McGill University

COMP 767

Probabilistic Graphical Models - Final Project Report

Combining Deterministic Inference with Monte Carlo Methods

December 17, 2019

Author:
Antonios Valkanas

Student Number:
260672034

1 Introduction

1.1 Background and Motivation

Typically, inference problems involve a large number of latent variables which makes inference hard. The use of structure allows us to improve the efficiency of inference, [Jordan (2004)]. Probabilistic Graphical Models (PGMs) are a natural method of leveraging the structure of probability distributions.

We can split inference methods for PGMs into two groups; deterministic approaches and Monte Carlo approaches. Both methods have their own merits and disadvantages. Deterministic approaches are swift and quite accurate but suffer from biases that are not easily measurable. On the other hand, Monte Carlo based algorithms are unbiased and consistent estimators. Nevertheless, these properties are asymptotic making the required number of samples before convergence large and hard to know in advance, typically resulting in lengthy computation times.

In [Lindsten et al. (2018)] a new algorithm is proposed offering to provide the advantages of both deterministic and sampling based inference. The key idea is to use sequential Monte Carlo in conjunction with deterministic methods. The authors of the same paper were able to demonstrate the result of their proposed algorithm by making use of deterministic algorithms such as loopy belief propagation, expectation propagation, and Laplace approximations.

1.2 Main Contribution

The aim of this project was to extend the results of [Lindsten et al. (2018)] from Sequential Monte Carlo (SMC) to Markov Chain Monte Carlo (MCMC). The SMC sampler proposed by [Lindsten et al. (2018)] makes use of twisting functions that are the outputs of a deterministic inference algorithm to provide a look-ahead ability for their method.

The idea proposed in in this project relies on using these same functions as an estimate of the high probability density region of the state space to initialize the Markov chain of the MCMC in a good region. This in turn should decrease the burn-in time of the chain and improve the overall effective sample size for a given number of steps. Both methods are compared.

2 Relevant Work

2.1 Stochastic Inference Algorithms

Monte Carlo methods have been applied for decades due to their ability to estimate probabilistic quantities from repeated sampling. There exist a number of families of algorithms that fall under the Monte Carlo umbrella such as Sequential Monte Carlo (SMC), Markov Chain Monte Carlo (MCMC) and more recently particle filtering algorithms such as the marginal particle filter, also known as the Blackwell-Rao filter.

These algorithms have been successfully used in continuous and discrete domains for decades. However, with the advent of graphical models there has been a need to modify them to be suitable for PGM structured problems.

The use of MCMC for Bayesian networks dates back to [Pearl 1987]. The algorithm provides a consistent and unbiased estimate of the distribution. However, there is a burn-in period at the start of the algorithm during which we must discard all samples. In general running MCMC can be a very time consuming process. Additionally, determining convergence of the Markov chain is a difficult task.

Another problem that can naturally arise in various applications are Dynamic Bayesian Networks (DBN). In that case, a method that has distinguished itself is the Blackwell-Rao filter [Doucet et al. 2000]. This method is able to apply said particle filter to a graph that is changing over time.

As mentioned in the previous section, recently, Sequential Monte Carlo has been proposed as a way of constructing a graph via successively adding nodes and edges to a sub-graph and sampling the sub-graph to get intermediate distributions. There is a clear similarity between creating a sub-graph and sampling it and inference in a Dynamic Bayesian Network setting.

The main advantage of the Blackwell-Rao filter is that it allows for the marginalization of all variables that are not being sampled. Essentially, using the Kalman filter, junction tree algorithm, or any other finite dimensional optimal filter it can increase the efficiency of sampling.

A drawback of Blackwell-Rao is that sampling in high-dimensional spaces can be inefficient. In some cases, however, the model has tractable substructure (defined in section 2.3), which can be analytically marginalized out. In that case other algorithms such as Naive Mean Field in conjunction with SMC using the twisting function approach (defined in section 2.4) can yield much quicker results.

