Local Quantum Computing for Fast Probably MAP Inference in Graphical Models

Charles Fox, Iead Rezek, Stephen Roberts
Robotics Research Group
Department of Engineering Science
University of Oxford
charles@robots.ox.ac.uk

Abstract

Maximum a Posteriori (MAP) inference in graphical models is a fundamental task but is generally NP-hard. We present a quantum computing algorithm to speed up this task. The algorithm uses only small, local operators, and is based on a physical analogy of striking the nodes in a net with superposed ‘coolants’. It may be viewed as a quantum version of the Gibbs sampler, making use of entanglement and superposition to represent the entire joint over \( N \) variables using only \( N \) physical nodes in \( 2^N \) states of superposition, and exploring all MCMC trajectories simultaneously. It requires information to be carried away by the coolants, so making essential use of decoherence – which is usually thought of as a hindrance rather than a feature in quantum computation. Phase information is not used.

We work with an undirected, possibly loopy, pairwise Markov Random Field (MRF) with \( N \) boolean-valued nodes \( X_i \). Any graphical model can be converted into this form (e.g. [11]). Let nodes have potentials \( \phi_{ii} \) and links have potentials \( \phi_{ij} \) so that the joint is

\[
P(x_1:N) = \frac{1}{Z} \prod_i \phi_{ii} \prod_{ij} \phi_{ij}
\]

where \( Z \) is a normalization coefficient. We restrict our discussion to potentials which may be defined by integer-valued energies \( E_{ii} \) and \( E_{ij} \) such that

\[
\phi_{ii} = \exp(-E_{ii}), \phi_{ij} = \exp(-E_{ij})
\]

We wish to find the MAP configuration \( \hat{x}_{1:N} \) to maximize \( P(x_{1:N}) \). This is generally NP-hard [17].

1.1 Overview of quantum computing

We briefly review some key concepts in quantum computation. For a full introduction see [13].

A classical register of bits exists in a single state, for example, a three-bit register could store a configuration of three boolean MRF nodes such as 101 (also notated as the decimal 5). Such classical states may also be thought of as basis vectors in \( 2^3 \)-dimensional boolean space. To emphasize this view we use ‘ket’ notation:

\[
|101\rangle \equiv [5] \equiv [0000100]^T
\]
Operations that map register states to register states (such as a particular state-change in a Gibbs sampler) may then be represented by a $2^N$ permutation matrix.

When two registers are brought together and considered as a single system, the tensor product of their state vectors gives the state vector for the combined system, e.g.:

$$|101⟩ ⊗ |001⟩ \equiv |101001⟩$$

Quantum computing allows systems to exist at any point on the unit radius hypersphere in the state space, rather than just the classical basis vectors. For example a register can be in a superposition of basis states $|000⟩$ and $|110⟩$:

$$|Ψ⟩ = α_{000}|000⟩ + α_{110}|110⟩ \\text{where the α are chosen to describe a hypersphere location, } \sum_1 |α_i|^2 = 1, \text{ and each } |α_i|^2 \text{ describes the probability of an observation collapsing the register into state } |i⟩. \text{ In general the } α_i \text{ may be complex, but our algorithm uses positive real values only. General unitary matrices may operate on the state vectors rather than just the special permutation cases. (Unitary matrices have a ‘reversible’ or ‘detailed balance’ property, and may be thought of as rotations on the hypersphere.)}$$

Our notation uses upper and lower case letters in bare and ket forms as follows: $A$ refers to a state space; $|A⟩$ is a general vector in that space; $a$ is an integer (in decimal or binary notation) denoting a basis state; $|a⟩$ is the basis vector in $A$ with coding $a$. (Such binary codings will be used to represent boolean MRF configurations.) We write true and cooled model probabilities as $P(a)$ and $Q(a)$ in contrast to quantum observation probabilities $Pr_{obs}(a) = |α_a|^2$.

