Method For Discovering Structure of a Bayesian Network Via a Quantum Computer

Robert R. Tucci

P.O. Box 226

Bedford, MA 01730

tucci@ar-tiste.com

March 24, 2014

CROSS REFERENCES TO RELATED APPLICATIONS

The following related patent applications are to be filed on the same day as this one:

- "Method For Calculating Symmetrized Functions Via a Quantum Computer", by R.R. Tucci
- "Method For Calculating Mobius-like Transforms Via a Quantum Computer", by R.R. Tucci
- "Method For Calculating Mean Values Via a Quantum Computer", by R.R. Tucci

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH AND DEVELOPMENT

Not Applicable

REFERENCE TO COMPUTER PROGRAM LISTING

A computer program listing consisting of a single file entitled `ArQ-Src1-6.txt`, in ASCII format, is included with this patent application.

BACKGROUND OF THE INVENTION

(A)FIELD OF THE INVENTION

The invention relates to a quantum computer; that is, an array of quantum bits (called qubits). More specifically, it relates to methods for using a classical computer to generate a sequence of operations that can be used to operate a quantum computer.

(B)DESCRIPTION OF RELATED ART

Henceforth, we will allude to certain references by codes. Here is a list of codes and the references they will stand for.

Ref.Bra1 is G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation", arXiv:quant-ph/0005055

Ref.Bra2 is G. Brassard, F. Dupuis, S. Gambs, and A. Tapp, "An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance", arXiv:1106.4267

Ref.CoHe is G.F. Cooper, and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data", Machine learning 9.4 (1992): 309-347.

Ref.Dev is S. Devitt, Kae Nemoto, and W. Munro, "Quantum error correction for beginners", arXiv:0905.2794.

Ref.ElWo is B. Ellis, and Wing Hung Wong, "Learning causal Bayesian network structures from experimental data", Journal of the American Statistical Association 103.482 (2008).

Ref.FrKo is N. Friedman, and D. Koller, "Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks", Machine learning 50.1-2 (2003): 95-125.

Ref.GPat is Lov K. Grover, "Fast Quantum Mechanical Algorithms", US Patent 6,317,766

Ref.HeTi is Ru He, Jian Tian, "Bayesian Learning in Bayesian Networks of Moderate Size by Efficient Sampling". Unpublished.

Ref.Koi is M. Koivisto, "Advances in exact Bayesian structure discovery in Bayesian networks", , arXiv:1206.6828

Ref.KoSo is M. Koivisto, and K. Sood, "Exact Bayesian structure discovery in Bayesian networks", The Journal of Machine Learning Research 5 (2004): 549-573.

Ref.Tuc-qJen is R.R. Tucci, "Quantum Circuit For Discovering from Data the Structure of Classical Bayesian Networks". Unpublished. Copy included as an appendix to this patent application.

Ref.Tuc-qLis is R.R. Tucci, "qSym, qMobius, qMargi, qMean and qJennings, 5 Code Generators for Generating Quantum Circuits that Perform Some Artificial Intelligence Related Tasks". Unpublished. Copy included as an appendix to this patent application.

Ref.Tuc-qMob is R.R. Tucci, "Quantum Circuit for Calculating Mobius-like Transforms Via Grover-like Algorithm". Unpublished. Copy included as an appendix to this patent application.

Ref.Tuc-qSym is R.R. Tucci, "Quantum Circuit for Calculating Symmetrized Functions Via Grover-like Algorithm". Unpublished. Copy included as an appendix to this patent application.