2.2 Sequential Monte Carlo in Graphs

Given a graphical model defined by the graph $G = \{V, E\}$, with vertex set V and edge set E, we can get the joint probability density function (JPDF) of the set of random variables indexed by V, $X_V = [x_1, ..., x_{|V|}]^T$, by a product of factors over the cliques of the graph:

$$p(X_V) = \frac{1}{Z}\Pi_{c\epsilon C}\psi_c(Xc)$$

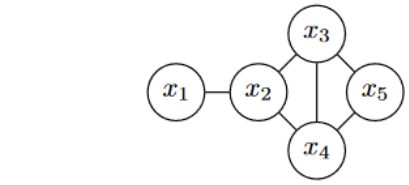
where C is the set of cliques in G, ψ_c is the factor for clique C, and $Z = \int \Pi_{c\epsilon C}\psi_c(Xc)dX_V$ is the partition function.

To be able to use SMC for inference in PGMs a sequence of target distributions is required. However, these target distributions do not have to be the marginal distributions of each one of our variables. As noted in [Naesseth et. al (2014)] it suffices to contruct a sequence of

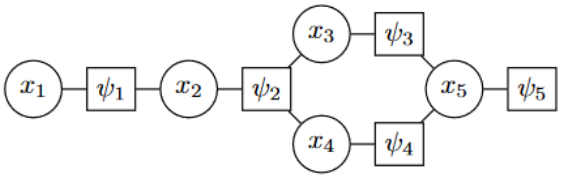
target distributions such that in the final iteration we have the marginal we are looking for, namely X_V. Having said that, all the intermediate target distributions do not matter and choosing one of the many possible ways of decomposing the graphs is valid. In [Naesseth et. al (2014)] the following method is explained from figures 1,2.

The sequential decomposition of the PGM is then based on $\tilde{\gamma}$ which is the unnormalized factor product which is estimated by the SMC particles at each iteration.

Then, $\tilde{\gamma}_k(X_k) := \Pi_{l=1}^{K}\psi_k(X_{I_k})$, where $I_k \subseteq \{1,..,|V|\}$. $L_l := \cup_{l=1}^{k} I_l$, for $k\epsilon\{1,...,K\}$. By definition, $L_K = V$ and the JPDF $p(X_{L_K})$ will be proportional to $\tilde{\gamma}_k(X_{L_k})$).



(a) Undirected graph.



(b) Factor graph.

Figure 1: A Markov network and its factor graph. [Naesseth et. al (2014)]

Figure 1 represents a simple example of a Markov network and its corresponding factor graph and Figure 2 shows two sample decompositions of the factor graph.

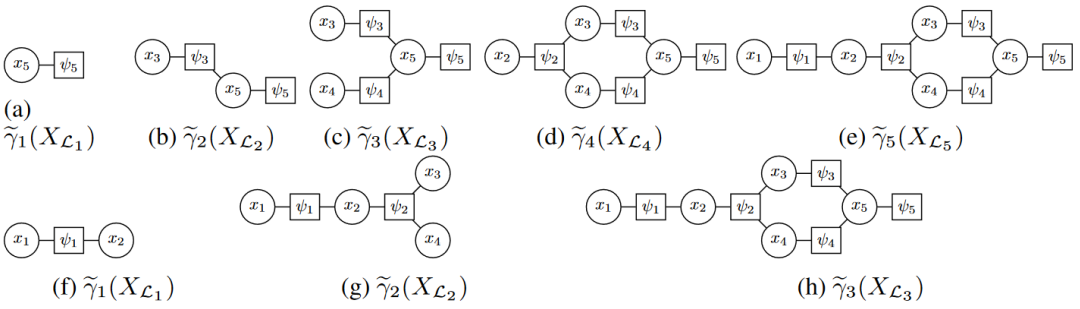


Figure 2: Two different factor decompositions of the graph. [Naesseth et. al (2014)]

Sequential Monte Carlo is a family of algorithms that employ importance sampling to a

sequence of target distributions. As shown in Algorithm 1, in the first step we sample from a proposal distribution q, compute the weights and normalize them.

Then the SMC sampler performs resampling. Resampling is necessary because it solves the problem of disappearing particles (particles that are multiplied by numbers close to zero in low probability regions). This is better than the alternative solution of using a huge number of particles in the initial iteration so that enough particles can still "make it" to the high probability regions even if most end up in close to zero probability regions.

