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 it up. The algorithm uses only small, local operators, and is based on a physical analogy of striking the nodes in a network with superposed 'coolants'. It may be viewed as a quantum version of the Gibbs sampler, utilising 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. Information is carried away by the coolants, making essential use of decoherence – often thought of as a hindrance rather than a benefit to quantum computation. The algorithm creates a superposition whose amplitudes represent *model* probabilities, then cools it to leave a sharp peak at the MAP solution. As quantum *observation* probabilities are *squares* of these amplitudes, the probability of observing the MAP solution is amplified quadratically over the classical case. Proof-of-concept simulation is presented.

Introduction

Quantum computing has become increasingly popular in the physics community and others. However, there is little work framing probabilistic inference tasks in a quantum computational setting. We present a quantum computational algorithm capable of performing MAP inference in in graphical models. The algorithm quickly finds a probably MAP or near-MAP configuration with a quadratic speedup over classical Gibbs sampling, in the sense that after the same number of sampling steps it yields observation probabilities $\frac{1}{Z}Pr_{obs}(a)^2$ of states a where $Pr_{obs}(a)$ are the classical Gibbs observation probabilities and Z is a normalising coefficient. The algorithm uses only small local operators and makes 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. (Myllymaki & Tirri 1993)). Let nodes have potentials ϕ_{ii} and links have potentials ϕ_{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 normalisation 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 maximise $P(x_{1:N})$. This is generally NP-hard (Shimony 1994).

Overview of quantum computing

We briefly review some key concepts in quantum computation. For a full introduction see (Nielsen & Chuang 2000).

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 emphasise this view we use 'ket' notation:

$$|101\rangle \equiv |5\rangle \equiv [00000100]'$$

where the rightmost term is a column vector (the prime symbol notates matrix transposition) whose elements are the eight classical states from $|0\rangle$ to $|7\rangle$, so they are all zero except for the sixth element which represents the integer 5. Operations that map register states to register states (such as a particular state-change in a Gibbs sampler) may then be represented by a 2^3 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.:

$$|101\rangle \otimes |001\rangle \equiv |101\rangle |001\rangle \equiv |101001\rangle$$

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 $|000\rangle$ and $|110\rangle$:

$$|\Psi\rangle = \alpha_{000}|000\rangle + \alpha_{110}|110\rangle$$

where the α are chosen to describe a hypersphere location, $\sum_i |\alpha_i|^2 = 1$, and each $|\alpha_i|^2$ describes the probability of an observation collapsing the register into state $|i\rangle$. In general the α_i 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 computing' property, and may be pictured 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; $|A\rangle$ is a general vector in that space; a is an integer (in decimal or binary notation) denoting a basis state; $|a\rangle$ 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 $Pr_{obs}(a) = |\alpha_a|^2$.

Representing joints by superpositions

A quantum register of N qubits ('quantum bits') is able to store and represent amplitudes of 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 normalisation; recall that our probabilities normalise to 1 but amplitudes normalise so that $\sum_i |\alpha_i|^2 = 1$), requiring only N qubits instead of exponential resources. We will try to set the α_i so that $\alpha_i = \frac{1}{\sqrt{Z}} Q(i)$ where Q is some probability distribution of interest over the space of binary registers I, and Z is chosen for correct normalisation of the amplitudes. Distributions Q of interest include the true joint $Q(i) = P(i)$ and the MAP-Dirac Delta joint, $Q(i) = \delta(i; \hat{i})$. The latter is of interest because it guarantees that an observation on the system will yield the MAP state, $|\hat{i}\rangle$. We may move gradually from a flat $Q(i)$ to $Q(i) = P(i)$ then to $Q(i) = \delta(i; \hat{i})$ via a series (parametrised by simulated 'temperature' T) 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.

Grover's method (Grover 1996) performs quantum unstructured search over N items in $O(\sqrt{N})$. However this algorithm requires the use of large operations over the whole space, applied in series. It also only finds target values whose target-ness is a function of their value only, rather than of their relationship to other values as in optimisation problems. In contrast, we present an algorithm using small local operators which may be applied in parallel during each overall sequential step, as in the local node updates of an annealed Gibbs sampler or Boltzmann machine (Hinton & Sejnowski 1986); and which seeks targets that are global and near-global MAP states.

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 is that quantum operators must be unitary. This means that as much amplitude must flow out of each state as flows in (similar to 'global balance' in Markov Chain theory). Initially this may seem to prevent our approach being possible, as the high-energy states must ultimately flow back to low-energy states.

