted data, and detect/correct errors if they happen. An optimal decoding algorithm was used for this class of codes introduced by Viterbi [10,11]. This algorithm solves the searching problem in the trellis to obtain the maximum likelihood path that best represents the transmitted data. This search grows rapidly with the size of the tree and the length of the data frame. Faster search algorithms will speed up the overall speed of the decoding algorithm. Quantum search algorithm introduced by Grover suggested an optimal searching method that can be used in Viterbi algorithm to speed its execution.
Viterbi decoder uses a tree search procedure to optimally detect the received sequence of data. It performs maximum likelihood (ML) decoding. It calculates a measure of similarity between the received signal and all the trellis paths entering each state at time. Remove all the candidates that are not possible based on the maximum likelihood choice. When two paths enter the same state, the one with the best path metrics (the sum of the distance of all branches) along the path is chosen. This path is called the *surviving path*. This selection of surviving paths is done for all the states and makes decisions to eliminate some of the least likely paths in early calculation stages to reduce the decoding complexity.
The Viterbi algorithm (VA) calculates the branch metrics at each signalling interval and searches for the minimum branch metric. It searches for the maximum possible correct branch out of all possible branches. The total number of searches for each path (for L signalling intervals) is given by [8,9]:

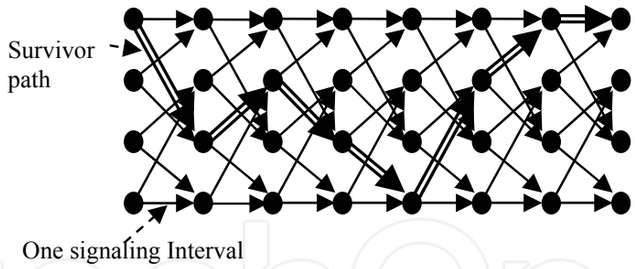

Fig. 1. Trellis Diagram Showing the Survivor Path.

$$N_T = L\,2^{k+m} \tag{7}$$

where 2^k is the number of possible branches from each state and 2^m is the number of trellis states. Note that, these searchs must be conducted even in the fastest classical implementation to VA. It is worth mentioning here that the larger the number of states the better the performance of the code (i.e. less Bit Error Rate (BER)) [10, 11]. For large number of states and longer signalling intervals the processing delay become a limiting factor in the implementation of such codes, especially the Turbo-Codes where more computations and searching is required [12]. Hence the use of Quantum search algorithm might be the solution for higher dimensionality codes as it promises in the Encryption field.

We will use an example to explain the Viterbi search technique. In this example, we transmit a sequence of L data bits (say: 1011000) over a noisy channel. Some bits will be corrupted during transmission, just like when you misinterpret a few words when listening to a lousy phone connection or a noisy radio transmission. In such case, instead of receiving the above L-sequence of bits (called hidden states), the receiver might obtain a new erroneous sequence (called the observed sequence).

To overcome such a problem, the transmitter (called convolutional encoder) operates in a state machine pattern, in which it can exist in a finite number of states, and instead of transmitting the above sequence of bits, the transmitter uses that bit sequence to drive it through the state machine. The encoder tells the receiver (the decoder) about its movement through the state machine by transmitting a codeword (a new bit sequence) that is a result of the state machine transitions.

The Viterbi Algorithm at the decoder side operates on that state machine assumption, and even though the transmitted codeword (representing how the encoder went though the different states) might be corrupted, the Viterbi Algorithm examines all possible sequences of states (called paths) to find the one that is the most likely transmitted sequence. In VA terminology, this is called the *survivor path*.

Let us consider the (rate 1/2, m = 3) convolutional code, the state machine of which is shown in Figure 2. The notation (rate n/k, m) is widely used for convolutional codes, where the parameter k represents the number of input bits that control movement in the state machine, the parameter n represents the number of output bits resulting from each state machine transition, and finally the parameter $k(m - 1)$ (called the *constraint length* of the code) is related to the number of states S in the state machine, where $S = 2^m$. In our example code, we have $S = 2^3 = 8$ different states.