We then have a propagation step where the particles are propagated forward by simulating from the proposal distribution q conditioned on the previous steps: $q_t(x_t|x_{1:t-1})$. In practice we compute this up to a constant using $q_t(x_t|x_{1:t-1}) \propto \frac{\gamma_t(x_t)}{\gamma_{t-1}(x_{1:t-1})}$. Finally, we recompute the importance weights $w_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1})} q_t(x_t|x_{1:t-1})$.

This method after k iterations allows us to estimate both the target final distribution as well as the partition function.

Algorithm 1 Sequential Monte Carlo (all steps are for $i = 1, \ldots, N$)

1. Sample $x_1^i \sim q_1(x_1)$, set $\widetilde{w}_1^i = \gamma_1(x_1^i)/q_1(x_1^i)$ and $w_1^i = \widetilde{w}_1^i / \sum_{j=1}^N \widetilde{w}_1^j$.
2. **for** $t = 2, \ldots, T$:
   (a) *Resampling:* Simulate ancestor indices $\{a_t^i\}_{i=1}^N$ with probabilities $\{\nu_{t-1}^i\}_{i=1}^N$.
   (b) *Propagation:* Simulate $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_t^i})$ and set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$.
   (c) *Weighting:* Compute $\widetilde{w}_t^i = \omega_t(x_{1:t}^i) w_{t-1}^{a_t^i} / \nu_{t-1}^{a_t^i}$ and $w_t^i = \widetilde{w}_t^i / \sum_{j=1}^N \widetilde{w}_t^j$.

To get the target distribution $P(x_t)$ use the empirical distribution $\Sigma_{i=1}^N \tilde{w}_i^t x_i$. The normalization constant is equal to $\hat{Z}_t = \Pi_{s=1}^t \frac{1}{N} \Sigma_{i=1}^N \tilde{w}_s^i$. The fact that the SMC estimator is unbiased and consistent is shown in [Del Moral et al. 2006].

2.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo is a family of algorithms used for sampling from a probability distribution. In general, MCMC works by constructing a Markov chain that has the desired distribution as its equilibrium distribution. To sample the distribution we record the current state of the chain and take a step. The steps are proposed from a random walk process. As a rule, the more steps we take the better the algorithm's results.

In this work we will use one of the most well known MCMC algorithms, the Metropolis-Hastings algorithm. The Hastings algorithm works as follows:

Algorithm - 2 MCMC

1. Set initial state x_0 and set iteration counter $k = 0$ and the maximum number of steps K.

2. Repeat until $k = K$:
   (a) Generate a random candidate x' and according to the proposal distribution $g(x'|x_t)$.
   (b) Calculate acceptance probability $A(x_t, x') = \min\{1, \frac{P(x')g(x_t|x')}{P(x_t)g(x'|x_t)}\}$.
   (c) Draw a sample $u_t \sim [0, 1]$ uniformly.
      i. If $u_t \leq A(x_t, x')$ accept proposal and set $x_{t+1} = x'$.
      ii. If $u_t > A(x_t, x')$ reject proposal and set $x_{t+1} = x_t$.
   (d) Set k = k + 1.

Some key points in evaluating the performance of MCMC methods are the auto-correlation of samples as well the effective sample size.

It is clear that a high correlation between samples weakens an estimator. Suppose we had two estimators; one using n uncorrelated data points, and one using n correlated data points with the correlation between two samples that are separated by time lag k being characterized by the function $\rho(k)$. We would expect that the quality of the first estimator would be better. However, how exactly can we quantify that?

To do that we define the effective sample size (ESS) which tells us roughly what amount of uncorrelated datapoints would do as good a job at modelling the posterior as the correlated datapoints we have. Obviously ESS $\leq n$. The effective sample size is given by the following formula:

$$ESS = \frac{n}{1 + 2\sum_{k=0}^{\infty} \rho(k)}$$

Despite having a number of diagnostic tools it can be hard to be certain as to whether an MCMC run has converged to the true distribution. Therefore it is common to run multiple chains from various starting points and checking to verify that they converged to the same distribution.