### 1.2 Representing joints by superpositions

A quantum register of $N$ qubits (‘quantum bits’) is able to store and represent amplitudes all its $2^N$ possible configurations simultaneously. A fundamental problem in machine learning is that the joint distribution of $N$ variables has exponential size. The structure of quantum amplitudes is very similar to that of joints so we can use the $α_i$ to store our probabilities (up to normalization; recall that our probabilities normalize to 1 but amplitudes normalize so that $\sum_1 |α_i|^2 = 1$), requiring only $N$ qubits instead of exponential resources. We will try to set the $α_i$ so that $α_i = \frac{1}{\sqrt{Z}}Q(i)$ where $Q$ is some probability distribution of interest over the space of binary registers $I$, and $Z$ is chosen for correct normalization of the amplitudes. Distributions $Q$ of interest include the true joint $Q(i) = P(i)$ and the MAP-Dirac Delta joint, $Q(i) = δ(i; i)$. The latter is of interest because it guarantees that an observation on the system will yield the MAP state, $|i⟩$. We may move gradually from a flat $Q(i)$ to $Q(i) = P(i)$ then to $Q(i) = δ(i; i)$ via a series of cooled distributions, $Q(i) = P(i)^{1/T}$.

We have seen that it is possible to represent joints in linear resources on a quantum register. The problem is how to bring the register into such a representational state. It has been shown [18] that the maximum quantum speedup for unstructured search is $O(\sqrt{N})$. Algorithms based on Grover’s method [6] achieve this bound using large operations on the whole space, applied in series. In contrast, we present an algorithm using small local operators which may be applied in parallel, reminiscent of a Gibbs sampler or Boltzmann machine [8].

### 1.3 Side-stepping the unitary requirement by extending the state space

The key idea is to construct and apply operators which transfer low-probability (high-energy) states to high-probability (low-energy) states. A difficulty with this approach is that quantum operators are required to be unitary. This means that as much amplitude must flow out of each state as flows in. Initially this may seem to prevent our approach being possible, as the high-energy states must ultimately flow back to low-
energy states. This is indeed the case when a fixed-size system is considered. Consider fig. 1(top) and fig. 1(bottom), which respectively illustrate unitary operations acting on single qubits to flip the left and right qubits of a two-qubit register. In general, single-qubit operators may have some portion of self-transition as well, shown by the thin loops. The widths of the arrows illustrate the transition probabilities. (Note that this state transition diagram is not an MRF, but could represent moves between configurations of a boolean MRF with two nodes.) Suppose the two variables are elements from an MRF whose configuration energies satisfy

\[ E(|00\rangle) > E(|01\rangle) = E(|10\rangle) > E(|11\rangle) \]

where \( E(|\psi\rangle) = \sum_i E_{ii}(|\psi\rangle) + \sum_{i,j} E_{ij}(|\psi\rangle) \) with the second sum over linked pairs of nodes.

In this case we would like to design operators that tend to make the probability amplitudes flow away from the high-energy state \(|00\rangle\) and down towards the low-energy state \(|11\rangle\). This is not possible with ordinary unitary operators on the 2-qubit space because unitary operators must have as much flow out of states as into them.

However by adding new bits to the system at every operation we will be able to make the state space grow and keep on providing new places for the accumulating high-probability states to flow to. The algorithm is reminiscent of – and inspired by – the idea of blowing a constant stream of coolant particles over a machine to cool it. The particles then flow away as garbage (or ‘exhaust’), carrying away information (cf. Feynman’s heat computers, [4]). They can then be ignored for the final measurement, but their existence is the key part of the algorithm. This process of spreading the superposition across a large garbage space is a form of decoherence (e.g. see [13]).

As we are considering only operators which – like the Gibbs sampler – flip only single qubits (and/or don’t flip them) we may depict whole collections of such operators by a single diagram, as in fig. 2. Depending which of the bits is chosen to be considered next, one may read off the appropriate flow arrows without ambiguity.

Fig. 3 shows how the 2-variable system from fig. 2 may be extended with a garbage qubit to achieve the desired flows. This qubit is notated by the rightmost bit in the state labels, and is initialized to \(|0\rangle\). Our node operators now act on the combined space of the node together with the garbage bit – i.e. the garbage bit’s value may change as well as the node of interest. The figure shows how the low-energy state \(|11\rangle|0\rangle\) may now act an attractor: the flow out of \(|11\rangle|0\rangle\) does equal its flow in, but most of this flow is into \(|11\rangle|1\rangle\), which maintains the \(|11\rangle\) state of the nodes. We don’t care about the resulting value of the garbage bit. For each operation a new, fresh garbage qubit, initialized to \(|0\rangle\), is introduced to the system.