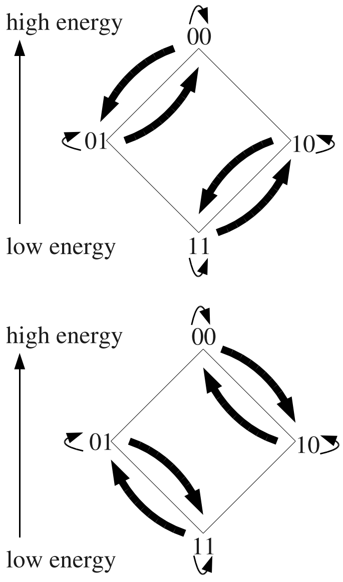

Figure 1: Top: A unitary operator that maps states to a superposition of themselves and the state reached by flipping their rightmost bit. Bottom: A similar operator for the leftmost bit.

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 probability amplitudes flow away from the high-energy state $|00\rangle$ down towards the low-energy state $|11\rangle$. This is impossible 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, (Feynman 1998)). 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

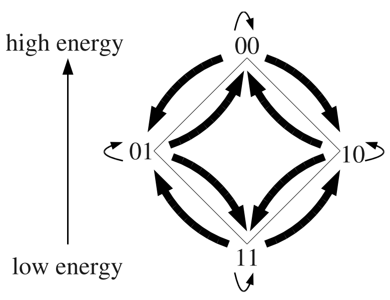

Figure 2: The same operator as fig. 1, plotted as a single figure. Note that unitarity is equivalent to the requirement that there is an equal amount of total flow into and out of each node.

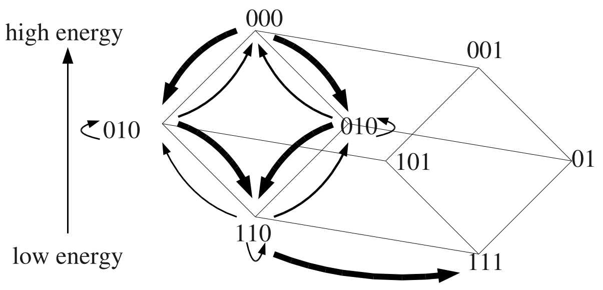

Figure 3: The rightmost qubit is new, introduced to extend the state space.

of *decoherence* (e.g. see (Nielsen & Chuang 2000)).

As we consider only operators which (like the Gibbs sampler) flip only single qubits (and/or don't flip them) we may depict whole collections of 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 initialised 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 $|11\rangle$ 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, initialised 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 realised as initial states, only as resulting states.

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 neighbours) 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 coolant's energy is exactly the amount to raise the atom/node to a *higher* energy state, then the node flips and {absorption} occurs. If the coolant has energy but not the exact amount then {no absorption} occurs.[1]

The terms in braces above refer to cases in algorithm 1. 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 make the low-energy states more likely. Note that use of the term 'temperature' is analogical: the algorithm itself may run on a standard quantum computer.

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 realised as inputs, but exist as outputs; they are a book-keeping device to allow unitarity to be maintained. Algorithm 1 is a constructive proof that this is possible.

Formal algorithm description

We consider Boolean MRFs so each node may exist in state $|0\rangle$ (false) or $|1\rangle$ (true). Initially, the system state space Ψ_0 is just the state space of the nodes:

$$\Psi_0 = X_1 \otimes X_2 \otimes ... \otimes X_N$$

This system state is initialised by an N-bit Hadamard transform H_N to a superposition of all possible configurations

[1] There 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 behaviour as discussed in (Neal 1995).

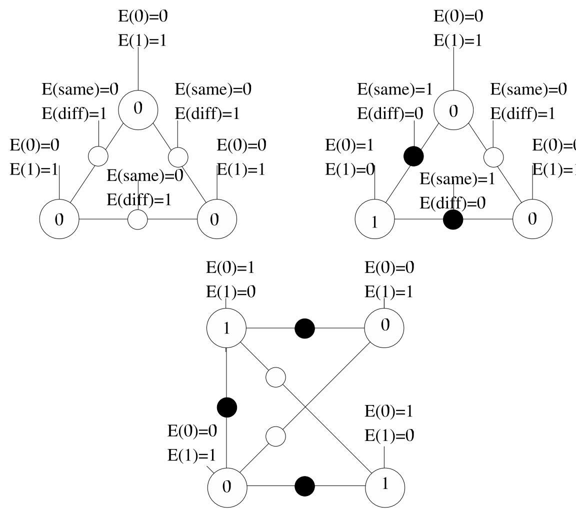

Figure 4: Top left: the $\hat{x} = 000$ test MRF. Top right: $\hat{x} = 010$ test. Bottom: $\hat{x} = 1010$ test. White and black links prefer same and differing nodes respectively. Nodes are labelled with optimal values.