In practice avoiding a bad starting point for the algorithm can be hard so we disregard all the information given by the chain before a number of steps is reached. This is referred to as the burn in period.

2.4 Deterministic Inference Algorithms

There exist many inference algorithms for PGMs. To name a few we have Expectation Propagation, Laplace Approximation, Belief Propagation and many others.

2.4.1 Belief Propagation

The Belief Propagation (BP) algorithm is a method for computing marginals. As noted earlier, computing marginal probability functions can require summing an exponentially large number of terms. The computed marginal probability functions will be exact if the factor graph has no cycles, but the BP algorithm is still well defined and a lot of the time approximates well the true marginals even when the factor graph does have cycles as explained in [Yedidia 2005].

BP works by message passing between adjacent factor nodes. The message $\delta_{i,j}$ from the factor node i to the variable node j is an estimate of the relative probability that j is in a particular state:

$$\delta_{i,j}(S_{i,j}) = \Sigma_{C_i - S_{i,j}} \psi_i(C_i) \Pi_{k \epsilon N b_{i-j}} \delta_{k,i}(S_{i,k})$$

where $S_{i,j}$ is a sepset, ψ_i is the factor i and Nb_{i-j} is the neighborhood of i excluding j.

The marginal at each cluster C_i is then proportional to the belief β at the cluster:

$$\beta(C_i) = \psi_i(C_i) \Pi_{k \epsilon N b_i} \delta_{k,i}(S_{i,k}).$$

2.4.2 Naive Mean Field

Given a graphical model based on a graph G, we base our description of mean field methods on the notion of a tractable subgraph defined a subset of connected nodes I where it is feasible to perform exact calculations. We then approximate the distribution on the tractable graph.

Our objective can be thought of as minimizing the KL divergence between the simplified distribution of the tractable graph and the original distribution. More concretely, we wish to find $\arg\min_{q \epsilon Q} D(q||p)$, where q is the product of the factors of the graph and p is the product over a subset of the factors.

This is clearly an optimization problem. There exists a closed form solution as shown in [Koller & Friedman 2012] for iterative updating where at each step the update is:

$$q_i(x_i) \propto exp(\mathbf{E}_{MB(i)} \Sigma_{I|i \epsilon I} ln \phi_I(x_I))$$

This update rule is repeated until convergence.

2.5 Twisting Functions

As previously discussed, SMC works by successively updating a set of intermediate distributions. However, there is no reason why the intermediate distributions should match the final distribution. Therefore, during the application of SMC there is no way of predicting the output of our algorithm even if we have already calculated multiple intermediate distributions.

A nice way of providing the algorithm a "look-ahead" capability is proposed in [Lindsten et al. (2018)]. The introduction of a twisting function that we multiply our intermediate

factor product by can provide us with the ability to bias the intermediate distributions towards the final one.

$$\tilde{\gamma}_k^{\psi}(X_k) := \psi_k(X_k)\tilde{\gamma}_k(X_k)$$

Clearly, the optimal $\psi(X_k)$ would be the product of the factors that our intermediate distribution has not been multiplied with yet. That way we would just go directly to the final distribution which is equal to the product of all the factors. Obviously, we cannot find that optimal $\psi(X_k)$ without solving the original problem so we need to use other functions.

While many functions can be used for $\psi(X_k)$, a very compelling choice would be to use deterministic inference methods can be used to approximate the optimal twisting function. Indeed, [Lindsten et al. (2018)] demonstrate that the use of Belief Propagation, Expectation Propagation and Laplace Approximation all yield reasonable estimates of the optimal twisting function that do in fact provide a good "look-ahead" effect on the SMC algorithm resulting in better convergence for a given number of particles.

3 A New Approach - Informed MCMC

Drawing inspiration from the SMC framework in conjunction with deterministic inference we propose combining deterministic inference methods with MCMC. More specifically, in the Informed Markov Chain Monte Carlo (IMCMC) we use information provided by a deterministic inference method to get the marginals of each node. Using these marginals we estimate a high-probability state of the system and use that state as the initial state x_0 of the Markov Chain.