Ref.TucAFGA is R.R. Tucci, "An Adaptive, Fixed-Point Version of Grover's Algorithm", arXiv:1001.5200

Ref.TucAFGApat is R.R. Tucci, "Method for Driving Starting Quantum State to Target One", US Patent 8,527,437

Ref.TucQuibbs is R.R. Tucci, "Quibbs, a Code Generator for Quantum Gibbs Sampling", arXiv:1004.2205

Ref.TucSimAnn is R.R. Tucci, "Code Generator for Quantum Simulated Annealing", arXiv:0908.1633

Ref.WiKaSv is N. Wiebey, A. Kapoor, and K. Svore, "Quantum Nearest-Neighbor Algorithms for Machine Learning", arXiv:1401.2142

This invention deals with quantum computing. A quantum computer is an array of quantum bits (qubits) together with some hardware for manipulating those qubits. Quantum computers with several hundred qubits have not been built yet. However, once they are built, it is expected that they will perform certain calculations much faster that classical computers. A quantum computer follows a sequence of elementary operations. The operations are elementary in the sense that they act on only a few qubits (usually 1, 2 or 3) at a time. Henceforth, we will sometimes refer to sequences as products and to operations as operators, matrices, instructions, steps or gates. Furthermore, we will abbreviate the phrase "sequence of elementary operations" by "SEO". SEOs for quantum computers are often represented by quantum circuits. In the quantum computing literature, the term "quantum algorithm" usually means a SEO for quantum computers for performing a desired calculation. Some quantum algorithms have become standard, such as those due to Deutsch-Jozsa, Shor and Grover. One can find on the Internet many excellent expositions on quantum computing.

Our algorithm utilizes the original Grover's algorithm (see **Ref.GPat**) or any variant thereof, as long as it accomplishes the task of driving a starting state $|s\rangle$ towards a target state $|t\rangle$. However, we recommend to the users of our algorithm that they use a variant of Grover's algorithm called AFGA (adaptive fixed point Grover's algorithm) which was first proposed in **Ref.TucAFGA** and **Ref.TucAFGApat** .

This invention gives a quantum circuit for calculating the probability $P(G|D)$ of a graph G given data D. G together with a transition probability matrix for each node of the graph, constitutes a Classical Bayesian Network, or CB net for short. Bayesian methods for calculating $P(G|D)$ have been given before (the so called structural modular and ordered modular models), but these earlier methods were designed to work on a classical computer. The goal of this invention is to "quantum computerize" those earlier methods.

Often in the literature, the word "model" is used synonymously with "CB net" and the word "structure" is used synonymously with the bare "graph" G, which is the CB net without the associated transition probabilities.

The Bayesian methods for calculating $P(G|D)$ that we will discuss in this patent assume a "meta" CB net to predict $P(G|D)$ for a CB net with graph G. The meta CB nets usually assumed have a "modular" pattern. Two types of modular meta CB nets have been studied in the literature. We will call them in this patent unordered modular and ordered modular models although unordered modular models are more commonly called structural modular models.

Calculations with unordered modular models require that sums $\sum_G$ over graphs G be performed. Calculations with ordered modular models require that, besides sums $\sum_G$, sums $\sum_\sigma$ over "orders" σ be performed. In some methods these two types of sums are performed deterministically; in others, they are both performed by doing MCMC sampling of a probability distribution. Some hybrid methods perform some of those sums deterministically and others by sampling.

One of the first papers to propose unordered modular models appears to be **Ref.CoHe** by Cooper and Herskovits. Their paper proposed performing the $\sum_G$ by sampling.

One of the first papers to propose ordered modular models appears to be **Ref.FrKo** by Friedman and Koller. Their paper proposed performing both $\sum_G$ and $\sum_\sigma$ by sampling.

Later on, **Ref.KoSo** , **Ref.Koi** by Koivisto and Sood proposed a way of doing $\sum_G$ deterministically using a technique they call fast Mobius transform, and performing $\sum_\sigma$ also deterministically by using a technique they call DP (dynamic programming).

Since the initial work of Koivisto and Sood, several workers (see, for example, **Ref.ElWo** , **Ref.HeTi**) have proposed hybrid methods that use both sampling and the deterministic methods of Koivisto and Sood.