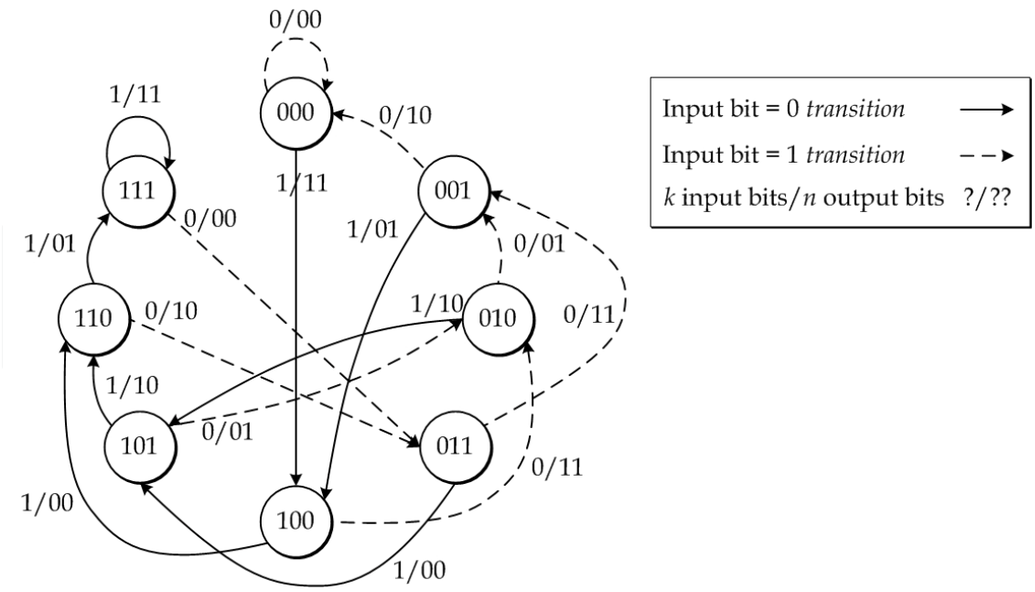

Fig. 2. The state diagram for (rate 1/2, m = 3) convolutional code.

As illustrated in Figure 2, the transmitter can exist in only one state at any given time. The transmitter always starts from state 000. To transmit one bit from the input data sequence, the transmitter looks up the *only* two possible transitions available from the current state. For example, if the transmitter is currently in state 100, and the input data bit is 1, the transmitter follows the solid-line transition to state 110 and sends the codeword 00 on the output channel. However, if the input bit is 0, the transmitter follows the dotted-line transition to state 010 and sends the codeword 11 to the decoder.

The transmitter remains in the new state until it is time to transmit the next bit in the input data sequence. The output codeword for any transition is decided by the numbers on the state diagram which are predetermined by certain generating polynomials. The ones we use in Figure 2 are G1 = [1 1 0 1] and G2 = [1 1 1 0]. Notice that one transition occurs in the state machine for each input bit (k = 1), which results in two bits being transmitted by the encoder over the channel (n = 2), resulting in the code rate of k/n = 1/2. If we use the input data sequence of 1011000 in the above state machine, we get the transmitted codeword of 11111010101110.

One might think that implementing the above state machine using hardware is a complex task, but it is actually very simple since it can be implemented using a shift register of size (K)(k) bits and an n group of XOR gates (to give the n output bits). The shift register and the XOR gates are connected based on the desired generating polynomials. A total of k input bits are shifted into the register each time tick to force the state machine into another state.

Since the transmitted codeword is based on a pattern (transitions in the state machine), the receiver can predict the movement in that machine even if some of the transmitted bits are corrupted by noise in the channel. To do that, the receiver uses the help of a *trellis diagram* as shown in Figure 3.

Figure 3 illustartes how the reciver decodes two incoming bits each time tick to decide which of the two possible transitions (the solid-line or the dotted-line) to be undertaken. Figure 4 shows the trasnitions that occur in the trellis for the transmitted codeword of 11111010101110, which corresponds to the input data sequcen of 1011000.