For the method to work we make two assumptions. Firstly, we assume that a MAP assignment of the nodes based on their marginals will lead to a high probability joint distribution. Secondly, we assume that either the high probability regions are connected together or the posterior distribution is uni-modal.

As shown in Algorithm 3 - step 1, it does not matter which deterministic inference algorithm we choose to use as all we care about is a MAP assignment of the marginals. This means that unlike in the twisted SMC algorithm a less accurate method such as Naive Mean Field will not necessarily perform worse than Belief Propagation in our setup as long as they have the same MAP assignment.

Algorithm - 3 IMCMC

1. Approximate the marginals using a deterministic inference method (e.g. LBP, Naive Mean field) and set each component of x_0 to the state with the highest marginal.
2. Set initial state x_0 and set iteration counter $k = 0$ and the maximum number of steps K.
3. Repeat until $k = K$:

(a) Generate a random candidate x' and according to the proposal distribution $g(x'|x_t)$.

(b) Calculate acceptance probability $A(x_t, x') = \min\{1, \frac{P(x')g(x_t|x')}{P(x_t)g(x'|x_t)}\}$.

(c) Draw a sample $u_t \sim [0, 1]$ uniformly.

i. If $u_t \leq A(x_t, x')$ accept proposal and set $x_{t+1} = x'$.

ii. If $u_t > A(x_t, x')$ reject proposal and set $x_{t+1} = x_t$.

(d) Set k = k + 1.

The expected advantages of this method include a smaller burn in time for chain. The reason for this is rather obvious, if we start in a high probability density region the chain will not have to take a potentially large number of steps to get to a high probability density region because it's already there and under assumption 2 (stated earlier in this section) we know that all high probability regions are connected. Given a good starting location the chain should immediately start to oscillate around the mean of the random variable and sample sequences will have a small auto-correlation for a time lag k.

On the other hand, if we started away from the mean then we would expect that initially the algorithm would be taking consistent steps towards the mean of the distribution. But this would mean that the sample sequences would have a high auto-correlation.

Clearly, the effect of the start point would diminish as the number of steps got very large however the difference of a good starting point would be noticeable before a lot of steps were taken. For a small number of steps a good starting point would reduce the auto-correlation and therefore significantly increase the effective sample size.

4 Empirical Evaluation

To be able to compare the results with the twisted SMC (TSMC) sampler of [Lindsten et al. (2018)] we use the same setup as in their paper's Ising model experiment.

The 2D-Ising model models a square lattice of particles that form a grid where each particle can have a spin up (+1) or down (-1). Each particle represents a node on the graph and its spin depends on its neighbors' spin and a self potential. Most nodes have four neighbors up, down, left, right unless they are boundary nodes in which case they could have fewer neighbors.

These relations can be summarized in the masked adjacency matrix A where $A_{i,i}$ represents the self potential of a node and $A_{i,j}$ denotes the potential between nodes and is non-zero only if the nodes are adjacent.

$$p(x_i) = \frac{1}{Z}\exp(\sum_i A_{i,i}x_i + \sum_{i,j<i} x_i A_{i,j} x_j), \quad x_i \in \{-1, +1\}$$

In [Lindsten et al. (2018)] the self potentials at each node are drawn from a uniform distribution $A_{i,i} \sim \text{Uniform } [0, 1]$ and the dependence on the neighbor nodes is set to the same constant for all nodes $A_{i,j} = J$ with $J = 0.44$.

A quantity that we use to measure the quality of the MCMC method we propose is the magnetization of the grid. The magnetization is just the sum of the expectations of all the nodes: $M = \sum_{i,j} p(x_{i,j})x_{i,j}, \quad x_{i,j} \in \{-1, +1\}$.

