Use of a Quantum Computer and the Quick Medical Reference To Give an Approximate Diagnosis

Robert R. Tucci
P.O. Box 226
Bedford, MA 01730
tucci@ar-tiste.com

July 29, 2008

Abstract

The Quick Medical Reference (QMR) is a compendium of statistical knowledge connecting diseases to findings (symptoms). The information in QMR can be represented as a Bayesian network. The inference problem (or, in more medical language, giving a diagnosis) for the QMR is to, given some findings, find the probability of each disease. Rejection sampling and likelihood weighted sampling (a.k.a. likelihood weighting) are two simple algorithms for making approximate inferences from an arbitrary Bayesian net (and from the QMR Bayesian net in particular). Heretofore, the samples for these two algorithms have been obtained with a conventional “classical computer”. In this paper, we will show that two analogous algorithms exist for the QMR Bayesian net, where the samples are obtained with a quantum computer. We expect that these two algorithms, implemented on a quantum computer, can also be used to make inferences (and predictions) with other Bayesian nets.

1 Introduction

Trying to make inferences based on incomplete, uncertain knowledge is a common everyday problem. Computer scientists have found that this problem can be handled admirably well using Bayesian networks (a.k.a. causal probabilistic networks)\[1\].
Bayesian nets allow one to pose and solve the inference problem in a graphical fashion that possesses a high degree of intuitiveness, naturalness, consistency, reusability, modularity, generality and simplicity.

This paper was motivated by a series of papers written by me, in which I define some nets that describe quantum phenomena. I call them “quantum Bayesian nets” (QB nets). They are a counterpart to the conventional “classical Bayesian nets” (CB nets) that describe classical phenomena. In particular, this paper gives an example of a general technique, first proposed in Ref. [2], of embedding CB nets within QB nets. The reader can understand this paper easily without having to read Ref. [2] first. He might consult Ref. [2] if he wants to understand better the motivation behind the constructs used in this paper and how they can be generalized.

![Figure 1: Bayesian network of the same form as the QMR Bayesian network, but with considerably fewer parent ("diseases") and children ("findings") nodes.](image)

The Quick Medical Reference (QMR) is a compendium of statistical knowledge connecting diseases to symptoms. The original version of QMR was compiled by Miller et al[3]. Shwe et al[4] designed a CB net based on the information of Ref.[3]. The QMR CB net of Shwe et al is of the form shown in Fig[1]. It contains two layers: a top layer of ≈600 parent nodes corresponding to distinct diseases, and a bottom layer of ≈4,000 children nodes corresponding to distinct findings. The inference problem (or, in more medical language, giving a diagnosis) for the QMR is to, given some findings, find the probability of each disease, or at least the more likely diseases.

Making an inference with a CB net usually requires summing over the states of a subset S of the set of nodes of the graph. If each node in S contains just 2 states, a sum over all the states of S is a sum over 2^{|S|} terms. These sums of exponential size are the bane of the Bayesian network formalism. It has been shown that making exact[5] (or even approximate[6]) inferences with a general CB net is NP-hard. In 1988, Lauritzen and Spiegelhalter (LS) devised a technique[7] for making inferences with CB nets for which the subset S is relatively small (for them, S = S_{LS} = the maximal clique of the moralized graph). This led to a resurgence in the use of CB nets, as it allowed the use of nets that hitherto had been prohibitively expensive computationally. According to Ref.[8], for the QMR CB net, |S_{LS}| ≈ 150, so the LS technique does not help in this case. Researchers have found(Ref.[8] gives a nice review of their work) many exact and approximate algorithms for making inferences
from the QMR CB net. Still, all currently known algorithms require performing an exponential number of operations.

Rejection sampling and likelihood weighted sampling (a.k.a. likelihood weighting) are two simple algorithms for making approximate inferences from an arbitrary CB net (and from the QMR CB net in particular). Heretofore, the samples for these two algorithms have been obtained with a conventional “classical computer”. In this paper, we will show that two analogous algorithms exist for the QMR CB net, where the samples are obtained with a quantum computer. We will show that obtaining each sample, for these two algorithms, for the QMR CB net, on a quantum computer, requires only a polynomial number of steps. We expect that these two algorithms, implemented on a quantum computer, can also be used to make inferences (and predictions) with other CB nets.

2 Notation

In this section, we will define some notation that is used throughout this paper. For additional information about our notation, we recommend that the reader consult Ref.[9]. Ref.[9] is a review article, written by the author of this paper, which uses the same notation as this paper.

Let $\mathbb{B} = \{0, 1\}$. As usual, let $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ represent the set of integers (negative and non-negative), real numbers, and complex numbers, respectively. For integers $a$, $b$ such that $a \leq b$, let $\mathbb{Z}_{a,b} = \{a, a+1, \ldots, b-1, b\}$. For any set $S$, let $|S|$ be the number of elements in $S$. The power set of $S$, i.e., the set of all subsets of $S$ (including the empty and full sets), will be denoted by $2^S$. Note that $|2^S| = 2^{|S|}$.

We will use $\theta(S)$ to represent the “truth function”; $\theta(S)$ equals 1 if statement $S$ is true and 0 if $S$ is false. For example, the Kronecker delta function is defined by

$$
\delta_{xy} = \delta(x,y) = \theta(x = y).
$$

Random variables will be represented by underlined letters. For any random variable $x$, $\text{val}(x)$ will denote the set of values that $x$ can assume. Samples of $x$ will be denoted by $x^{(k)}$ for $k \in \mathbb{Z}_{1,N_{sam}}$.

Consider an $n$-tuple $\vec{f} = (f_1, f_2, \ldots, f_n)$, and a set $A \subset \mathbb{Z}_{1,n}$. By $(\vec{f})_A$ we will mean $(f_i)_{i \in A}$; that is, the $|A|$-tuple that one creates from $\vec{f}$, by keeping only the components listed in $A$. If $\vec{f} \in \mathbb{B}^n$, then we will use the statement $\vec{f} = 0$ to indicate that all components of $\vec{f}$ are 0. Likewise, $\vec{f} = 1$ will mean all its components are 1.

