Fast Mixing Monte-Carlo For Biological Networks

Dan Barker,[1] Sach Mukherjee,[2] and Mario Nicodemi[3]
[1] *Complexity Science, University Of Warwick, Coventry, CV4 7AL, UK*
[2] *Dept. Statistics, University of Warwick, Coventry, CV4 7AL, UK*
[3] *Dept. Physics, University of Warwick, Coventry, CV4 7AL, UK*
(Dated: June 18, 2009)

Inferring biological networks from data is currently an area of much interest. Understanding novel biology based upon these networks has important implications for studying many diseases. We examine two Monte Carlo schemes which it was hoped would provide a speed up over a widely used Metropolis-Hastings scheme for inferring the networks: Tempering (both simulated and parallel) and a novel 'Tunnelling' scheme. Analysis and empirical results provide a clearer understanding of the nature of the MCMC problem and insight into the computational exploration of network space. A parallel variant of Simulated Tempering called Parallel Tempering is found to provide gains relative to other schemes on a 10 node problem with a vast state space. In addition the basis of a possible application of Swendesen-Wang type cluster algorithms is detailed, though this was not actually implemented.

I. INTRODUCTION

Thanks to modern experimental techniques, such as DNA microarrays & protein arrays, researchers have an abundance of biological data on which quantitative analysis can be performed. One particular method that has received a lot of attention is probabilistic graphical models. One can measure, for example, the expression of many genes or proteins under various conditions and using probabilistic graphical models infer from data relationships between these molecular components. Such a network is shown in Figure 1, which shows interactions between some proteins that are involved in the Epidermal Growth Factor Receptor system, a system of particular interest in cancer biology.

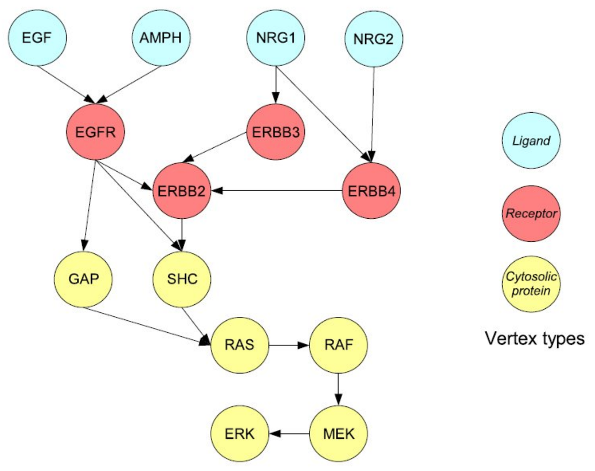

FIG. 1: The network relating the various components of the Epidermal Growth Factor Receptor (EGFR) system which was used to simulate data [1].

These biological networks can be represented by graphs. A graph $G = (V, E)$ is a mathematical object consisting of a vertex set V and an edge set E. The vertices (nodes) represent each of the molecular components in the biological system of interest and the edges connecting them denote interactions between these components. Since the relations between the proteins are causal the edges must be directed, i.e. A causes B or B causes A. We apply the additional constraint that G must be acyclic. That is there exist no paths following edges such that one can start on any node and end up back there. Such graphs are called Directed Acyclic Graphs (DAGs). This is required to yield a well defined probabilistic model, but precludes modelling of feedback loops from steady state data. Feedback can be modelled using dynamic variants of the models used herein.

Associated with each node i is a random variable, which we label X_i. A link from node i to node j in G implies (loosely speaking) a dependence of X_j on X_i. Thanks to the acyclic structure of G we can factorise the whole joint distribution $P(X_1...X_p|G)$

$$P(X_1...X_p|G) = \prod_{i=1}^{p} P(X_i|\mathbf{Pa}_G(X_i)) \tag{1}$$

where $\mathbf{Pa}_G(X_i)$ is the set of parents of X_i in G and $p = |V|$ is the number of nodes in G. For a detailed discussion of probabilistic semantics see [2]. Root nodes have no parents and so their probabilities are specified as marginal distributions. In general the conditional probabilities can be any suitable distribution but as in [1] we are working with multinomial local conditionals.

We are interested in the posterior probability of any particular graph G *given* the measured data $\mathbf{X}$. We use Bayes' theorem to write this probability in terms of the likelihood $P(\mathbf{X}|G)$ and a prior distribution $P(G)$ on DAGs.

$$P(G|\mathbf{X}) \propto P(\mathbf{X}|G)P(G) \tag{2}$$

The prior represents our belief about which graphs are more likely *a priori*. We can use the prior to specify any biology which is known about the system or what we believe to be unlikely in a realistic network. For example consider the set of components shown in Figure 1, based on current biological knowledge we might for example want to penalise graphs which have direct interactions between Ligands (cyan) and Cytosolic proteins (yellow). This can be achieved using the prior. Since priors are not the main focus of this project we will be using a uniform prior $P(G) = \frac{1}{|\mathcal{G}|}$ where $\mathcal{G}$ is the space of all DAGs.

The likelihood $P(\mathbf{X}|G)$ is simply how likely the data is given a specified network structure. By integrating over the multinomial local conditional model parameters $\mathbf{\Theta}$ we can write

$$P(\mathbf{X}|G) = \int P(\mathbf{X}|G, \mathbf{\Theta}) P(\mathbf{\Theta}|G)\, d\mathbf{\Theta} \tag{3}$$

$P(\mathbf{X}|G, \mathbf{\Theta})$ are the multinomial conditional distributions chosen for (1) and $P(\mathbf{\Theta}|G)$ is a prior on the parameters of the multinomials. Since the Dirichlet distribution is conjugate to the multinomial conditional distributions choosing $P(\mathbf{\Theta}|G)$ to be Dirichlet distributed yields a closed form solution for the likelihood:

$$P(\mathbf{X}|G) = \prod_{i=1}^{p} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})} \tag{4}$$

As is often the case it is much easier to work in log space so we can write the log-likelihood

$$\begin{aligned} \log P(\mathbf{X}|G) = \\ \sum_{i=1}^{p} \sum_{j=1}^{q_i} \Bigg(&\log \Gamma(N'_{ij}) - \log \Gamma(N_{ij} + N'_{ij}) \\ &+ \sum_{k=1}^{r_i} \left[\log \Gamma(N_{ijk} + N'_{ijk}) - \log \Gamma(N'_{ijk}) \right] \Bigg) \end{aligned} \tag{5}$$

In (4) and (5) N_{ijk} is the number of observations in which X_i takes value k and has parent configuration j. N'_{ijk} are the parameters for the Dirichlet prior distribution, i.e. the model hyperparameters. r_i is the number of possible values X_i can take and q_i is the number of possible parent configurations.

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}, \quad N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$$

Once we have written down the likelihood as in (4) we can ask any question about which graph is the most probable or what are the chances of seeing individual features. However to deal with the missing normalisation constant in (2) we must enumerate the whole space $\mathcal{G}$. The number of directed graphs is 2^{p^2}. This can be seen by representing the graph as an adjacency matrix, which is equivalent to a binary number with p^2 bits. While the number of directed *acyclic* graphs (DAGs) is not as large as the number of directed graphs it still grows super-exponentially [3] so enumerating all DAGs becomes a practical impossibility. For illustration by the time we reach 11 nodes there are almost as many DAGs as there are stars in the known universe. Despite this inability to enumerate $|\mathcal{G}|$ we can still *estimate* the posterior probability distribution $P(G|\mathbf{X})$ by using the Monte Carlo method. The Monte Carlo method is capable of correctly sampling from probability distributions which are known up to some normalising constant.

Metropolis-Hastings [4, 5] is currently the most widely used scheme for inferring network structures from data. It is guaranteed to converge asymptotically but in practice this convergence (mixing) can take a long time, especially if the number of nodes p is large. Thus there is a strong desire for Monte-Carlo schemes which mix faster than the Metropolis-Hastings scheme.