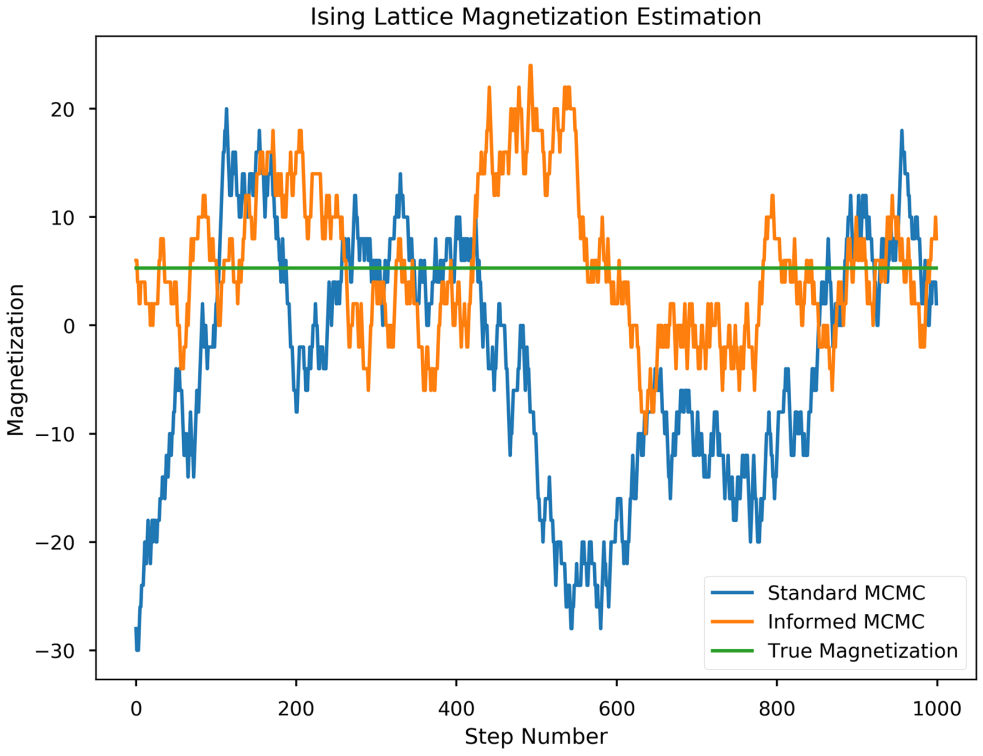


Figure 3: MCMC vs IMCMC performance on the same Ising grid when computing magnetization of the lattice.

We observe that the IMCMC chain starts close to the true magnetization value. On the other hand, the standard MCMC algorithm needs to take a number of steps before it starts oscillating around the mean. Figure 3 illustrates that a good starting position reduced the burn-in time by about 200 steps.