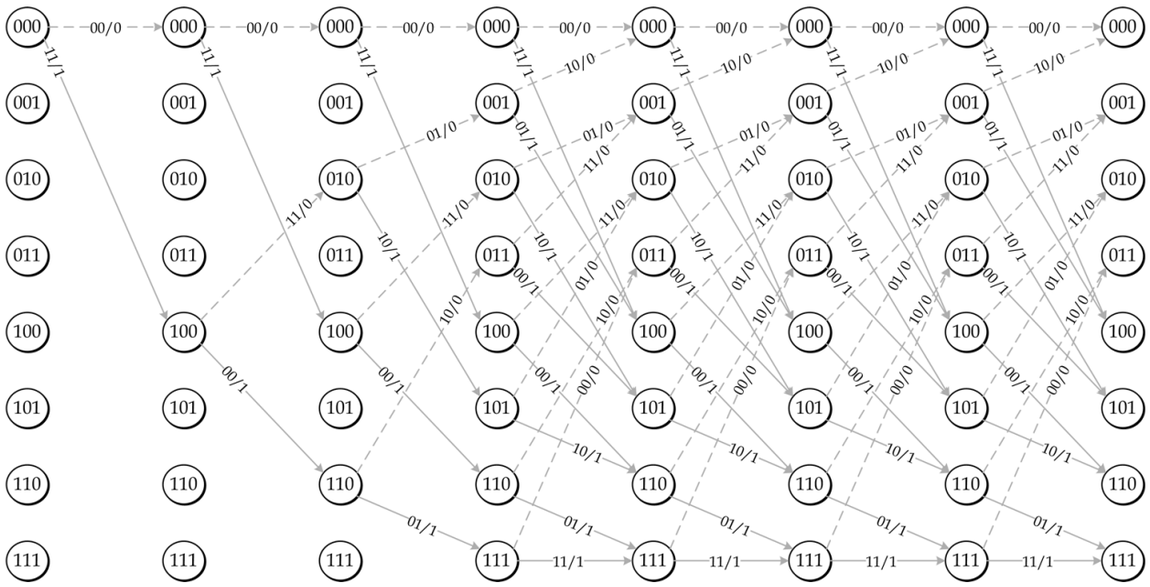

Fig. 3. Decoder trellis diagram for the convolutional code (rate 1/2, m = 3).

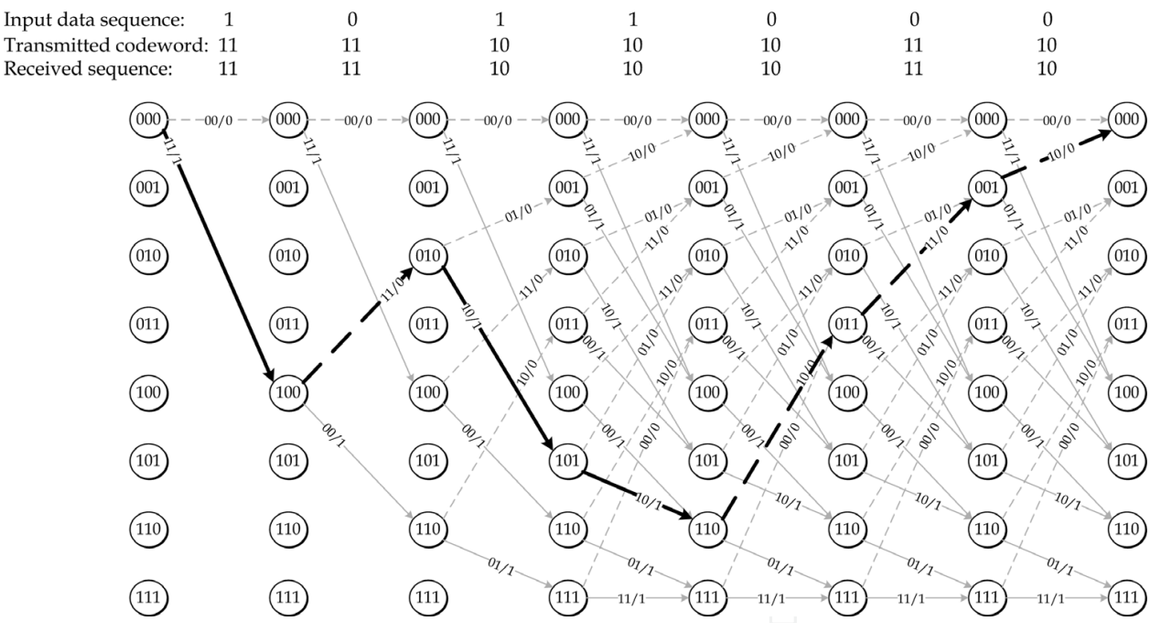

Fig. 4. Decoder trellis diagram showing the path for the input codeword of 11111010101110.

If no errors occurs while transmitting the encoder-generated codeword, the decoder will traverse the trellis without any problems and will be able to read the original data sequence of 1011000. However, due to errors in the channel, the receiver might read a different bit sequence. Since the input data sequence (1011000) is composed of 7 bits, the challenge the decoder faces is to know which of the 2^7 possibilities did the transmitter intend to send.

The idea is that because each 7 bit code will generate a unique trasnmitted codeword, the decoder will search through all the 2^7 possible codewords that can be generated by the state machine to see which one is closest to the received sequence. This is called the *brute-force*

search method, which is computationally expensive. For example, imagine the case of 256-bit data bit sequence, which will force the receiver to comapre the received codeword againsts $2^{256} \approx 1.2 \times 10^{77}$ possibilities.