Because we are introducing new qubits at each step, we are able to control all of their initial states – setting them to \(|0\rangle\). So we know that the initial complete state will have all its amplitude in the left half of the diagram. We can set up our unitary operator to have large flows from the left half to the right half, and vice versa. But we know that the flows from right to left (called ‘shadow flows’ and not shown on the diagram) will never actually be used. The states in the right half are called ‘shadow states’ and are never realized as initial states, only as resulting states.

2 Intuitive Explanation

The algorithm is based on a simplified analogy to physical atoms being bombarded by photons. Each new coolant is like a photon; each node is like an atom. At each step, one coolant and one node (and its neighbors) interact, and like Gibbs sampling, the node may or may not flip as a result.
If the energy of the coolant/photon is exactly zero and an energy emission is possible from the atom/node, then the node flips and energy (emission) occurs. If the coolant has zero energy but the node is unable to emit energy then \{no emission\} occurs.

If the energy of the coolant is exactly the right amount to raise the atom/node to a higher energy state, then the node flips and \{absorption\} occurs. If coolant has energy but not the exact amount then \{no absorption\} occurs. ¹

The terms in braces above refer to cases in the algorithm pseudocode in section 3.2. Note that they refer to transitions between basis configurations of nodes and coolants: in the general quantum setting, nodes and coolants will be in a superposition of states, so all of these cases can occur simultaneously. As in classical Gibbs sampling, we would like to encourage the emissions to occur more often, and absorptions to occur less often, whilst still allowing energy increases to escape from local minima. We can control the rates of emissions and absorptions by manipulating the composition of the coolant superpositions. Initially we create ‘hot’ coolants with relatively high amplitudes of high energy states. Then we reduce the temperature to 'cool' coolants with relatively high amplitudes of high energy but not the exact amount then \{no absorption\} occurs.²

As discussed above, all operators are required to be unitary. As well as ‘energy bits’, our coolants contain ‘change bits’ which act as in fig. 2 to construct shadow states and transitions. As before these states are never realized as inputs, but exist as outputs; they are a book-keeping device to allow unitarity to be maintained. The algorithm in section 3.2 is a constructive proof that this is possible.

3 Formal algorithm description

We consider boolean MRFs so each node may exist in state |0⟩ (false) or |1⟩ (true). Initially, the system state space \(\Psi_0\) is just the state space of the nodes:

\[
\Psi_0 = X_1 \otimes X_2 \otimes ... \otimes X_N
\]

This system state is initialized by an N-bit Hadamard transform \(H_N\) to a superposition of all possible configurations (this may be thought of as the limit of high

¹There is one special subcase where flipping the node would ‘emit’ zero energy, i.e. flipping makes no energy difference. In this case we make a superposed transition, \{both flip and not flip\} simultaneously. Amplitude thus disperses over equiprobable states, and this is required to escape from plateaux in the pdf. An alternative would be to always flip in such cases, but this would lead to undesirable random-walk MCMC behavior as discussed in [12].

²This is to encourage more energy emission than absorption.

At each step \(s\) of the algorithm we extend the state space with a new coolant state space \((\Gamma_s \otimes C_s)\), so as the algorithm progresses, the space grows very large:

\[
\Psi_s = \Psi_0 \otimes (\Gamma_1 \otimes C_1) \otimes (\Gamma_2 \otimes C_2) \ldots \otimes (\Gamma_s \otimes C_s)
\]

Each coolant has two parts: \(\Gamma\) is generally made of several qubits, and is called the energy register. It is analogous to a fresh particle being blown over the system to heat or cool it; or to a photon striking an atom to transfer energy in or out of it. \(C\) is a single qubit and is called the change bit because we will use it as a book-keeping device to record whether the node changed state (allowing us to create ‘shadow states’ and to make our operators unitary). Initially each of the coolants are created in the state:

\[
|\Gamma C⟩ = \frac{1}{\sqrt{Z}} \sum_{\gamma=0}^{Δ_{max}} E^{−1} \exp(−\gamma/T) |\gamma⟩ |0⟩
\]

(where as usual \(Z\) is chosen so that the sum of squared amplitude moduli is unity). The amplitudes follow a Boltzmann distribution. \(T\) is a temperature parameter, and is chosen so that \(\sum_{\gamma=0}^{Δ_{max}} E^{−1} \exp(−\gamma/T) > 1\).²