For any matrix $A \in \mathbb{C}^{p \times q}$, $A^*$ will stand for its complex conjugate, $A^T$ for its transpose, and $A^\dagger$ for its Hermitian conjugate. When we write a matrix, and leave some of its entries blank, those blank entries should be interpreted as zeros.

For any set $\Omega$ and any function $f : \Omega \rightarrow \mathbb{R}$, we will use $f(x)/(\sum_{x \in \Omega} \text{numerator})$ to mean $f(x)/(\sum_{x \in \Omega} f(x))$. This notation is convenient when $f(x)$ is a long expression that we do not wish to write twice.
Next we explain our notation for quantum circuit diagrams. We label single qubits (or qubit positions) by a Greek letter or by an integer. When we use integers, the topmost qubit wire is 0, the next one down is 1, then 2, etc. *Note that in our quantum circuit diagrams, time flows from the right to the left of the diagram.* Careful: Many workers in Quantum Computing draw their diagrams so that time flows in the opposite direction. We eschew their convention because it forces one to reverse the order of the operators every time one wishes to convert between a circuit diagram and its algebraic equivalent in Dirac notation.

## 3 The QMR CB Net

In this section, we describe the QMR CB net.

![Diagram](image)

### Figure 2: (a) CB net with n parent nodes (“diseases”) for all pointing into a single child node (“finding”). (b) Noisy-or CB net, a special case or an approximation of the CB net of Figure (a).

Before describing the QMR CB net, let us describe the noisy-or CB net (invented by Pearl in Ref.[10]). Consider a CB net of the form of Fig[2](a), consisting of n parent nodes (“diseases”), \( d_j \in \text{Bool} \) with \( j \in \mathbb{Z}_{1,n} \), all pointing into a single child node (“finding”), \( f \in \text{Bool} \). The CB net of Fig[2](a) represents a probability distribution \( P(f, \overline{d}) \) that satisfies:

\[
P(f, \overline{d}) = P(f | \overline{d}) \prod_{j=1}^{n} P(d_j). \tag{1}
\]

We say the probability distribution of Eq.(1) and Fig[2](a) is a **noisy-or** if it also satisfies:

...
\[
P(f, \vec{d}) = \left\{ \sum_{\vec{d}'} P(f|\vec{d}') \prod_j P(d'_j|d_j) \right\} \prod_j P(d_j), \tag{2a}
\]

with
\[
P(f|\vec{d}') = \delta(f, d'_1 \lor d'_2 \lor \cdots \lor d'_n). \tag{2b}
\]

For example, when \(n = 2\),
\[
P(f|\vec{d}') = \begin{cases}
00 & 01 & 10 & 11 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
\end{cases}
\]
\[
= \delta(f, d'_1 \lor d'_2). \tag{3}
\]

Eqs. (2) are represented by Fig. 2(b). Sometimes, one also restricts the distributions \(P(d'_j|d_j)\) to have the special form:
\[
P(d'_j|d_j) = \begin{cases}
d_j \rightarrow \\
d'_j \rightarrow 01 & 10 & 11 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
\end{cases}
\]
\[
= (1 - q_{1j})^d_j \delta^{0}_{d'_j} + (q_{1j}d_j)\delta^{1}_{d'_j}, \tag{5}
\]

where \(q_{1j} \in [0, 1]\). A general distribution \(P(d'_j|d_j)\) would contain 2 degrees of freedom whereas Eq. (6) contains only one, namely \(q_{1j}\). Note that
\[
P(f = 0|\vec{d}) = \prod_j (1 - q_{1j})^{d_j} = e^{-\sum_j \theta_{1j}d_j}, \tag{7}
\]

where \(\theta_{1j} = -\ln(1 - q_{1j}) \geq 0\). The inference problem for the noisy-or CB net consists in calculating \(P(\vec{d}|f = 0)\) and \(P(\vec{d}|f = 1)\); that is, the probability of diseases having the value \(\vec{d}\), given that \(f\) is 0 or 1. This is given by Bayes rule:
\[
P(\vec{d}|f = 0) = \frac{P(f = 0|\vec{d})\prod_j P(d_j)}{P(f = 0)} \tag{8}
= \frac{\prod_j \{(1 - q_{1j})^{d_j}P(d_j)\}}{\sum_{\vec{d}} \text{numerator}}. \tag{9}
\]

Note that the sum in the denominator of Eq. (9) is over \(2^n\) terms.
Now that we understand the noisy-or CB net, it’s easy to understand the QMR CB net. The QMR CB net consists of multiple noisy-or CB nets, one for each finding. Suppose the QMR CB net has $N_D$ diseases (parent nodes), $d_j \in \text{Bool}$ for $j \in Z_{1,N_D}$, and $N_F$ findings (children nodes), $f_i \in \text{Bool}$ for $i \in Z_{1,N_F}$. Then, for each $i \in Z_{1,N_F}$, one has

$$P(f_i = 0|(\vec{d})_{\text{pa}(f_i)}) = \prod_{j \in \text{pa}(f_i)} \left\{(1-q_{ij})^{d_j}\right\} = e^{-\sum_{j \in \text{pa}(f_i)} \theta_{ij}d_j}, \quad (10)$$

where $q_{ij} \in [0, 1]$ and $\text{pa}(f_i) \subset Z_{1,N_D}$ is the set of parents of node $f_i$. Let $I_0$, $I_1$, and $I_{\text{unk}}$ constitute a disjoint partition of $Z_{1,N_F}$. (“unk” stands for unknown.) The inference problem for the QMR CB net consists in calculating $P[\vec{d} | (\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1]$. By Bayes rule,

$$P[\vec{d} | (\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1] = \frac{P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1 | \vec{d}] P(\vec{d})}{P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1]} \quad \text{(11)}$$

$$= \frac{\Pi_1 \Pi_0 P(\vec{d})}{P_{I_1,I_0}}, \quad (12)$$

where

$$\Pi_0 = \prod_{i \in I_0} \prod_{j \in \text{pa}(f_i)} \left\{(1-q_{ij})^{d_j}\right\}, \quad (13)$$

$$\Pi_1 = \prod_{i \in I_1} \left\{1 - \prod_{j \in \text{pa}(f_i)} \left\{(1-q_{ij})^{d_j}\right\}\right\}, \quad (14)$$

and

$$P_{I_1,I_0} = P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1] \quad \text{(15)}$$

$$= \sum_{\vec{d}} P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}] \quad \text{(16)}$$