A more efficient method compared to the brute-force search is the Viterbi algorithm. In such method, and using the trellis shown in Figure 3, we narrow the investigated codewords (called paths) systematically each signalling interval. The algorithm goes like this: As the decoder examines an entire received codeword of a given length, the decoder computes a metric for each possible path in the trellis. The metric is cumulative across the whole path. All paths are followed until two paths converge at a trellis state. Then the path with the higher metric is kept (called the survivor path) and the one with the lower metric is discarded. Since the path with the highest metric is kept, the decoder is classified as a maximum-likelihood receiver.

There are different metrics that can be used to compare the received codeword with different valid codewords, the smiplest of which is the *Hamming distance*, which is the dot product between the received codeword and the original codeword (i.e., the bit agreements between the two bit sequences). The table below shows some exmaples of how to calculate the Hamming distance.

Data Bit Sequence	**Corresponding Codeword**	**Received Sequence (*due to errors*)**	**Bit Agreement (Hamming metric)**
0101111	00 11 11 10 10 01 11	10 11 10 10 10 11 10	10/14
0101100	00 11 11 10 10 10 11	10 11 10 10 10 11 10	10/14
1011000	11 11 10 10 10 11 10	10 11 10 10 10 11 10	13/14
1010110	11 11 10 01 10 10 10	10 11 10 10 10 11 10	10/14
1011011	11 11 10 10 10 00 10	10 11 10 10 10 11 10	11/14

To illustrate the idea of Viterbi search, we show in Figure 5 the case of decoding the codeword 11111010101110 corresponding to the data bit sequcen 1011000. An error occurs in the second bit, thus leading to the received bit sequence of **10**111010101110.

The decoder starts at state 000, just like the encoder. From this point it has two possible paths available, which should be decided by the incoming bits. Unfortunately, the two incoming bits are 01, which do not match 00 or 11. Hence, the decoder computes the path metric (Hamming distance) for both transitions and continues along both of these paths. The metric for both paths is now equal to 1, which means that *only one* of the two bits was matched with the incoming 01 sequence.

As the algorithm progresses while traversing multiple paths, we notice that certain paths start to converge at different states of the trellis (see the fourth, fifth, sixth, … signalling intervals). The metrics are shown in Figure 5. Since maximum likelihood is employed, the decoder discards the path with the lower metric because it is least likely. This discarding of paths at each trellis state helps to reduce the number of paths that have to be examined and gives the Viterbi method its efficiency. The survivor paths are shown in darker color in Figure 5.

Once the whole codeword is traversed through the trellis, the path with the highest metric is chosen as the final path, which is 11111010101110 in our example corresponding to the input data sequence of 1011000.