II. MONTE CARLO SCHEMES

Loosely speaking the Monte-Carlo method works by moving around our space of graphs $\mathcal{G}$ randomly but in such a way that the number of times we visit a particular graph is proportional to its posterior probability. It achieves this by proposing a new graph from some 'proposal distribution', accepting or rejecting this new graph according to an 'acceptance probability'. Crucially the acceptance probability is chosen so that the number of times a graph is sampled is representative of its probability. For T samples our estimate of the probability of a graph G is given by

$$\hat{P}(G|\mathbf{X}) = \frac{1}{T} \sum_{t=1}^{T} I(g^{(t)} = G) \tag{6}$$

where $g^{(t)}$ is the t^{th} sampled graph and $I(\cdot)$ is the indicator function which equals one if its argument is true and 0 otherwise.

The differences between Monte Carlo schemes considered here lie in the exact nature of the proposal distributions and corresponding acceptance probabilities. We shall consider several different schemes for which the baseline is taken to be the widely used Metropolis-Hastings. All the schemes described technically use the Metropolis-Hastings algorithm but from herein we shall refer to the single edge flipping Metropolis-Hastings scheme in wide use simply as the Metropolis-Hastings (MH) scheme and the others by their respective names.

A. Metropolis-Hastings

There are two appealing aspects to the Metropolis-Hastings scheme; firstly the proposal distribution Q is very simple and secondly it is easy to show that the resulting Markov chain satisfies detailed balance. Detailed balance ensures that the Markov chain converges to the correct *unique* stationary distribution (see §A 1).

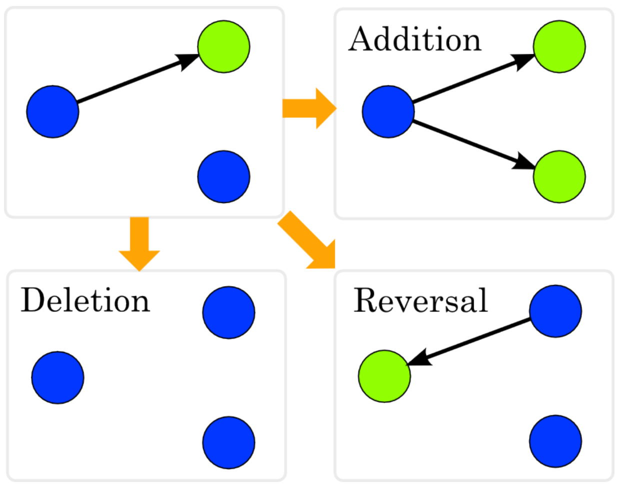

FIG. 2: The neighbourhood $\eta(G)$ of a graph G is defined as any DAG reachable from G by adding, deleting or flipping only a single edge.

The proposal distribution Q involves picking so-called 'neighbours' with uniform probability. The neighbourhood $\eta(G)$ of a graph G is defined to be all other graphs G' which are reachable by either removing, adding or flipping only one edge while maintaining acyclicity. The proposal distribution is then

$$Q(G \to G') = \begin{cases} \frac{1}{|\eta(G)|} & \text{if } G' \in \eta(G) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Obviously this proposal distribution has only short range support. The acceptance probability has the form

$$A = \min\{1, \alpha\} \tag{8}$$

where

$$\alpha = \frac{P(\mathbf{X}|G')Q(G' \to G)}{P(\mathbf{X}|G)Q(G \to G')} = \frac{P(\mathbf{X}|G')|\eta(G)|}{P(\mathbf{X}|G)|\eta(G')|} \tag{9}$$

Naturally we wish to reject as few moves as possible so α is made to be as large as possible while still giving the correct posterior distribution. The form of α shown in (9) has this property.

B. Tunnelling

The Metropolis-Hastings scheme is guaranteed to converge given enough time. However, in practice the mixing time can be large. This because the resultant Markov chain can get stuck in regions of locally high scoring graphs (large $P(G|\mathbf{X})$). The tunnelling Monte-Carlo scheme is designed to overcome this obstacle by allowing the process to jump or 'tunnel' between regions of high scoring graphs, thus allowing it to sample more of $\mathcal{G}$ in a shorter time [6]. This scheme is similar in essence to Metropolis-Hastings but it has a slightly modified proposal distribution.

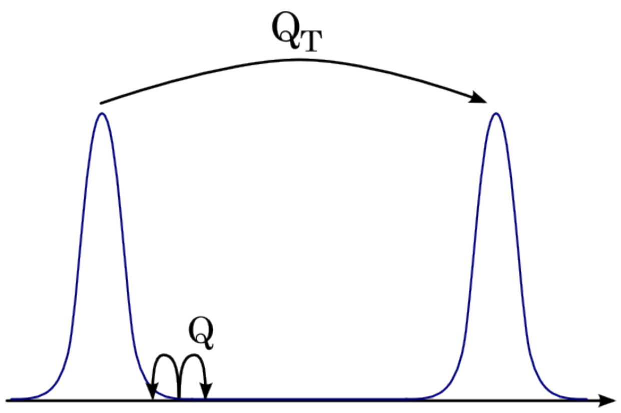

FIG. 3: Tunnelling Monte Carlo works by supplementing local moves from the Metropolis proposal distribution Q with long range jumps between high scoring graphs from the tunnelling proposal distribution Q_T. This prevents the Markov chain getting stuck in local maxima.

In order to jump between high scoring regions we first have to know which regions of $\mathcal{G}$ these are. Before performing any Monte-Carlo Steps (MCS) a set of highly probably graphs $\mathcal{G}_T \subset \mathcal{G}$ (the modes) is pre-computed. The modes must not be neighbours, that is

$$\forall\ G, G' \in \mathcal{G}_T \quad G \notin \eta(G') \tag{10}$$

In practice this is computed using steepest ascent with multiple random starting DAGs. By construction this satisfies condition (10).

When considering the proposal distribution we now have two possibilities to account for, either the current graph is not a mode ($G \notin \mathcal{G}_T$) or it is ($G \in \mathcal{G}_T$). If the former is the case then we simply use the Metropolis-Hastings proposal distribution

$$Q(G \to G') = \begin{cases} \frac{1}{|\eta(G)|} & \text{if } G' \in \eta(G) \\ 0 & \text{otherwise} \end{cases}$$

If the proposed new graph G' is a mode so that $G \notin \mathcal{G}_T \wedge G' \in \mathcal{G}_T$ then we summarily reject G' with a probability p, this ensures detailed balance (see §A 2). If however we are on a mode then with probability p we propose a move using the new tunnelling proposal distribution Q_T otherwise with probability $(1-p)$ we use the Metropolis-Hastings proposal Q. Using Q_T we uniformly choose another mode s.t.

$$Q_T(G \to G') = \begin{cases} \frac{1}{|\mathcal{G}_T|-1} & \text{if } G' \in \mathcal{G}_T \wedge G \neq G' \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Once we have proposed the new graph we accept it with the usual Metropolis-Hastings probability given by (8). It is important to note that if we used proposal distribution Q_T then α now becomes just

$$\alpha_T = \frac{P(\mathbf{X}|G')Q_T(G' \to G)}{P(\mathbf{X}|G)Q_T(G \to G')} = \frac{P(\mathbf{X}|G')}{P(\mathbf{X}|G)} \tag{12}$$

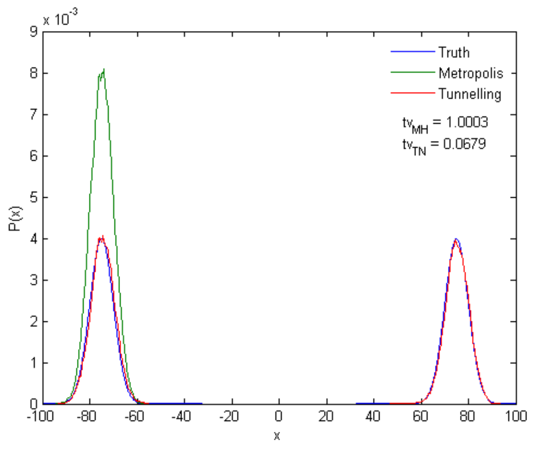

FIG. 4: For a simple 1d example of two highly separated Gaussians tunnelling Monte-Carlo (red) correctly estimates the true distribution (blue) whereas a Metropolis sampler (green) fails.

Since $Q_T = \frac{1}{|\mathcal{G}_T|-1}$ is the same for all modes. As the modes all have high scores, if we propose a tunnelling move then it stands a reasonable chance of being accepted. Thus the tunnelling scheme strikes a balance between short range movements and longer jumps.