Note that the numerator of Eq.(12) can be calculated in a polynomial number of steps, but its denominator (i.e., $P_{I_1,I_0}$) is expressed in Eq.(16) as a sum over

---

1 It’s possible to include “leakage” in the definitions of noisy-or and QMR nets, but we won’t include it since it can be ignored without loss of generality. One can add a leakage node $L_i \in \text{Bool}$ pointing into each $f_i$ node, for each $i \in Z_{1,N_F}$. These leakage nodes behave just like disease nodes that are always “turned on” (i.e., set to 1). Then, instead of Eq.(10), one has

$$P(f_i = 0|L_i = 1, (\vec{d})_{\text{pa}(f_i)}) = (1-q_{i0}) \prod_{j \in \text{pa}(f_i)} \left\{(1-q_{ij})^{d_j}\right\} = e^{-\theta_{i0} - \sum_{j \in \text{pa}(f_i)} \theta_{ij}d_j}.$$
2^{ND} terms. Calculating \( P_{I_1,I_0} \) naively, by summing numerically those \( 2^{ND} \) terms, is unfeasible when \( ND \) is large.

At the end of this paper are 4 appendices. Reading them is not a prerequisite to understanding the rest of this paper, but they might be of interest to some readers.

In Appendix A we show that

\[
P_{I_1,I_0} = \sum_{S \subset I_1} (-1)^{|S|} T_{S,I_0},
\]

where \( T : 2^{I_1} \times 2^{I_0} \rightarrow \mathbb{R} \) is some function that can be calculated in a polynomial number of steps. Thus, \( P_{I_1,I_0} \) can be calculated by summing numerically over \( 2^{|I_1|} \) terms, regardless of \( |I_0| \) size. This is better than \( 2^{ND} \) terms, but still unfeasible for \( |I_1| \) large.

Eq. (17) can be inverted. For more on this see Appendix B.

Rejection sampling and likelihood weighted sampling are two algorithms for making approximate inferences from an arbitrary CB net (and from the QMR CB net in particular). Heretofore, the samples for these two algorithms have been obtained with a conventional “classical computer”. In case the reader is not familiar with these two algorithms, in the manner they have been implemented heretofore on a classical computer, see Appendices C and D for an introduction to them. In the next section, we will show that two analogous algorithms exist for the QMR CB net, where the samples are obtained with a quantum computer.

4 Diagnosis Via Quantum Computer

In this section, we will describe a method for making inferences from the QMR using a quantum computer.

A slight change of notation: the parameter \( q_{ij} \in [0,1] \) of the previous section will be replaced in this section by a sine squared. Let

\[
q_{ij} = \sin^2 \alpha_{ij} = S_{ij}^2, \quad 1 - q_{ij} = \cos^2 \alpha_{ij} = C_{ij}^2,
\]

for some real number \( \alpha_{ij} \). We will also abbreviate \( S_{ij} \) by \( S_j \) and \( C_{ij} \) by \( C_j \).

We begin by considering the simple case of a CB net consisting of two diseases pointing to one finding, as displayed in Fig 3(a). We will next show that Fig 3(b) is a quantum circuit that can generate some of the same probability distributions as the CB net Fig 3(a). The state vectors \( |\psi_1\rangle, |\psi_2\rangle \), and the unitary transformations \( A_1, A_2, A_{OR} \) that appear in the quantum circuit of Fig 3(b) are defined as follows.

For \( j = 1,2 \), define \( |\psi_j\rangle \) by

\[
|\psi_j\rangle = U_j |0\rangle,
\]

where
For those familiar with Ref. [2], note that the probability amplitude $A_j(d'_j, \tilde{d}_j|d'_j, d_j)$ is a q-embedding of the probability distribution $P(d'_j|d_j)$ defined in Eq. (6). Note also that source and sink nodes are denoted by letters with tildes over them.

Figure 3: (a) CB net consisting of two diseases pointing to one finding. (b) Quantum circuit that generates some of the same probability distributions as the CB net of Figure (a).

\[
U_j = \begin{bmatrix}
\sqrt{P_{d'_j}(0)} & -\sqrt{P_{d'_j}(1)} \\
\sqrt{P_{d_j}(1)} & \sqrt{P_{d_j}(0)}
\end{bmatrix}, \quad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.
\]

(20)

For $j = 1, 2$, let

\[
A_j(d'_j, \tilde{d}_j|d'_j, d_j) = \begin{cases} \tilde{d}'_j, d_j \to \\
00 & 1 \\
01 & 0 \\
10 & C_j \\
11 & -S_j
\end{cases}
\]

(21)

\[
= \left( (C_j^{d_j})\delta_{d_j}^{d'_j} \delta_{d_j}^{\tilde{d}'_j} + (S_j d_j) \delta_{d_j}^{d'_j} \delta_{d_j}^{\tilde{d}'_j} \right) \delta_{d_j}^{0} + [\ldots] \delta_{d_j}^{1}.
\]

(22)

The matrix given by Eq. (21) is a 2 qubit unitary transformation. Such transformations can be decomposed (compiled) into an expression containing at most 3 CNOTs, using a method due to Vidal and Dawson [11] (For software that performs this decomposition, see Ref. [12]).
In particular, when $f$, note that the probability amplitude $A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \tilde{f}, d'_1, d'_2)$ is a q-embedding of the probability distribution $P(f | d'_1, d'_2)$ defined in Eq. (4).