As in classical Gibbs annealing, \(T\) will be gradually reduced. It should be follow a similar schedule as used in classical Gibbs; these schedules are generally
heuristic. We will show that the quantum algorithm produces a speedup for the same schedule as its classical counterpart. \( \Delta_{\text{max}}E \) is chosen to be the first power of two above the highest possible change in network energy that would occur from flipping one boolean node, \( \log_2 \Delta_{\text{max}}E = \max_{x_i,m_i} \log_2 \Delta E \), with \( \Delta E = |E_i(x_i,m_i) - E_i(\bar{x}_i,m_i)| \), as described in section 3.2.

### 3.1 Algorithm

First construct an operator \( U_i \) for each node \( X_i \) as detailed in section 3.2. At each iteration \( k \), choose a random node ordering (as in classical Gibbs sampling). For each node \( X_i \) in the ordering, create a new coolant and add it to the system. The size of the state space thus keeps growing. Apply the node’s associated operator \( U_i \) to the subspace consisting of \( X_i \) and its neighbors \( M_i \). Ignore old coolants – they are garbage. When all nodes in the ordering have been operated on, lower the temperature according to the cooling schedule and progress to the next iteration. With appropriate hardware, each iteration’s operations could be applied in parallel instead of random series.

The system converges to a superposition of local minima, and usually to a global minimum with high amplitude. Unlike the classical Gibbs sampler, all possible histories of proposal acceptances and rejections are explored in parallel by the superposition.\(^3\)

### 3.2 Constructing the local operators

Before running the quantum iterations we first define one operator \( U_i \) corresponding to each node \( X_i \). These operators are local in the sense that they act only on the reduced state space \( \Psi_i \) comprising the node \( X_i \), its neighbors \( M_i = \bigotimes_{X_j \in \text{neigh}(X_i)} X_j \), and a coolant system (\( \Gamma \otimes C \)):

\[
\Psi_i = X_i \otimes M_i \otimes \Gamma \otimes C
\]

Define \( E_i(x_i,m_i) \) to be the energy contribution due to the prior on \( X_i \) and the links to its neighbors:

\[
E_i(x_i,m_i) = E_{ii}(x_i) + \sum_{X_j \in \text{neigh}(X_i)} E_{ij}(x_i,x_j)
\]

Each \( U_i \) is constructed as follows (refer to section 2 for descriptions of the cases):

\[\text{for each } x_i \text{ configuration; each } m_i \text{ configuration; each } \gamma = 0 : \Delta_{\text{max}}E \text{ do} \]
\[
\Delta E := E_i(\bar{x}_i,m_i) - E_i(x_i,m_i)
\]
\{
\text{network’s energy gain by flipping}
\}
\[
\text{if } \Delta E \neq 0 \text{ then}
\]
\[
\text{if } \gamma = 0 \text{ then}
\]
\{
\text{coolant carries no energy – possible cooling}
\}
\[
\text{if } \Delta E < 0 \text{ then}
\]
\[
U [x_i,m_i,0,0] := |\bar{x}_i,m_i,-\Delta E,1\rangle
\]
\{
\text{emission}
\}
\[
U [x_i,m_i,-\Delta E,1] := |x_i,m_i,0,0\rangle
\]
\{
\text{shadow transition}
\}
\[
\text{end if}
\]
\[
\text{if } \Delta E > 0 \text{ then}
\]
\[
U [x_i,m_i,0,0] := |x_i,m_i,0,0\rangle
\]
\{
\text{no emission}
\}
\[
U [x_i,m_i,0,1] := |x_i,m_i,0,1\rangle
\]
\{
\text{shadow transition}
\}
\[
\text{end if}
\]
\[
\text{else}
\]
\{
\text{coolant carries energy – possible absorb}
\}
\[
\text{if } \Delta E = 0 \text{ then}
\]
\[
U [x_i,m_i,0] := |\bar{x}_i,m_i,0\rangle
\]
\{
\text{absorption}
\}
\[
U [x_i,m_i,0] := |x_i,m_i,0\rangle
\]
\{
\text{shadow transition}
\}
\[
\text{else}
\]
\[
U [x_i,m_i,0] := |x_i,m_i,0\rangle
\]
\{
\text{no absorption}
\}
\[
U [x_i,m_i,0] := |x_i,m_i,0\rangle
\]
\{
\text{shadow transition}
\}
\[
\text{end if}
\]
\[
\text{else}
\]
\{
\text{equi-energy special case – flipping node has no energy effect}
\}
\[
U [x_i,m_i,\gamma,0] := \frac{1}{\sqrt{2}} |x_i,m_i,\gamma,0\rangle + \frac{1}{\sqrt{2}} |\bar{x}_i,m_i,\gamma,0\rangle
\]
\{
\text{flip and not flip}
\}
\[
U [x_i,m_i,\gamma,1] := \frac{1}{\sqrt{2}} |x_i,m_i,\gamma,1\rangle - \frac{1}{\sqrt{2}} |\bar{x}_i,m_i,\gamma,1\rangle
\]
\{
\text{shadow transitions}
\}
\[
\text{end if}
\]
\[
\text{end if}
\]
\[
\text{end for}
\]