The tunnelling scheme is expected to work in situations with highly separated peaks where Metropolis-Hastings fails. As a simple example of this consider a mixture two Gaussians $N(\mu_1, \sigma)$ and $N(\mu_2, \sigma)$ such that $P(x) \propto N(\mu_1, \sigma) + N(\mu_2, \sigma)$ with $|\mu_1 - \mu_2| >> \sigma$. Figure 4 shows that unlike the Tunnelling sampler a simple Metropolis sampler fails to estimate the distribution correctly. The Metropolis sampler will pick up either peak randomly but once it has found one it cannot escape to sample the other peak correctly.

C. Tempering

Simulated Tempering (ST) is similar in spirit to tunnelling. It also aims help the Markov chain escape local maxima. The ST scheme does not achieve this (as the tunnelling scheme does) by supplementing its proposal distribution with longer range support. Instead it combines MH with higher temperature moves. Marinari and Parisi [7] and Geyer and Thompson [8] have shown that the ST scheme can provide large benefits with regards to converge to particularly pathological probability distributions however it does have the disadvantage that only a subset of visited graphs can actually be taken as samples.

In this statistical application there is no equivalent to the physical temperature however we can introduce an analogue by writing

$$\alpha = \left(\frac{P(\mathbf{X}|G')|\eta(G)|}{P(\mathbf{X}|G)|\eta(G')|} \right)^{\beta} \tag{13}$$

Here $\beta = T^{-1}$ is the inverse temperature. Clearly as $\beta \to 0$ (infinite temperature) $\alpha = 1$ and so we have the uniform distribution one would expect. Similarly as $\beta \to \infty$ (zero temperature) $\alpha = \infty$ if G' is more likely or $\alpha = 0$ if G is more likely and we recover steepest ascent.

We supplement the state (in this case our graph G) with an index $i \in \mathcal{M} = \{1, ...m\}$ so that now the state of the system is the pair (G, i). Each index i labels a particular inverse temperature β_i. The Markov chain now performs its walk through this extended state space $\mathcal{G} \times \mathcal{M}$. To perform this walk the following algorithm is applied

(1) The graph G is updated *at* the current temperature β_i according to any scheme for which there exists a unique stationary distribution. For simplicity here we use MH.

(2) Set $j = i \pm 1$ with probabilities $q_{i,j}$. Reflecting boundary conditions imply $q_{1,2} = q_{m,m-1} = 1$ and $q_{i,i\pm 1} = \frac{1}{2}$ for $1 < i < m$.

(3) Accept or reject this new temperature with the Metropolis probability $min\{1, r\}$ where

$$r = \frac{(P(\mathbf{X}|G)|\eta(G)|)^{\beta_j}}{(P(\mathbf{X}|G)|\eta(G)|)^{\beta_i}} \frac{\pi_j}{\pi_i} \frac{q_{j,i}}{q_{i,j}} \tag{14}$$

Here the π_i's are the pseudo-priors, so called because of the way they multiply $P(\mathbf{X}|G)$'s. They are simply numbers which have to be chosen in advance. The simplest choice for the pseudo-priors is uniform however in practice this leads to the distributions not mixing and we get stuck in the hottest distribution. It is possible to update the pseudo-priors as the simulation proceeds to give uniform acceptance rates. The method used here is 'Stochastic Approximation' [9]. It penalises distributions in which we spend a lot of time by lowering their pseudo-prior value π_i and raising those of all other distributions $\pi_{i' \neq i}$. This is achieved by updating the pseudo-priors according to

$$\begin{aligned} \pi_i^{(t+1)} &= \pi_i^{(t)} \exp(-c_0/(t+n_0)) \\ \pi_{i'}^{(t+1)} &= \pi_{i'}^{(t)} \exp(c_0/[m(t+n_0)]) \end{aligned}$$

c_0 and n_0 are user specified parameters so in a sense we have simply replaced the problem of specifying suitable pseudo-priors with that of choosing c_0 and n_0, however these parameters just set how quickly we update the pseudo-priors and hence how quickly we reach uniform acceptance rates for moving up and down the temperature ladder.

In the ST algorithm described above, step (1) moves around $\mathcal{G}$ and steps (2) and (3) move around $\mathcal{M}$, thus the extended state (G, i) can explore $\mathcal{G} \times \mathcal{M}$.

Parallel tempering (PT) is very similar to simulated tempering but it makes use of parallel computing: instead of performing a random walk on our temperatures we run the chains at different temperatures in parallel. Because of this we can run all m temperatures in the same time it takes to perform a MH run. Our state space in this case has been extended to $\mathcal{G}^m$. The algorithm is similar to the ST algorithm;

(1) With probability α_0 conduct a parallel step;

  (a) Update each graph G_i using the MH scheme at temperature β_i.

(2) else conduct an exchange step;

  (a) Randomly choose a neighbouring pair i and $j = i \pm 1$. Propose swapping G_i with G_j and vice versa.

  (b) Accept the swap with probability $min\{1, r\}$

$$r = \frac{P(\mathbf{X}|G_j)^{\beta_i}\ P(\mathbf{X}|G_i)^{\beta_j}}{P(\mathbf{X}|G_i)^{\beta_i}\ P(\mathbf{X}|G_j)^{\beta_j}} \tag{15}$$

Naturally there is only benefit here if one has access to parallel computing facilities but these are becoming more and more available with even laptops having 2 or more cores.

III. SIMULATION

In order to proceed further data on which the inference will be performed are required. Since we are working with a network we know, we can simulate data from the network and use this for inference. To keep things simple we only considered binary random variables so $X_i \in \{0, 1\}\ \forall X_i$. We know from (1) how the probability distributions (in this case Bernoulli) factorise. To introduce the conditionality of child nodes we set the Bernoulli success parameter to be either θ or $(1 - \theta)$ depending on the result of a boolean function of the node's parents' values. The algorithm for simulating a single datum is then as follows:

(1) For each root node set its value to 1 with probability θ_0 and 0 otherwise.

(2) For all children of the these nodes if any of their parents' values is 1 set their value to 1 with probability θ and 0 with probability $1 - \theta$, otherwise set their value to 1 with probability $1 - \theta$ and 0 with probability θ.

(3) Repeat (2) until all nodes have been set.

For the data simulated in the project we used OR as the boolean function and set $\theta_0 = 0.5$ and $\theta = 0.8$

IV. SOME COMPUTATIONAL ISSUES

Before performing the Monte Carlo runs it serves to consider some computational issues. For each scheme there is quite a lot of computation which must be carried out for each iteration, and a large number of iterations are necessary for larger problems. Because of the intensive nature of the computations required C++ was chosen over MATLAB because of its speed. The main issues are common across schemes, namely generating the neighbourhoods $\eta(G)$ & $\eta(G')$ and computing the two log-likelihoods $P(\mathbf{X}|G)$ & $P(\mathbf{X}|G')$.

Thankfully there are several ways of ameliorating these issues. The most obvious is to store the $\eta(G)$ and $P(\mathbf{X}|G)$ from one step to the next. This saves having to compute two neighbourhoods and log-likelihoods at each iteration. If we accept the new graph G' it becomes G and we simply update the stored values for $\eta(G)$ and $P(\mathbf{X}|G)$. This provides a constant $O(1)$ computational saving of roughly a factor two.

A. Acyclicity Checking

To calculate $\eta(G)$ we must loop over every edge, which is p^2 operations, then we must try either adding, deleting or flipping the edge as appropriate. Once we have done this we must check that the new graph is acyclic, itself an expensive operation. If done naively this scales as $O(p^3)$. It is possible to keep track of the so called 'reachability matrix' for our graph so that when we come to create a new graph G' it is easy to check whether it is acyclic. The reachability matrix is a binary matrix similar to the adjacency matrix, which indicates whether there is a path from one to node to another. The algorithm for updating the reachability matrix after performing an elementary edge operation is quite complicated but it avoids the necessity of performing many matrix multiplications which are required for the naive acyclicity checking [10]. Naturally it is possible to choose G' (with the correct probabilities) without computing all graphs in $\eta(G)$ but we still need to know how many graphs there are in $\eta(G)$ for the Hastings factor in our acceptance probabilities.