The matrix given by Eq. (23) can be compiled as follows:

$$A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \tilde{f}, d'_1, d'_2) = \begin{pmatrix}
000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$

(23)

$$= [i^f \delta_{f}^d \delta_{d'_1}^f \delta_{\tilde{e}_2}^\tilde{f}] \delta_{\tilde{f}}^0 \cdot [\ldots] \delta_{\tilde{f}}^1.$$  

(24)

For those familiar with Ref. [2], note that the probability amplitude $A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \tilde{f}, d'_1, d'_2)$ is a q-embedding of the probability distribution $P(f | d'_1, d'_2)$ defined in Eq. (4).

The matrix given by Eq. (23) can be compiled as follows:

$$[A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \tilde{f}, d'_1, d'_2)] = e^{i \frac{\pi}{2} \sigma_X \otimes \sum_{(b, b') \in Bool^2} (P_{b, b'})}$$

(25)

$$= e^{i \frac{\pi}{2} \sigma_X \otimes I_4} e^{-i \frac{\pi}{2} \sigma_X \otimes I_4}$$

(26)

$$= i \sigma_X (2) \left[ -i \sigma_X (2) \right]^{n(0)n(1)}.$$  

(27)

The probability $P(f, \tilde{e}_1, \tilde{e}_2, \tilde{d}_1, \tilde{d}_2)$ for the quantum circuit Fig. 3(b) is given by:

$$P(f, \tilde{e}_1, \tilde{e}_2, \tilde{d}_1, \tilde{d}_2) =$$

$$= \left| \sum_{d'_1, d'_2, d_1, d_2} A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \tilde{f} = 0, d'_1, d'_2) \prod_{j=1,2} \left( A_j(d'_j, \tilde{d}_j | \tilde{d}_j = 0, d_j) \sqrt{P(d_j)} \right) \right|^2$$

(28)

$$= \left| \sum_{d'_1, d'_2, d_1, d_2} i^f \delta_{f}^d \delta_{d'_1}^f \delta_{\tilde{e}_2}^{\tilde{e}} \delta_{\tilde{d}_2}^{\tilde{d}} \prod_{j=1,2} \left( [(C_j^{d_j}) \delta_{d_j}^d \delta_{d'_j}^f + (S_j d_j) \delta_{d_j}^0 \delta_{d'_j}^f] \sqrt{P(d_j)} \right) \right|^2.$$  

(29)

In particular, when $f = 0$,

$$P(f = 0, \tilde{e}_1, \tilde{e}_2, \tilde{d}_1, \tilde{d}_2) = \prod_{j=1,2} C_j^{2\tilde{d}_j} P(\tilde{d}_j) \delta_{\tilde{e}_j}^0.$$  

(30)

If $\tilde{e}_1$ and $\tilde{e}_2$ are not observed, we may sum over them to get

$$P(f = 0, \tilde{d}_1, \tilde{d}_2) = \prod_{j=1,2} C_j^{2\tilde{d}_j} P(\tilde{d}_j).$$  

(31)
If we replace $\tilde{d}_j$ by $d_j$, $P(f = 0, \tilde{d}_1, \tilde{d}_2)$ for the quantum circuit Fig 3(b) is identical to $P(f = 0, d_1, d_2)$ for the CB net Fig 3(a). This is no coincidence. The quantum circuit was designed from the CB net to make this true. In a sense defined in Ref. [2], the CB net is embedded in the quantum circuit.

![Diagram](image)

**Figure 4:** (a) QMR-like CB net with two diseases and three finding. (b) Quantum circuit that generates some of the same probability distributions as the CB net of Figure(a).

One can easily generalize this example with $N_D = 2$ and $N_F = 1$ to arbitrary $N_D$ and $N_F$. Fig. 4 gives an example with $N_D = 2$ and $N_F = 3$. In the example with $N_D = 2, N_F = 1$, we set:

$$[A_{OR}(f, \tilde{e}_1, \tilde{e}_2 | \vec{f}, d'_1, d'_2)] = i\sigma_X(2)[-i\sigma_X(2)]^{n_0 n_1}. \quad (32)$$

For arbitrary $N_D, N_F$, this equation can be generalized to:

$$[A_{OR}(f_i, \{\tilde{e}_j\}_{j \in pa(f_i)}, \vec{f}_i, \{d'_j\}_{j \in pa(f_i)}] = i\sigma_X(\tau_i)[-i\sigma_X(\tau_i)]^{\prod_{k \in K_i} n_k}. \quad (33)$$

for $i \in Z_{1,N_F}$, where $\tau_i$ is the qubit label of qubit $f_i$, and $K_i$ is the set of qubit labels for the parents of qubit $f_i$.

For arbitrary $N_D, N_F$, we can generalize this construction to obtain a quantum circuit that yields probabilities $P(\vec{f}, \vec{e}, \vec{d})$. If the external outputs $\vec{e}$ are not observed, then we measure $P(\vec{f}, \vec{d})$. If we replace $\vec{d}$ by $\vec{d}'$, the probability $P(\vec{f}, \vec{d})$ for the quantum circuit is identical to the probability $P(\vec{f}, \vec{d}')$ for the CB net that was embedded in that quantum circuit. As discussed previously, the inference problem for the CB net is to find $P[\vec{d}(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1]$. This probability equals $P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}]$ divided by $P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}]$ can
be calculated exactly numerically on a conventional classical computer. Not so the
denominator $P[(f)_{I_0} = 0, (f)_{I_1} = 1]$, at least not for large $|I_1|$. Here is where the
quantum computer shows its mettle. One can run the quantum circuit many times,
in either of two modes, to get a so called empirical distribution that approximates
$P[\hat{d}[(f)_{I_0} = 0, (f)_{I_1} = 0]$. The empirical distribution converges to the exact one. The
two modes that we are referring to are rejection sampling and likelihood weighted
sampling. We describe each of these separately in the next two sections.