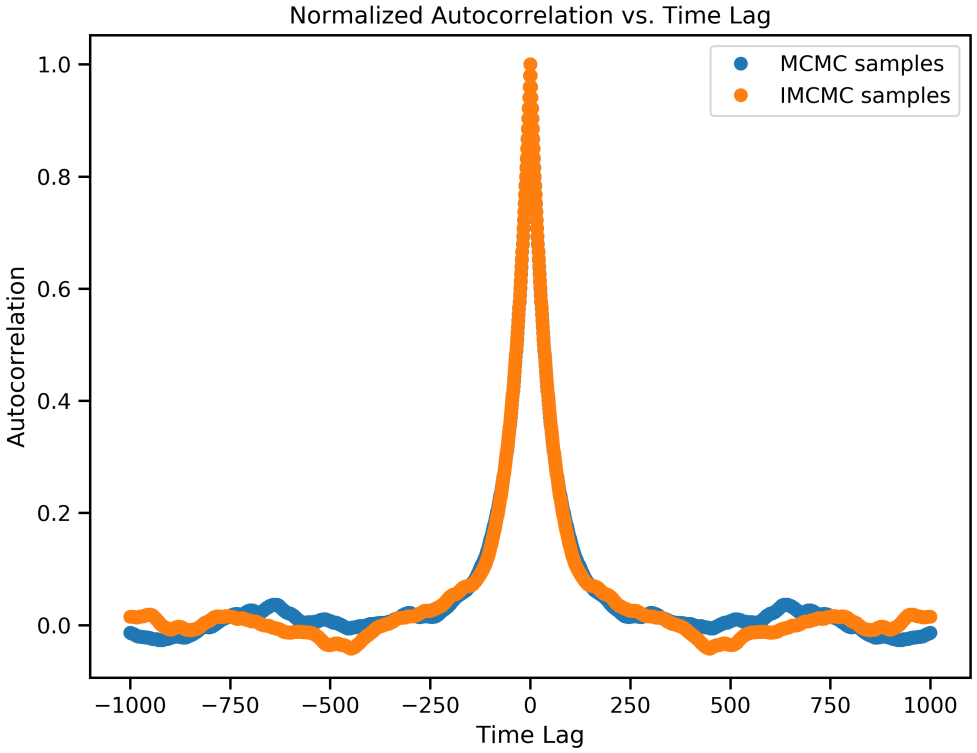


Figure 4: MCMC vs IMCMC performance on the same Ising grid when computing magnetization of the lattice.

While the example run in Fig. 4 may not seem significant after averaging 1000 runs that each take 1000 steps for various grid sizes we observe that on average the ESS increases by significant amount.

Ising Grid Size	**Mean Factor Increase of ESS**
3	1.19
4	1.34
5	2.61
8	4.24
10	6.32

Table 1: Grid size vs. Ratio of IMCMC ESS to MCMC ESS

The data shown in Table 1 clearly show that the ESS can be significantly increased for the first 1000 steps of the algorithm by making use of a information of the marginal

distributions. It is important to note that the variance of the factor increase is high. The reason for this is that when the random start of the MCMC is good then there is little gain to be made. However, when the random start is particularly bad it can take a very long time to reach the high probability density region hence causing a high auto-correlation and a small ESS.

5 Discussion

5.1 Comparison between IMCMC and TSMC

As shown in the previous section IMCMC offers some improvements over standard MCMC. An interesting relationship is the strong relation between grid size and the mean factor of ESS increase. This can be explained by the fact that when the grid size grows so does the burn in period and as a result the prior information we have from the marginals becomes more valuable.

A question that we could naturally ask is how do SMC and MCMC compare when calculating the partition function. The partition function is useful as an estimate of the quality of the performance of each inference method.

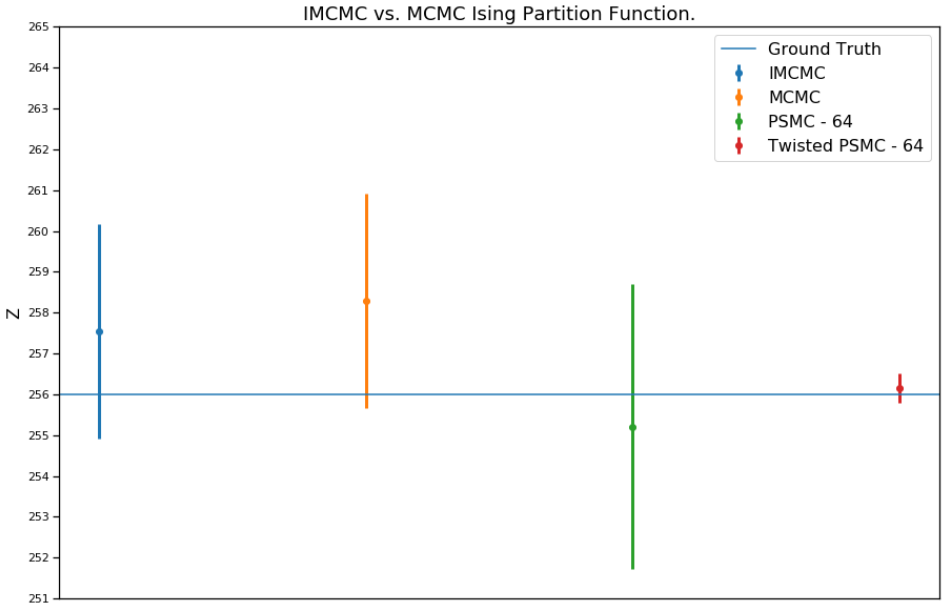


Figure 5: IMCMC and MCMC runs for 10000 steps compared with PSMC and TPSMC with 64 particles tested on a 16 by 16 Ising grid.