B. Caching

The next saving can be found by eliminating repeated calculations of the likelihoods. Once we see a graph G and calculate its likelihood we can store this so that next time we see the graph G we do not have to re-calculate it. The log-likelihoods are indexed by the graphs that represent them. It is difficult to say what order the computational saving provided by this method is, since in theory there is a *small* chance that we never encounter the same graph twice so we gain no saving and an equally small chance that we never move off the graph we start on in which case the savings would be enormous. Amortized complexity analysis would give a better understanding of the saving but is beyond the scope of this project. In practice this simple addition provides a considerable a speed-up.

Naively storing the likelihoods in a C++ `std::vector` means to search through the list of seen graphs takes $O(n)$ operations where n is the number of encountered graphs. However if we impose a strict ordering on graphs then we can reduce this to $O(\log n)$ by using `std::set`, thus further reducing computation time. An obvious ordering arises from the fact there is a bijection between graphs and binary numbers. This bijection is clear if we represent the graphs by their adjacency matrices. Unfortnately we cannot simply convert the adjacency matrix to an integer and use the computer's built in comparison operations. The reason for this is that for, say $p = 10$ we have 90 possible edges which is more than we could store in either the 32 or 64 bits available on current machines. Thus we must directly compare the adjacency matrices as if they were binary numbers. To implement a less than operator for the adjacency matrices we take each consecutive entry to represent a higher power of two, then the highest power of 2 where the adjacency matrices differ will tell us which graph is represented by the smaller binary number.

It should be noted that inserting a graph into a `std::set`, which can be anything from $O(1)$ to at worst $O(n)$, is slower than adding one to the end of a `std::vector` which is always $O(1)$. Fortunately we perform many more look up operations than insertions and so the savings here outweigh the extra cost of insertion. We can also use `std::set` to store the graphs we have sampled, which has the same benefits as above. In practice runs with $p = 10$, $N = 200$, $T = 100,000$ took 20 minutes and 10 minutes for the `std::vector` and `std::set` implementations respectively. This is a big saving, which conveniently becomes more pronounced as T increases.

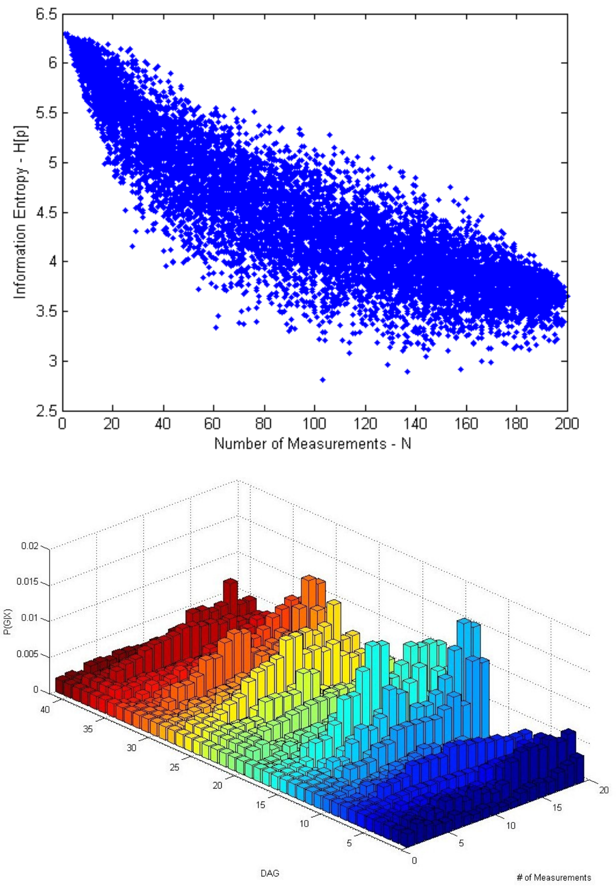

FIG. 5: (*top*) The information entropy of many subsets of 200 measurements plotted against the size of the subsets for 4 nodes. We can see a slight decrease but with large variance. (*bottom*) As we increase the number of measurements we can see that the distribution becomes more peaked around certain modal graphs. Note proximity of DAGs to each other in this chart implies nothing about whether they really are in each-others' neighbourhoods.

V. RESULTS

A. Preliminaries

The graph space in which we perform our Monte Carlo estimation is vast and with a highly variable posterior distribution. We can attempt to gain a better picture of the problem in hand by looking at two things; the information entropy and the distances between graphs.

The information entropy H is a measure of how highly peaked a distribution is. Mathematically it is defined as

$$H[p] = -\sum_i p_i \ln p_i. \tag{16}$$

A higher information entropy means the distribution is flatter. If we plot $H[p]$ against the number of measurements we can see from Figure 5 that although the entropy does decrease with increasing N it is still quite high even for $N = 200$ and the variation is quite high. To see lower entropies still we used the method described in §III to simulate data sets of varying size. The largest of which was $N = 2000$ data points. This large amount of data might seem too large to be obtained using something like a DNA Microarray but it can be obtained using flow cytometry, albeit for a smaller number of molecular components.

The distances between graphs is also an important factor governing graph space. In this setting the distance d between two graphs G and G' is defined to be the minimum number of elementary edge operations required to get from G to G', that is edge additions, deletions and flips. If we represent the two graphs as binary adjacency matrices A_G and $A_{G'}$ we can write

$$d = ||A_G \oplus A_{G'}||_F^2 - ||A_G \wedge A_{G'}^T||_F^2 \tag{17}$$

where the boolean operations $\oplus$ (XOR) and $\wedge$ (AND) are applied element-wise to the matrices and $||\cdot||_F^2$ (the Frobenius norm squared) is equal to the number of nonzero elements. The first term counts how many edges are different and the second term accounts for reversal moves.

Since we are interested in how well these schemes are estimating the target posterior distribution in a given time we need a way to measure this. For small systems (less than 6 nodes) we can enumerate the whole space $\mathcal{G}$ which means we know the true posterior. Thus we can use the total variation norm ΔP to measure the difference between our estimated distribution and the truth.

$$\Delta P = \sum_{g \in \mathcal{G}} |p_g - q_g| \tag{18}$$

Where p_g is the true probability of graph g and q_g is the probability from sampling the graph with our Monte Carlo schemes (i.e. the number of times we see the graph divided by the total number of samples). ΔP ranges from 0 which indicates a perfect estimate to 2 which tells us our estimate is completely wrong.

While this provides a very clear measure of convergence for small systems we will typically be interested in larger systems where it is no longer applicable. In this case we can examine the variance of probabilities of individual edges for signs of convergence. The edge probability $P(e)$ is the probability of a specific edge e appearing and is calculated in a very similar way to the estimated posterior on graphs

$$P(e) = \frac{1}{T} \sum_{t=1}^{T} I(e \in g^{(t)}) \tag{19}$$

Again I is the indicator function which returns 1 if e is in the edge set of graph $g^{(t)}$ and 0 if it is not.

When we run multiple MC estimations, if our Markov chains are converging to the correct posterior then we would expect the edge probabilities to converge to the same values. Thus the variance on the edge probabilities gives us as indication of how well our Markov chains are doing. If one scheme has consistently lower edge probability variances then it is likely doing better.

With the exception of Metropolis-Hastings all of the schemes have at least one parameter to set. For the Tunnelling scheme there is just one; the probability p that we propose a tunnelling jump if we are on a modal graph. A range of values of p were examined though they were restricted to $p \in (0, 1)$. The reason for this is that if $p = 0$ we simply recover the Metropolis-Hastings scheme and if $p = 1$ the state space is no longer irreducible. If $p = 1$ when we start on a mode we clearly only sample modes, and when we do not start on a mode we never sample the modes, either way our estimate of the posterior will be terrible.

Simulated tempering has several parameters which must be specified. Thankfully replacing specification of the pseudo-priors with the stochastic approximation update scheme makes this somewhat easier but we still have to specify the following

1. The number of temperatures m

2. The temperatures themselves β_i

3. The stochastic approximation parameters c_0 and n_0

As noted by Geyer and Thompson [8] this is one of the principle draw-backs of ST. Because of this a wide range of parameters were examined. We tried $m = 2, 3, 5$ & 10 in combination with a maximum temperature of $T_{max} = 1.2 - 5.0$.

Parallel tempering is related to simulated tempering and as such has a similar set of parameters. We must still specify the number of different temperatures m and the temperatures themselves β_i. Instead of the parameters c_0 and n_0 we must now set α_0 the probability to make a 'parallel' step instead of an 'exchange' step. Again a range of temperatures and number of temperatures were tested.