4.1 Rejection Sampling

Assume that we are given the number of samples $N_{sam}$ that we intend to collect,
and the sets $I_0, I_1, I_{unk}$ which are a disjoint partition of $Z_{1,N_F}$. Then the rejection
sampling algorithm goes as follows (expressed in pseudo-code, pidgin C language):

```
For all $\vec{d}$ \{$W(\vec{d}) = 0; \}
W_{tot} = 0;
For samples $k = 1, 2, \ldots, N_{sam}$ \{ 
    Generate ($d^{(k)}, f^{(k)}$) with quantum computer;
    If ($f^{(k)}_{I_0} = 0$ and $f^{(k)}_{I_1} = 1$) { //rejection here
        If $\vec{d}^{(k)} = \vec{d}$ \{W($\vec{d}) + +; \}
        W_{tot} + +;
    }
\} //k loop (samples)
For all $\vec{d}$ \{$P[\vec{d} | (f)_{I_0} = 0, (f)_{I_1} = 1] = \frac{W(\vec{d})}{W_{tot}}; \}$
```

A convergence proof of this algorithm goes as follows. For any function $g : \text{Bool}^{N_D+N_F} \rightarrow \mathbb{R}$, as $N_{sam} \rightarrow \infty$, the sample average $g(\vec{d}^{(k)}, f^{(k)})$ tends to:

$$g(\vec{d}^{(k)}, f^{(k)}) = \frac{1}{N_{sam}} \sum_k g(\vec{d}^{(k)}, f^{(k)}) \rightarrow \sum_{\vec{d}, \vec{f}} P(\vec{d}, \vec{f}) g(\vec{d}, \vec{f}) . \quad (34)$$

Therefore,

$$\frac{W(\vec{d})}{W_{tot}} = \frac{1}{N_{sam}} \sum_k \delta^0_{(f^{(k)})_{I_0}} \delta^1_{(f^{(k)})_{I_1}} \delta \vec{d} \delta \vec{f}^{(k)} \rightarrow \sum_{\vec{d}, \vec{f}} P(\vec{d}, \vec{f}) \delta^0_{(f)_{I_0}} \delta^1_{(f)_{I_1}} \delta \vec{d} \delta \vec{f} \quad (35)$$

$$\rightarrow P[\hat{d}[f]_{I_0} = 0, (f)_{I_1} = 1] . \quad (37)$$
4.2 Likelihood Weighted Sampling

For likelihood weighted sampling, the quantum circuit must be modified as follows: We assume that all gates in the quantum circuit are elementary; that is, either single-qubit transformations or controlled elementary gates (like CNOTs or multiply-controlled NOTs or multiply-controlled phases).

1. For any qubit \( f_i \) with \( i \in I_1 \), initialize the qubit to state \( |1\rangle \). (For any qubit \( f_i \) with \( i \in I_0 \), initialize the qubit to state \( |0\rangle \), same as before.)

2. For any qubit \( f_i \) with \( i \in I_0 \cup I_1 \), remove those elementary gates that can change the state of \( f_i \). In particular, remove any single-qubit gates acting on \( f_i \), and any controlled elementary gates that use \( f_i \) as a target. Do not remove controlled elementary gates that use \( f_i \) as a control only.

Assume that we are given the number of samples \( N_{sam} \) that we intend to collect, and the sets \( I_0, I_1, I_{unk} \) which are a disjoint partition of \( Z_{1,N_F} \). Then the likelihood weighted sampling algorithm goes as follows (expressed in pseudo-code, pidgin C language):

```
For all \( \vec{d} \) \{ \( W(\vec{d}) = 0 \); \\
    \( W_{tot} = 0 \); \\
    For samples \( k = 1, 2, \ldots, N_{sam} \) \{ \\
        Generate \( (\vec{d}^{(k)}, \vec{f}^{(k)}) \) subject to \( (f)_I = 0 \), \( (f)_I = 1 \) with quantum computer; \\
        \( L = \prod_{i \in I_0} P[f_i^{(k)} = 0 | (\vec{d}^{(k)})_{\text{pa}(f_i)}] \prod_{i \in I_1} P[f_i^{(k)} = 1 | (\vec{d}^{(k)})_{\text{pa}(f_i)}] \); \\
        If \( \vec{d}^{(k)} = \vec{d} \{ W(\vec{d}) + = L; \} \)
        \( W_{tot} + = L; \) \\
    } //k loop (samples) \\
For all \( \vec{d} \) \{ \( P[\vec{d} | (\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1] = \frac{W(\vec{d})}{W_{tot}} \); \}
```

A convergence proof of this algorithm goes as follows. Define the likelihood functions \( L_{evi} \) and \( L_{unk} \) by (“evi” stands for evidence and “unk” for unknown):

\[
L_{evi}(\vec{d}) = \prod_{i \in I_0} P[f_i = 0 | (\vec{d})_{\text{pa}(f_i)}] \prod_{i \in I_1} P[f_i = 1 | (\vec{d})_{\text{pa}(f_i)}], \quad (38)
\]

and

\[
L_{unk}(\vec{d}, \vec{f}) = \prod_{i \in I_{unk}} P[f_i | (\vec{d})_{\text{pa}(f_i)}]. \quad (39)
\]