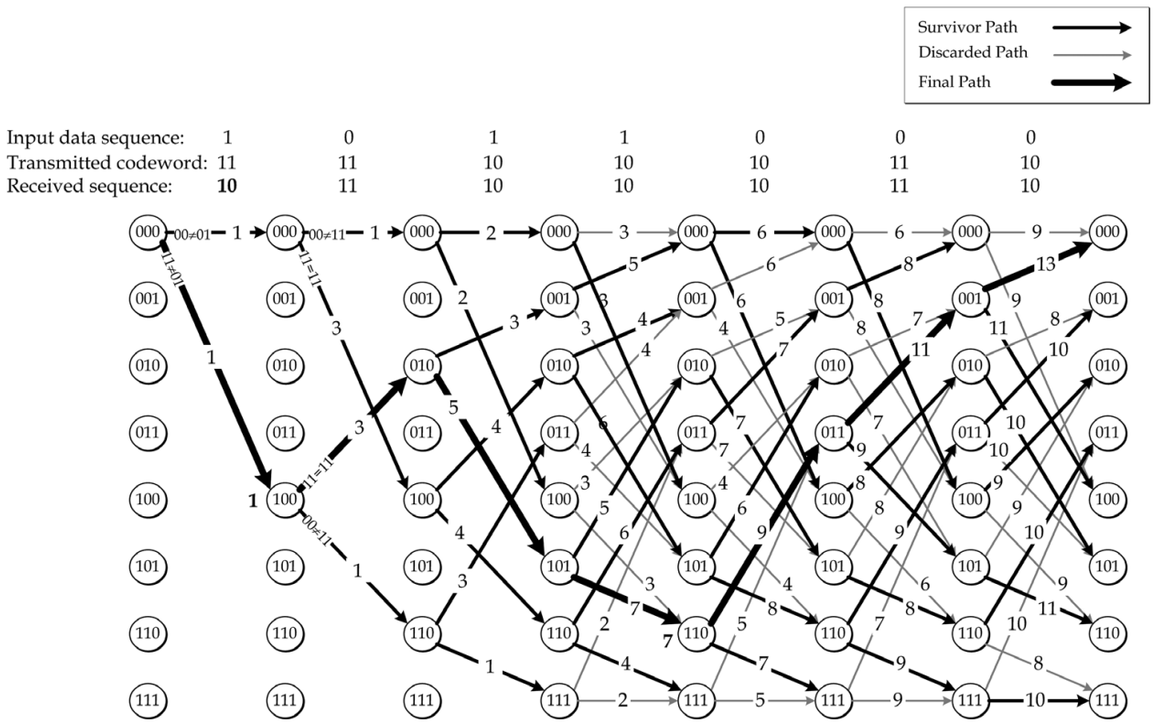

Fig. 5. Decoder trellis diagram showing the path for the input codeword of **10**111010101110.

It is worth mentioning that building the receiver hardware is much more difficult compared to the transmitter hardware. This is because we need to save path history for the algorithm to work. The length of the trellis diagram decides the memory storage requirements for the decoder, which is (in classical hardware) usually reduced by a truncation process. Truncation also reduces latency since decoding need not be delayed until the end of the transmitted codeword. Such truncation can be avoided if a parallel search is done through multiple paths at the same time.

4. The quantum Grover's algorithm

Grover's search algorithm in quantum computing gives an optimal, quadratic speedup in the search for a single object in a large unsorted database [1-4]. It can be generalized in a Hilbert-space framework for both continuous and discrete time cases. As shown in Figure 6, Grover's algorithm initializes the search space by creating an equal superposition of n qubits $\left|\varphi_1^n\right\rangle$ (the superscript denotes n qubits) by applying the Hadmard Transform **H**. Then it applies the function ***f(x)*** (The Oracle) that is equivalent to a gate operator called the **C** gate. This function is given by:

$$f(x)=\begin{cases}0 & \left|\varphi_1\right\rangle \neq \left|x_r\right\rangle \\ 1 & \left|\varphi_1\right\rangle = \left|x_r\right\rangle\end{cases} \tag{8}$$

where $\left|x_r\right\rangle$ is the required search result. The **C** operator will reverse the sign of the state (rotate by π) $\left|\varphi_1\right\rangle$ that most represents the required search result *(this argument will be used*

later in defining the oracle function used in our proposed implementation) and leave all other states unchanged.

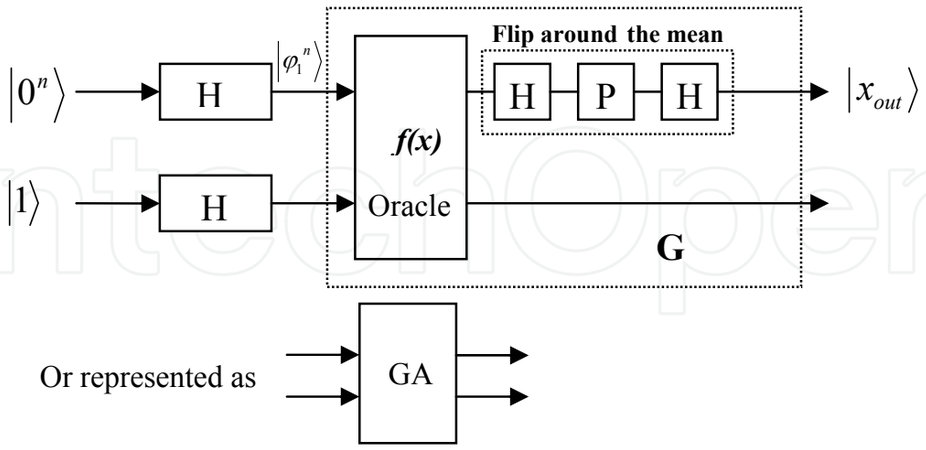

Fig. 6. Block Diagram of One Stage Basic Grover's Algorithm.

After applying **C** the result is flipped around the mean by the **H** and **P** operators. **H** is the Hadamard gate and **P** is the controlled phase shift operator gate, which for arbitrary ϕ_1 and ϕ_2 is given by:

$$P = \begin{bmatrix} e^{j\phi_1} & 0 \\ 0 & e^{j\phi_2} \end{bmatrix} \tag{9}$$

Then in matrix form we can see that:

$$|x_{out}\rangle = U^N |\varphi_1\rangle \tag{10}$$

Where:

$$U^N = H^N P^N H^N C^N \tag{11}$$

The superscript *N* denotes that it is an *N*x*N* operator. Applying the Grover's gate (**G)** *r* times to find the final search result, where:

$$r = \left\lfloor \frac{\cos^{-1}\left(\frac{1}{\sqrt{N_{paths}}}\right)}{2\sin^{-1}\left(\frac{1}{\sqrt{N_{paths}}}\right)} \right\rfloor \tag{12}$$

we see that as $N_{paths} \to \infty$ the searching order $r \to O(\sqrt{N_{paths}})$. Next we discuss the use of Grover's algorithm for implementing Viterbi algorithm.