B. Empirical Analysis of Monte Carlo Schemes

To begin with a simplified network of 4 nodes was studied. The reasons for beginning with this subset of the full network are (a) computation times are significantly smaller on small networks and (b) The space of 4 node DAGs contains only 543 graphs, so we can enumerate all DAGs and compare our Monte Carlo results to the real distribution. If we compare the actual scores of all 543 4-node DAGs and the scores obtained by using a long Metropolis-Hastings (MH) run with $T = 5000 \approx 10 \times |\mathcal{G}|$ MCS we see there is a nice agreement between the distributions (Figure 6). The total variation for these two distributions is 0.217 which is indicates a good, but not fantastic, estimate.

Now comparing how Tunnelling performs against Metropolis-Hastings on this 4 node system we can see that there is no noticeable difference in performance between the two. Since this is a small system we can measure convergence using the total variation. In Figure 7 the total variation against number of Monte Carlo steps (MCS) was averaged over 50 independent realisations. It also appears here that varying the probability to make a tunnelling jump p does not have any effect on the convergence. Except of course where $p = 1$ which we already know does not estimate the correct distribution. Again the results were averaged over 50 independent realisations.

Next a slightly larger system of $p = 5$ nodes with $N = 2000$ data points was examined. This is the largest system for which we can compute the true posterior. The mode set was computed using steepest ascent. Starting from 500 initial random graphs we see 154 distinct modes after the steepest ascent has been run, which suggests that many of the initial graphs are converging to the same peaks. If we look at Figure 8 we can see that the fraction of accepted jumps which are tunnelling jumps between modes levels off. This shows our samplers are making tunnelling jumps consistently throughout the run. Despite the fact the Markov chains are making these jumps

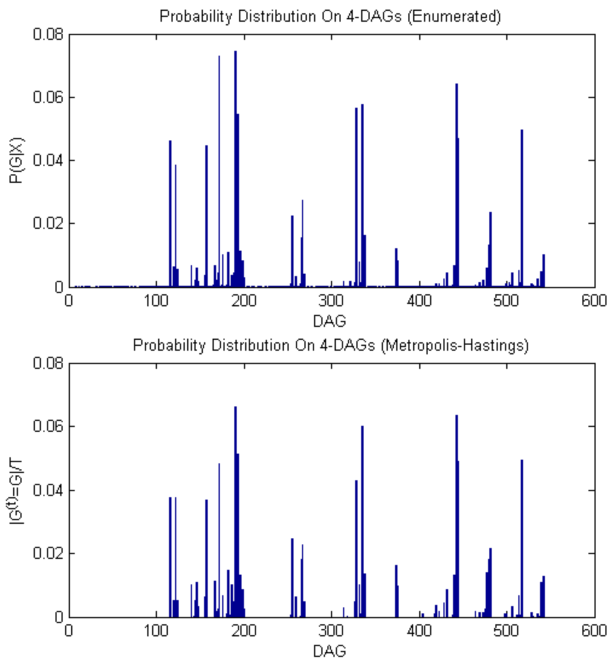

FIG. 6: The scores of all 543 4-node DAGs, and their MH estimated scores. The total variation for these two distributions is small at $\Delta P = 0.22$. Note that the positions of the graphs on the x-axis implies nothing about their locality.

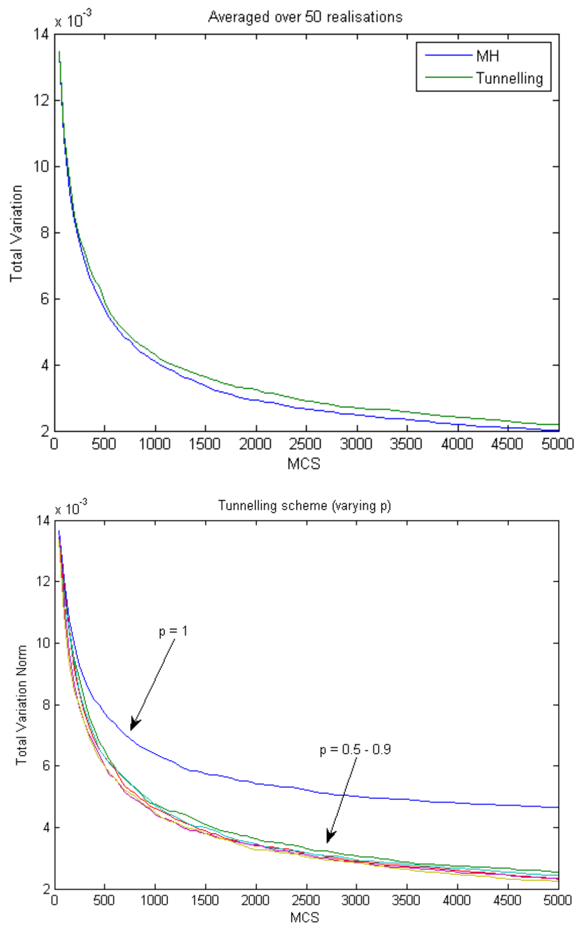

FIG. 7: (*top*)The total-variation against Monte Carlo Steps (MCS) for the Metropolis-Hastings and Tunnelling schemes. Averaged over 50 realisations. (*bottom*) The performance of the tunnelling does not depend on our choice of p in this case.

we do not see a gain (in terms of a lower total variation) over the standard Metropolis-Hastings scheme.

One possible reason for this is that the mode set is poorly specified. Figure 9 shows the true posterior distribution for 5 nodes with 2000 data points in blue. The graphs that steepest ascent picked out as modes are highlighted in red. We can see that it has failed to find some of the highest scoring graphs but has got stuck in many low scoring but locally maximal graphs.

Next it was decided to take the top ranked 100 graphs under the true posterior to be the mode set in an attempt to see if tunnelling would offer a speed up. Due to condition (10) only 35 of the top 100 could be kept. Unfortunately even this did not cause tunnelling to out perform Metropolis-Hastings. Again we still consistently made tunnelling jumps throughout the runs.

Examining the structure of the graph space provides some clues as to why Metropolis-Hastings performs as well as tunnelling. If we plot the distances between the top 50 graphs as defined by (17) we can see that the most likely graphs (upper left quadrant of Figure 10) are actually quite close together with respect to the single edge operations. This is further highlighted if we look at which graphs are in each others' neighbourhoods. The graphs which are in each others' neighbour hoods are highlighted in red.

Additionally almost all of the probability mass is contained in the first twenty graphs. This is shown in Figure 11 which shows the cumulative posterior probability against the number of DAGs ranked in descending order according to their posterior probability.

Next a larger 10 node system was examined. We used plots similar to those in Figures 10 and 11 to determine that the intermediate case with $N = 1000$ data points was most likely to show an advantage to tunnelling.

Looking at the performance of simulated tempering on the same systems we have tested tunnelling on reveals that it actually performs worse than the Metropolis-Hastings scheme. Figure 12 shows a single run of a simulated tempering sampler with typical parameters compared to a Metropolis-Hastings sampler. We can see that it fares far worse in terms of total variation. Indeed in Figure 13 we can see a possible reason for this, this plot shows how many samples were generated in total for each hundred samples which were generated at the correct temperature. It is clear that in many cases we are generating up to 100 times as many samples as are kept. It is possible to alleviate this problem through the use of the related parallel tempering scheme [11]. This poor performance of the ST scheme was found with all parameter values and for all problems we applied it to. That is

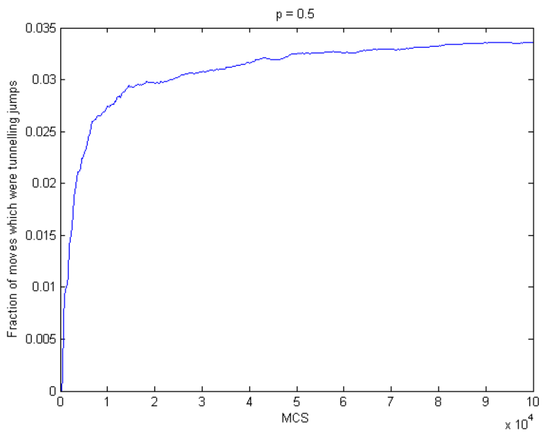

FIG. 8: The fraction of accepted moves which were tunnelling jumps as a function of MCS

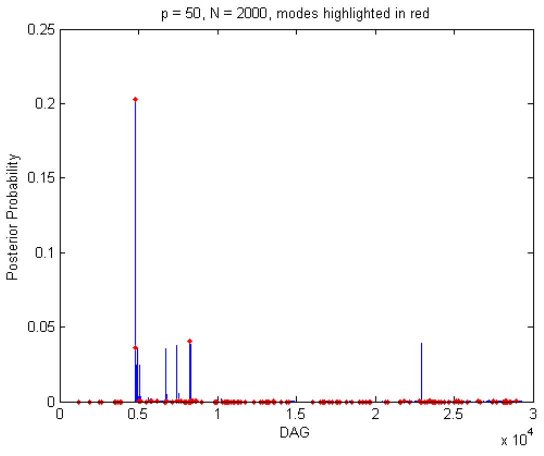

FIG. 9: The true posterior distribution for 5 nodes with 2000 data points. The modes computed using steepest ascent are highlighted in red.