Clearly,

\[
P(\vec{d}, \vec{f}) = L_{evi}(\vec{d})L_{unk}(\vec{d}, \vec{f})P(\vec{d}). \quad (40)
\]
For any function $g : \text{Bool}^{N_D + N_F} \to \mathbb{R}$, as $N_{\text{sam}} \to \infty$, the sample average $g(\vec{d}^{(k)}, \vec{f}^{(k)})$ tends to:

$$g(\vec{d}^{(k)}, \vec{f}^{(k)}) = \frac{1}{N_{\text{sam}}} \sum_k g(\vec{d}^{(k)}, \vec{f}^{(k)}) \to \sum_{\vec{d}, \vec{f}} \delta^0 (\vec{f})_{I_0} \delta^1 (\vec{f})_{I_1} L_{\text{unk}}(\vec{d}, \vec{f}) P(\vec{d}) g(\vec{d}, \vec{f}) .$$

Therefore,

$$W(\vec{d}) = \frac{1}{N_{\text{sam}}} \sum_k L_{\text{evi}}(\vec{d}^{(k)}) \delta_{\vec{d}^{(k)}} \to \frac{1}{N_{\text{sam}}} \sum_k L_{\text{evi}}(\vec{d}^{(k)})$$

$$\to \sum_{\vec{d}, \vec{f}} P(\vec{d}, \vec{f}) \delta^0 (\vec{f})_{I_0} \delta^1 (\vec{f})_{I_1} \delta_{\vec{d}}$$

$$\to P[\vec{d}((\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1] .$$

### A Appendix: Summing $P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}]$ over $\vec{d}$

In this appendix, we will sum $P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}]$ over $\vec{d}$. This is like performing a multidimensional integral.

Recall that $P_{I_1, I_0}$ was defined as:

$$P_{I_1, I_0} = \sum_{\vec{d}} P[(\vec{f})_{I_0} = 0, (\vec{f})_{I_1} = 1, \vec{d}] .$$

For all $i \in Z_{1,N_F}$ and $j \in Z_{1,N_D}$, let

$$(\vec{h}_i)_j = \theta_{ij} \theta[j \in pa(f_j)] = \begin{cases} \theta_{ij} & \text{if } j \in pa(f_i) \\ 0 & \text{otherwise} \end{cases} .$$

For all $j \in Z_{1,N_D}$, we can always find $\alpha_j, \beta_j \in \mathbb{R}$ so that $P(d_j)$ can be expressed as:

$$P(d_j) = e^{-\alpha_j - \beta_j d_j} .$$

Now $\Pi_0$ (defined by Eq.(13)), $\Pi_1$ (defined by Eq.(14)), and $P(\vec{d})$ can be expressed as:

$$\Pi_0 = \prod_{i \in I_0} e^{-\vec{h}_i \cdot \vec{d}},$$

$$\Pi_1 = \prod_{i' \in I_1} \{1 - e^{-\vec{h}_{i'} \cdot \vec{d}}\} ,$$

$$P(\vec{d}) = e^{-\vec{h} \cdot \vec{d}} ,$$

$$P(\vec{d}) = e^{-\vec{h} \cdot \vec{d}} .$$
and
\[ P(\vec{d}) = e^{-\alpha - \vec{\beta} \cdot \vec{d}} \text{, where } \alpha = \sum_{j=1}^{N_D} \alpha_j \text{.} \] (50)

Thus
\[
P_{I_1, I_0} = \sum_{\vec{d}} \Pi_0 \Pi_1 P(\vec{d})
= e^{-\alpha} \sum_{\vec{d}} e^{-\vec{\beta} \cdot \vec{d} - \sum_{i \in I_0} \vec{h}_i \cdot \vec{d}} \prod_{i' \in I_1} \left\{ 1 - e^{-\vec{h}_{i'} \cdot \vec{d}} \right\} \text{.} \] (52)

Consider any set \( \Omega \) and any function \( f: \Omega \to \mathbb{R} \). When \( \Omega = \{a, b\} \),
\[
(1 - e^{-f(a)})(1 - e^{-f(b)}) = 1 - e^{-f(a)} - e^{-f(b)} + e^{-f(a) - f(b)} \text{.} \] (53)

This generalizes to
\[
\prod_{x \in \Omega} \left\{ 1 - e^{-f(x)} \right\} = \sum_{S \in 2^\Omega} (-1)^{|S|} e^{-\sum_{x \in S} f(x)} \text{.} \] (54)

Using identity Eq.(54), Eq.(52) yields
\[
P_{I_1, I_0} = e^{-\alpha} \sum_{S \subset I_1} (-1)^{|S|} \sum_{\vec{d}} e^{-\vec{\beta} \cdot \vec{d} - \sum_{i \in I_0 \cup S} \vec{h}_i \cdot \vec{d}} \text{.} \] (55)

For any \( j \in Z_{1,N_D} \) and \( A \subset Z_{1,N_F} \), define
\[
\phi_j(A) = \beta_j + \sum_{i \in A} (\vec{h}_i)_j \text{.} \] (56)

Also define a function \( t: \mathbb{R} \to \mathbb{R} \) by
\[
t(\phi) = \frac{1}{2} \sum_{d=0}^{1} e^{-\phi d} = \frac{1}{2} (1 + e^{-\phi}) \text{.} \] (57)

Using these definitions, Eq.(55) yields
\[
P_{I_1, I_0} = e^{-\alpha} \sum_{S \subset I_1} (-1)^{|S|} 2^{N_D} \prod_{j \in Z_{1,N_D}} t[\phi_j(I_0 \cup S)] \text{.} \] (58)
\[
= \sum_{S \subset I_1} (-1)^{|S|} T_{S,I_0} \text{.} \] (59)

\( T \) is a function \( T: 2^{I_1} \times 2^{I_0} \to \mathbb{R} \) defined by the last equation.
B Appendix: Mobius Inversion Theorem and Eq.(17)

In this Appendix, we discuss the application of the Mobius Inversion Theorem\cite{13} to Eq.(17).

\begin{align*}
\left[ P_{S_1, S_0} \right] &= S_1 \varepsilon 2^{I_1} \\
&\quad \rightarrow S_0 \varepsilon 2^{I_0}
\end{align*}

Figure 5: The matrix $P_{S_1, S_0} = P[(\vec{f})_{S_1} = 1, (\vec{f})_{S_0} = 0]$ for all $S_1 \in 2^{I_1}$ and $S_0 \in 2^{I_0}$.