So how can one quantum computerize to some extent the earlier classical computer methods for calculating $P(G|D)$? One partial way is to replace sampling with classical computers by sampling with quantum computers (QCs). An algorithm for sampling CB nets on a QC has been proposed by Tucci in **Ref.TucQuibbs** . A second possibility is to replace the deterministic summing of $\sum_G$ or $\sum_\sigma$ by quantum summing of the style discussed in **Ref.Tuc-qSym** , **Ref.Tuc-qMob** , wherein one uses Grover's algorithm and the technique of targeting two hypotheses. This second possibility is what will be discussed as the preferred embodiments of this invention, for both types of modular models.

Finally, let us mention that some earlier workers (see, for example, **Ref.WiKaSv** and references therein) have proposed using a quantum computer to do AI related calculations reminiscent of the ones being tackled by this invention. However, the methods proposed by those workers differ greatly from the ones in this invention. Those workers either don't use Grover's algorithm, or if they do, they don't use our techniques of targeting two hypotheses and blind targeting.

BRIEF SUMMARY OF THE INVENTION

A preferred embodiment of the invention is qJennings, a computer program written in Java. Source code for qJennings1.6 is included as an appendix to this patent. qJennings is a "code generator" for generating quantum circuits. The quantum circuits generated by qJennings can be used to calculate the probability $P(G|D)$ of a graph (aka structure) G given data D, for a classical Bayesian network with graph G. qJennings calculates such probabilities for so called ordered modular models. Other embodiments of the invention calculate such probabilities for so called unordered modular models.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG.1 shows a block diagram of a classical computer feeding data to a quantum computer.

FIG.2 shows equations that describe technique of targeting two hypotheses.

FIG.3 shows equations that define certain quantities that arise in the circuit of FIG.**4**.

FIG.4 shows quantum circuit used to generate starting state $|s\rangle$ used in AFGA.

FIG.5 shows equations that describe properties of starting state $|s\rangle$ generated by the circuit of FIG.**4**.

FIG.6 shows quantum circuit of FIG.**4** if one assumes that $\ell \leq 1$, or, equivalently, that no node of the graph G has more than 1 parent, where G is the graph whose probability we are seeking to estimate.

FIG.7 shows **Control Window** of qJennings.

DETAILED DESCRIPTION OF THE INVENTION

This section describes in detail a preferred embodiment of the invention and other possible embodiments of the invention. For a more detailed description of possible embodiments of this invention, see **Ref.Tuc-qJen** , **Ref.Tuc-qLis** , **Ref.TucAFGApat** and references therein.

A preferred embodiment of the invention is qJennings, a computer program written in Java. Source code for qJennings1.6 is included as an appendix to this patent. qJennings is a "code generator" for generating quantum circuits. The quantum circuits generated by qJennings can be used to calculate the probability $P(G|D)$ of a graph (aka structure) G given data D, for a classical Bayesian network with graph G. qJennings calculates such probabilities for so called ordered modular models. Other embodiments of the invention calculate such probabilities for so called unordered modular models.

FIG.**1** is a block diagram of a classical computer feeding data to a quantum computer. Box **100** represents a classical computer. qJennings1.6 software runs inside Box **100**. Box **100** comprises sub-boxes **101**, **102**, **103**. Box **101** represents input devices, such as a mouse or a keyboard. Box **102** comprises the CPU, internal and external memory units. Box **102** does calculations and stores information. Box **103** represents output devices, such as a printer or a display screen. Box **105** represents a quantum computer, comprising an array of quantum bits and some hardware for manipulating the state of those bits.

The remainder of this section is divided into 3 subsections. Subsection (A) describes the quantum circuit generated by qJennings. Subsection (B) describes qJennings's user interface. Subsection (C) discusses other possible embodiments of the invention.

(A)qJennings: Quantum Circuit

In this section, we describe the quantum circuit generated by qJennings. For a more detailed description of the circuit, see **Ref.Tuc-qJen** .

The full algorithm utilizes the original Grover's algorithm or any variant thereof, as long as the algorithm drives a starting state $|s\rangle$ to a target state $|t\rangle$. For concreteness, we will assume for the preferred embodiment of this invention that we are using a variant of Grover's algorithm called AFGA, described in **Ref.TucAFGA** and **Ref.TucAFGApat** .