5. Implementation of VA using Grover's Search Algorithm

Viterbi decoder searches all possible branches at each signalling interval in the code tree. This requires storage of all paths including the survivor path, computation of each branch metric and a decision making to select the survivor path. In quantum theory the components are somehow different, we have a register containing a certain number of qubits, an arbitrary transformation applied on these qubits and a measuring mechanism that measures the state of each particular qubit.

The quantum VA can be viewed as a global search in all the possible paths in the decoding trellis. This means that for a classical Convolutional Code with (*n,k,m)* parameters, *L* signalling symbols and the number of trellis states is 2^m, from equation (7) we see that the total number of possible paths is $N_T = L\,2^{m+k}$ when using the VA. We propose a single search quantum viterbi algorithm (*SSQVA)* and a multi search quantum viterbi algorithm *(MSQVA)*.

6. Single Search Quantum Viterbi Algorithm (SSQVA)

In this algorithm all the N_T paths are searched at once for the closest state to the received signal qubits producing a search order of $O(\sqrt{N_T})$. Then the total number of searches is given by:

$$N_{SSQVA} = \sqrt{N_T} = \sqrt{L}\,2^{\frac{m+k}{2}} \tag{13}$$

This requires large storage (especially for huge number of signalling intervals and number of trellis states). Figure 7 shows the block diagram of the *SSQVA*.

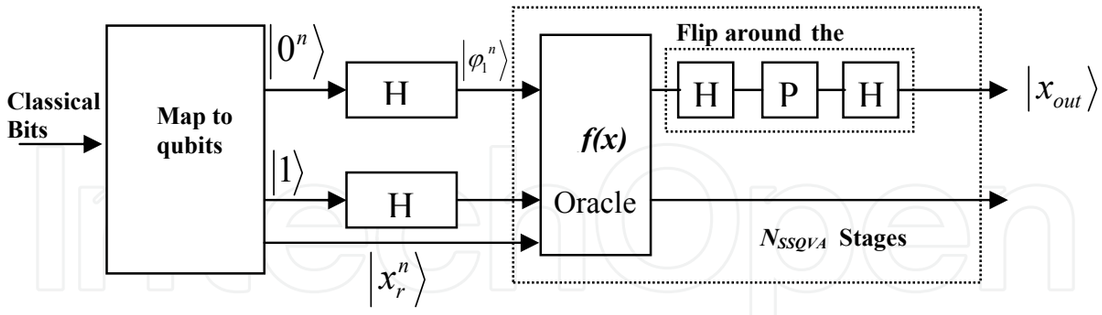

Fig. 7. Block Diagram of Single Search Quantum Viterbi Algorithm (*SSQVA*).

The *SSQVA* can be summarized as:

1. *Convert the received classical data bits into qubits* $|x_r^n\rangle$.
2. *Initialize the Grover's search engine.*
3. *Apply the search on all possible paths.*
4. *Measure the output and map it back to classical bits.*

Following the nature of the classical VA, where multi-stages are implemented, we devise the Multi Search version (MSQVA) as follows:

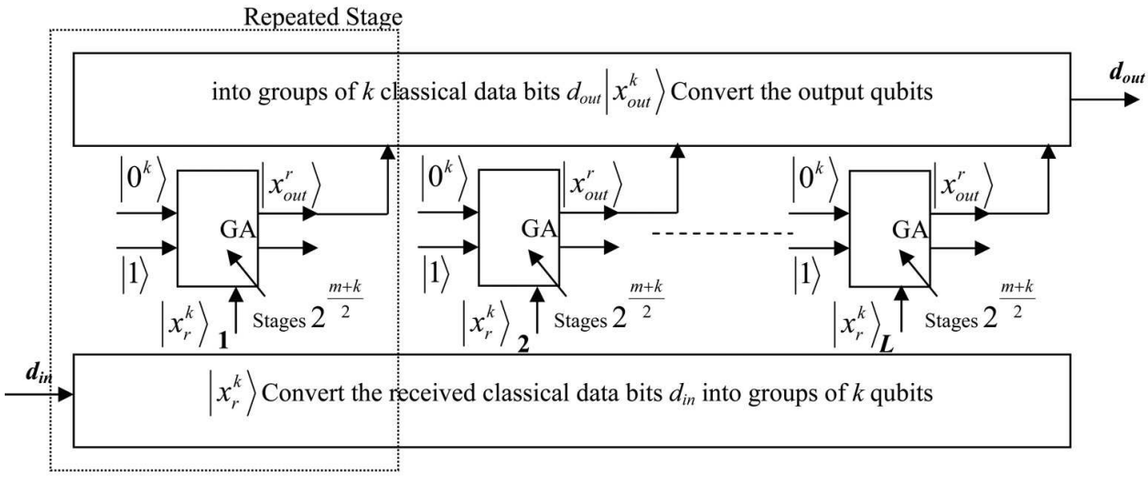

Fig. 8. Block Diagram of Multi Search Quantum Viterbi Algorithm.

7. Multi Search Quantum Viterbi Algorithm (MSQVA)

The N_T paths can be searched for the closest state to the received signal qubits in distributed fashion (Multi Search). For each stage we search only the corresponding tree branches and then combine the searches to the survivor path. This will produce a search order of $O(N_{MSQVA})$. Where the total number of searches is given by:

$$N_{MSQVA} = L\ 2^{\frac{m+k}{2}} \tag{14}$$

This requires less storage but as we can see larger number of searches. Figure 8 shows the block diagram of the *MSQVA*.

The MSQVA can also be implemented iteratively using the same one stage (the repeated stage shown in Figure 8).

The *MSQVA* can be summarized as:

1. *Convert the received classical k data bits into groups of k qubits* $|x_r^k\rangle$.
2. *Initialize the Grover's search engine.*
3. *Apply the search on all possible paths.*
4. *Measure the output and map it back to classical bits.*
5. *Go to step 1.*

The MSQVA version is implementing the VA iteratively as its classical nature; therefore, it has higher searching order. The advantage of the *MSQVA* is that the same hardware setup is used every iteration.

In both *SSQVA* and *MSQVA* the function ***f(x)*** is the black box of the algorithm. It differs according to the application of the VA. For example if VA is used as a Convolutional Code Decoder the function ***f(x)*** will decide in favour of the state that is closest to the received state. This means that the equality might not hold for all the received qubits. Then the function becomes:

$$f(x) = \begin{cases} 0 & |\varphi_1\rangle \neq |x_r\rangle \\ 1 & |\varphi_1\rangle \cong |x_r\rangle \end{cases} \tag{15}$$

Then, the flipping around the mean transformation will amplify the most probable state and hence the result will be the maximum likelihood decision. Note that, ***f(x)*** still has the same implementation as in Grover's algorithm, since Grover's algorithm produces the most probable search result.

The implementation of quantum algorithms is in its first steps; therefore, only theoretical performance evaluation is possible. Figure 9 shows a comparison of the classical VA and the two devised algorithms at different frame lengths and number of states.

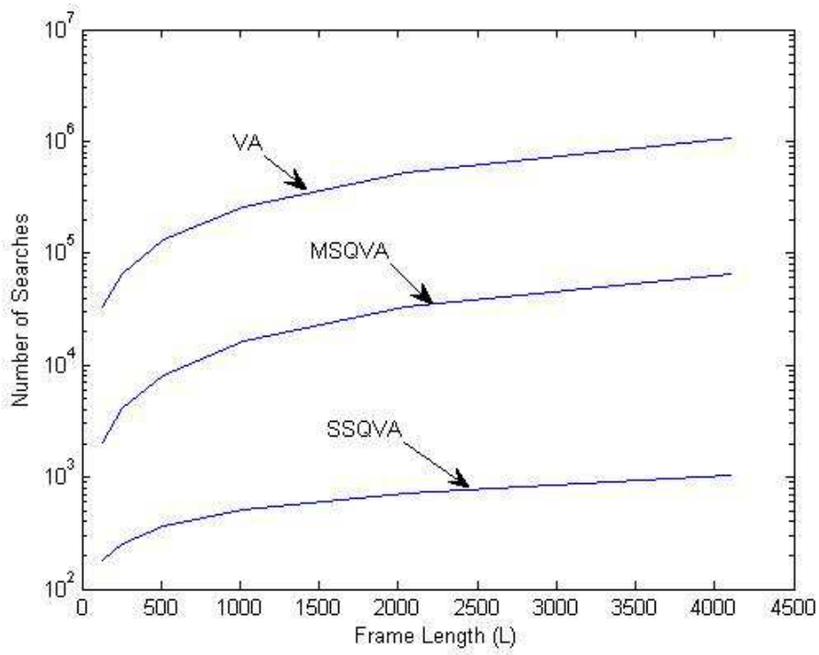

Fig. 9. Search Order as a function of Frame Length (*L*) for the Classical VA, *MSQVA* and *SSQVA* for *m*=4 and *k*=4.