Fig 5 shows the matrix $P_{S_1, S_0} = P[(\vec{f})_{S_1} = 1, (\vec{f})_{S_0} = 0]$ for all $S_1 \in 2^{I_1}$ and $S_0 \in 2^{I_0}$, assuming large $|I_1|$ but arbitrarily $|I_0|$. We label the rows and columns of $P_{S_1, S_0}$ in order of increasing set size. The top-left corner entry is $P_{\emptyset, \emptyset} = 1$ and the bottom-right corner entry is $P_{I_1, I_0}$. Note that $P_{S_1, S_0} \geq P_{I_1, I_0}$ for all $S_1 \subset I_1$, $S_0 \subset I_0$. The shaded top part (corresponding to small or moderate $|S_1|$) of this matrix can be calculated numerically with a classical computer. But not the unshaded bottom part (corresponding to large $|S_1|$). An empirical approximation of the bottom part can be obtained with a quantum computer.

Consider any set $J$ and any functions $f, g : 2^J \rightarrow \mathbb{C}$. The Mobius Inversion Theorem\cite{13} states that

\begin{align*}
g(J) &= \sum_{J' \subset J} (-1)^{|J-J'|} f(J') \quad \iff \quad f(J) = \sum_{J' \subset J} g(J') \ . \quad (60)
\end{align*}

Using the fact that when $J' \subset J$, $|J-J'| = |J| - |J'|$, and replacing $g(J)$ by $(-1)^{|J|}g(J)$ in the previous equation, we get

\begin{align*}
g(J) &= \sum_{J' \subset J} (-1)^{|J'|} f(J') \quad \iff \quad f(J) = \sum_{J' \subset J} (-1)^{|J'|} g(J') \ . \quad (61)
\end{align*}

We showed in Appendix A that

\begin{align*}
P_{I_1, I_0} &= \sum_{S_1 \subset I_1} (-1)^{|S_1|} T_{S_1, I_0} \ . \quad (62)
\end{align*}
Thus, by virtue of Eq. (61),

\[ T_{I_1,I_0} = \sum_{S_1 \subset I_1} (-1)^{|S_1|} P_{S_1,I_0}. \]  

(63)

More generally, if \( S'_1 \subset I_1, S_0 \subset I_0, \) and

\[ M_{S'_1,S_1} = (-1)^{S_1}, \]  

(64)

then

\[ P_{S'_1,S_0} = \sum_{S_1 \subset S'_1} M_{S'_1,S_1} T_{S_1,S_0}, \]  

(65)

and

\[ T_{S'_1,S_0} = \sum_{S_1 \subset S'_1} M_{S'_1,S_1} P_{S_1,S_0}. \]  

(66)

Eq. (63) implies

\[ P_{I_1,I_0} = (-1)^{|I_1|} \left\{ T_{I_1,I_0} - \sum_{S_1 \not\subset I_1} (-1)^{|S_1|} P_{S_1,I_0} \right\}. \]  

(67)

To approximate \( P_{I_1,I_0} \), one can estimate the right hand side of the last equation. \( T_{I_1,I_0} \) and \( P_{S_1,I_0} \) for small and moderate \(|S_1|\), can be calculated exactly numerically on a classical computer. \( P_{S_1,I_0} \) for large \(|S_1|\) can be obtained empirically on a quantum computer.

**C Appendix: Rejection Sampling for CB Nets on a Classical Computer**

In this Appendix, we review the rejection sampling algorithm for arbitrary CB nets on a classical computer.

Consider a CB net whose nodes are labeled in topological order by \( Z_1, Z_2, \ldots, Z_{N_{\text{nds}}} \). Assume that \( E \) (evidence set) and \( H \) (hypotheses set) are disjoint subsets of \( Z_{1,N_{\text{nds}}} \), with \( Z_{1,N_{\text{nds}}} - E \cup H \) not necessarily empty. Assume that we are given the number of samples \( N_{\text{sam}} \) that we intend to collect, and the prior evidence \((x)_E\). Then the rejection sampling algorithm goes as follows (expressed in pseudo-code, pidgin C language):
For all \((x)_H\) \(\{W[(x)_H] = 0;\}\)
\(W_{\text{tot}} = 0;\)
For samples \(k = 1, 2, \ldots, N_{\text{sam}}\)\
For nodes \(i = 1, 2, \ldots, N_{\text{nds}}\)\
Generate \(x_i^{(k)}\) from \(P[x_i|(x)^{(k)}_{\text{pa}(x_i)}];\)\
//Note that \(\text{pa}(x_i) \subset Z_{1,i-1}\) so\
//\((x)^{(k)}_{\text{pa}(x_i)}\) has been calculated at this point\
} //\(i\) loop (nodes)
If \((x)^{(k)}_E = (x)_E\) { //rejection here
If \((x)^{(k)}_H = (x)_H\{W[(x)_H] + +;\}\)
\(W_{\text{tot}} + +;\)
}
} //\(k\) loop (samples)
For all \((x)_H\) \(\{P[(x)_H|(x)_E] = \frac{W[(x)_H]}{W_{\text{tot}}};\}\)

A convergence proof of this algorithm goes as follows. For any function \(g : \text{val}(x) \rightarrow \mathbb{R}\), as \(N_{\text{sam}} \rightarrow \infty\), the sample average \(g(x^{(k)})\) tends to:

\[
\overline{g(x^{(k)})} = \frac{1}{N_{\text{sam}}} \sum_k g(x^{(k)}) \rightarrow \sum x' g(x').
\] (68)

Therefore,

\[
\frac{W[(x)_H]}{W_{\text{tot}}} = \frac{1}{N_{\text{sam}}} \sum_k \delta[(x)_E \cup H, (x^{(k)}_E \cup H)]
\]

\[
= \frac{1}{N_{\text{sam}}} \sum_k \delta[(x)_E, (x^{(k)}_E)]
\]

\[
\rightarrow \sum x' P(x') \delta[(x)_E \cup H, (x')_E \cup H]
\]

\[
= \sum x' P(x') \delta[(x)_E, (x')_E]
\]

\[
\rightarrow \frac{P[(x)_E \cup H]}{P[(x)_E]}.
\] (71)

**D Appendix: Likelihood Weighted Sampling for CB Nets on a Classical Computer**

In this Appendix, we review the likelihood weighted sampling algorithm for arbitrary CB nets on a classical computer\[14, 15\].