### 4 Results

Our simulations are performed using the QCF quantum computing simulator for Matlab [5]. Due to the exponential resources required to simulate quantum devices, we are restricted in the present work to giving only proof of concept demonstrations. We consider the 3-node and 4-node boolean MRFs in fig. 4. The first net has unit energy penalties for nodes with

\[\text{\textsuperscript{3}}\text{This is reminiscent of an ‘exact particle filter’, which adds new particles at each step to ensure all possible histories are explored. Information carried away by coolants could be used to recover the trajectory histories.}\]
value 1 and for links whose neighborhoods are different, i.e. $E_i = x_i$, $E_{ij} = \Delta(x_i, x_j)$, so its MAP state is 000. The second net has asymmetric penalties and MAP state 010. The third is a more complex net with MAP state 1010. Fig. 5 shows the three operators constructed for the $\hat{x} = 010$ task.

Note that a naive state-vector-based computer simulation would require resources exponential in the number of iterations, due to the expanding state space from the new coolants (e.g. of size about $2^N$ after 4 iterations of the 4-node net). However as we don’t care about the resulting states of the coolant – they are garbage – we may make use of the density matrix formulation of quantum computing (e.g. see [13]), which allows us to ‘trace out’ over these coolant states. This essentially provides us with the analogous operator to the ‘sum’ in the ‘sum-product’ algorithm (the ‘products’ corresponding to the unitary operators). Note that it is the lack of ability to sum out states – which reduces information – that prevents us from using quantum algorithms without garbage bits to do inference. Summing these traces uses the bulk of computing time on a classical computer simulation.

The performance of the quantum algorithm was compared to the analogous classical Gibbs sampler, whose acceptance probabilities are $\frac{1}{2} \exp(-E/T)$. The same arbitrary cooling schedule was used in both algorithms, and was chosen prior to the experiments: at the $k^{th}$ iteration we set $T = \frac{1}{k+1}$. Each classical sampler was run for 100,000 trials, each trial initialized with a random configuration. The table below shows the empirical probability of the classical sampler (C) reaching the MAP solution after the final iteration, compared with the (deterministic) quantum observation probabilities (Q), for the three tasks with MAP targets 000, 010 and 1010. The quantum algorithm appears to lag the classical sampler for the first two iterations before overtaking.

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>010</th>
<th>1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>step</td>
<td>C/Q</td>
<td>C/Q</td>
<td>C/Q</td>
</tr>
<tr>
<td>1</td>
<td>.68/.51</td>
<td>.68/.43</td>
<td>.63/.35</td>
</tr>
<tr>
<td>2</td>
<td>.80/.76</td>
<td>.81/.70</td>
<td>.84/.72</td>
</tr>
<tr>
<td>3</td>
<td>.84/.84</td>
<td>.82/.86</td>
<td>.89/.92</td>
</tr>
<tr>
<td>4</td>
<td>.84/.87</td>
<td>.84/.92</td>
<td>.92/.98</td>
</tr>
<tr>
<td>5</td>
<td>.84/.89</td>
<td>.84/.94</td>
<td>.94/.99</td>
</tr>
<tr>
<td>6</td>
<td>.85/.89</td>
<td>.84/.96</td>
<td>.95/.999</td>
</tr>
</tbody>
</table>