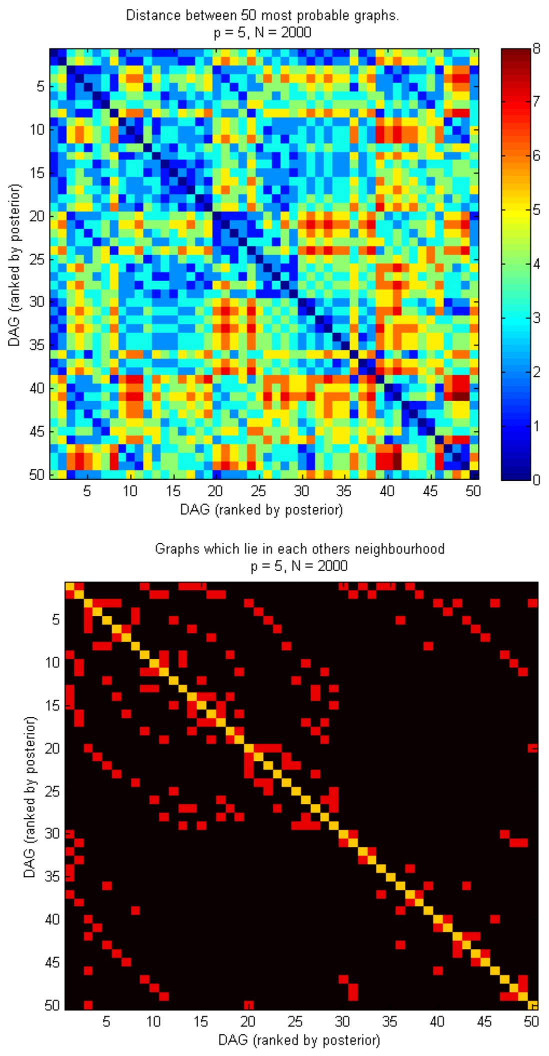

FIG. 10: (*top*) The distances between the 50 most probable graphs for 5 nodes with 2000 data. The distance is the minimum number of elementary operations needed to move between the two graphs. (*bottom*) Shows those graphs which are in each others' neighbourhoods highlighted in red. We can see that lots of the highest scoring graphs (including the top two) are in each others' neighbourhoods.

not to say that ST has no applications, indeed Geyer & Thompson showed that it can correctly estimate particularly pathological distributions where MH struggles [8].

Returning now to the larger 10 node problem we turn to the variances of the edge probabilities to indicate convergence of our Markov chains. There are 90 possible edges for the 10 node system since by definition we cannot have edges which loop back to their parent node (the diagonal entries of A_G must be zero). Since we have a reasonably large number of variances here we can use a box plot to compare the performance of different scheme. Figure 14 shows the standard deviation of edge probabilities across 50 realisations for 10 different samplers. Sampler 1 is our baseline, the Metropolis-Hastings scheme, samplers 2 - 7 are parallel tempering samplers and samplers 8 - 10 are tunnelling samplers. The parameter values for each sampler are shown in Table I. We can see that still tunnelling performs comparably to Metropolis-Hastings but encouragingly the standard deviations of the edge probabilities are lower for all of the parallel tempering schemes.

VI. SWENDSEN-WANG & CLUSTERING ALGORITHMS

In statistical physics there exists a class of Monte Carlo algorithms called 'cluster algorithms', which overcome the problem of critical slowing down for systems of interacting spins. For Ising type spins the two best known are the Swendsen-Wang and Wolff algorithms [12–14] . They are based on earlier work by Kasteleyn and For-

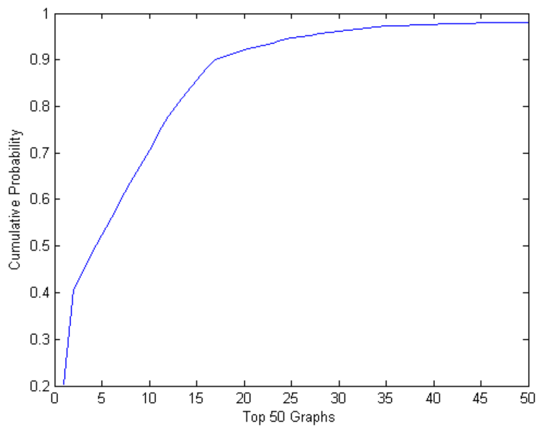

FIG. 11: The cumulative probability of ranked graphs for $p = 5$ with $N = 2000$. We have seen from Figure 10 that the highest scoring graphs are close together and from this plot we can see that they contain almost all of the probability mass.

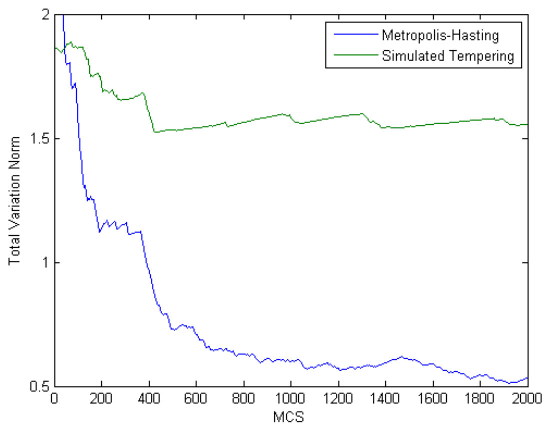

FIG. 12: A typical run of the ST scheme shown in comparison to a typical MH run for $p = 5$ nodes and $N = 2000$ data. The many wasted samples lead to a much slower convergence.

tuin [15, 16]. A particular spin s_i can be either up $(+1)$ or down (-1). A set of spins $\{s_i\}$ has an energy given by the Hamiltonian

$$H(\{s_i\}) = -\sum_{\langle ij\rangle} J_{ij}(s_i s_j - 1) \tag{20}$$

Using classical equilibrium statistical mechanics we can write the probability of seeing a particular set of spins as

$$P(\{s_i\}) = \frac{e^{-\beta H(\{s_i\})}}{Z} = \frac{e^{\beta \sum_{\langle ij\rangle} J_{ij}(s_i s_j - 1)}}{Z} \tag{21}$$

This model displays a phase transition at a critical temperature β_c. When the inverse temperature β is close to β_c long range spacial and temporal correlations arise. This causes the usual single spin flipping Metropolis algorithm to slow dramatically. The reason being that if

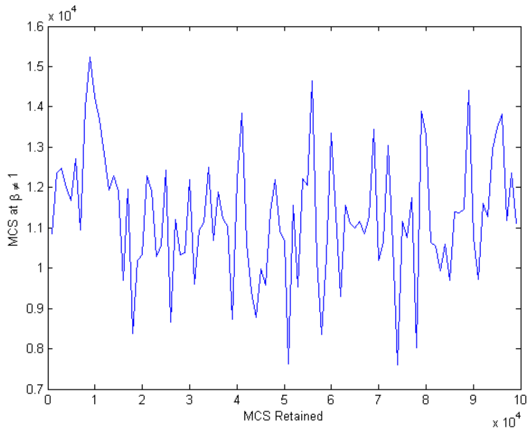

FIG. 13: In order to generate samples with the ST scheme we must discard many graphs sampled at the wrong temperatures. This plot shows the total number of samples generated to generate each 100 retained samples of a single ST run.