Consider FIG.**2**.

FIG.**2** describes what we will call "targeting two hypotheses". Targeting two hypotheses is a trick that can sometimes be used when applying Grover's original algorithm or some variant thereof. Sometimes it is possible to arrange things so that the target state is a superposition $a_0|0\rangle + a_1|1\rangle$ of two orthonormal states $|0\rangle$ and $|1\rangle$, so that if we know a_0, we can infer a_1, a type of hypothesis testing with 2 hypotheses. If the target state were just proportional to say $|0\rangle$, then its component along $|0\rangle$ would be 1 after normalization so one wouldn't be able to do any type of amplitude inference.

Suppose z_0, z_1 are complex numbers and $|\chi\rangle$ is an unnormalized state that satisfy **201**. Define p and q by **202**.

Let μ, ν and ω label subsystems. Assume the states $|\psi_0\rangle_\mu$ and $|\psi_1\rangle_\mu$ are orthonormal, the states $|0\rangle_\nu$ and $|1\rangle_\nu$ are orthonormal, and the states $|0\rangle_\omega$ and $|1\rangle_\omega$ are orthonormal.

We want to use AFGA with a starting state given by **203** and a target state given by **204**.

It's easy to check that **205** and **206** are true. $|t\rangle$ only appears in AFGA within the projection operator $|t\rangle\langle t|$, and this projection operator always acts solely on the space spanned by $|t\rangle$ and $|s\rangle$. But $|t\rangle\langle t|$ and $|0\rangle\langle 0|_\omega$ act identically on that space. Hence, for the purposes of AFGA, we can replace $|t\rangle\langle t|$ by $|0\rangle\langle 0|_\omega$. We will call $|0\rangle_\omega$

the "sufficient" target state to distinguish it from the full target state $|t\rangle_{\mu,\nu,\omega}$.

Recall that AFGA converges in order $1/|\langle t|s\rangle|$ steps. From the definitions of $|s\rangle$ and $|t\rangle$, one finds **207**.

Once system (μ,ν,ω) has been driven to the target state $|t\rangle_{\mu,\nu,\omega}$, one can measure the subsystem ν while ignoring the subsystem (μ,ω). If we do so, the outcome of the measurements of ν can be predicted from the partial density matrix **208**. From this density matrix, one gets **209** and **210**.

At first sight, it seems that Grover-like algorithms and AFGA in particular require knowledge of $|\langle t|s\rangle|$. Next, we will describe a technique called "blind targeting" for bypassing that onerous requirement.

For concreteness, we will assume in our discussion below that we are using AFGA and that we are targeting two hypotheses, but the idea of this technique could be carried over to other Grover-like algorithms in a fairly obvious way.

According to **207**, when targeting two hypotheses, $|\langle t|s\rangle| = \sqrt{p}$. Suppose we guess-timate p, and use that estimate and the AFGA formulas of **Ref.TucAFGA** to calculate the various rotation angles α_j for $j = 0,1,\ldots,N_{Gro}-1$, where N_{Gro} is the number of Grover steps. Suppose N_{Gro} is large enough. Then, in the unlikely event that our estimate of p is perfect, as $j \to N_{Gro}-1$, $\hat{s}_j$ will converge to $\hat{t}$. On the other hand, if our estimate of p is not perfect but not too bad either, we expect that as $j \to N_{Gro}-1$, the point $\hat{s}_j$ will reach a steady state in which, as j increases, $\hat{s}_j$ rotates in a small circle in the neighborhood of $\hat{t}$. After steady state is reached, all functions of $\hat{s}_j$ will vary periodically with j.