(this may be though of as the limit of high temperature, the flat distribution):

$$|\Psi_0\rangle := H_N |00...0\rangle = \frac{1}{\sqrt{Z}} \sum_{\psi=0}^{2^N-1} |\psi\rangle$$

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:

$$\Psi_s = \Psi_0 \otimes (\Gamma_1 \otimes C_1) \otimes (\Gamma_2 \otimes C_2) ... \otimes (\Gamma_s \otimes C_s)$$

Each coolant has two parts: $|\Gamma\rangle$ 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. $|C\rangle$ 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:

$$|\Gamma C\rangle = \frac{1}{\sqrt{Z}} \sum_{\gamma=0}^{\Delta_{max}E-1} \exp(-\gamma/T) |\gamma\rangle |0\rangle$$

The amplitudes follow a Boltzmann distribution. T is a temperature parameter, and is chosen so that $\sum_{\gamma=1}^{\Delta_{max}E-1} \exp(-\gamma/T) > 1$.[2] As in classical 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. $\Delta_{max}E$ 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, $\log_2 \Delta_{max}E = \max_{i,x_i,m_i} \lceil \log_2 \Delta E \rceil$, with $\Delta E = |E_i(x_i, m_i) - E_i(\bar{x_i}, m_i)|$, described below.

[2]This is to encourage more energy emission than absorption.

Algorithm

First construct an operator U_i for each node X_i as detailed in the next section and algorithm 1. 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 neighbours 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]

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 Ψ_i comprising the node X_i, its neighbours $M_i = \bigotimes_{X_j \in neigh(X_i)} X_j$, 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 neighbours:

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

Each U_i is constructed by algorithm 1.

Results

Our simulations are performed using the QCF quantum computing simulator for Matlab (Fox 2003). 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 value 1 and for links whose neighbours 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.

[3]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.

Algorithm 1 Pseudo-code for the construction of the U_i operator matrices. This is a classical algorithm which simply inserts numbers into matrices. A quantum algorithm will later apply these operators to the nodes and coolants. Refer to the text for descriptions of the various cases. QCF/Matlab versions of both classical and quantum algorithms are available from the author.

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

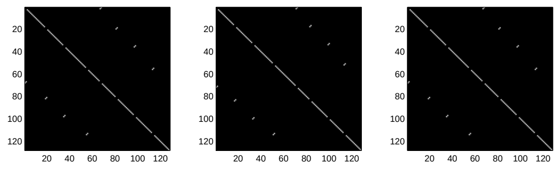

Figure 5: Operators U_0, U_1 and U_2 constructed for the $\hat{x} = 010$ task by the algorithm 1. There is one operator for each node. In this simple case the matrix representations of the operators are all of size 120x120. Matrix elements are always 0 or 1 (except in special equi-energy cases). After the operators are constructed, the quantum computing itself may run. Applying U_i to node i and its neighbours is analogous to a classical Gibbs step of updating one node.

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^{50} 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 (Nielsen & Chuang 2000)), 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 $\frac{1}{Z}\exp(-E/T)$. 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 initialised 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.

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

Discussion

Comparison to other quantum methods

Quantum optimisation methods have featured in the physics literature in recent years (e.g. (Hogg & Yanik 1998), (Durr & Hoyer 1996), (Castagnoli & Finkelstein 2003)). 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 optimisation (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 (Ricks & Ventura 2004) which apply Grover's algorithm to parameter search.

A recent development in quantum optimisation is the *adiabatic* method (Farhi *et al.* 2000), which uses no garbage bits and matches Grover's $\sqrt{N}$ unstructured search speedup (Roland & Cerf 2002). 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 optimisation 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 (Rezek & Roberts 2003) and from 'quantum annealing' (Kadowaki 2002), a quantum-inspired search method for classical computers.