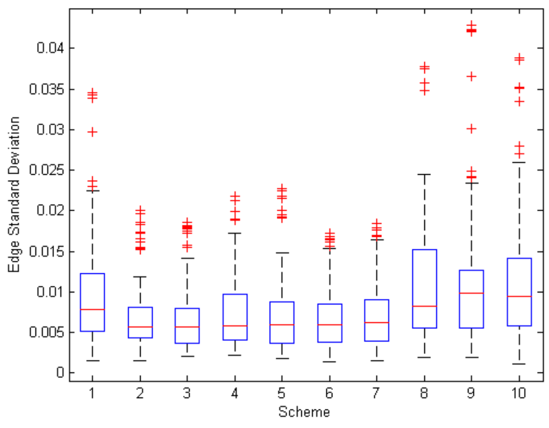

FIG. 14: A box plot of the edge probability standard deviations from 50 realisations for the 90 possible edges for the 10 nodes system with 1000 data points. The schemes and their parameters are detailed in table I. From this we can see that it appears that PT offers at least some gains in all cases where as TN appears to perform much the same as MH.

one spin is flipped it is very likely that it will belong to a cluster of similarly aligned spins and thus this will be energetically unfavourable. It is therefore very likely this spin will flip back long before we flip the whole cluster.

The cluster algorithms overcome this problem by mapping the Ising model to an equivalent model where the constant interactions J_{ij} are replaced by either infinite interactions or no interactions. Spins with $J'_{ij} = \infty$ are forced to be parallel and spins with $J'_{ij} = 0$ are completely independent. By adding these bonds with the correct probability p we ensure that the spin configurations $\{s_i\}$ have the same statistical weight as in the original Ising model. Since the clusters are now completely independent we are free to chose their orientations ran-

Sampler	Type	Parameters
1	MH	-
2	PT	$m = 5$, $T_{max} = 2.5$, $\alpha_0 = 0.9$
3	PT	$m = 3$, $T_{max} = 1.5$, $\alpha_0 = 0.9$
4	PT	$m = 5$, $T_{max} = 2.5$, $\alpha_0 = 0.7$
5	PT	$m = 5$, $T_{max} = 1.5$, $\alpha_0 = 0.5$
6	PT	$m = 10$, $T_{max} = 1.5$, $\alpha_0 = 0.7$
7	PT	$m = 5$, $T_{max} = 1.5$, $\alpha_0 = 0.9$
8	TN	$p = 0.25$, $N_{modes} = 200$
9	TN	$p = 0.50$, $N_{modes} = 200$
10	TN	$p = 0.75$, $N_{modes} = 200$

TABLE I: The various samplers and their parameters used in the 10 nodes, 1000 data case. See Figure 14

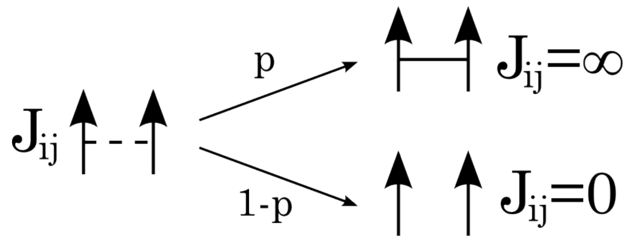

FIG. 15: We map the Ising model with interactions J_{ij} to an equivalent model with interactions $J'_{ij} = \{0, \infty\}$. Choosing p correctly ensures the two models are equivalent.

domly allowing us to update many spins at once.

We can draw a direct analogy between this problem and our graph inference problem. Just as in this application we are trying to estimate a probability distribution where we know the probability as a function of our state but we only know this up to a constant of proportionality. In this statistical physics application the state is $\{s_i\}$ which is equivalent to our G and the *unknown* constant of proportionality Z is equivalent to our missing constant in Bayes' theorem (2). The analogy holds further appeal when it is noted that G can be represented as a binary adjacency matrix A_G. The entries of A_G directly correspond to the spins in the Ising model. Of course there are several complications to consider and overcome.

One of the keys to the Swendsen-Wang formalism is that it is possible to factorise $P(\{s_i\})$ in terms of the interacting pairs, that is

$$P(\{s_i\}) \propto e^{\beta} \prod_{\langle ij \rangle} e^{J_{ij}(s_i s_j - 1)} = \prod_{\langle ij \rangle} p_{ij} \tag{22}$$

The reason for this is that the Hamiltonian (20) is written in terms of a sum over interacting pairs. We are interested in the posterior probability $P(G|\mathbf{X})$ which is equivalent in the Ising model to $\exp(-\beta H)/Z$ thus it is clear that our *log-likelihood* is directly equivalent to $-\beta H$.

Thankfully our log-likelihood (5) is also written as a sum over 'interacting' terms. In its present form this might not be obvious as the dependence of (5) on G, and hence A_G, is implicit. It is hidden in the N_{ijk}'s. It is non-trivial to write the log-likelihood explicitly as a function of the adjacency matrix A_G. This is where one of the complications arises.

A node can have anywhere from zero to $(p-1)$ parents: we must not only consider pairs of spins but pairs of spins, triples of spins and so on up to $(p-1)$-tuples of spins. Each grouping of spins has a different probability of begin bound together to ensure the models are equivalent. Careful, here p is the number of nodes not the probability of adding a bond between spins. To further complicate matters there is no sense of locality in A_G in the same fashion there is in $\{s_i\}$ so we must take into account all possible combinations of these edges. Extending the right hand side of (22) to account for these facts yields

$$P(G|\mathbf{X}) \propto \prod_{j=1}^{p-1} \prod_{\langle i_1 \dots i_j \rangle} p_{i_1 \dots i_j} \tag{23}$$

Writing the factors of the posterior $p_{i_1 \dots i_j}$ explicitly in terms of the entries i_1 through i_j in the adjacency matrix is a remaining challenge. However in principle once this is achieved the mapping is complete.

VII. DISCUSSION

We began §V B by looking at a small 4 node subset of the full network shown in Figure 1 with 200 data points. The results show that in this small case Metropolis-Hastings performs admirably providing a good estimate of the posterior distribution with a total variation of $\Delta P = 0.217$. Indeed in this case we found that tunnelling offers no benefit over Metropolis-Hastings for any value of the jump probability p. Figure 7 shows that our tunnelling performance does not depend on p, however it is churlish to conclude that the tunnelling scheme is insensitive to our choice of p. In this case it seems obvious that its performance will not depend on p since tunnelling confers no advantage. If there is a case where tunnelling provides a noticeable speed up then presumably the amount of speedup must depend on p, for if we reduce p to zero then we just recover Metropolis-Hastings and any benefit is lost.

Looking at the information entropy for the $N = 200$ case we concluded that tunnelling would not offer any benefit here for two reasons: firstly the size of the system is simply too small and secondly the distribution was too dispersed. Here we have 4 nodes which means there are only 12 possible edges thus it is only a short distance between any two graphs. Also because of the high information entropy there is at least some mass on most graphs thus there exist paths between the most likely graphs which Metropolis-Hastings can take.

On the basis of this we decided to examine systems in a much lower entropy regime with $N = 2000$. In addition we looked at 5 nodes, which while not an especially large system does mean we can still use total variation to measure convergence. Here we still found that tunnelling did not mix any quicker than Metropolis-Hastings. One possible factor we thought could be hindering tunnelling was the choice of mode set. Starting from 500 initial graphs steepest ascend converged to just 154 modal graphs, i.e. lots of graphs were approaching the same peaks. Given that the size of the state space $|\mathcal{G}|$ is roughly 30,000 the modal set is still a very small subset of the whole space.

The fact that many reach the same peaks might then seem to be encouraging but if we highlight the 154 modes on the true posterior distribution (Figure 9) we can see immediately that we have missed several of the highest scoring graphs and picked up many low scoring graphs. Figure 8 shows that we consistently make tunnelling jumps throughout the run so it would seem we spend a lot of time jumping between relatively low scoring graphs which will not help us considerably with our estimate of $P(G|\mathbf{X})$.