Suppose we do AFGA with p fixed and with $N_{Gro} = (N_{Gro})_0 + r$ Grover steps where $r = 0,1,\ldots N_{tail}-1$. Call each r a "tail run", so p is the same for all N_{tail} tail runs, but N_{Gro} varies for different tail runs. Suppose that steady state has already been reached after $(N_{Gro})_0$ steps. For any quantity Q_r where $r = 0,1,\ldots N_{tail}-1$, let $\langle Q\rangle_{LP}$ denote the outcome of passing the N_{tail} values of Q_r through a low pass filter that takes out the AC components and leaves only the DC part. For example, $\langle Q\rangle_{LP}$

might equal $\sum_r Q_r/N_{tail}$ or $[\max_r Q_r + \min_r Q_r]/2$. By applying the SEO of tail run r to a quantum computer several times, each time ending with a measurement of the quantum computer, we can obtain values $P_r(0)$ and $P_r(1)$ of $P(0)$ and $P(1)$ for tail run r. Then we can find $\left\langle \sqrt{P(1)/P(0)} \right\rangle_{LP} = \langle |z_1|/|z_0| \rangle_{LP}$. But we also expect to know $|z_0|$, so we can use $\langle |z_1|/|z_0| \rangle_{LP} |z_0|$ as an estimate of $|z_1|$. This estimate of $|z_1|$ and the known value of $|z_0|$ yield a new estimate of $p = |z_1|^2 + |z_0|^2$, one that is much better than the first estimate we used. We can repeat the previous steps using this new estimate of p. Every time we repeat this process, we get a new estimate of p that is better than our previous estimate. Call a "trial" each time we repeat the process of N_{tail} tail runs. p is fixed during a trial, but p varies from trial to trial.

Next consider FIG.**3**.

301 defines a graph G (aka a structure) as an n-tuple of sets pa_j which are subsets of the set $\{0, 1, \ldots, n-1\} = \{0..n-1\}$, where n is the number of vertices (aka nodes) of the graph G that we are trying to discover from the data D. Let $\mathcal{F}$, called a feature set, be a set of graphs G.

302 defines the probability $P(\mathcal{F}|D)$ for unordered modular models. In **302**: An $1_S(x)$ denotes an indicator function: it equals 1 if $x \in S$ and 0 otherwise. $\theta(X)$ is a truth function: it equals 1 if X is true and 0 otherwise. The product $\prod_j$ is over all vertices from 0 to $n-1$. The feature set $\mathcal{F}$ is a cartesian product of sets $\mathcal{F}_j \in 2^{\{0..n-1\}}$. The symbol $\{< j\}$ denotes all integers from 0 to $j-1$. The functions $\beta_j()$ are specified by the user.

303 defines the probability $\overline{P}(\mathcal{F}|D)$ for ordered modular models. In **303**: We put a line over the P for the ordered modular case to distinguish it from the P for the unordered modular case. Sym_n denotes the set of permutations on n letters. x^σ or $\sigma(x)$ denotes the outcome of applying the permutation σ to x. Given any set S and permutation σ, S^σ equals a new set obtained by applying the permutation σ to each element of the original set S. The functions $h()$ in **303** are specified by the user.

Suppose j, S and ℓ satisfy **304**. By $\{0..n-1 \backslash j\}$ we mean all elements of set $\{0..n-1\}$ except for j. Then $h(j|S)$, $\theta_{j|S}$ and $R_y^{j|S}$ are related by **305**. It is convenient to define the parameters $N_2()$ and ϵ by **306**.

The V operators used in FIG.**4** are not unique. Any definition that satisfies **307** and **308** will work in the preferred embodiment of the invention. **Ref.Tuc-qSym** gives a specific definition of the V operators that satisfies **307** and **308**.

FIG.**4** uses multiply controlled rotations with halfmoon vertices. The half-moon vertices are defined by **309**, where H stands for a one-qubit Hadamard matrix.

The goal of the invention is to give a method whereby a user can calculate (1) the probability $P(\mathcal{F}|D)$ given by **302** for unordered modular models and (2) the probability $\overline{P}(\mathcal{F}|D)$ given by **303** for ordered modular models.

302 for unordered modular models consists of a product over j of Mobius transforms. Such Mobius transforms can be calculated using the method described in a previous patent application by Tucci and in the paper **Ref.Tuc-qMob** by Tucci.