Consider a CB net whose nodes are labeled in topological order by \(x_1, x_2, \ldots, x_{N_{\text{nds}}} = x\). Assume that \(E\) (evidence set) and \(H\) (hypotheses set) are disjoint subsets of \(Z_{1,N_{\text{nds}}}\), with \(Z_{1,N_{\text{nds}}} - E \cup H\) not necessarily empty. Let \(X^c = Z_{1,N_{\text{nds}}} - X\) for any \(X \subset Z_{1,N_{\text{nds}}}.\) Assume that we are given the number of samples \(N_{\text{sam}}\) that we intend to collect, and
the prior evidence \((x)_E\). Then the likelihood weighted sampling algorithm goes as follows (expressed in pseudo-code, pidgin C language):

```
For all \((x)_H\) \{\(W[(x)_H] = 0;\)\}
\(W_{tot} = 0;\)
For samples \(k = 1, 2, \ldots, N_{sam}\)\{
    \(L = 1;\)
    For nodes \(i = 1, 2, \ldots, N_{nds}\)\{
        If \(i \in E^c\)\{
            Generate \(x_i^{(k)}\) from \(P[x_i | (x)_E^{pa(x_i)}]\);
            //Note that \(pa(x_i) \subset Z_{1,i_{-1}}\) so
            //\((x)_E^{pa(x_i)}\) has been calculated at this point
        }else if \(i \in E\)\{
            \(x_i^{(k)} = x_i;\) //\((x)_E\) known
            \(L* = P[x_i | (x)_E^{pa(x_i)}];\)
        }
    } //i loop (nodes)
If \((x)_H = (x)_H\{W[(x)_H] += L;\}\)
\(W_{tot} += L;\)
} //k loop (samples)
For all \((x)_H\) \{\(P[(x)_H | (x)_E] = \frac{W[(x)_H]}{W_{tot}};\)\}
```

A convergence proof of this algorithm goes as follows. Define the likelihood function:

\[
L_A(x) = \prod_{i \in A} P[x_i | (x)_{pa(x_i)}] \tag{72}
\]

for any \(A \subset Z_{1,N_{nds}}\). Clearly,

\[
P(x) = L_E(x)L_{E^c}(x) . \tag{73}
\]

For any function \(g : val(x) \rightarrow \mathbb{R}\), as \(N_{sam} \rightarrow \infty\), the sample average \(\overline{g(x^{(k)})}\) tends to:

\[
\overline{g(x^{(k)})} = \frac{1}{N_{sam}} \sum_k g(x^{(k)}) \rightarrow \sum_{x'} L_{E^c}(x')\delta[(x)_E, (x')_E]g(x') . \tag{74}
\]

Therefore,
\[
\frac{W[(x)_H]}{W_{tot}} = \frac{1}{N_{sam}} \sum_k L_E(x^{(k)}) \delta[(x)_H, (x^{(k)})_H] \tag{75}
\]

\[
\rightarrow \sum_{x'} P(x') \delta[(x)_{E \cup H}, (x')_{E \cup H}] \tag{76}
\]

\[
\rightarrow \frac{P[(x)_{E \cup H}]}{P[(x)_E]} \tag{77}
\]

**References**

[1] M. Jordan (Editor) *Learning in Graphical Models* (1999, The MIT Press)

[2] R.R. Tucci, “Quantum Computer as a Probabilistic Inference Engine”, [arXiv:quant-ph/0004028 v2](http://arxiv.org/abs/quant-ph/0004028).

[3] R.A. Miller, F.E. Fasarie, J.D. Myers, *Medical Computing* 3(1986)34

[4] M. Shwe, G. Cooper, *Computers and Biomedical Research* 24(1991)453

[5] G. Cooper, *Artif. Intelligence* 42(1990)393

[6] P. Dagum, M. Luby, *Artif. Intelligence* 60(1993)141

[7] S. Lauritzen, D. Spiegelhalter, *Jour. of the Royal Statistical Society* B50(1988)157

[8] T.S. Jaakkola, M.I. Jordan, *Jour. of Artif. Int. Reasearch* 10(1999)291.

[9] R.R.Tucci, “QC Paulinesia”, [quant-ph/0407215](http://arxiv.org/abs/quant-ph/0407215).

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, (1988, Morgan Kaufmann, San Mateo, CA)

[11] G. Vidal, C.M. Dawson, “A Universal Quantum Circuit for Two-qubit Transformations with 3 CNOT Gates”, [quant-ph/0307177](http://arxiv.org/abs/quant-ph/0307177).

[12] R.R. Tucci, “An Introduction to Cartan’s KAK Decomposition for QC Programmers”, [arXiv:quant-ph/0507171](http://arxiv.org/abs/quant-ph/0507171). Software available at [www.ar-tiste.com](http://www.ar-tiste.com) in the m-fun section.

[13] R.R. Tucci, “Factorization of Density Matrices According to Bayesian and Markov Networks”, [arXiv:quant-ph/0701201](http://arxiv.org/abs/quant-ph/0701201).

[14] R. Fung, KC Chang, Proceedings of the Fifth Conference on Artificial Intelligence (UAI-89), pg 475

[15] R. Shachter, M. Peot, Proceedings of the Fifth Conference on Artificial Intelligence (UAI-89), pg 311