As shown in Fig. 5 1000 steps MCMC methods are close to the true distribution but not quite as good as the SMC methods. The MCMC methods improve as we allow the chain to run longer and the SMC methods improve as more particles are used. While that may be true, this comes at the expense of extra complexity.

5.2 Future Work

One possible modification that should improve the convergence of the algorithm would be to modify the Metropolis-Hastings acceptance rule. In both MCMC and IMCMC the standard $q \sim U[0, 1]$ uniform probability acceptance rule is used when a change of a spin in the Ising lattice increases the system's total energy. However, we could in principle use the marginals such that spins in e.g. the (-1) state that flip to (+1) and increase the total energy of the system in its current state but have a high marginal probability of being (+1) are passed with a higher probability. This would provide a look-ahead effect in the chain given that it would not have to wait until the neighbors of that particular node changed to (+1) to encourage the state of said node to also flip. In principle, this would increase the acceptance rate and allow the chain to mix quicker, further reducing the burn-in period and improving convergence.

Another idea would be to extend this technique to PMCMC and create a PIMCMC. As we saw in the PMC framework, using particle methods leads to better estimates of the partition function. Using a Particle based IMCMC should give us the best of both worlds, quick convergence due to a good start, better accuracy and reduced variance.

6 Conclusion

The goal of the project was to extend the link between Monte Carlo methods and deterministic inference algorithms. After replicating the results of recent publications we attempted to formulate a new method. The idea of using the marginals already calculated in the TSMC approach as twisting function was used as motivation to use these same marginals as a MAP estimate of the high probability region of the target distribution. That estimate was then used to initialize the state of the MCMC algorithm yielding an algorithm we called IMCMC.

The proposed algorithm was tested on a simple 2D Ising model and it was shown that it provides better estimates of the partition function than the standard MCMC Metropolis-Hastings algorithm. Additionally, a good starting location reduces the burn-in time and reduces the auto-correlation of the initial hundreds of samples. This leads to a higher effective sample size and hence explains the observation of better accuracy of IMCMC versus MCMC in Fig. 5.

7 Appendix: A note on Software

I was able to access the code base published by [Lindsten et al. (2018)] but there were problems with the code. In fact, the code could not run because a class was missing. It was the class containing the re-sampling methods. I had to read the code then find out what was missing and fix it to get it to work but this took a lot of time away from the rest of the project. The SMC methods are written in MATLAB.

As far as the IMCMC software is concerned, I wrote the MCMC and IMCMC implementations in a IPython notebook. The LBP and Naive Mean Field methods were adapted from my earlier assignments for the course. I also used the Junction Tree algorithm as method for getting the exact solutions to compare to my approximate solutions.

Link to Github repository: `https://github.com/AntonValk/PGM`

8 References

[1] M. I. Jordan. Graphical models.Statistical Science, 19(1):140–155, 2004.

[2] Fredrik Lindsten, Jouni Helske, Matti Vihola. Graphical model inference: Sequential Monte Carlo meets deterministic approximations. In Advances in Neural Information Processing Systems (NIPS) 31. 2018.

[3] Pearl, J. (1987). Evidential reasoning using stochastic simulation. Artificial Intelligence, 32, 245–257.

[4] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. 2000. Rao-blackwellised particle filtering for dynamic Bayesian networks. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (UAI'00), Craig Boutilier and Moisés Goldszmidt (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 176-183.

[5] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms," IEEE Transactions on Information Theory, vol. 51, no. 7, pp. 2282–2312, 2005.

[6] D. Koller and N. Friedman, Probabilistic graphical models principles and techniques. Cambridge, MA: MIT Press, 2012.

[7] C. A. Naesseth, F. Lindsten, and T. B. Schön. Sequential Monte Carlo methods for graphical models. In Advances in Neural Information Processing Systems (NIPS) 27, pages 1862–1870. 2014.

[8] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. Journal of the Royal Statistical Society: Series B, 68(3):411–436, 2006.