5 Discussion

5.1 Comparison to existing work

Quantum optimization methods have featured in the physics literature in recent years (e.g. [9], [2], [1]). However this work has not generally been framed in the context of discrete local quantum computation and the MAP-MRF task. The novel contribution of the present work is to give a purely quantum-computational algorithm for graphical model MAP optimization (for example there are no continuous Hamiltonians, integrals, or physical temperatures in this paper). Recall that any graphical model can be reduced to a boolean MRF which the algorithm may then operate upon. Our algorithm uses only local operators, in contrast to inference methods such as [15] which apply Grover’s algorithm to parameter search.

A recent development in quantum optimization is the adiabatic method [3], which uses no garbage bits and matches Grover’s $\sqrt{N}$ unstructured search speedup [16]. This speedup is dependent on prior information on the difference in probability between the first and second most likely states, and would not be achieved for general optimization such as MAP inference. It is an open question how it could be exploited in those particular inference tasks where we do have some such information. All the above should be distinguished from operator formulations of classical message passing [14] and ‘quantum annealing’ [10], a quantum-inspired search method for classical computers.

5.2 Comparison to classical Gibbs sampling

A fundamental problem in machine learning is that the joint over $N$ nodes generally requires exponential resources to represent it – whether as a brute-force table or the classical Gibbs sampler’s use of ensembles of $N$-node configurations over exponential time. We have shown how by using quantum amplitudes to represent joint probabilities, it is possible to represent the exponential-sized joint of $N$ nodes instantaneously using only $N$ physical nodes. However, moving the state vector into this representation is non-trivial and we...
showed how to use a unitary quantum analog of Gibbs sampling to get there. Further, we showed how it can be cooled towards the MAP solution, \( Q(\tau) \rightarrow \delta(x; \hat{x}) \).

The algorithm does not make use of phase cancellations (all amplitudes are positive real), so trajectories in the state space starting from each of the superposed initial states evolve almost completely independently of each other.\(^5\) When we make a measurement after the cooling, we are therefore selecting just a single trajectory history to have occurred, and discarding the others.\(^6\) It may be asked what advantage this has over the classical Gibbs method of simply sampling a single \( s \)-step trajectory \( \tau = \{x_{1:N}\}_{1:s} \) without the need for quantum hardware – what is the use of the unobserved trajectories? The answer is that as we are representing cooled model probabilities \( Q(\tau) \) by amplitudes, \( \alpha_\tau \), our probability of observing a particular \( \tau \) is given by the (renormalized) squared probability of its occurrence in the corresponding classical sampler, \( Pr_{obs}(\tau) = \alpha_\tau^2 = Q(\tau)^2 \). This gives a quadratic amplification of the probabilities \( Q(\tau) \) from the classical sampler. This quadratic speed-up appears to fit in spirit with the result [18] that the optimal quantum search speedup is \( O(\sqrt{N}) \), however full mathematical analysis of the algorithm’s nature could form the basis of future work. Such analysis could also try to explain why the first two iterations give poorer results than the classical sampler: we believe this is due to the sub-optimal nature of our sampling, as for simplicity we excluded the possibility of low but non-zero energy states absorbing energy.

5.3 Implications

We present the algorithm primarily as a potential quantum method to speed up the machine learning community’s general task of MAP inference in graphical models, and to illustrate the quantum ability to represent complete \( N \) node joints instantaneously using only \( N \) physical nodes. Unlike many quantum algorithms it requires only small local operators that would be easy to implement. Also unusual is the essential requirement for decoherence into coolants – rather than being a hindrance, decoherence is necessary to disperse information: due to the reversibility requirement, uncertainty is transferred out of the cooled nodes and into the coolants. The algorithm may be of interest to quantum biologists, as if it does provide a useful computational speedup then it seems likely that evolution would have found a way to use it – though whether this is in fact the case is of course an empirical question. Highly speculatively, for example, Hameroff’s sub-neuronal computation theory [7] could be extended to perform this kind of MRF inference, making use of biological coolants flowing through microtubules to provide the necessary decoherence.

5.4 Acknowledgements

Thanks to Anuj Dawar, David Deutsch, Roger Penrose and Amos Storkey for discussions. This work was supported by an EPSRC studentship.

References