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, $Q(x) \to \delta(x; \hat{x})$.[4]

If amplitudes are considered to represent probabilities, then the algorithm can be thought of as behaving exactly as a classical annealing Gibbs sampler, with transition probabilities identical to the classical Gibbs sampler's. The difference between classical 'annealing' and our 'cooling' is that in the quantum domain, information cannot simply be discarded as in a classical computer, but must be accounted for within the unitary operations. The coolants are a method for producing annealing behaviour in the quantum domain, and achieve this by extending the state space at each step.

[4] $Q(x) \approx P(x)^{1/T}$ is represented implicitly as the marginal of the explicit $Q(\tau) = Q(x, \gamma_{1:s}, c_{1:s})$. The coolant space was introduced as it was not possible to carry out marginalisation explicitly by unitary operators.

Comparison to variational inference

Although we have presented the algorithm largely as a quantum extension of the classical Gibbs sampler, an alternative and interesting view is as a quantum extension of classical variational methods (Bishop 2006) such as Variational Bayes and loopy (Bethe) message passing. Such methods approximate the true joint $P(x_{1:N})$ by a product $Q(x_{1:N}) = \prod_{i=1}^{N} Q_i(x_i)$ which ignores correlation structure but is typically more tractable to compute. In the Gibbs analogy, we view our algorithm as a multitude of Gibbs samplers, running in 'parallel worlds'. An alternative view is to consider the amplitudes over individual node subspaces as being like variational Q_i factors. However they are extended from classical variational methods because they are entangled with the other nodes, so representing the correlation structure. A re-derivation of the algorithm from this perspective could be interesting future work.

The source of the quadratic speedup

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*, α_τ, our probability of *observing* a particular τ is given by the (renormalised) *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 Grover's quadratic speedup, and suggests some rationale for the quadraticty: we may work linearly in the amplitude domain but then obtain a free quadratic amplification at observation. Full mathematical analysis of this idea could form the basis of future work. Such analysis could also try to explain why our 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.

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

[5] The unimportant exception being the special plateaux case.

[6] The computation is reversible, so reachable basis states including coolants specify historical trajectories.

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. Highly speculatively, the algorithm may be of interest to quantum biologists, as if it does provide a useful computational speedup then it seems *likely* (in the Bayesian sense) that evolution would have found a way to use it – though whether this is in fact the case is of course an empirical question.

Acknowledgements

Many thanks to Anuj Dawar, David Deutsch, Stuart Hameroff, Stephen N.P. Smith and Amos Storkey for discussions; and two reviewers for comments. This work was supported by an EPSRC studentship.

References

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. Springer.

Castagnoli, G., and Finkelstein, D. R. 2003. Quantum-statistical computation. *Proceedings of the Royal Society A* 459(2040):3099–3108.

Durr, C., and Hoyer, P. 1996. A quantum algorithm for finding the minimum, `quant-ph/9607014`.

Farhi, E.; Goldstone, J.; Gutmann, S.; and Sipser, M. 2000. Quantum computation by adiabatic evolution.

Feynman, R. P. 1998. *Feynman Lectures on Computation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Fox, C. 2003. QCF: Quantum computing functions for Matlab. Technical Report PARG-03-02, Robotics Research Group, Oxford University.

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–219.

Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: foundations*. Cambridge, MA, USA: MIT Press. 282–317.

Hogg, T., and Yanik, M. 1998. Local search methods for quantum computers, `quant-ph/9802043`.

Kadowaki, T. 2002. *Study of Optimization Problems by Quantum Annealing*. Ph.D. Dissertation, Department of Physics, Tokyo Institute of Technology.

Myllymaki, P., and Tirri, H. 1993. Massively parallel case-based reasoning with probabilistic similarity metrics. In K. Althoff, M. R., and Wess, S., eds., *Proceedings of the First European Workshop on Case-Based Reasoning*.

Neal, R. 1995. Suppressing random walks in Markov Chain Monte Carlo using ordered overrelaxation. Technical report, Dept. of Statistics, University of Toronto.

Nielsen, M., and Chuang, I. 2000. *Quantum computation and quantum information*. Cambridge.

Rezek, I., and Roberts, S. 2003. An operator interpretation of message passing. Technical Report PARG-03-01, Robotics Research Group, University of Oxford.

Ricks, B., and Ventura, D. 2004. Training a quantum neural network. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Roland, J., and Cerf, N. J. 2002. Quantum search by local adiabatic evolution. *Phys. Rev. A* 65(4):042308.

Shimony, S. E. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence* 68(2):399–410.