Looking at the distances between the 50 most likely graphs for p=5, N=2000 we can see that steepest ascent fails to pick up some of the high scoring graphs because many of them are in fact in the neighbourhood of the most probable graph (Figure 10). The distances between the 15 most likely graphs are all a few steps at most and thus Metropolis-Hastings does not struggle to get between these and there is no benefit to be had from tunnelling. Figure 11 shows that the first 20 most likely graphs contain over 90% of the probability mass and so once we find this peak with what ever scheme, we will have a reasonable estimate. By going to $N = 2000$ we have, if anything, gone too far and picked out a single peak.

When considering the larger 10 node system we also estimated plots like Figures 10 and 11 for various values of N and found that the mid range value of $N = 1000$ looked to have the more desired property of relatively spaced out peaks.

Before moving to the larger system let us consider the simulated tempering scheme. In all cases tested and for all parameter values the simulated tempering performed worse than Metropolis-Hastings. It is entirely possible that since we make moves at higher temperatures and only drop down occasionally to sample from the desired distribution that the Markov chain is exploring regions of $\mathcal{G}$ which contain low scoring graphs and then when we sample at the correct temperature $\beta = 1$ we are sampling a relatively low scoring graph. However in practice this effect is almost entirely dominated by a second effect which was alluded to in the previous sentence: we can only keep samples at $\beta = 1$. Thus any samples we make at $\beta \neq 1$ must be discarded.

This problem becomes worse as m increases and also if we poorly specify the temperatures. The neighbouring temperatures must be fairly close for the acceptance rates for moving between them to be appreciable. Thus if we want a high maximum temperature we must have lots of temperatures which leads to many wasted samples. In this graph setting no regime was found where the gains of simulated tempering outweighed this disadvantage.

Returning now to the 10 node problem we saw that we must estimate whether we think our chains have converged by examining the standard deviation across 50 realisations of the 90 different edge probabilities (Figure 14). Using this as a measure for how well our samplers were estimating the posterior distribution we found that even in this case, which looked the most promising, that tunnelling still performs much the same as Metropolis-Hastings for a range of jump probabilities.

The fact that we have failed to locate a regime where tunnelling offers an advantage would suggest that MCMC on graphs is not as difficult as previously believed and provides added confidence in results based upon standard Metropolis-Hastings estimation. In addition it appears that in this situation the parallel tempering scheme does offer an advantage. The edge standard deviations are noticeably lower for all choices of parameters. Although parallel tempering still discards many samples we can see the benefits in the same time as a Metropolis-Hastings sampler since it utilises parallel computing facilities to run in the same time. This is a pleasing result which suggests that if one wants to be as confident as possible in the estimate of $P(G|\mathbf{X})$ then it is possible to do better than Metropolis-Hastings.

Finally we discussed possible application of Swendsen-Wang type algorithms to the graph problem. While there remain several complications and considerations to overcome the analogy is clear and direct. Our log likelihood can be seen directly to play the role of the Hamiltonian and the entries of the adjacency matrix the role of the spins. Whether in practice this provides a speed up remains to be seen and would probably only be seen in larger systems where we have the usual problems of diagnosing convergence. However this in an obvious extension to the work here.

Acknowledgments

I would like to thank Dr Sach Mukherjee and Dr Mario Nicodemi for supervising this project. This work was funded by the EPSRC.

APPENDIX A

1. Proof of Detailed Balance For The Metropolis-Hastings Scheme

Detailed balance implies

$$P(G|\mathbf{X})T(G \to G') = P(G'|\mathbf{X})T(G' \to G) \tag{A1}$$

where T, the transition matrix, is the probability of moving to G' from G. One can write T in terms of the proposal distribution and acceptance probability such that

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{T(G' \to G)}{T(G \to G')} = \frac{|\eta(G)|}{|\eta(G')|}\frac{\min\{1,\alpha^{-1}\}}{\min\{1,\alpha\}} \quad \text{(A2)}$$

Without loss of generality this is

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{|\eta(G)|}{|\eta(G')|}\frac{1}{\alpha} \quad \text{(A3)}$$

By substituting in for α from (8)

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{|\eta(G)|}{|\eta(G')|}\frac{P(G|\mathbf{X})|\eta(G')|}{P(G'|\mathbf{X})|\eta(G)|} \quad \text{(A4)}$$

$$= \frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} \quad \text{(A5)}$$

□

2. Proof of Detailed Balance For The Tunnelling Scheme

To show that (A1) is satisfied for the Tunnelling scheme we consider the three possible cases where neither, one or both of G and $G\prime$ are in $\mathcal{G}_T$. If neither are modal graphs then we simply make a Metropolis-Hastings move and the proof in §A 1 holds.

Now consider the case where both graphs are in $\mathcal{G}_T$. Condition (10) ensures that we cannot move between these two using the usual Metropolis-Hastings proposal Q such that

$$T(G \to G') = p \quad \frac{1}{|\mathcal{G}_T| - 1} \quad \min\{1, \alpha_T\} \quad \text{(A6)}$$

where we have α_T given by (12) so that

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{T(G' \to G)}{T(G \to G')} = \frac{p\,(|\mathcal{G}_T| - 1)}{p\,(|\mathcal{G}_T| - 1)}\frac{\min\{1,\alpha_T^{-1}\}}{\min\{1,\alpha_T\}} \quad \text{(A7)}$$

cancelling and without loss of generality

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{1}{\alpha_T} = \frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} \quad \text{(A8)}$$

□

The last case to consider is that when only one of G or G' is a modal graph. Assume G' is the modal graph. We can only move onto this graph by a local Metropolis-Hastings move which leads to

$$T(G \to G') = (1-p)|\eta(G)|^{-1}\min\{1,\alpha\} \quad \text{(A9)}$$

Now the reverse can again only occur via local moves, but since G' is a mode this is only invoked with probability $(1-p)$, thus

$$T(G' \to G) = (1-p)|\eta(G')|^{-1}\min\{1,\alpha^{-1}\} \quad \text{(A10)}$$

Combining these two gives

$$\frac{P(G|\mathbf{X})}{P(G'|\mathbf{X})} = \frac{T(G' \to G)}{T(G \to G')} = \frac{(1-p)}{(1-p)}\frac{|\eta(G)|}{|\eta(G')|}\frac{\min\{1,\alpha^{-1}\}}{\min\{1,\alpha\}} \quad \text{(A11)}$$

Cancelling the $(1-p)$'s we can see from (A2) that this satisfies (A1). □

[1] S. Mukherjee and T.P. Speed. Network Inference Using Informative Priors. *PNAS*, 105(38):14313–14318, Sept. 2008.

[2] S.L. Lauritzen. *Graphical Models.* O.U.P., Oxford, U.K., 1996.

[3] R.W. Robinson. *Counting Labeled Acyclic Digraphs*, pages 239–273. New Directions in the Theory of Graphs. Academic Press, 1973.

[4] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of State Calculations by Fast Computing Machines. *Journal of Chem. Phys.*, 21(6):1087–1092, 1953.

[5] W.K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

[6] S. Mukherjee. Tunnelling Monte Carlo. March 2009.

[7] E. Marinari and G. Parisi. Simulated Tempering: a New Monte Carlo Scheme. *Europhys. Lett.*, 19(6):451–458, July 1992.

[8] C.J. Geyer and E.A. Thompson. Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference. *Journal of the American Statistical Association*, 90(431):909–920, Sept. 1995.

[9] M.T. Wasan. *Stochastic Approximation.* C.U.P., Cambridge, U.K., 1969.

[10] S. Hill. MCMC Single Edge Change Acyclicity Checks. May 2009.

[11] J.S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer Series in Statistics. Springer, New York, USA, 2008.

[12] R.H. Swendsen and J.S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.*, 58(2):86–88, 1987.

[13] U. Wolff. Collective Monte Carlo Updating for Spin Systems. *Phys. Rev. Lett.*, 62(4):361–364, 1989.

[14] M.E.J. Newman and G.T. Barkema. *Monte Carlo Methods in Statistical Physics.* O.U.P., Oxford, U.K., 1999.

[15] P.W. Kasteleyn and C.M. Fortuin. On The Random Cluster Model, I: Introduction And Relation To Other Models. *Physica*, 57:536–564, 1972.

[16] M. Nicodemi. Percolation and cluster formalism in continuous spin systems. *Physica A*, 238:9–22, 1997.