8. Results and conclusions

In this chapter we discussed the use of quantum search algorithm introduced by Grover to speed the execution of Viterbi's algorithm. Two methods were discussed to implement VA using quantum search algorithm: The first is the *SSQVA* where the search domain contains all possible paths in the code tree. The number of paths depends on the signalling length (received data frame length). This domain becomes very large when long frames are used. The *SSQVA* is optimal in the global sense, because it uses all possible solutions and obtains the best one that has the minimum number of differences. The second method is the *MSQVA* where the search is divided into multiple stages just like in the classical algorithm. This method produces a sub optimal solution from speed point of view, since it uses the search algorithm partially at each signalling interval.

9. References

[1] L. K. Grover. "A fast quantum mechanical algorithm for database search." *In the Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*. pp:212–219, May 1996.

[2] L. K. Grover. "Quantum mechanics helps in searching for a needle in a haystack." *Phys. Rev. Lett.* 79(2):325–328, July 1997.

[3] L. K. Grover. "Quantum computers can search rapidly by using almost any transformation." *Phys. Rev. Lett.* 80(19):4329–4332, 1998.

[4] P. W. Shor. "Quantum Computing." *Documenta Mathematica*, Extra Volume ICM 1998 I, pp. 467-480.

[5] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge ; New York : Cambridge University Press, 2000.

[6] J. Gruska, *Quantum Computing*. McGraw Hill, 1999.

[7] Isaac L Chuang, Neil Gershenfeld and Mark Kubinec. "Experimental Implementation of Fast Quantum Searching," *Phys. Rev. Lett.* 80(15):3408–3411, April 1998.

[8] John G. Proakis & Massoud Salehi, "Digital Communications," 5th Edition, McGraw Hill Higher Education, 2008.

[9] Bernard Sklar, "Digital Communications : Fundamentals and Applications", 2nd Edition, Prentice Hall PTR, 2001.

[10] Shu Lin and Daniel J. Costello, Jr. *Error Control Coding, Fundamentals and Applications*. Prentice Hall, 1982.

[10] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[11] G. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[12] L. Hanzo, T.H. Liew, B.L. Yeap, *Turbo Coding, Turbo Equalisation and Space-Time Coding*, John wiley, 2002.

[13] Elton Ballhysa, "A Generalization of the Deutsch-Jozsa Algorithm and the Development of a Quantum Programming Infrastructure." M.S. Thesis, Boğaziçi University, 2004.

[14] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, Daniel A. Spielman. "Exponential algorithmic speedup by a quantum walk." STOC'03, 59-68. 2003.

[15] A. Yu. Kitaev, A. H. Shen, M. N. Vyalyi. "Classical and Quantum Computation." American Mathematical Society, 2002.

Bibliography

Jamal Rahhal received the B.S. degree from the University of Jordan in 1989, the M.S. and Ph.D. from Illinois Institute of Technology, Chicago IL, in 1993, 1996 respectively. He is currently an assistant professor at the University of Jordan. His current research areas including; CDMA\OFDMA cellular communication systems, array processing, quantum computing and optimization.

D. I. Abu-Al-Nadi received BS from Yarmouk University- Jordan, MS from Oklahoma State University-USA, and PhD from the University of Bremen-Germany all in Electrical Engineering, in 1987, 1991, and 1999, respectively. He is currently an associate professor at the University of Jordan. His current research areas are array processing, quantum computing, pattern recognition, and applications of Neural Networks and Fuzzy Systems.

Mohammed Hawa received his B.S. degree from the University of Jordan in 1997, the M.S. from University College London in 1999 and Ph.D. from University of Kansas, USA in 2003. He is currently an Assistant Professor at the University of Jordan. His current research areas include: communication networks, quantum computing and Quality-of-Service.

Evolutionary Computation

Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7
Hard cover, 572 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jamal S. Rahhal, Dia I. Abu-Al-Nadi and Mohammed Hawa (2009). Evolutionary Computation in Coded Communications: an Implementation of Viterbi Algorithm, Evolutionary Computation, Wellington Pinheiro dos Santos (Ed.), ISBN: 978-953-307-008-7, InTech, Available from: http://www.intechopen.com/books/evolutionary-computation/evolutionary-computation-in-coded-communications-an-implementation-of-viterbi-algorithm

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen