Quantum Factor Graphs

Matthew.G.Parker [1]

Inst. for Informatikk, University of Bergen,

5020 Bergen, Norway,

`matthew@ii.uib.no`

`http://www.ii.uib.no/~matthew/MattWeb.html`

Biography: Matthew Parker is currently a Postdoctoral Researcher in the Code Theory Group at the University of Bergen, Norway. Prior to this he was a Postdoctoral Researcher in the Telecommunications Research Group at the University of Bradford, UK. His research interests are Iterative Computation, Sequence Design, Quantum Computation, and Coding Theory.

[1]This work was funded by NFR Project Number 119390/431, and was presented in part at 2nd Int. Symp. on Turbo Codes and Related Topics, Brest, Sept 4-7, 2000

Abstract: *The natural Hilbert Space of quantum particles can implement maximum-likelihood (ML) decoding of classical information. The 'Quantum Product Algorithm' (QPA) is computed on a Factor Graph, where function nodes are unitary matrix operations followed by appropriate quantum measurement. QPA is like the Sum-Product Algorithm (SPA), but without summary, giving optimal decode with exponentially finer detail than achievable using SPA. Graph cycles have no effect on QPA performance. QPA must be repeated a number of times before successful and the ML codeword is obtained only after repeated quantum 'experiments'. ML amplification improves decoding accuracy, and Distributed QPA facilitates successful evolution.*

Keywords: Factor Graphs, Quantum Computation, Quantum Algorithms, Sum Product Algorithm, Graph Algorithms

1 Introduction

Recent interest in Turbo Codes [2] and Low Density Parity Check Codes [4, 6] has fuelled development of Factor Graphs and associated Sum-Product Algorithm [5, 1] (SPA), with applications to error-correction, signal processing, statistics, neural networks, and system theory. Meanwhile the possibility of Quantum Computing has sparked much interest [9, 10], and Quantum Bayesian Nets have been proposed to help analyse and design Quantum Computers [12, 11]. This paper links these areas of research, showing that quantum resources can achieve maximum-likelihood (ML) decoding of classical information. The natural Hilbert Space of a quantum particle encodes a probability vector, and the joint-state of quantum particles realises the 'products' associated with SPA. SPA summary is omitted as quantum bits (qubits) naturally encode the total joint-probability state. Dependencies between vector indices become 'entanglement' in quantum space, with the Factor Graph defining dependency (entanglement) between qubits. Graph function nodes are implemented as unitary matrix [2] -vector products followed by quantum measurement. This is the Quantum Product Algorithm (QPA). As QPA avoids summary it avoids problems encountered by SPA on graphs with short cycles. Moreover, whereas SPA is iterative, using message-passing and activating each node more than once, QPA does not iterate but must successfully activate each node only once. However the (severe) drawbacks with QPA are as follows: 1) Each function node must be repeatedly activated until it successfully 'prepares' it's local variable nodes (qubits) in the correct entangled state - any activation failure destroys evolution in all variable nodes already entangled with local variables. 2) Once a complete Factor Graph has successfully evolved, final quantum measurement only delivers the ML codeword with a certain (largest) probability. Repeated successful evolutions then determine the ML codeword to within any degree of confidence. This second drawback can be overcome by suitable "ML Amplification" of QPA output, prior to measurement.

Section 2 presents QPA, highlighting its ability to deliver the optimal output joint-state, unlike SPA. Quantum systems describe the exact joint-state by appropriate 'entanglement' with and measurement of ancillary qubits. Section 3 considers a simple example of QPA on Quantum Factor Graphs, showing that iteration on graphs with cycles is unnecessary because QPA avoids premature summary. Section 4 shows how to amplify the likelihood of measuring the ML codeword from QPA output. Unfortunately QPA must be repeated many times and/or executed in parallel to have a hope of successful completion. Suitable distributed QPA scheduling is discussed in Section 5, and it is argued that successful QPA completion is conceivable using

[2] 'Unitary' means that $\mathbf{U}$ satisfies $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$, where † means 'conjugate transpose'.

asynchronous distributed processing on many-node Factor Graphs. This paper does not deal with phase properties of quantum computers. It is expected that the inclusion of phase and non-diagonal unitary matrices will greatly increase functionality of the Quantum Factor Graph.

The aim of this paper is not to propose an immediately realisable implementation of a quantum computer. Rather, it is to highlight similarities between graphs for classical message-passing, and graphs that 'factor' quantum computation. The paper also highlights the differences between the two graphs: whereas classical graphs can only ever compute over a tensor product space, the quantum graph can compute over the complete entangled (tensor-irreducible) space.

2 The Quantum Product Algorithm (QPA)

2.1 Preliminaries

Consider the Factor Graph of Fig 1.

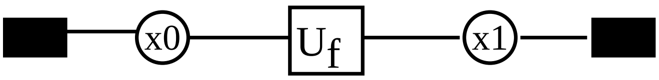

Figure 1: Two-Qubit Factor Graph

Let $\mathbf{U_f} = \begin{pmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_1 & 0 & 0 \\ 0 & 0 & f_2 & 0 \\ 0 & 0 & 0 & f_3 \end{pmatrix}$, and $\mathbf{U_g} = \begin{pmatrix} g_0 & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_3 \end{pmatrix}$, where $|g_k|^2 = 1 - |f_k|^2$, and $f_k^* g_k + f_k g_k^* = 0$, $\forall k$. '$*$' means complex conjugate.

Let $\mathbf{U_{fg}} = \begin{pmatrix} \mathbf{U_f} & \mathbf{U_g} \\ \mathbf{U_g} & \mathbf{U_f} \end{pmatrix}$. $\mathbf{U_{fg}}$ is unitary, and the $\mathbf{U_f}$ of Fig 1 and subsequent figures always implies the action of $\mathbf{U_{fg}}$ together with the measurement of an ancillary qubit, z, as described below. A qubit, x_i, can be in states 0 or 1 or in a statistical superposition of 0 and 1. Let qubits x_0, x_1 be initialised (by the black boxes) to states $x_0 = (\alpha_0, \beta_0)^T$ and $x_1 = (\alpha_1, \beta_1)^T$, where α_i, β_i are complex probabilities such that $|\alpha_i|^2 + |\beta_i|^2 = 1$. For instance, x_0 is in states 0 and 1 with probabilities $|\alpha_0|^2$ and $|\beta_0|^2$, respectively. Let an ancillary qubit, z, be initialised to state 0, i.e. $z = (1, 0)$. Then the initial joint probability product-state of qubits x_0, x_1, z is $\mathbf{A} = (\alpha_0, \beta_0)^T \otimes (\alpha_1, \beta_1)^T \otimes (1, 0)^T = (\alpha_0\alpha_1, \beta_0\alpha_1, \alpha_0\beta_1, \beta_0\beta_1, 0, 0, 0, 0)^T = (s_0, s_1, s_2, s_3, 0, 0, 0, 0)^T$, where $|s_0|^2 + |s_1|^2 + |s_2|^2 + |s_3|^2 = 1$, and '$\otimes$' is the tensor product. The element at vector index v is the probability that the qubits are in state v. For instance, qubits $x_0 x_1 z$ are in joint-state 010 with probability $|s_2|^2$. Subsequent measurement of a subset of the qubits projects the measured qubits to a fixed substate with a certain probability, and 'summarises' the vector for the remaining non-measured qubits. Thus QPA is as follows,

- Compute $\mathbf{S} = \mathbf{U_{fg}A}$.

- Measure qubit z. With probability $p_f = |s_0 f_0|^2 + |s_1 f_1|^2 + |s_2 f_2|^2 + |s_3 f_3|^2$ we collapse z to 0, and x_0, x_1 to joint-state $\mathbf{S_f} = \mu_0(s_0 f_0, s_1 f_1, s_2 f_2, s_3 f_3)^T$. With probability $p_g = |s_0 g_0|^2 + |s_1 g_1|^2 + |s_2 g_2|^2 + |s_3 g_3|^2$ we collapse z to 1, and x_0, x_1 to joint-state $\mathbf{S_g} = \mu_1(s_0 g_0, s_1 g_1, s_2 g_2, s_3 g_3)^T$. μ_0 and μ_1 are normalisation constants. $p_f + p_g = 1$. $\mathbf{S_f}$ is our desired QPA result. Successful QPA completion is self-verified when we measure $z = 0$.

In contrast, classical SPA computes $\mathbf{S_f} = \mathbf{U_f A}$ (with probability 1) and must then perform a subsequent 'summary' step on $\mathbf{S_f}$ before returning a result for each variable separately. This result is,

$x_0 = |\mu_0|^2(|s_0 f_0|^2 + |s_2 f_2|^2, |s_1 f_1|^2 + |s_3 f_3|^2)^T$, $x_1 = |\mu_1|^2(|s_0 f_0|^2 + |s_1 f_1|^2, |s_2 f_2|^2 + |s_3 f_3|^2)^T$. For instance, for $x_0 = 0$ we sum the two **classical** [3] probabilities of $\mathbf{S_f}$ where $x_0 = 0$ to get $|s_0 f_0|^2 + |s_2 f_2|^2$. Similarly, for $x_0 = 1$ we summarise to $|s_1 f_1|^2 + |s_3 f_3|^2$. It is in this sense that SPA is a 'tensor-approximation' of QPA.

We identify the following successively accurate computational scenarios (decoding modes) for a space of N binary-state variables:

- **Hard-Decision** operates on a probability space,
  $(\alpha_0, \beta_0) \otimes (\alpha_1, \beta_1) \otimes \ldots \otimes (\alpha_{N-1}, \beta_{N-1}), \qquad \alpha, \beta \in \{0, 1\}$

- **Soft-Decision** operates on a probability space,
  $(\alpha_0, \beta_0) \otimes (\alpha_1, \beta_1) \otimes \ldots \otimes (\alpha_{N-1}, \beta_{N-1}), \qquad \alpha, \beta \in \{\text{Real Numbers } 0 \rightarrow 1\}$

- **Quantum Soft-Decision** operates on a probability space,
  $(\alpha_0, \beta_0) \otimes (\alpha_1, \beta_1) \otimes \ldots \otimes (\alpha_{N-1}, \beta_{N-1}), \qquad \alpha, \beta \in \{\text{Complex Numbers}\}$

- **Entangled-Decision** operates on a probability space,
  $(s_0, s_1, s_2, \ldots, s_{2^N-1})$, $s \in \{\text{Complex Numbers}\}$

All four of the above Decision modes satisfies the probability restriction that the sum of the magnitude-squareds of the vector elements is 1. Both Quantum Soft-Decision and Entangled-Decision make use of the natural quantum statistical properties of matter, including the property of Superposition. Moreover, Entangled-Decision operates over exponentially larger space.

[3] Classical SPA probabilities in this paper are always represented as the magnitude-squared of their quantum counterparts

Classical SPA operates in Soft-Decision mode. QPA operates in Entangled-Decision mode. In the previous discussion it was assumed that the QPA was operating on input of the form, $(\alpha_0, \beta_0)^T \otimes (\alpha_1, \beta_1)^T \otimes (1, 0)^T$. More generally, QPA can operate on input and deliver output in Entangled-Decision mode. This is in strong contrast to SPA which must summarise both input and output down to Soft-Decision mode. It is this approximation that forces SPA to iterate and to sometimes fail on graphs with cycles.

Consider the following example. If the diagonal of $\mathbf{U_f}$ is $(1, 0, 0, 1)$, then $\mathbf{U_f}$ represents XOR, and Fig 1 decodes to codeset $\mathbf{C} = \{00, 11\}$ (i.e. $x_0 + x_1 = 0$, mod 2). $\mathbf{C}$ has distance 2, which is optimal for length 2 binary codes: in general if $\mathbf{U_f}$ cannot be tensor-decomposed then it represents a code $\mathbf{C}$ with good distance properties. Initially, let $x_0 = (\sqrt{0.4}, \sqrt{0.6})^T$, $x_1 = (\sqrt{0.6}, \sqrt{0.4})^T$. Then $\mathbf{A} = (\sqrt{.24}, 0.6, 0.4, \sqrt{0.24}, 0, 0, 0, 0)^T$, and $\mathbf{S_f} = \frac{1}{\sqrt{2}}(1, 0, 0, 1)^T$. $p_f = 0.48$, so, on average, 48 $\mathbf{S_f}$ outputs are computed for every 100 QPA attempts. The ML codeword is both 00 and 11, and when $\mathbf{S_f}$ is measured, 00 and 11 are equally likely to be returned. In contrast, classical SPA for the same input returns $x_0 = x_1 = (\frac{1}{2}, \frac{1}{2})$, implying (wrongly) an equally likely decode to any of the words $00, 01, 10, 11$. So even in this simplest example the advantage of QPA over SPA is evident.

2.2 Product Space for Classical SPA

Because x_0 and x_1 are separated in Fig 1, their classical joint-state only represents tensor **product** states (Soft-Decision mode). An equivalent Factor Graph to that of Fig 1 could combine x_0 and x_1 into one quaternary variable which would reach all non-product quaternary states. But this requires 'thickening' of graph communication lines and exponential increase in SPA computational complexity. Consequently only limited variable 'clustering' is desirable, although too little clustering 'thins out' the solution space to an insufficient highly-factored product space. This is the fundamental Factor Graph trade-off - good Factor Graphs achieve efficient SPA by careful variable 'separation', ensuring the joint product space is close enough to the exact (non-summarised) non-product space.

2.3 Entangled Space for QPA

In contrast, although x_0 and x_1 are physically separated in Fig 1, quantum non-locality must take into account correlations between x_0 and x_1. Their joint-state now occurs over the union of product and (much larger) non-product (entangled) space (Entangled-Decision mode). An entangled joint-state vector cannot be tensor-factorised over constituent qubits. QPA does not usually output to product space because the joint-state of output qubits is usually entangled. In fact QPA is algorithmically simpler than SPA, as SPA is a subsequent tensor approximation of QPA output at each local function.

2.4 Example

Let the diagonal of $\mathbf{U_f}$ be $(1,0,0,1)$. Initialise x_0 and x_1 to joint-product-state, $x_0 = \frac{1}{\sqrt{3}}(1,\sqrt{2})^T$, $x_1 = \frac{1}{\sqrt{2}}(1,1)^T$. With probability $p_f = 0.5$ QPA measures $z = 0$ and computes the joint-state of x_0, x_1 as $\mathbf{S_f} = \frac{1}{\sqrt{3}}(1,0,0,\sqrt{2})^T$. A final measurement of qubits x_0 and x_1 yields codewords 11 and 00 with probability $\frac{2}{3}$, and $\frac{1}{3}$, respectively. In contrast SPA summarises $\mathbf{S_f}$ to $x_0 = x_1 = \frac{1}{3}(1,2)$. Although a final 'hard-decision' on x_0 and x_1 chooses, correctly, the ML codeword $x_0 = x_1 = 1$, the joint-product-state output, $\frac{1}{3}(1,2)^T \otimes \frac{1}{3}(1,2)^T = \frac{1}{9}(1,2,2,4)^T$ assigns, incorrectly, a non-zero probability to words 01 and 10.

2.5 A Priori Initialisation

To initialise x_0 to $(\alpha_0,\beta_0)^T$, we again use QPA. Let the diagonal of $\mathbf{U_f}$ (for the left-hand black box of Fig 1) be (α_0,β_0). Then the diagonal of $\mathbf{U_g}$ is $\pm i(\frac{\alpha_0\sqrt{1-|\alpha_0|^2}}{|\alpha_0|}, \frac{\beta_0\sqrt{1-|\beta_0|^2}}{|\beta_0|})^T$. Measurement of $z = 0$ initialises x_0 to $(\alpha_0,\beta_0)^T$, and this occurs with probability $p_f = 0.5$. x_1 is initialised likewise.

2.6 Comments

The major drawback of QPA is the significant probability of QPA failure, occurring when z is measured as 1. This problem is amplified for larger Quantum Factor Graphs where a different z is measured at each local function; **QPA evolution failure at a function node not only destroys the states of variables connected with that function, but also destroys all states of variables entangled with those variables.** QPA is more likely to succeed when input variable probabilities are already skewed somewhat towards a valid codeword. Section 3 shows how QPA can operate successfully even when SPA fails.

3 Quantum Product Algorithm on Factor Graphs with Cycles

This section shows that graph cycles do not compromise QPA performance. Consider the Factor Graph of Fig 2.

Functions $\mathbf{U_{f0}}$ and $\mathbf{U_{f1}}$ are both 8×8 XOR diagonal matrices with diagonal elements (10010110). Acting on the combined four-qubit space, $x_0x_1x_2x_3$, they are the functions $\mathbf{U_{f0}} \otimes \mathbf{I_2}$

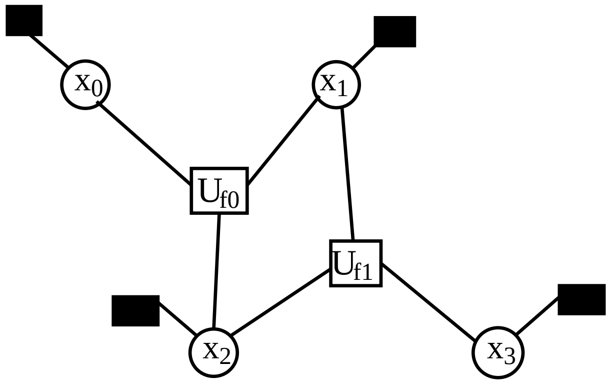

Figure 2: Factor Graph with a Cycle

and $\mathbf{I_2} \otimes \mathbf{U_{f1}}$, respectively, with diagonal elements (1001011010010110) and (1100001100111100), respectively, where $\mathbf{I_2}$ is the 2×2 identity matrix. QPA on Fig 2 performs the global function $\mathbf{U_F} = (\mathbf{U_{f0}} \otimes \mathbf{I_2})(\mathbf{I_2} \otimes \mathbf{U_{f1}})$ on four-qubit space, with diagonal elements (1000001000010100), forcing output into codeset $\mathbf{C} = \{0000, 0110, 1011, 1101\}$. Functions $\mathbf{U_{f0}}$, $\mathbf{U_{f1}}$, and $\mathbf{U_F}$ 'sieve' the input joint-state, where $\mathbf{U_F}$ is the combination of two 'sub-sieves', $\mathbf{U_{f0}}$ and $\mathbf{U_{f1}}$. QPA iteration (i.e. successfully completing a sub-function more than once on the same qubits) has no purpose, as only one needs apply a particular sieve once. So graph cycles have no bearing on QPA. (However iteration may be useful to maintain the entangled result in the presence of quantum decoherence and noise). To underline cycle-independence, consider the action of SPA, then QPA on Fig 2.

Initialise as follows (using classical probabilities),

$$x_0x_1x_2x_3 = (0.1, 0.9)^T \otimes (0.6, 0.4)^T \otimes (0.6, 0.4)^T \otimes (0.6, 0.4)^T$$

Hard-decision gives $x_0x_1x_2x_3 = 1000$, which can then be decoded algebraically to codeword 0000. However optimal soft-decision would decode to either $x_3x_2x_1x_0 = 1011$ or 1101, with equal probability. Because of the small graph cycle SPA fails to decode correctly, and settles to the joint-product-state,
$x_0x_1x_2x_3 = (0.108, 0.892)^T \otimes (0.521, 0.479)^T \otimes (0.521, 0.479)^T \otimes (0.601, 0.399)^T$. A final hard-decision on this output gives non-codeword $x_0x_1x_2x_3 = 1000$ which can then be decoded algebraically, again to codeword 0000. In contrast, successful QPA outputs the optimal entangled joint-state,
$S_F = \frac{1}{\sqrt{2040}}(\sqrt{216}, 0, 0, 0, 0, 0, \sqrt{96}, 0, 0, 0, 0, \sqrt{864}, 0, \sqrt{864}, 0, 0)^T$. Final measurement of S_F always outputs a codeword from $\mathbf{C}$, and with probability $\frac{2*864}{2040}$ outputs either 1011 or 1101. QPA evolves on Fig 2 correctly with probability 0.204. Therefore 1000 attempts produce around 204

correctly entangled joint-states.

To underline QPA advantage, consider the single variable extension of Fig 2 in Fig 3, where x_4 is initialised to $(\sqrt{0.5}, \sqrt{0.5})^T$.

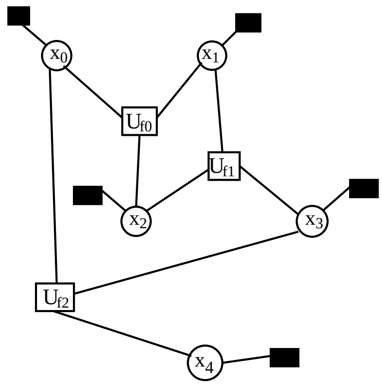

Figure 3: Extended Factor Graph with a Cycle

As $x_4 = x_0 \oplus x_3$, and our original code, $\mathbf{C} = \{0000, 0110, 1011, 1101\}$, always had $x_0 = x_3$, then x_4 should always be 0. But SPA on Fig 3 computes $x_4 = (0.421, 0.579)^T$ and subsequent hard-decision gives $x_4 = 1$. In contrast, successful QPA computes the optimal non-product joint-state,

$$S_{F'} = \frac{1}{\sqrt{2040}}(\sqrt{216}, 0, 0, 0, 0, 0, \sqrt{96}, 0, 0, 0, 0, \sqrt{864}, 0, \sqrt{864}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$$

Final measurement of $S_{F'}$ **always** outputs $x_4 = 0$. QPA evolves on Fig 3 correctly with probability $0.204 * 0.5 = 0.114$.

4 Maximum-Likelihood (ML) Amplification

4.1 Preliminaries

The ML codeword is the one most likely to be measured from QPA output, with probability, p_M, say. For instance, if QPA output of Fig 1 is $S_f = \frac{1}{\sqrt{3}}(1, 0, 0, \sqrt{2})^T$, say, then 11 is the ML codeword, and it is measured with probability $p_M = \frac{2}{3}$. Numerous executions of QPA on the same input will verify that 11 is, indeed, the ML codeword. However these numerous executions must output to a length 2^N final averaging probability vector (for N qubits). We do not want to store such an exponential vector. Instead, therefore, we 'amplify' the statistical advantage of 11 over 00 prior to measurement, thereby making 11 significantly more likely to be read. This is achieved by computing the square of each quantum vector element as follows. Consider two independent QPA executions on the same input, both outputting S_f. Associate these outputs with qubits $x_{0,0}, x_{1,0}$, and $x_{0,1}, x_{1,1}$. The joint-state of qubits $x_{0,0}, x_{1,0}, x_{0,1}, x_{1,1}$ is,

$$V_0 = S_f \otimes S_f = \frac{1}{3}(1, 0, 0, \sqrt{2}, 0, 0, 0, 0, 0, 0, 0, 0, \sqrt{2}, 0, 0, 2)^T$$

Consider the unitary permutation matrix

$$\mathbf{P} = \begin{pmatrix} 1000000000000000 \\ 0000010000000000 \\ 0000000000100000 \\ 0000000000000001 \\ 0100000000000000 \\ 0010000000000000 \\ 0001000000000000 \\ 0000100000000000 \\ 0000001000000000 \\ 0000000100000000 \\ 0000000010000000 \\ 0000000001000000 \\ 0000000000010000 \\ 0000000000001000 \\ 0000000000000100 \\ 0000000000000010 \end{pmatrix}$$

Only the '1' positions in the first four rows are important. Performing $\mathbf{P}$ on $x_{0,0}, x_{1,0}, x_{0,1}, x_{1,1}$, gives,

$$\mathbf{P}V_0 = \frac{1}{3}(1, 0, 0, 2, 0, 0, \sqrt{2}, 0, 0, 0, 0, 0, 0, \sqrt{2}, 0, 0)^T$$

We then measure qubits $x_{0,1}, x_{1,1}$. With probability $p_{a_0} = \frac{5}{9}$ we read $x_{0,1} = x_{1,1} = 0$, in which case $x_{0,0}$ and $x_{1,0}$ are forced into joint state $S_{f,1} = \frac{1}{\sqrt{5}}(1, 0, 0, 2)$, which is the element-square of S_f. A measurement of $S_{f,1}$ returns 11 with probability $p_M = \frac{4}{5}$, which is a significant improvement over $p_M = \frac{2}{3}$. Likewise we compute the element fourth-powers of S_f by preparing two independent qubit pairs in $S_{f,1}$ and permuting the (umeasured) joint state vector $V_1 = S_{f,1} \otimes S_{f,1}$ to give $\mathbf{P}V_1$, and then measuring the second pair of qubits. With probability $p_{a_1} = \frac{17}{25}$ we read this pair as 00, in which case the first two qubits are forced into the joint-state $S_{f,2} = \frac{1}{\sqrt{17}}(1, 0, 0, 4)$, which is the element fourth-power of S_f. A measurement of $S_{f,2}$ returns 11 with probability $p_M = \frac{16}{17}$, which is a further improvement over $p_M = \frac{2}{3}$. In this way we amplify the likelihood of measuring the ML codeword. To compute the element $2^{k\text{th}}$-power, $S_{f,k}$, we require, on average, $\frac{2}{p_{a_k}}$ independent preparations, $S_{f_{k-1}}$, each of which requires, on average, $\frac{2}{p_{a_{k-1}}}$ independent preparations, $S_{f_{k-2}}$, and so on.

We can perform QPA on large Factor Graphs, then amplify the result k times to ensure a high likelihood of measuring the ML codeword, as described above. However the above amplification acts on the complete graph with one operation, $\mathbf{P}$. It would be preferable to decompose $\mathbf{P}$ into 4×4 unitary matrices which only act on independent qubit pairs $x_{i,0}$ and $x_{i,1}$, thereby localising amplification. Consider, once again, Fig 1. From the point of view of

$x_{0,1}$, $x_{0,0}$ appears to be in summarised state [4] , $s_f = \frac{1}{\sqrt{3}}(1, \sqrt{2})^T$. Similarly, from the point of view of $x_{0,0}$, $x_{0,1}$ appears to be in state s_f. Thus $x_{0,0}, x_{0,1}$ appear to be in joint product state $v_0 = \frac{1}{3}(1, \sqrt{2}, \sqrt{2}, 2)^T$. Consider unitary permutation matrix,

$$\mathbf{Q} = \begin{pmatrix} 1000 \\ 0001 \\ 0100 \\ 0010 \end{pmatrix}$$

We compute $\mathbf{Q}v_0 = \frac{1}{3}(1, 2, \sqrt{2}, \sqrt{2})^T$ on qubits $x_{0,0}, x_{0,1}$ and measure qubit $x_{0,1}$. With probability $p_{a_0} = \frac{5}{9}$ we read $x_{0,1} = 0$, in which case $x_{0,0}$ is forced into joint state $s_{f,1} = \frac{1}{\sqrt{5}}(1, 2)$, which is the element-square of s_f. Due to the exact form of our joint-state vector, S_f, this single measurement is enough to also force $x_{0,0}x_{1,0}$ into joint state $S_{f,1}$. However, for a general function S_f, we should perform $\mathbf{Q}$ on every qubit pair, $x_{i,0}x_{i,1}$, then measure $x_{i,1}$ $\forall i$. This is equivalent to performing $\mathbf{P}' = \mathbf{Q} \otimes \mathbf{Q}$ on (re-ordered) joint-state vector $x_{0,0}x_{0,1}x_{1,0}x_{1,1}$, and this is identical to performing $\mathbf{P}$ on $x_{0,0}x_{1,0}x_{0,1}x_{1,1}$. The probability of measuring $x_{1,0} = x_{1,1} = 0$ is the same whether $\mathbf{P}$ or $\mathbf{Q}$ is used. The same process is followed to achieve element 2^kth powers.

4.2 The Price of Amplification

There is a statistical cost to qubit amplification. Let $s = (\alpha, \beta)^T$ be the initial state of a qubit x, where, for notational convenience, we assume that α and β are both real. Then $\alpha^2 + \beta^2 = 1$ and, given 2^k qubits all identically prepared in state s, the likelihood of preparing one qubit in (unnormalised) state $s_k = (\alpha^{2^k}, \beta^{2^k})^T$ is γ_k, where,

$$\gamma_k = \gamma_{k-1}^2 \frac{r_{k+1}}{r_k^2}, \qquad \gamma_0 = 1$$

and $r_k = \alpha^{2^k} + \beta^{2^k}$. For a qubit in state s_k, the probability of selecting the ML codebit is,

$$P_{Mk} = \frac{\alpha^{2^{k+1}}}{\alpha^{2^{k+1}} + \beta^{2^{k+1}}}$$

(assuming $\alpha \geq \beta$). We can plot γ_k against P_{Mk} for various α^2 as k varies, as shown in Fig 4.

Each of the 25 lines in Fig 4 refers to a different value of α^2, for α^2 from 0.5 up to 0.98 in steps of 0.2. The initial state, s, when $k = 0$, occurs with probability $\gamma_k = 1$, and is marked on the right-hand side of Fig 4 for each of the 25 lines. After one amplification step, $k = 1$, and another 25 points are marked on the graph to the left of the points for $k = 0$, indicating that a successful amplification step has occurred with probability $\gamma_k \leq 1$. Similarly points for $k = 2$, $k = 3$,..etc are marked successively to the left on Fig 4. The y-axis shows the ML

[4] $x_{0,0}$ is generally <u>not</u> in this summarised state, due to phase considerations, but the viewpoint is valid for our purposes as long as subsequent unitary matrix operations on $x0$ only have one non-zero entry per row.

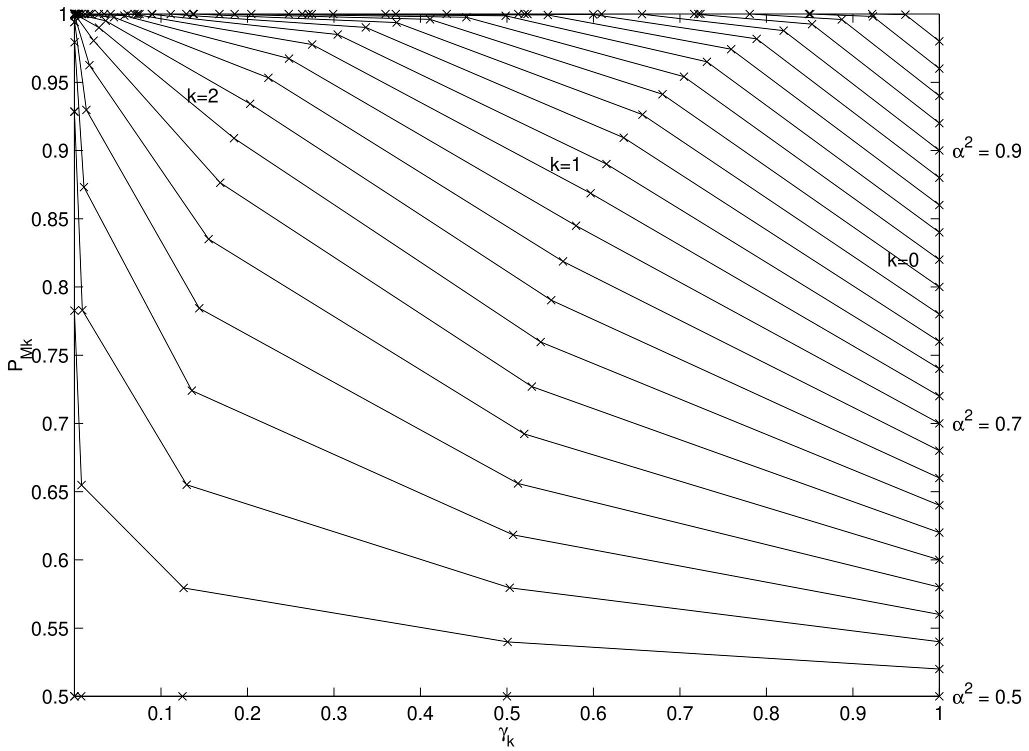

Figure 4: Amplification Success Probability, γ_k, v ML Advantage, P_{Mk}

advantage, P_{Mk}, which can be achieved with probability γ_k after k steps for each value of α^2. For instance, when $s = (\alpha, \beta)^T = (\sqrt{0.62}, \sqrt{0.38})^T$, then an ML advantage of $P_{Mk} = 0.9805$ can be ensured after $k = 3$ steps, and this can be achieved with probability $\gamma_k = 0.0223$ given $2^3 = 8$ independently prepared qubits, all in state s. Amplification is more rapid if s already has significant ML advantage (i.e. when α is high). In contrast if $\alpha^2 = 0.5$ then no amplification of that qubit is possible. This is quite reasonable as, in this case, both states 0 and 1 are equally likely, so there is no ML state. Successive measurement of zero of all second qubits of each qubit pair self-verifies that we have obtained successful amplification. If, at any step, k, the second qubit of the qubit pair is measured as one then amplification fails and the graph local to this qubit which has been successfully entangled up until now is completely destroyed.

5 Distributed QPA on Many-Node Factor Graphs

5.1 Preliminaries

In classical systems it is desirable to implement SPA on Factor Graphs which 'tensor-approximate' the variable space using many small-state variables (e.g. bits), linked by small-dimensional constituent functions, thereby minimising computational complexity. In quantum systems it is similarly desirable to implement QPA on Factor Graphs using many small-state variables (e.g. qubits), linked by small-dimensional constituent unitary functions. Any Quantum Computation can be decomposed into a sequence of one or two-bit 'universal' gate unitary operations [3][5] . Computational complexity is minimised by using small-dimensional unitary matrices for constituent functions. Moreover, fine granularity of the Factor Graph allows distributed node processing. This appears to be essential for large Quantum Factor Graphs to have acceptable probability of successful global evolution, as we will show. Distributed QPA allows variable nodes to evolve entanglement only with neighbouring variable nodes so that, if a local function measurement or amplification is unsuccessful, only local evolution is destroyed. Remember that local evolution is OFTEN unsuccessful, as failure occurs when a local ancillary qubit, z, is measured as 1, or when a local amplifying qubit is measured as 1. Therefore node localities with high likelihood of successful evolution (i.e. with positively skewed input probabilities) are likely to evolve first. These will then encourage other self-contradictory node localities to evolve successfully. In contrast, non-distributed QPA on large Factor Graphs using one large global function is very unlikely to ever succeed, especially for graphs encoding low-rate codes.

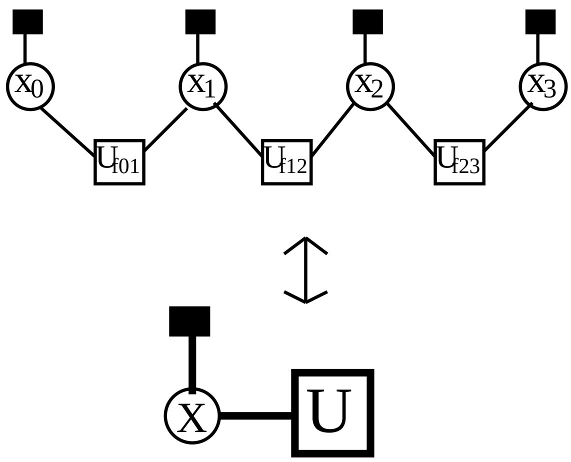

Figure 5: Distributed QPA (top) Non-Distributed QPA (bottom), 4-bit code

To illustrate the advangtage of distributed QPA, consider the low rate code of Fig 5, where $\mathbf{U_{fij}} = \mathrm{diag}(1, 0, 0, 1)$. Both top and bottom graphs represent the code $\mathbf{C} = \{0000, 1111\}$, where $\mathbf{U}$ is a combination of XOR sub-matrices, $\mathbf{U_{f01}}$,$\mathbf{U_{f12}}$, and $\mathbf{U_{f23}}$. The top graph distributes processing. We allow $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$ to operate independently and in parallel. Moreover, if $\mathbf{U_{f01}}$ fails to establish, then it does not destroy any successful evolution of $\mathbf{U_{f23}}$, as the two localities are not currently entangled. Once both $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$ have completed successfully,

[5]This also implies that any classical Factor Graph can be similarly decomposed.

the subsequent probability of successful completion of $\mathbf{U_{f12}}$ is, in general, likely to increase. So distributing QPA increases likelihood of successful evolution of the complete Factor Graph. We now demonstrate this graphically. Let qubits x_0, x_1, x_2, x_3 of Fig 5 initially be in states $x_0 = (\alpha_0, \beta_0)^T$, $x_1 = (\alpha_1, \beta_1)^T$, $x_2 = (\alpha_2, \beta_2)^T$, $x_3 = (\alpha_3, \beta_3)^T$, where, for notational convenience, we assume all values are real. Then $\alpha_i^2 + \beta_i^2 = 1$, $\forall i$. The probability of successful completion of $\mathbf{U_{f01}}$ is $p_{f01} = (\alpha_0\alpha_1)^2 + (\beta_0\beta_1)^2$, and probability of successful completion of $\mathbf{U_{f23}}$ is $p_{f23} = (\alpha_2\alpha_3)^2 + (\beta_2\beta_3)^2$. Therefore the probability of successful completion of both $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$ after exactly q parallel attempts (no less) is,

$$p_{0-3}(q) = (1-p_{f01})^{q-1}(1-p_{f23})^{q-1}p_{f01}p_{f23}+(1-p_{f01})^{q-1}(1-(1-p_{f23})^{q-1})p_{f01}+(1-p_{f23})^{q-1}(1-(1-p_{f01})^{q-1})p_{f23}$$

Given successful completion of $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$, the probability of **subsequent** successful completion of $\mathbf{U_{f12}}$ is,

$$p'_{f12} = \frac{(\alpha_0\alpha_1\alpha_2\alpha_3)^2 + (\beta_0\beta_1\beta_2\beta_3)^2}{p_{f01}p_{f23}}$$

Therefore the probability of successful completion of $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$, immediately followed by successful completion of $\mathbf{U_{f12}}$ is, $p_{0\to 3}(q) = p_{0-3}(q-1)p'_{f12}$, and the probability of successful completion of $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$, immediately followed by completion failure of $\mathbf{U_{f12}}$ is, $\overline{p_{0\to 3}}(q) = p_{0-3}(q-1)(1-p'_{f12})$. Therefore the probability of successful completion after exactly t steps of $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$ in parallel, followed by $\mathbf{U_{f12}}$, is,

$$p_e(t) = \sum_{q=2}^{t} p_{0\to 3}(q) \sum_{\mathbf{v}\in\mathbf{D(t-q)}} \prod_{u\in\mathbf{v}} \overline{p_{0\to 3}}(u)$$

where $\mathbf{D(k)}$ is the set of unordered partitions of k. Therefore the probability of successful completion after at most t steps of $\mathbf{U_{f01}}$ and $\mathbf{U_{f23}}$ in parallel, followed by $\mathbf{U_{f12}}$, is,

$$p_m(t) = \sum_{i=2}^{t} p_e(i)$$

In contrast, for non-distributed QPA, the probability of successful completion, after at most t steps, of $\mathbf{U}$, (the bottom graph of Fig 5) is $P(t) = 1 - (1 - (\alpha_0\alpha_1\alpha_2\alpha_3)^2 - (\beta_0\beta_1\beta_2\beta_3)^2)^t$. Figs 6 and 7 show plots of $p_m(t)$ and $P(t)$ versus t for $\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = w$ as w varies, and $\alpha_0 = u, \alpha_1 = \alpha_2 = \alpha_3 = w = 0.9$ as u varies, respectively. For Fig 7, low values of u indicate a contradiction between x_0 and the other three variables. In particular the contradiction is so pronounced when $\alpha_0 = 0.0$ that successful QPA completion is highly unlikely. More generally, this indicates that severe internal Factor Graph contradictions are fatal to QPA (as they are for SPA). Both Fig 6 and 7 indicate that, due to initial latency of distributed processing, non-distributed QPA appears marginally faster for the first few steps. However, after a few steps distributed QPA in general becomes marginally faster. In fact results are unfairly biased towards the non-distributed case, as it is assumed that attempts to complete $\mathbf{U}$ and $\mathbf{U_{fij}}$ have the same space-time-complexity cost, whereas $\mathbf{U}$ is far more costly. Hence, even for this smallest example, Distributed QPA outperforms non-Distributed QPA.

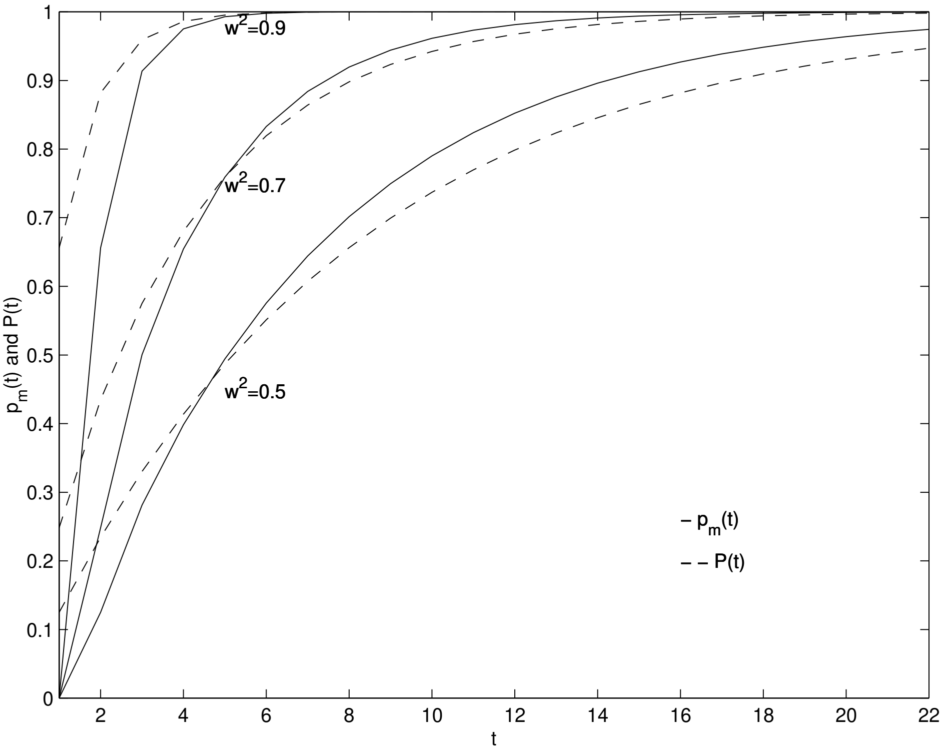

Figure 6: No of Steps v Non-Distributed and Distributed QPA: Completion Probabilities

The example of Fig 5 only achieves marginal advantage using Distributed QPA because the example has so few nodes. The advantage is more pronounced in Fig 8.

Fig 8 represents the code $\mathbf{C} = \{000000000, 111111111\}$ [6] , where $\mathbf{U_{ijk}} = \mathrm{diag}(1, 0, 0, 1, 0, 1, 1, 0)$. We allow $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ to operate independently and in parallel. If $\mathbf{U_{f012}}$ fails to establish, then it does not destroy any successful evolution of $\mathbf{U_{f345}}$ or $\mathbf{U_{f678}}$, as the three localities are not currently entangled. Once $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ have completed successfully, the probability of successful subsequent completion of $\mathbf{U_{f258}}$ is, in general, amplified. Let qubits $x_i, 0 \leq i < 9$ of Fig 8 initially be in states $x_i = (\alpha_i, \beta_i)^T$, where, for notational convenience, we assume all values are real. Then $\alpha_i^2 + \beta_i^2 = 1$, $\forall i$. Let the probability of successful completion

[6] This code is trivial but demonstrates a 'worst-case' low-rate scenario. In general, codes of higher rate, with or without cycles, decode more quickly.

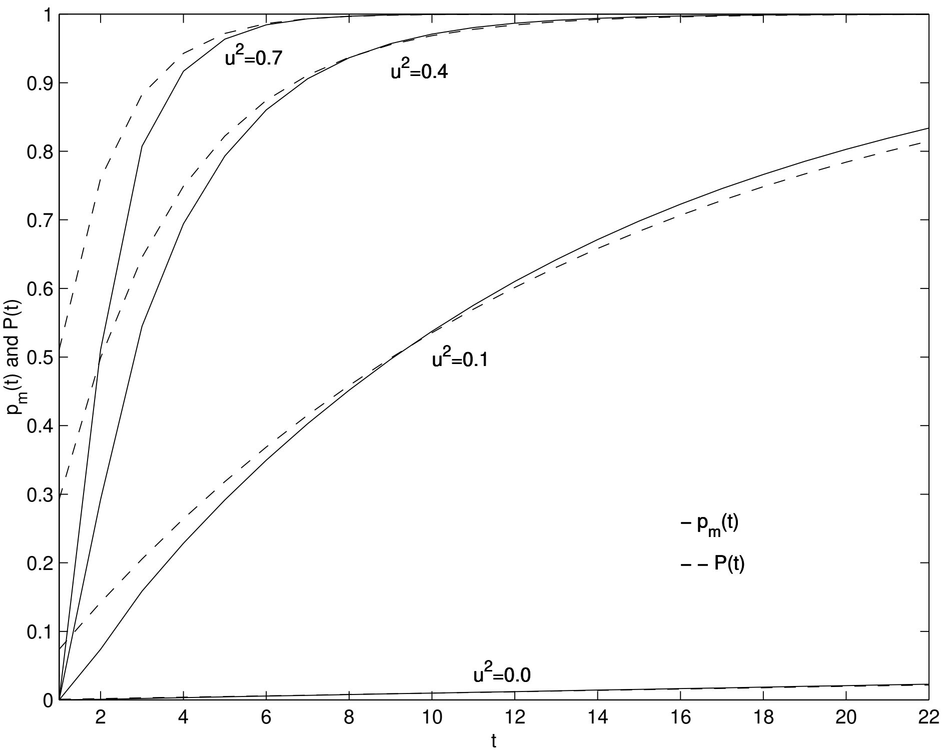

Figure 7: No of Steps v Non-Distributed and Distributed QPA: $w^2 = 0.9$, α_0 varies

after at most t steps of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ in parallel, followed by $\mathbf{U_{f258}}$, be $p_m(t)$, and the probability of successful completion, after at most t steps, of a non-distributed version of Fig 8 be $P(t)$. Appendix A derives $p_m(t)$ and P_t for this case. Figs 9 and 10 show plots of $p_m(t)$ and $P(t)$ versus t for $\alpha_i = w$, $\forall i$, as w varies, and $\alpha_0 = u, \alpha_i = w = 0.9$, $\forall i$, $i \neq 0$, as u varies, respectively. For Fig 10 low values of u indicate contradiction between x_0 and the other eight qubits. The contradiction is so pronounced when $\alpha_0 = 0.0$ that successful QPA completion is highly unlikely.

Figs 11 and 12 show plots of $p_m(t)$ and $P(t)$ versus t for $\alpha_0 = \alpha_1 = u, \alpha_i = w = 0.9$, $\forall i$, $i \neq 0, 1$, and $\alpha_0 = \alpha_8 = u, \alpha_i = w = 0.9$, $\forall i$, $i \neq 0, 8$, respectively, as u varies. Both figures indicate contradictions between two qubits and the rest, but the scattered nature of contradictions for Fig 12 (x_0 and x_8 are connected to different local functions) enhances Distributed QPA performance compared to Fig 11.

Figs 6-12 indicate that distributed QPA completes significantly faster than non-distributed QPA, in particular for cases requiring many steps, t. Even more so as the presented results are unfairly biased towards the non-distributed case, as it is assumed that attempts to complete non-distributed $\mathbf{U}$ or each constituent $\mathbf{U_{fijk}}$ have the same space-time-complexity cost, whereas $\mathbf{U}$ is far more costly. We conclude that Distributed QPA is essential for large Quantum Factor

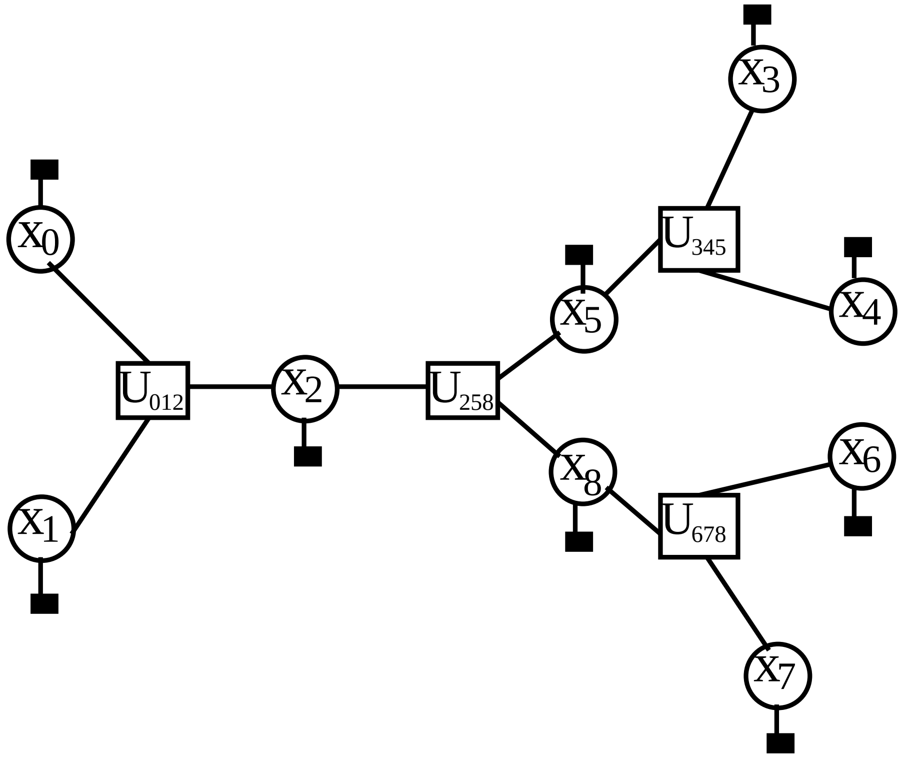

Figure 8: Distributed QPA, 9-qubits

Graphs.

5.2 Free-Running Distributed QPA

Consider the notional Factor Graph of Fig 13. Each (square) function node activates time-independently on its local (circular) variable nodes. Functions successfully completed are marked with an 'X'. After a certain time, say, three 'areas of success' evolve, due to general agreement between input variable states at these localities. This means that variables on the perimeter of each region of success are 'encouraged' to agree with the 'general view' of the associated region of success. Unfortunately, in the bottom left of the graph is a variable (dark circle) which strongly contradicts with the rest of the graph. No area of success evolves around it, and it is difficult for other areas of success to 'swallow' it. Assuming the contradiction is not too strong then, eventually, after numerous attempts, the complete graph is marked with 'X's and the Graph evolves successfully. At this point the contents of each qubit variable can be amplified, and final measurement of all qubits provides the ML codeword with high probability. The advantage of a free-running strategy, where each function node is free to activate asynchronously, is that regions of general agreement develop first and influence other areas of the graph to 'follow their opinion'. Fig 13 also shows that one 'bad' (contradictory) qubit can be a

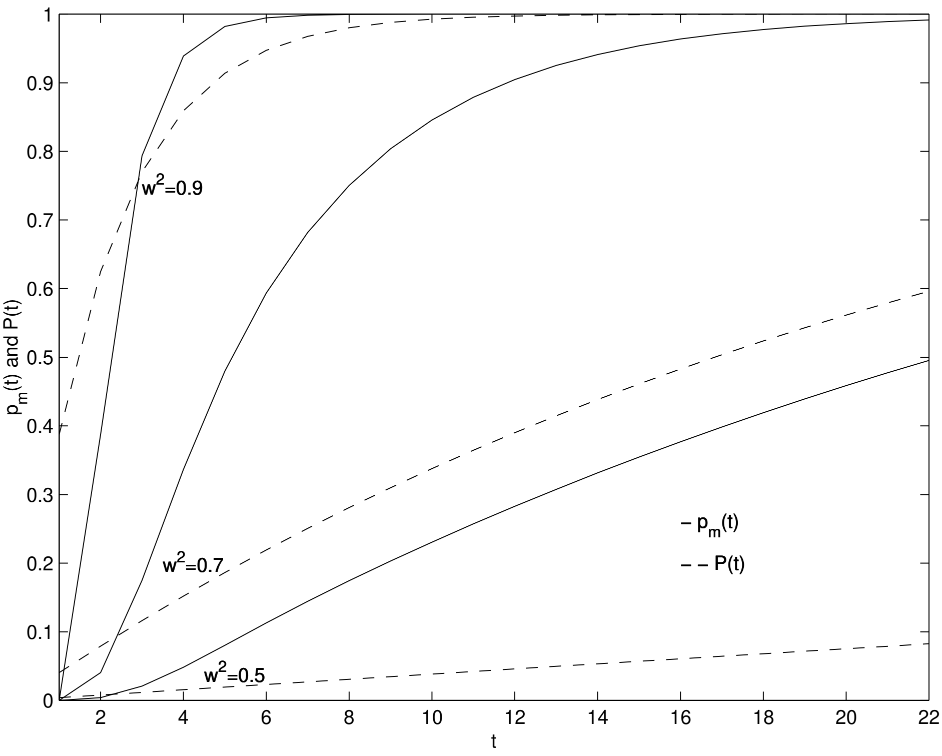

Figure 9: No of Steps v Non-Distributed and Distributed QPA, 9 qubits

fatal stumbling block to successful evolution of the whole graph (as it can for SPA on classical graphs). Thus Distributed QPA requires Fault-Tolerance, where only an arbitrary subset of entangled nodes are required as a final result (node redundancy). The free-running schedule of Fig 13 naturally avoids the 'bad' qubits, but sufficient evolution occurs when enough function nodes complete. Alternatively, bad qubits could be set to $(\sqrt{0.5}, \sqrt{0.5})$ after a time-out. A more detailed proposal of Fault-Tolerant QPA is left for future work.

Fig 13 also serves to illustrate the 'template' for a Reconfigurable Quantum Graph Array. One can envisage initialising an array of quantum variables so that two local variables can be strongly or weakly entangled by identifying the mutual square function nodes with strongly or weakly-entangling matrices, respectively. In particular, two neighbouring nodes may be 'locally disconnected' by setting the function node joining them to a tensor-decomposable matrix, (i.e. zero-entangling). The quantum computer is then measurement-driven. The concept of measurement-driven quantum computation has also recently been pursued in [8], where a uniform entanglement is set-up throughout the array [7] prior to computation via measurement.

Fig 14 shows the system view of QPA. A continual stream of pure qubits needs to be initialised and then entangled, and then amplified, so as to ensure at least one successful entangled

[7]It is interesting that this entanglement is strongly related to Rudin-Shapiro and quadratic constructions [7]

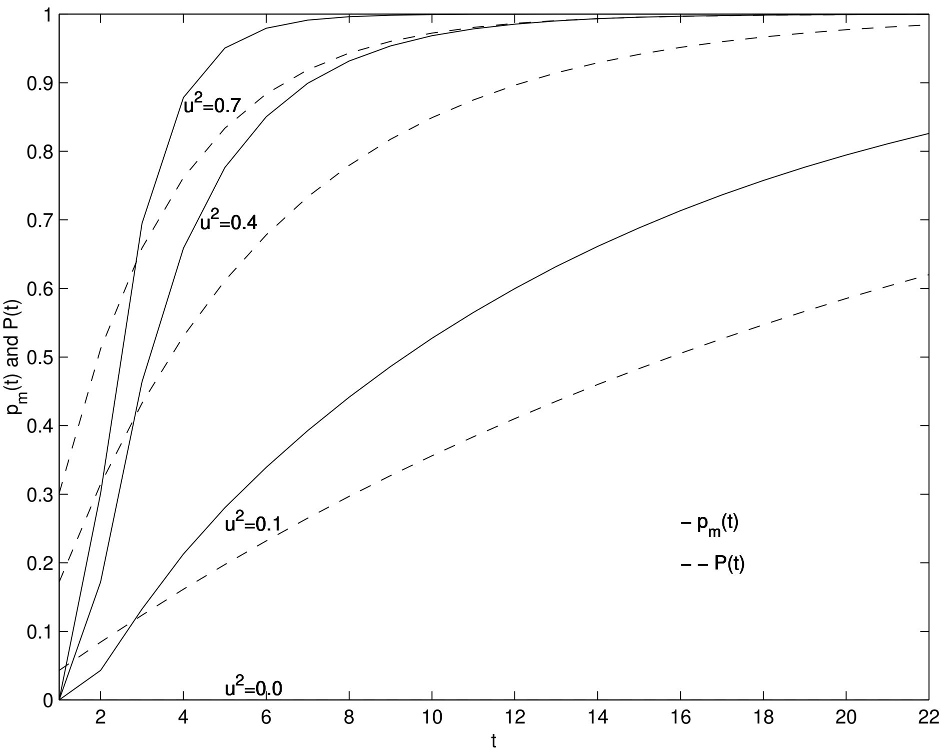

Figure 10: No of Steps v Non-Distributed and Distributed QPA: $w^2 = 0.9$, α_0 varies, 9 qubits

and amplified output from the whole apparatus.

6 Phase QPA

The above discussions have ignored the capacity of Quantum Systems to carry phase information. In fact QPA, as presented so far, is immune to phase modification, as classical probabilities have no phase component. However QPA should be generalised to cope with phase shift in order to decode quantum information. This is the subject of ongoing research.

7 Conclusion and Discussion

The Quantum Product Algorithm (QPA) on a Factor Graph has been presented for Maximum-Likelihood (ML) Decoding of Classical 'soft' information using quantum resources. The relationship of QPA to the Sum-Product Algorithm (SPA) has been indicated, where avoidance of summary allows QPA to overcome small graph cycles. Quantum Factor Graphs use small unitary matrices which each act on only a few qubits. QPA is measurement-driven and is only statistically likely to succeed after many attempts. The ML codeword is obtained with maximum likelihood by measuring the entangled vector resulting from successful QPA. To en-

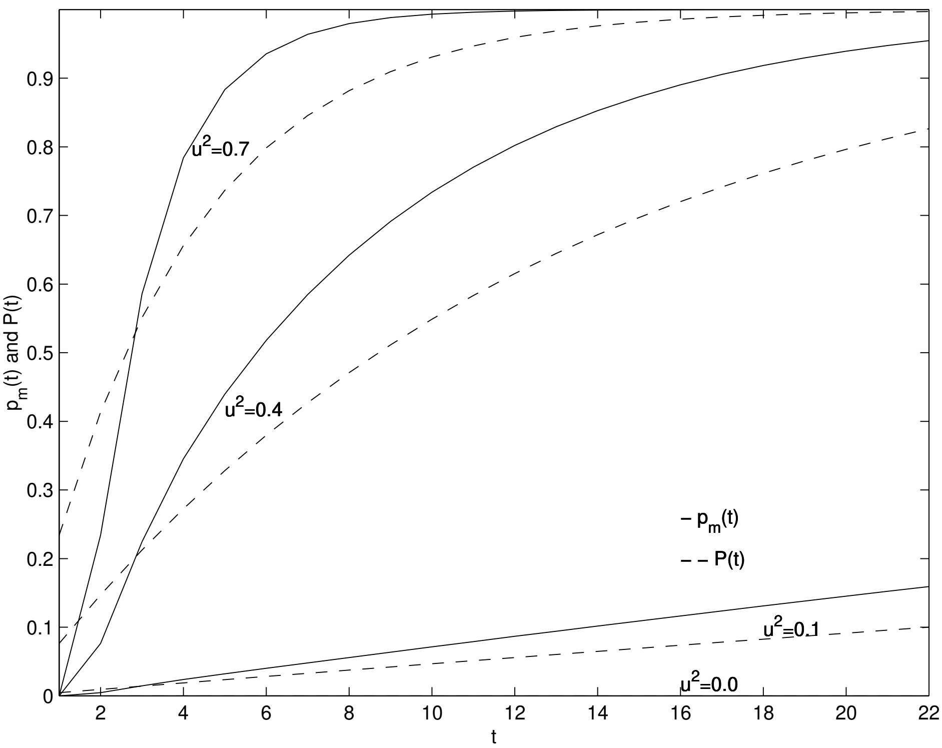

Figure 11: No of Steps v Non-Distributed and Distributed QPA: $w^2 = 0.9$, $\alpha_0 = \alpha_1$ varies, 9 qubits

sure a high probability of measuring the ML codeword QPA output can be amplified prior to measurement. The complete ML decoder is only successful after many attempts. Finally, free-running Distributed QPA is proposed to improve the likelihood of successful QPA completion. The free-running distributed structure suggests further benefit will be obtained by introducing Fault-Tolerance in the form of redundant function and variable nodes. Phase aspects of QPA have yet to be explored. This paper has been written to demonstrate the exponential capacity of quantum systems, and their natural suitability for graph decompositions such as the Factor Graph. The paper has not tried to deal with quantum noise and quantum decoherence, but one can expect the Factor Graph form to 'gracefully' expand to cope with the extra redundancy necessary to protect qubits from decoherence and noise. When viewed in the context of entangled space, it is surprising how successful classical message-passing algorithms are, even though they are restricted to operate in tensor product space. This suggests that methods to improve the likelihood of successful QPA completion may include the possibility of hybrid QPA/SPA graphs, where SPA operates on non-cyclic and resolvable parts of the graph, leaving QPA to cope with small cycles or unresolved areas of the graph.

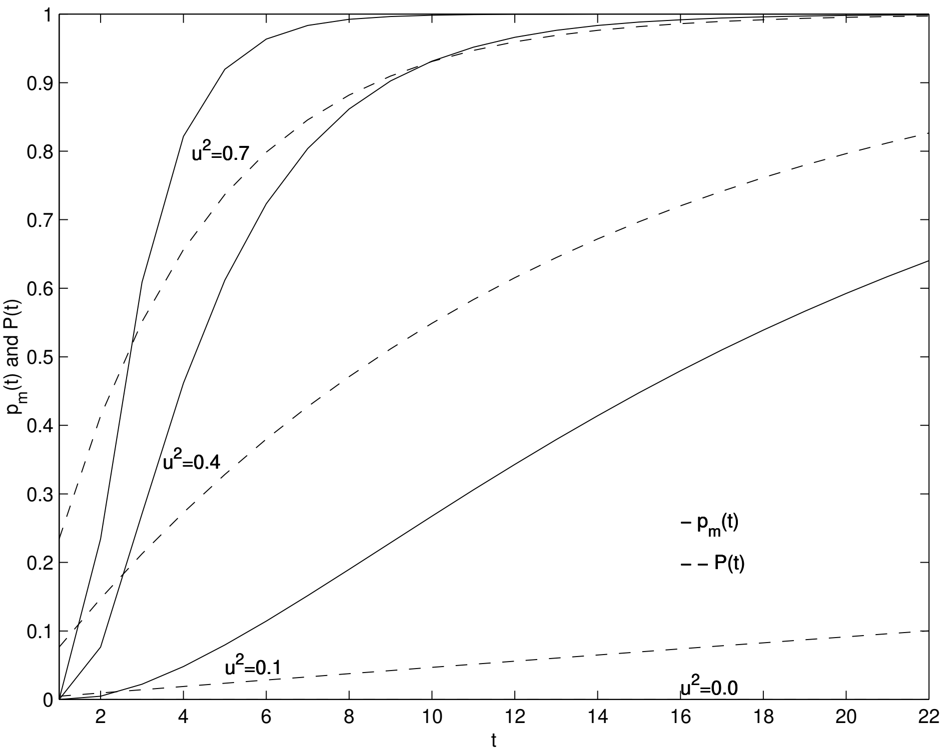

Figure 12: No of Steps v Non-Distributed and Distributed QPA: $w^2 = 0.9$, $\alpha_0 = \alpha_8$ varies, 9 qubits

8 Appendix A - Deriving $p_m(t)$ and $P(t)$ for Fig 8

The probability of successful completion of $\mathbf{U_{f012}}$, is $p_{f012} = (\alpha_0\alpha_1\alpha_2)^2 + (\beta_0\beta_1\beta_2)^2$, and similarly for p_{f345} and p_{f678}. Let $h_{012} = (1-p_{f012})^{q-1}$, $h_{345} = (1-p_{f345})^{q-1}$, $h_{678} = (1-p_{f678})^{q-1}$. Then the probability of successful completion of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ after exactly q parallel attempts is,

$$\begin{aligned}
p_{0-3-6}(q) = h_{012}h_{345}h_{678}p_{f012}p_{f345}p_{f678} + (1-h_{012})h_{345}h_{678}p_{f345}p_{f678} \\
+h_{012}(1-h_{345})h_{678}p_{f012}p_{f678} + h_{012}h_{345}(1-h_{678})p_{f012}p_{f345} \\
+(1-h_{012})(1-h_{345})h_{678}p_{f678} + (1-h_{012})h_{345}(1-h_{678})p_{f345} \\
+h_{012}(1-h_{345})(1-h_{678})p_{f012}
\end{aligned}$$

Given successful completion of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$, the probability of **subsequent** successful completion of $\mathbf{U_{f258}}$ is,

$$p'_{f258} = \frac{(\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7\alpha_8)^2 + (\beta_0\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8)^2}{p_{f012}p_{f345}p_{678}}$$

Therefore the probability of successful completion of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$, immediately followed by successful completion of $\mathbf{U_{f258}}$ is, $p_{0\to 8}(q) = p_{0-3-6}(q-1)p'_{f258}$, and the probability

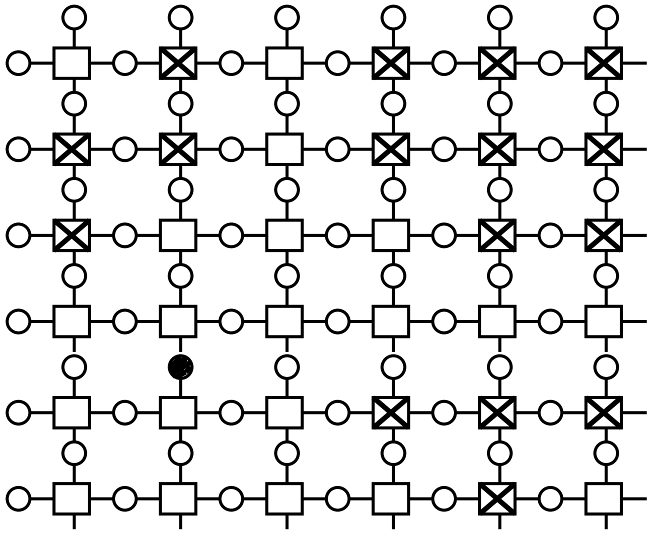

Figure 13: Free-Running Distributed QPA with one 'Bad' Variable

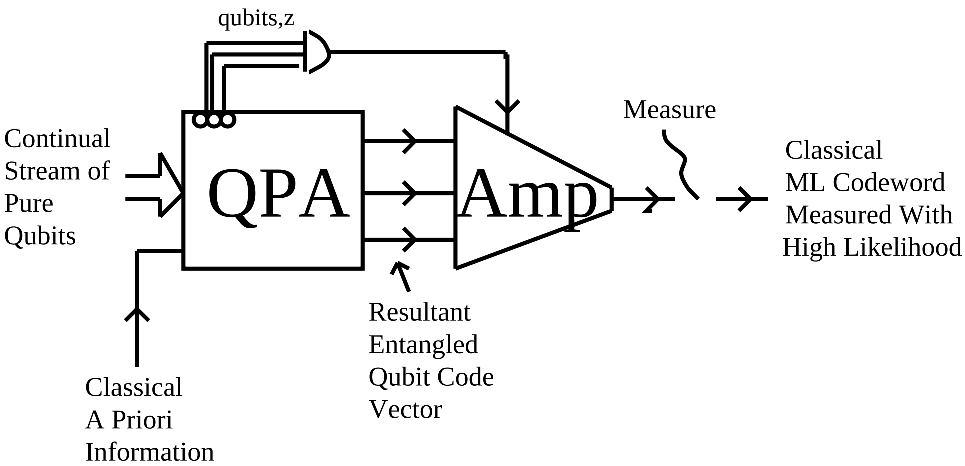

Figure 14: QPA with Amplification

of successful completion of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$, immediately followed by completion failure of $\mathbf{U_{f258}}$ is, $\overline{p_{0\to 8}}(q) = p_{0-3-6}(q-1)(1-p'_{f258})$. The probability of successful completion after exactly t steps of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ in parallel, followed by $\mathbf{U_{f258}}$, is,

$$p_e(t) = \sum_{q=2}^{t} p_{0\to 8}(q) \sum_{\mathbf{v}\in\mathbf{D(t-q)}} \prod_{u\in\mathbf{v}} \overline{p_{0\to 8}}(u)$$

where $\mathbf{D(k)}$ is the set of unordered partitions of k. Therefore the probability of successful completion after at most t steps of $\mathbf{U_{f012}}$, $\mathbf{U_{f345}}$, and $\mathbf{U_{f678}}$ in parallel, followed by $\mathbf{U_{f258}}$, is,

$$p_m(t) = \sum_{i=2}^{t} p_e(i)$$

In contrast, the probability of successful completion, after at most t steps, of a non-distributed version of Fig 8 is $P(t) = 1 - (1 - (\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7\alpha_8)^2 - (\beta_0\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8)^2)^t$.

References

[1] AJI (S.M.), McELIECE (R.J.), "The Generalized Distributive Law", *IEEE Trans. Inform. Theory*, Vol. IT-46, pp. 325-343, March 2000

[2] BERROU (C.),GLAVIEUX (A.),THITIMAJSHIMA (P.), "Near Shannon-Limit Error-Correcting Coding and Decoding: Turbo-Codes", *Proc. 1993 IEEE Int. Conf. on Communications (Geneva, Switzerland),*, pp. 1064-1070, 1993

[3] DiVINCENZO (D.P.), "Two-Bit Gates are Universal for Quantum Computation", *Phys. Rev. A 51*, Vol 1015, 1995, Also *LANL arXiv:cond-mat/9407022*

[4] GALLAGER (R.G.), "Low Density Parity Check Codes," *IRE Trans. Inform. Theory*, Vol IT-8, pp 21-28, Jan. 1962

[5] KSCHISCHANG (F.R.),FREY (B.J.),LOELIGER (H-A.), "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inform. Theory*, Vol 47, No 2, pp 498-519, Feb. 2001

[6] MACKAY (D.J.C.),NEAL (R.M.), "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electronics Letters*, Vol 32, No 18, pp 1645-1646, Aug 1996. Reprinted Vol 33, No 6, pp 457-458, Mar 1997

[7] PARKER (M.G.), RIJMEN (V.), "The Quantum Entanglement of Bipolar Sequences", *Sequences and their Applications, SETA'01, Bergen, Norway*, 13-17 May, 2001 Also http://www.ii.uib.no/∼matthew/MattWeb.html

[8] RAUSSENDORF (R.),BRIEGEL (H.J.), "Quantum Computing via Measurements Only", *LANL arXiv:quant-ph/0010033*, 7 Oct 2000

[9] SHOR (P.W.), "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM J. Computing*, Vol 26, pp 1484-, 1997 *First appeared in Proceedings of the* 35^{th}*Annual Symposium on the Theory of Computer Science, ed. S. Goldwasser, IEEE Computer Society Press, Los Alamitos, CA*, pp 124-, 1994, Expanded Version: *LANL: quant-ph/9508027*

[10] STEANE (A.M.), "Quantum Computing," *Rept.Prog.Phys.*, Vol 61, pp 117-173, 1998, Also *LANL: quant-ph/9708022*

[11] TUCCI (R.R.), "How to Compile a Quantum Bayesian Net", *LANL: quant-ph/9805016*, 7 May, 1998

[12] TUCCI (R.R.), "Quantum Information Theory - A Quantum Bayesian Net Perspective", *LANL: quant-ph/9909039*, 13 Sep, 1999