The preferred embodiment of this invention is a Java applet called qJennings v1.6 included as an appendix to this patent. qJennings outputs a quantum circuit which can be used to calculate **303** for ordered modular models.

Next consider FIG.**4**.

Our preferred method for calculating $\overline{P}(\mathcal{F}|D)$ consists of applying AFGA using the techniques of targeting two hypotheses and blind targeting. When we apply AFGA, we will use a sufficient target $|0\rangle_\omega$. All that remains for us to do to fully specify our circuit for calculating $\overline{P}(\mathcal{F}|D)$ is to give a circuit for generating $|s\rangle$. That is what FIG.**4** does. FIG.**4** assumes that n, the number of nodes of the graph G being discovered, equals 3 for concreteness. That figure uses fairly standard quantum circuit notation except for the V operators and the "half-moon vertices" which have been defined already in FIG.**3**. The H stands for Hadamard matrix, σ_X for the X Pauli matrix, etc. More detailed explanations of the symbols in FIG.**4** can be found in **Ref.Tuc-qJen** . Note that every horizontal line of FIG.**4** is a qubit.

Let α include all alpha qubits in FIG.**4**. Let β include all beta qubits in FIG.**4**.

Next consider FIG.**5**.

Assuming that the circuit of FIG.**4** is correct, then that circuit will generate the state $|s\rangle$ given by **501**, where $|\chi\rangle$ is an unnormalized state and where **502** through **506** are satisfied. (If there is some small mistake in the circuit of FIG.**4**, then we should be able to find that mistake in the future and make small amendments to FIG.**4** with the goal of generating an $|s\rangle$ that satisfies **501**).

Next consider FIG.**6**.

A serious problem with using the circuit of FIG.**4** for large n is that the number of β qubits grows exponentially with n so the circuit FIG.**4** is too expensive for large n's. However, one can make an assumption which doesn't seem too restrictive, namely that the in-degree (number of parent nodes) ℓ of all nodes of the graph G is $\leq \ell_{max}$, where the bound ℓ_{max} does not grow with n. For example, if $\ell \leq \ell_{max} = 1$ in FIG.**4**, then we can omit all the $\beta_{;2}$ qubits, and the $R_y^{a|\{b,c\}}$ rotations for $a, b, c \in \{0, 1, 2\}$. In other words, FIG.**4** can be simplified to FIG.**6**.

(B)qJennings: User Interface

In this section, we describe qJennings's user interface. For a more detailed description of the interface, see **Ref.Tuc-qLis** .

(B1)Input Parameters

qJennings expects the following inputs:

***n*:** The number of nodes of the graph G being discovered.

ℓ_{max}: An upper bound on the number of parents any node of G is allowed to have.

a subroutine that returns the value of $h(j|S)$ for any $j \in \{0..n-1\}$ and $S \subset \{0..n-1 \backslash j\}$. The demonstration version of qJennings uses a trivial, inconsequential function $h(j|S)$, but this can be changed easily by rewriting or overriding the method that defines $h(j|S)$.

an estimate of $p = |\langle t|s\rangle|^2$

(B2)Output Files

qJennings outputs 3 types of files: a Log File, an English File and a Picture File.

A Log File records all the input and output parameters displayed in the **Control Window** (see section entitled "Control Window"), so the user won't forget them.

An English File gives an "in English" description of a quantum circuit. It completely specifies the output SEO. Each line in it represents one elementary operation, and time increases as we move downwards in the file.

A Picture File partially specifies the output SEO. It gives an ASCII picture of the quantum circuit. Each line in it represents one elementary operation, and time increases as we move downwards in the file. There is a one-to-one onto correspondence between the rows of corresponding English and Picture Files.

English and Picture Files are used in many of my previous computer programs. I've explained those files in detail in previous papers so I won't do so again here. See, for example, **Ref.TucQuibbs** for a detailed description of the content of those files and how to interpret that content.

(B3)Control Window

FIG.**7** shows the **Control Window** for qJennings. This is the main and only window of qJennings (except for the occasional error or advice message window). This window is open if and only if qJennings is running.

The **Control Window** allows the user to enter the following inputs:

File Prefix: Prefix to the 3 output files that are written when the user presses the **Write Files** button. For example, if the user inserts `test` in this text field, the following 3 files will be written:

- `test_qJen_log.txt` This is a Log File.

- `test_qJen_eng.txt` This is an English File
- `test_qJen_pic.txt` This is a Picture File.

Number of Nodes: This equals n.

Maximum Number of Parents: This equals ℓ_{max}. Must have $\ell_{max} \leq n-1$. For demonstration purposes, this Java applet only allows a maximum n of 5 and a maximum ℓ_{max} of 4. However, the applet is based on a class called `JenMain` which does not have these limitations.

Estimate of |*z*_1|^2/|*z*_0|^2: This equals the user's initial estimate of $|z_1|^2/|z_0|^2$.

Maximum Number of Grover Steps: qJennings will stop iterating the AFGA if it reaches this number of iterations.

Gamma Tolerance (degs): This is an angle given in degrees. qJennings will stop iterating the AFGA if the absolute value of γ_j becomes smaller than this tolerance. (γ_j is an angle in AFGA that tends to zero as the iteration index j tends to infinity. γ_j quantifies how close the AFGA is to reaching the target state).

Delta Lambda (degs): This is the angle $\Delta\lambda$ of AFGA, given in degrees.

The **Control Window** displays the following output text boxes.

|*z*_0|^2: This equals $|z_0|^2$, the probability of the "null" hypothesis of the two hypotheses being targeted.

Starting Gamma (degs): This is γ_0, the first γ_j, the γ_j for the first Grover iteration, given in degrees.

Final Gamma (degs): This is the γ_j for the final Grover iteration, given in degrees.

Number of Grover Steps: This is N_{Gro}, the total number of Grover iterations that were performed. It must be smaller or equal to the **Maximum Number**

of Grover Steps. It will be smaller if the **Final Gamma (degs)** reached the **Gamma Tolerance (degs)** before the **Maximum Number of Grover Steps** was reached.

Number of Qubits: This is the total number of qubits for the output quantum circuit.

Number of Elementary Operations: This is the number of elementary operations in the output quantum circuit. Since there are no LOOPs in qJennings v1.6, this is the number of lines in the English File, which equals the number of lines in the Picture File.

(C)Other Embodiments

In this section, we describe other possible embodiments of the invention.

A standard definition in the field of quantum computation is that a qu(d)it is a quantum state that belongs to a d dimensional vector space and a qubit is a qu(d)it with $d = 2$. In quantum error correction (see **Ref.Dev** for an introduction), one distinguishes between 2 types of qu(d)its, physical and logical. A logical qu(d)it consists of a number of physical qu(d)its. It goes without saying that the qu(d)its in the quantum circuit FIG.**4** (or variant thereof) can always be interpreted as logical qu(d)its, and additional gates can be added to FIG.**4** (or variant thereof) with the purpose of performing error correction.

For convenience, the quantum circuits generated by an embodiment of this invention may include gates that act on more than 3 qubits at a time. Such "fat" gates might be judged by some not to be elementary gates as defined earlier in this patent. However, such fat gates should be allowed inside the SEO's covered by this invention for cases in which they are trivially expandable (TE) fat gates. By TE fat gates we mean, fat gates for which there are well known, expanding methods for replacing them by a sequence of gates that are strictly elementary, in the sense that they act on just one or two qubits at a time. Multi-controlled rotations and multiplexors are examples of TE fat gates. In fact, see the Java classes `MultiCRotExpander` and `MultiplexorExpander` and related classes included in the code listing appendix to this patent. These classes automate such expanding methods for multi-controlled rotations and multiplexors.

So far, we have described some exemplary preferred embodiments of this invention. Those skilled in the art will be able to come up with many modifications to the given embodiments without departing from the present invention. Thus, the inventor wishes that the scope of this invention be determined by the appended claims and their legal equivalents, rather than by the given embodiments.