11-26-2018

# Parallel Adaptive Collapsed Gibbs Sampling

Craig Nathan Kelly

# PARALLEL ADAPTIVE COLLAPSED GIBBS SAMPLING

by

Craig N. Kelly

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

December 2018

# Abstract

Rao-Blackwellisation is a technique that provably improves the performance of Gibbs sampling by summing-out variables from the PGM. However, collapsing variables is computationally expensive, since it changes the PGM structure introducing factors whose size is dependent upon the Markov blanket of the variable. Therefore, collapsing out several variables jointly is typically intractable in arbitrary PGM structures. This thesis proposes an adaptive approach for Rao-Blackwellisation, where additional parallel Markov chains are defined over different collapsed PGM structures. The collapsed variables are chosen based on their convergence diagnostics. Adding chains requires re-burn-in the chain, thus wasting samples. To address this, new chains are initialized from a mean field approximation for the distribution, that improves over time, thus reducing the burn-in period. The experiments on several UAI benchmarks shows that this approach is more accurate than state-of-the-art inference systems such as Merlin which have previously won the UAI inference challenge.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Probabilistic graphical models (PGM's) (Koller and Friedman 2009) are routinely used in several practical problems such as computer vision (Scharstein and Szeliski 2002), computational biology (Fishelson and Geiger 2004), medical diagnosis (Shwe et al. 1991), natural language processing (McCallum and Wellner 2004), etc. A core problem in PGM's is probabilistic inference, which is required both for learning graphical models as well as for prediction. However, exact probabilistic inference is typically intractable for most PGM structures. Therefore, approximate inference methods such as sampling (Liu 2001) or belief propagation (Yedidia, Freeman, and Weiss 2001) are almost always used for practical problems. Gibbs sampling (Geman and Geman 1984) is arguably one of the most popular MCMC sampling-based approaches for approximate inference in PGM's.

However, despite its widespread use, Gibbs sampling is well-known to have difficulties when the distribution has highly-correlated variables (Liu 2001, 130-131), since the sampler tends to get struck in local modes of the distribution. Rao-Blackwellisation (27-28) is a strategy that significantly improves the convergence of of Gibbs sampling on hard-to-sample multimodal distributions with correlated variables. However, though Rao-Blackwellised Gibbs sampling is provably better than ordinary Gibbs sampling (146-151), performing Rao-Blackwellisation effectively and in a scalable manner is a challenging problem. Specifically, Rao-Blackwellisation (or collapsing) involves summing-out variables from the PGM and implicitly changes the underlying structure of the PGM. That is, collapsing a variable creates a factor across all variables in its Markov blanket. Therefore, it is intractable to collapse variables arbitrarily from the PGM. At the same time, collapsing certain variables in the distribution could be more beneficial w.r.t convergence of the sampler, as compared to collapsing other variables. Previous approaches

(Hamze and de Freitas 2004; Paskin 2003; Bidyuk and Dechter 2007) have utilized the structure of the PGM to perform Rao-Blackwellisation tractably. For example, Hamze and Defreitas (Hamze and de Freitas 2004) partition checkerboard Markov Random Fields (MRF's) into disjoint trees, where they sample one tree and conditioned on this, estimate the joint distribution tractably over the other using belief propagation. Similarly, Bidyuk and Dechter (Bidyuk and Dechter 2007) sample a cutset of the PGM such that the induced width on the remaining variables is bounded by a constant. However, though these approaches leverage structural properties of the PGM, they do not exploit sampler history to determine the optimal variables to collapse. At the same time, adaptive sampling approaches (Andrieu and Thoms 2008) have been successful in improving convergence in MCMC-based samplers by exploiting sampler history. As a result, this thesis proposes a parallelized, adaptive sampler for Rao-Blackwellised Gibbs sampling that collapses variables in parallel based on their convergence properties.

The main idea is quite straightforward. Over time, the sampler adapts by collapsing variables whose marginal probabilities are slow to converge as compared to other variables. Specifically, the algorithm alternates between two steps. In the first step, the optimal variables are collapsed based on their convergence statistics, given tractability constraints. In the second step, new parallel Gibbs samplers are added with the selected variables collapsed out. The convergence statistics are re-computed and the process repeats. The final sampler is a mixture of parallel Markov chains where each Markov chain is constructed around a different collapsed PGM structure, but with all marginals converging to the same invariant distribution. A key issue with parallel sampling is that every time a new sampler is added, the new chain must be initialized. Typically, Gibbs samplers initialize parallel chains by sampling from a uniform distribution over its state space. However, in such a case, the benefit of adding new collapsed samplers is offset by time spent in the *burn-in* period of the new chain before useful samples can be generated from the collapsed sampler. To address this, the state of the sampler is initialized based on a mean field approximation of the distribution. The parameters of this approximation are improved progressively resulting in better initialization points of the sampler and smaller mixing times.

To evaluate this approach, an open source implementation was created. It was used to evaluate the approach against UAI benchmarks on marginal inference tasks. A comparison with Merlin (Marinescu 2016), a PGM inference system, clearly shows that this proposed approach is more accurate than state-of-the-art marginal inference solvers. The main contributions of this thesis are this new algorithm, an open source implementation of the algorithm, and experimental verification of the advantages of this new approach.

# Chapter 2

# Background

## Discrete Graphical Models

Probabilistic Graphical models (PGM's) (Pearl 1988; Darwiche 2009; Koller and Friedman 2009) build upon graph theory with probabilistic reasoning. Directed graphical models are known as Bayesian networks (Pearl 1988). If the network is undirected, it is known as a Markov network (Chellappa and Jain 1993). The rest of this thesis focuses on Markov networks. With regards to inference, directed and undirected networks are equivalent (Darwiche 2009; Koller and Friedman 2009), and similar algorithms are used in both networks.

Although probabilistic graphical models can be used with both continuous and discrete random variables, this research has focused on discrete graphical models. Specifically, this means that individual variables are discrete random variables and the joint distribution modeled by the entire network is discrete.

A discrete Markov network, denoted by $\mathcal{M}$ is a pair $\langle \mathbf{X}, \Phi \rangle$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of discrete variables and $\Phi = \{\phi_1, \ldots, \phi_m\}$ is a set of positive real-valued functions (or potentials). Each function is defined over one or more variables and the scope of a function, $S(\phi)$, is the union of all the variables occurring in $\phi$. These factors correspond to the cliques in an undirected graph (Jordan and Bach 1998).

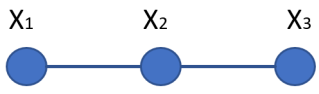



Figure 2.1: Sample 3 node Markov network

Consider the example in Fig. 2.1. There are three variables $\mathbf{X} = \{X_1, X_2, X_3\}$. In general,one would expect these factors to be $\Phi = \{\phi_1, \phi_2\}$ where $\phi_1 : (X_1, X_2) \to \mathbb{R}$ and $\phi_2 : (X_2, X_3) \to \mathbb{R}$. Note that given the factors the structure of the graph could be inferred, *or* given the specified structure the graph the factors one should expect could be listed.

The normalized product of all the factors in the model $\mathcal{M}$ represents a joint probability distribution called the Gibbs distribution. It is given by the following equation:

$$P(\overline{\mathbf{x}}) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi(\overline{\mathbf{x}}_{S(\phi)}) \tag{2.1}$$

where $\overline{\mathbf{x}}$ is an assignment of values to all variables in $\mathbf{X}$, $\overline{\mathbf{x}}_{S(\phi)}$ is the projection of $\overline{\mathbf{x}}$ on the scope $S(\phi)$ of $\phi$, and $Z$ is a normalization constant called the partition function. The thesis will often abuse notation and use $\phi(\overline{\mathbf{x}})$ to mean $\phi(\overline{\mathbf{x}}_{S(\phi)})$.

# Inference With Probabilistic Graphical Models

One of the main reasons for constructing a PGM is to perform inference over a model, possibly in the presence of evidence. Among the possible inference tasks, the most studied (and arguably the most useful) are computing the partition function $Z$ (and thus the probability of evidence), computing the most likely values of the variables in the model given evidence, and computing the marginal distributions of individual variables given evidence. (Koller and Friedman 2009)

As a result, there are known exact algorithms to perform these kinds of inference. A naive approach for any of the three techniques would involve a summation over all unknown terms in a particular query. Restricting the problem to binary discrete variables (although in practice the cardinality of discrete variables is frequently greater than two), this naive approach would need to loop over every possible instantiation of the unknown variables. Assume $\mathbf{X} = (X_1, X_2, \cdots)$ are the variables in the model and $\mathbf{Q} = (X_1, X_2, \cdots)$ are the query variables whose values are known (noting the $\mathbf{Q} \subset \mathbf{X}$). Then this naive approach would run in $2^{|X|-|Q|}$ time.

However, one of the great strengths of a structural model is that the specified structure may

be exploited for inference. There are algorithms that are somewhat better than above for certain classes and sizes of PGM's. For instance, variable elimination, originally introduced in Zhang and Poole (1994) takes advantage of the graph structure and a clever ordering to reduce the number of summations required. Variable elimination is still exponential, but in the width of the ordering of variables instead of the total numbers of variables. In practice, this width can be much lower than the graph size (Koller and Friedman 2009). However, it is still exponential. In addition, finding the optimal ordering for this algorithm is NP-hard (Arnborg, Corneil, and Proskurowski 1987). In fact, inference of this kind is NP-hard in general (Koller and Friedman 2009).

As with other fields plagued by NP-hard problems, there are also approximation techniques available whose accuracy can be arbitrarily increased with additional computation (Williamson and Shmoys 2011). One example is the family of algorithms known as "message passing" or "belief propagation". When originally introduced it was conceived as an exact algorithm on trees (Pearl 1982). In the case of general graphs, it is an approximate algorithm (Pearl 1988). In the general case, a variant known as "loopy belief propagation" (Ihler, Fisher, and Willsky 2005) is often a very competitive technique for approximate inference.

Unfortunately, many of these techniques still do not scale well in the general case as the number of variables increases. As a result, Monte Carlo sampling techniques have been used to perform approximate inference (Liu 2001). In particular, Markov Chain Monte Carlo (MCMC) techniques have been extremely useful. These techniques are based on the construction of a Markov chain designed so that, in the limit, sampling from the chain's equilibrium distribution is equivalent to sampling from the actual target distribution.

As as example, the extremely popular Stan probabilistic programming environment (Carpenter et al. 2017) uses both Hamiltonian Monte Carlo (HMC) (Neal 1993) and an extended version known as the "No U-Turn Sampler" (NUTS) (Hoffman and Gelman 2014). While extremely powerful, Hamiltonian techniques make use of the gradient for the distribution under consideration. Stan's remarkable performance is based on tree-based models (not general graphs) of continuous variables and automatic differentiation (Carpenter et al. 2017). To sample from a

joint distribution of discrete variables, a different technique known as Gibbs sampling is more useful.

## Gibbs Sampling

Gibbs sampling is one of the most widely used MCMC algorithms to date. Here, when sampling from an $n$-dimensional state space, each dimension is sampled one at a time from the conditional distribution for that dimension as described below.

Given a PGM $\mathcal{M} = \langle \mathbf{X}, \Phi \rangle$, Gibbs sampling begins by initializing all variables randomly, denoted by $\overline{\mathbf{x}}^{(0)}$. Then, at each iteration $j$, it randomly chooses a variable $X_i \in \mathbf{X}$ and samples a value $\overline{x}_i$ for it from the conditional distribution $P(X_i|\overline{\mathbf{x}}_{-i}^{(j-1)})$, where $\overline{\mathbf{x}}_{-i}^{(j-1)}$ denotes the projection of $\overline{\mathbf{x}}^{(j-1)}$ on all variables in the PGM other than $X_i$. The new sample is $\overline{\mathbf{x}}^{(j)} = (\overline{x}_i, \overline{\mathbf{x}}_{-i}^{(j-1)})$. The computation of the conditional distribution can be simplified by observing that in a PGM, a variable $X_i$ is conditionally independent of all other variables given its neighbors (or its *Markov blanket*) denoted by $MB(X_i)$. Formally, $P(X_i|\overline{\mathbf{x}}_{-i}) = P(X_i|\overline{\mathbf{x}}_{MB(X_i)})$. Thus, generating each sample is efficient in Gibbs sampling and therefore it is the arguably the method-of-choice for MCMC based inference in PGM's.

The Gibbs sampling procedure just described is called random-scan Gibbs sampling in literature. Another variation is systematic-scan Gibbs sampling in which samples are drawn along a particular ordering of variables. It is known that random-scan Gibbs sampling is statistically more efficient than systematic-scan Gibbs sampling (Liu 2001). The marginal probabilities of the variables can be computed using the Gibbs samples as,

$$\widehat{P}_T(\overline{x}_i) = \frac{1}{T} \sum_{t=1}^{T} P(\overline{x}_i|\overline{\mathbf{x}}_{-i}^{(t)}) \tag{2.2}$$

Collapsing is a technique for improving the accuracy of Gibbs sampling. Collapsing operates by eliminating or marginalizing out a subset of variables, say $\mathbf{C}$, from the PGM $\mathcal{M}$ yielding a collapsed PGM, $\mathcal{M}_{\mathbf{X}\setminus\mathbf{C}}$. Gibbs sampling is then performed on this smaller PGM and this

improves its accuracy (because only a sub-space is sampled). Collapsing a variable creates a clique in the PGM over all its neighbors.

## Superiority of Collapsed Gibbs Sampling

In the general case, it can be shown that collapsing has advantages over other techniques for improving Gibbs sampling. SpecConsider another variation of Gibbs sampling known as blocking. A blocked Gibbs sampler groups multiple variables together and samples from them simultaneously as if they were single variable (Liu, Wong, and Kong 1994).

Liu presents a theorem comparing the convergence rate of Gibbs samplers with these kinds of techniques (Liu 2001, 147-148,275-276):

Given a target distribution $\pi$, define $L^2(h)$ as the set of all functions $h()$ where $h$ is a function that is square integrable with respect to $\pi$ and therefore has finite variance. Define this set to be a Hilbert space via the inner product $\langle h, g\rangle = E_\pi[h(x)g(x)]$. As a result, $\|h\|$ is the variance of $h$ with respect to our target distribution $\pi$. Define a "forward operator" $\mathbf{F}h$ on $L^2(h)$ as:

$$\mathbf{F}h(\overline{\mathbf{x}}) = E[h(\overline{\mathbf{x}}^{(1)})|\overline{\mathbf{x}}^{(0)} = x] \tag{2.3}$$

Define the norm of $\mathbf{F}$ to be $\|\mathbf{F}\| = sup_h\langle \mathbf{F}h(x)\rangle$ over all functions where $E(h^2) = 1$.

If the state space of the target distribution is finite and the chain of samples is reversible, then $\mathbf{F}$ is compact and self-adjoint. If $\mathbf{F}$ is compact and self-adjoint, then the eigenvalue with the second largest absolute value characterizes the convergence rate for this case.

Further, define the subspace $L_0^2$ of $L^2(x)$ were $h$ is restricted to functions where $E[h(x)] = 0$. In this subspace then the largest eigenvalue by absolute value of $F$ is equal to the second largest absolute eigenvalue. Thus for 2 forward operators $F_1$ and $F_2$, $\langle \mathbf{F}_1\rangle \leq \langle \mathbf{F}_2\rangle$ implies that in the limit, the average convergence rate for samplers corresponding to $F_1$ will converge faster than samplers corresponding to $F_2$

Given this ordering by convergence, we can define an $F$ for various Gibbs samplers,

specifically random-scan, blocked, and collapsed. For random scans:

$$\mathbf{F}_R = \sum_{i=1}^{d} \alpha_i \text{var}[E\{h(\overline{\mathbf{x}})|\overline{\mathbf{x}}_{(-i)}\}] \tag{2.4}$$

Where $d$ is the number of variables (i.e. dimensions) and $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d)$ represents the sampling distribution over the variables. Now assume that a blocking sampler will block variables $d_1$ and $d_2$. Similarly, a collapsed sampler will collapse the same two variables. (Noting that the results are the same under any initial ordering of the variables and so apply to any two variables selected.) Then for blocking samplers:

$$\mathbf{F}_B = (\alpha_1 + \alpha_2)\text{var}[E\{h(\overline{\mathbf{x}})|\overline{\mathbf{x}}_{(-1,-2)}\}] + \sum_{i=3}^{d} \alpha_i \text{var}[E\{h(\overline{\mathbf{x}})|\overline{\mathbf{x}}_{(-i)}\}] \tag{2.5}$$

and for collapsed samplers:

$$\mathbf{F}_C = (\alpha_1 + \alpha_2)\text{var}[E\{h(\overline{\mathbf{x}})|\overline{\mathbf{x}}_{(-1,-2)}\}] + \sum_{i=3}^{d} \alpha_i \text{var}[E\{h(\overline{\mathbf{x}})|\overline{\mathbf{x}}_{(-1,-i)}\}] \tag{2.6}$$

Thus:

$$\|\mathbf{F}_C\| \leq \|\mathbf{F}_B\| \leq \|\mathbf{F}_R\| \tag{2.7}$$

As a result, in the general case collapsing Gibbs samplers converge faster than blocking samplers, which in turn converge faster than random-scan samplers.

# Chapter 3

# Related Work

Liu (2001) showed that Rao-Blackwellised Gibbs is provably better than random scan Gibbs sampling, and therefore samplers should always try to collapse variables when possible. Thus, several previous approaches have been proposed for Rao-Blackwellised Gibbs sampling in discrete PGM's. Most of these approaches are non-adaptive samplers that only exploit the structure of the Markov network to collapse a subset of variables tractably.

Specifically, Paskin (Paskin 2003) proposed sample propagation, an efficient Rao-Blackwellisation technique for PGM's. Sample propagation samples some variables and performs exact inference over the others using a junction tree by passing messages in an efficient manner. The passing strategy does not require running the full junction tree (over the collapsed variables) for each sample.

Hamze and Defreitas (Hamze and de Freitas 2004) build a Rao-Blackwellised sampler over Ising models. This sampler partitions the PGM into two parts. One is sampled; the exact marginal is computed over the remaining tree-structured induced graph conditioned on the first. This technique is not just blocking: it "is also a 2-block Rao-Blackwellised scheme" (248).

Bidyuk and Dechter (Bidyuk and Dechter 2007) proposed cutset sampling. A cutset is a subset of the variables in the model selected for sampling. Once selected, a Gibbs sampler is used to sample from the cutset. By conditioning on the cutset, the induced width of the remaining network is bounded. The marginal distributions of the variables not in the cutset can be solved exactly.

Adaptive approaches include the splash Gibbs sampler by Gonzalez et al. (Gonzalez et al. 2011), where they use a likelihood estimator to compute the optimal variables to sample jointly via blocking, also known as splashes. They run parallel chains to sample from different

splashes where the splashes are constructed to maintain ergodicity of the sampler. Although not collapsing, their parallel blocking Gibbs sampler performs well on models with highly correlated variables. This is in contrast to standard Gibbs samplers like the Random-Scan Gibbs Sampler which have known problems with highly correlated variables (Liu 2001).

Finally, Venugopal and Gogate (Venugopal and Gogate 2013) proposed an approach to learn correlations, and used these to collapse the sampler efficiently. However, they considered a single collapsed PGM structure as compared to this approach, where multiple collapsed structures are sampled in parallel.

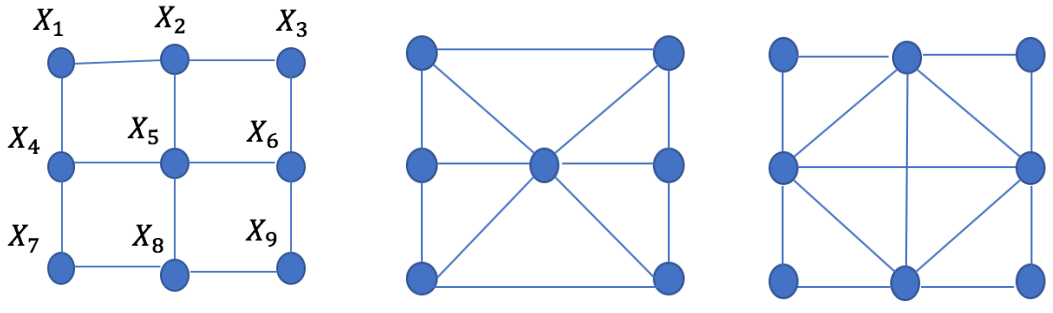
# Chapter 4

# Adaptive Rao-Blackwellisation

## Motivation and Overview

As a motivation example, consider a pairwise Markov network shown in Fig. 4.1 (a). Suppose, the variables $X_2$, $X_5$ and $X_8$ are correlated with ($X_1$, $X_3$), ($X_4$, $X_6$) and ($X_7$, $X_9$) respectively. In such a case, collapsing $X_2$, $X_5$ and $X_8$ is likely to improve the mixing time of the sampler. However, collapsing all three variables leads to factor with six variables. Specifically, after collapsing, there is a clique over the variables $X_1$, $X_3$, $X_4$, $X_6$, $X_7$ and $X_9$. Further, suppose a constraint is added that only factors of four variables or less can be constructed, then, clearly, $X_2$, $X_5$ and $X_8$ cannot be collapsed in the same chain. An alternate strategy is to then spawn two parallel Gibbs samplers corresponding to the Markov networks shown in Fig. 4.1 (b) and (c). Specifically, in Fig. 4.1 (b), $X_2$ and $X_8$ are collapsed, and in Fig. 4.1 (c) the variable $X_5$ is collapsed. The final sampler is now a mixture of the two collapsed samplers.

Collapsing samplers can provide better sampling in even simpler circumstances. Consider




(a) (b) (c)

Figure 4.1: (a) Original PGM (b) $X_2$ and $X_8$ collapsed (c) $X_5$ collapsed

Table 4.1: The 2 factors in a deterministic example.

| **A** | **B** | $\phi_1$ | **B** | **C** | $\phi_2$ |
|---|---|---|---|---|---|
| 0 | 0 | 0.5 | 0 | 0 | 0.25 |
| 0 | 1 | 0.0 | 0 | 1 | 0.25 |
| 1 | 0 | 0.0 | 1 | 0 | 0.25 |
| 1 | 1 | 0.5 | 1 | 1 | 0.25 |

Table 4.2: The joint distribution in a deterministic example.

| **A** | **B** | **C** | $P(A, B, C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0.25 |
| 0 | 0 | 1 | 0.25 |
| 0 | 1 | 0 | 0.00 |
| 0 | 1 | 1 | 0.00 |
| 1 | 0 | 0 | 0.00 |
| 1 | 0 | 1 | 0.00 |
| 1 | 1 | 0 | 0.25 |
| 1 | 1 | 1 | 0.25 |

a deterministic example Markov graphical model with three variables $A, B, C$ and two factors $\phi_1$ and $\phi_2$, as seen in table 4.1. Areas of zero probability violate one of the assumptions of Gibbs sampling, namely that all areas of the sample space are reachable. In addition to $\phi_1$, the joint distribution, shown in table 4.2, has zero entries. While even naive implementations of a Gibbs sampler will insure that zero value components of the state space are sampled with a non-zero value (typically some small number $\epsilon \leq 10^{-8}$), in practice these samplers have trouble converging or converge to a joint distribution with more error than an equivalent collapsing sampler.

In this model, $P(A = 0) = P(B = 0) = P(C = 0) = 0.5$, so the marginal estimate for each variable should be $(0.5, 0.5)$. However, simple Gibbs sampling methods have trouble finding these marginals. As one example, a naive Gibbs sampling failed to infer the correct marginal distributions as can be seen in table 4.3. However, a collapsing sampler quickly finds the correct answer.

Table 4.3: The marginal distributions in a deterministic example. A collapsed Gibbs sampler found the **Correct** probability. Incorrect inferences are *italicized*.

| **Variable** | **Actual Value** | **Naive Gibbs Inference** | **Collapsed Gibbs Inference** |
|---|---|---|---|
| $P(A) = 0$ | 0.5000 | *0.3881* | 0.5000 |
| $P(B) = 0$ | 0.5000 | *0.3884* | 0.5000 |
| $P(C) = 0$ | 0.5000 | 0.5000 | 0.5000 |

Formally, let $\mathcal{M}$ be a Markov network, with variables $\mathbf{X} = X_i \dots X_n$ and factors $\phi_1 \dots \phi_m$. Our task is to partition the variables into two subsets, $\mathbf{X}_s$ and $\mathbf{X}_c$, where $\mathbf{X}_s$ refers to all the variables that are sampled, and $\mathbf{X}_c$ refers to all the variables that are collapsed, such that the mixing time of the sampler is minimized. Specifically,

$$\min_{\mathbf{X}' \subseteq \mathbf{X}} t_{mix}(P_{-\mathbf{X}'}, \epsilon) \tag{4.1}$$

where $t_{mix}(P_{-\mathbf{X}'}, \epsilon)$ is the mixing time of the Gibbs sampler where $\mathbf{X}'$ is collapsed out of $\mathcal{M}$. In general, the mixing time can be defined as the minimum number of time steps before the

total variational distance between the true and approximate distribution is less than a constant. Specifically,

$$t_{mix}(P_{-\mathbf{X}'}, \epsilon) = \min \left\{ t : \max_{\mu} ||P^{(t)}_{-\mathbf{X}'}\mu - P_{\mathcal{M}}||_{TV} \leq \epsilon \right\} \tag{4.2}$$

where $\epsilon$ is a constant, $P^{(t)}$ is the transition matrix after $t$ time steps of the sampler defined on the PGM obtained by collapsing variables $\mathbf{X}'$ from $\mathcal{M}$, $||P^{(t)}_{-\mathbf{X}'}\mu - P_{\mathcal{M}}||_{TV}$ is the total variational distance between the approximate distribution estimated by the sampler after $t$ time steps, given by $P^{(t)}_{-\mathbf{X}'}\mu$ and the true stationary distribution $P_{\mathcal{M}}$. Thus, the task is to select the optimal subset of variables to minimize the mixing time of the sampler. However, it is notoriously hard to analytically derive the mixing time for arbitrary PGM's. Therefore, a standard approach used to analyze whether a sampler has mixed or not is to use convergence diagnostics derived from the drawn samples. In principle, any convergence diagnostic can be used for $f$.

This thesis adopted a popular approach known as the Gelman and Rubin (GR) diagnostic (Gelman and Rubin 1992). Specifically, given multiple chains from different starting points, for a parameter $\theta$ that is estimated by the sampler, estimate the between-chain as well as the within-chain variances, and combine the variances together. For mixed chains, the within as well as in-between variances for $\theta$ would be small. In this case, the algorithm uses the diagnostic to determine whether the marginal estimates for each of the variables in the sampler has mixed. However, since the class of models under study use discrete variables, it turns out that the standard GR diagnostic does not work effectively with limited sample-sizes (Cowles and Carlin 1996; Deonovic and Smith 2017). Since the aim is to get an accurate estimate of the convergence diagnostic as quickly as possible, the system uses variant of the GR scheme based on the methods proposed by Deonovic and Smith (Deonovic and Smith 2017). Specifically, for each variable, compute convergence within and between Markov chains using the Hellinger distance, which is a popular symmetric distance measure for discrete distributions. Then compute the PSRF (potential scale reduction factor) as in the GR diagnostic. Larger values will mean that the chain has not converged. Therefore, the goal is a subset of variables to collapse such that the GR diagnostics

computed after time $t$ over the marginal probability estimates for the remaining variables is minimized. Specifically,

$$\min_{\mathbf{X}' \subseteq \mathbf{X}} \sum_{X \in \mathbf{X} \setminus \mathbf{X}'} GR(P_{-\mathbf{X}'}(X)) \tag{4.3}$$

Solving the above optimization problem is clearly computationally hard. Specifically, one must enumerate all possible subsets of $\mathbf{X}$, and corresponding to each of the subsets, collapse the PGM and compute the GR statistics for all un-collapsed variables after running the sampler for $t$ time steps. Instead, the system uses a co-ordinate descent approach to optimize Eq. (4.3), where processing alternates between the following steps. The optimal variables to collapse are chosen using the modified, fixed GR diagnostics described above. Parallel samplers are added for the collapsed Markov network. Then the GR diagnostics are computed over the un-collapsed variables.

However, note that the unconstrained problem specified in Eqn. (4.3) can lead to a trivial solution where all the variables are selected for collapsing. As a result, there is a tractability constraint where the minimum *width* of the collapsed variables should be bounded by a constant. Specifically, given $\mathbf{X}' \subseteq \mathbf{X}$, the width is defined over an ordering of $\mathbf{X}'$, $\pi(\mathbf{X}')$ as the maximum clique-size in the induced graph obtained by collapsing variables in the order $\pi(\mathbf{X}')$. The minimum width is then smallest width over all orderings. Therefore, the algorithm has a constraint to Eq. (4.3) that the minimum width of the variables chosen for collapsing must be bounded by a constant. That is, $w(\mathbf{X}') \leq \alpha$. However, adding this constraint implies that we need to compute the minimum width by enumerating all possible orderings of the subset in the worst case. Instead, heuristics are used that yield an upper bound to $w(\mathbf{X}')$, such as the *min-degree* or the *min-fill* heuristic. Note that more accurate branch-and-bound based estimates (Gogate and Dechter 2004) for $w(\mathbf{X}')$ can be computed, however, they are computationally more expensive.

The approach to solve Eq. (4.3) with the tractability constraint proceeds as follows. To start with, assume that every variable in $\mathcal{M}$ converges at the same rate, i.e. the GR diagnostics for each marginal probability in $\mathbf{X}$ are equal. In iteration $t$, conditioned on the GR diagnostics,

partition $\mathbf{X}$ into $\mathbf{X}_s^{(t)}$ and $\mathbf{X}_c^{(t)}$ using a greedy approach. Specifically, select the variable $X$ that has maximum GR value and where $w(X) \leq \alpha$, and add it to $\mathbf{X}_c^{(t)}$. Add $K$ Gibbs samplers to sample from $P_{-\mathbf{X}_c^{(t)}}$, and, using samples from them, update the GR statistics for all variables in $\mathbf{X}_s^{(t)}$. Based on the new GR statistics, recompute $\mathbf{X}_c^{(t+1)}$ and $\mathbf{X}_s^{(t+1)}$.

# Estimating Marginals

Let $\bar{\mathbf{x}}_{ij}^{(t)}$ represent the $j$-th sample generated from the $i$-th sampler spawned in iteration $t$ after the sampler is assumed to have converged. The marginals for $\mathbf{X}_c^{(t)}$ are computed exactly, while the marginals for $\mathbf{X}_s^{(t)}$ are estimated from $\bar{\mathbf{x}}_{ij}^{(t)}$. Note that once a variable is collapsed in one iteration, that variable's exact marginal is known and no longer requires a sampling-based estimator. For variables that cannot be collapsed in any iteration, the marginal probabilities are estimated using a mixture estimator:

$$\hat{P}(X) = \frac{1}{T * K * M} \sum_{t=1}^{T} \sum_{i=1}^{K} \sum_{j=1}^{M} P_{-\mathbf{X}_c^{(t)}}(X | \bar{\mathbf{x}}_{ij}^{(t)} \setminus X) \tag{4.4}$$

where $K$ is the number of parallel chains in each iteration, $M$ is the number of samples collected per chain and $T$ is the number of iterations.

However, note that continuously adapting the transition kernel makes the overall sampler non-ergodic (Gonzalez et al. 2011). In this case, collapsing different variables in each chain changes the structure of the PGM and the underlying transition matrix. Therefore, the system uses *vanishing adaptation* (Gonzalez et al. 2011) to ensure that the transition matrix stabilizes over time. Specifically, let $GR(X)^{(t)}$ be the GR diagnostic for variable $X$ after iteration $t$, this is updated in iteration $t+1$ as $\beta\, GR(X)^{(t+1)} + (1-\beta)\, GR(X)^{(t)}$, and $\beta$ is decreased in each iteration. Thus, as $T \to \infty$, $GR(X)^{(t+1)} \to GR(X)^{(t)}$. this ensures that the greedy selection process selects the same set of variables to collapse as $T \to \infty$.

**Proposition 1.** *As $T \to \infty$, for each variable $X$ in $\mathcal{M}$, the estimated marginal $\hat{P}(X) \to P_{\mathcal{M}}(X)$, where $P_{\mathcal{M}}(X)$ is the true marginal distribution for $X$.*

**Algorithm 1:** Adaptive RB

**Input:** $\mathcal{M}$ ($\mathbf{X}$,$\Phi$), $\alpha$
**Output:** Marginal probabilities for each variable in the PGM
// Initialize GR stats
**for** *each* $X_i$ *in* $\mathbf{X}$ **do**
  $GR^{(1)}(X_i) = C$
Initialize burn-in $B$
Initialize adaptation parameter $\beta$
**for** $t = 1$ *to* $T$ **do**
  // Compute the optimal collapsed variables
  $\mathbf{X}_c^{(t)}$, $\mathbf{X}_s^{(t)}$ = Greedy-select based on $GR^{(t)}$ where $w(\mathbf{X}_c^{(t)}) \leq \alpha$
  Collapse $\mathcal{M}$ to represent $P_{-\mathbf{X}_c^{(t)}}(\mathbf{X}_s^{(t)})$
  Initialize $K$ parallel Gibbs samplers by sampling from the current marginal estimates for $\hat{P}(X_i) \ldots \hat{P}(X_n)$
  // Estimate the marginals
  **for** $i = 1$ *to* $M$ **do**
    **if** $i \geq B$ **then**
      **for** $X \in \mathbf{X}_s^{(t)}$ **do**
        Update $\hat{P}(X)$
  **for** $X \in \mathbf{X}_c^{(t)}$ **do**
    Compute exact marginals and store in $\hat{P}(X)$
  **for** $X \in \mathbf{X}_s^{(t)}$ **do**
    $GR^{(t)}(X) = \beta\, GR^{(t)}(X) + (1-\beta) GR^{(t-1)}(X)$
  Reduce $\beta$
  Reduce $B$
Return $\hat{P}(X_i) \ldots \hat{P}(X_n)$

*Proof.* Let $P_{\mathcal{M}}$ be the distribution represented by $\mathcal{M}$. In iteration $t$, the Gibbs sampler is constructed on the PGM with $\mathbf{X}_c^{(t)}$ collapsed. That is, samples are drawn from $P(\mathbf{X}_s^{(t)} = \sum_{\bar{\mathbf{x}} \in \mathbf{X}_c^{(t)}} P(\mathbf{X}_s^{(t)}, \bar{\mathbf{x}})$. There are two possible cases. Let $X$ be a variable that is collapsed in some iteration. This means, the exact marginal is computed for $X$. Therefore, $\hat{P}(X) = P_{\mathcal{M}}(X)$. Suppose $X$ is not collapsed in any iteration. Then, the estimate for $X$ is derived by the samples from $P(\mathbf{X}_s^{(1)}, \ldots P(\mathbf{X}_s^{(T)}$. Since each collapsed distribution leaves the marginal for $X$ invariant, and as $T \to \infty$, $\mathbf{X}_s^{(t-1)} = \mathbf{X}_s^{(t)}$ (vanishing adaptation), $\hat{P}(X) \to P_{\mathcal{M}}(X)$. □

## Initializing the Markov Chains

In each iteration, $K$ new Gibbs samplers are added based on a collapsed distribution. The common approach to initialize the samplers is to draw a state uniformly at random from the state-space. However, such an initialization would require a full burn-in before samples can used for estimation. Thus, as more samplers are added, more and more samples are wasted as part of a burn-in period, which affects scalability of the sampler. To address this problem, the system uses an importance distribution to initialize the sampler and learn the parameters of the importance distribution based on results obtained from the previous samplers. Specifically, it assumes that the importance function is fully factorized over the estimated marginal probabilities, i.e., a *mean-field* approximation of the joint distribution.

$$P_{\mathcal{M}}(\mathbf{X}) \approx Q_{\mathcal{M}}(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} \hat{P}(X_i) \tag{4.5}$$

In each iteration, the Gibbs sampler is initialized by sampling assignments from $\prod_{X \in \mathbf{X}} \hat{P}(X)$. As the marginal estimates become more accurate, the mean-field approximation for the joint distribution improves. Specifically, the KL-distance between the original distribution and the mean field approximation is given by,

$$KL(Q_{\mathcal{M}}||P_{\mathcal{M}}) = \sum_{\mathbf{X}} Q_{\mathcal{M}}(\mathbf{X}) \log \frac{P_{\mathcal{M}}(\mathbf{X})}{Q_{\mathcal{M}}(\mathbf{X})} \tag{4.6}$$

Using variational inference (Jordan et al. 1999), the optimal parameters for $Q_{\mathcal{M}}$ can be obtained by maximizing the Evidence-lower bound (ELBO) which is equivalent to minimizing the KL-divergence between the mean-field approximation and the original distribution. Typically, this is done using a co-ordinate descent procedure where the optimal distribution is computed for each variable in the mean-field approximation independent of the distributions over the other variables. Analytically computing the mean field parameters for a variable in the PGM requires the multiplication of all factors that the variable is involved in. Specifically, if a variable has $N$

variables in its Markov blanket, and is involved in $F$ factors and can take $K$ values, computing the distribution analytically has a complexity $O(KFN^K)$. For PGM's where variables have large Markov blankets, this can be an expensive operation. Therefore, this approach is a way to estimate the parameters of the approximation $Q_{\mathcal{M}}$ from the samples.

Formally, let $Q* = \min_{Q \in \mathcal{F}} KL(Q||P_{\mathcal{M}})$, where $\mathcal{F}$ is the set all possible mean-field approximations for $P_{\mathcal{M}}$. It can be shown that

**Proposition 2.** *As $T \to \infty$, $Q_{\mathcal{M}} \to Q*$*

*Proof.* Let $Q*(\mathbf{X}) = q_1(X_1) \dots q_n(X_n)$. The ELBO optimization problem is given by,

$$\min_{q_1 \dots q_n} \mathbb{E}_{Q*(\mathbf{X})}[\log \bar{P}_{\mathcal{M}}(\mathbf{X}) - \log Q*(\mathbf{X})] \tag{4.7}$$

where $\bar{P}_{\mathcal{M}}(\mathbf{X})$ is the un-normalized probability, which is product of all factors, i.e., $\prod_{i=1}^{m} \phi_i(\mathbf{X})$. Using the closed form solution (Jordan et al. 1999) to the above problem,

$$\log q_j(X_j) \propto \mathbb{E}_{q_{-j}}[log \bar{P}_{\mathcal{M}}(X_j)] \tag{4.8}$$

where $\mathbb{E}_{q_{-j}}$ denotes that the expectation is computed while keeping all the distributions except $q_j$ fixed. $\bar{P}_{\mathcal{M}}(x_j)$ is the product of all factors containing variable $x_j$. From Proposition 1, the sampled marginals approach the true marginals as $T \to \infty$. Thus, as $T \to \infty$, $\forall i$, $\hat{P}(X_i) \to P(X_i) \propto q_i(X_i)$. Therefore, $Q_{\mathcal{M}} \to Q*$. □

Thus, after each iteration, the KL divergence between the true distribution and the mean-field approximation decreases. This is used to set better initialization values for the sampler that reduces the burn-in time. Specifically, the burn-in time is reduced in each iteration and thus, fewer samples are needed as more parallel chains are added to the sampler. Algorithm 1 illustrates the complete sampler.

# Chapter 5

# Implementation

## Language Selection

To test the algorithm, a sampler (Kelly 2018) was written in the Go programming language (Griesemer, Pike, and Thompson 2012). Go was chosen because it provides a fast, compiled binary and modern higher-level programming conveniences like garbage collection and closures. However, the most important reason for selecting Go was it's strong support for concurrent programming.

In fact, Go provides an implementation of Hoare's Communicating Sequential Processes (Hoare 1985) in the form of lightweight "goroutines" (or green threads) and bi-directional synchronized communication channels. In addition, the standard library provides synchronization primitives. The Go runtime provides a complete and transparent work-stealing scheduling system that uses all available cores. (Deshpande, Sponsler, and Weiss 2012)

## Program Structure

As is standard for Go programs, the program is divided into packages, one of which provides a command line interface. This organizational structure is not just a matter of engineering and design for this thesis; it also allow others to reuse the code as library in their own applications. Of course, users can also use the same command line interface used to run the experiments validating this work. All packages are listed in Table 5.1 and are described below.

The Go programming language provides both a fast pseudo-random number generator (PRNG) and a cryptographically secure PRNG in the Go standard library (Griesemer, Pike, and

Table 5.1: Packages in Program Implementation. File count includes unit tests. Packages are listed in alphabetical order

| Package | Files | Description |
|---|---|---|
| *buffer* | 2 | Circular buffer designed for chains |
| *cmd* | 4 | Command line interface |
| *model* | 14 | Variables, functions, models, error functions |
| *rand* | 2 | Random number generation based on Mersenne Twister |
| *sampler* | 9 | Gibbs samplers, parallel chains, adaptive strategy |

Thompson 2012). The cryptographically secure PRNG is slow and not appropriate for large-scale sampling applications (by design). However, the fast PRNG is is a custom Linear Congruential Generator that hasn't been studied in depth; although the algorithm is credited to "DP Mitchell and JA Reeds" (Authors 2009), there is no formal citation. An online forum post from a Go team member indicates that it may be the Plan 9 PRNG and possibly the Ken Thompson's implementation (Cox 2011). Out of abundance of caution, it was decided to use a different PRNG.

The Mersenne Twister was specifically designed to provide an efficient PRNG with an extremely long period, which is exactly the design needed for large scale MCMC sampling (Matsumoto and Nishimura 1998). An open source implementation of the MT19937 algorithm in Go was used (Voss 2013). This PRNG is wrapped in a goroutine that fills a Go channel. By setting the buffering limit on the channel, a balance can be struck between parallel random number generation and memory use. In addition, specifying a random seed is much easier because there is just a single seed needed to start the parallel PRNG. However, it is important to note that due to the stochastic nature of thread scheduling, re-using the same seed does not guarantee the exact same results.

The software includes a model package that provides an implementation of random variables, function (factors) of those variables, a model representation using variables and functions, and support for using a model. A reader interface is included that can read models, evidence, and inference solutions in the UAI inference competition format (Gogate 2014). In

addition there is a suite of error (or distance) metrics for evaluating models. Currently there are four metrics: Mean Absolute Error, Maximum Absolute Error, Jensen-Shannon Divergence, and Hellinger Distance. Hellinger Distance is described in chapter 6.

Jensen-Shannon (JS) Divergence (Fuglede and Topsoe 2004) is a generalization of the Kullback–Leibler (KL) Divergence (Kullback and Leibler 1951). The KL Divergence is a popular and powerful measurement of the relative entropy between two distributions, but it is not symmetric. That is, $KL(A||B)$ does not necessarily equal $KL(B||A)$. However, the JS Divergence *is* symmetric: $JS(A, B) = JS(B, A)$ (Fuglede and Topsoe 2004).

All sampling logic is in the package named sample. The core abstraction is the sampler: an object that (optionally) maintains state and can draw a single sample from the complete joint distribution under study (represented by a model). A random-scan Gibbs sampler is provided (called a "simple" Gibbs sampler in the source code). In addition, a sampler using Rao-Blackwellisation is provided, called a "collapsed" Gibbs sampler in the source code. The collapsed sampler defers sampling to an internal simple Gibbs sampler, but also provides the ability to collapse a variable.

Sequences of draws are represented by a chain object. Given a model and a sampler, a chain is able to take an arbitrary number of samples asynchronously. Thus a list or array of chains will all operate in parallel, scheduled by the Go scheduler across all available cores. The chain object also maintains a history of samples sufficient to detect per-variable within-chain error as necessary for the Gelman-Rubin diagnostic as described in chapter 4. The chain module exports a function that, given a list of chains and an error measure, calculates a Gelman-Rubin diagnostic for the chains

The chain module also has a merge function that combines all current chains' marginal estimates. This is used for final output, but it is also key for performing the mean field approximation necessary to skip burn-in of newly created chains. When a new chain is created, the merged output of all chains is used to estimate the new starting position of the chain. If no merged chains were available, then this technique would collapse to selecting a starting position

uniformly from the sample space of the full joint distribution (which is not the same as sampling directly from the joint distribution). This is how pre-burn-in sample positions are selected when initializing the non-adaptive Gibbs samplers.

The final piece of the implementation is the adaptation using the GR diagnostic as described in chapter 4. This is accomplished with a strategy pattern (Gamma et al. 1994). The interface for an adaptation strategy defines a function that accepts a list of chains and the expected number of chains after the adaptation step; this function should return a new list of chains after performing necessary changes. When using running random-scan or collapsed Gibbs sampling (where the collapsed variables are chosen randomly before sampling begins), an "identity" strategy is used that does nothing and leaves the current list of chains unchanged. A convergence sampler strategy creates new chains with collapsed variables chosen via the Adaptive RB algorithm described in chapter 4.

# Execution

All the above functionality is instantiated by a simple command line interface. After startup and initialization, execution begins. After a specified number of iterations ($c$), a convergence score is calculated per variable, select $a$ variables to collapse, and create new chains with those variables collapsed. This adaptive process continues for $\frac{s}{2}$ seconds, where $s$ is the maximum time in seconds for sampling. From that time until $s$ seconds (or until the sampler converges), the chains are not altered and sampling continues normally with no adaptive steps. (This is accomplished by simply ceasing to use the adaptive strategy object.) In the tests performed, the settings used were $b = 2$, $a = 4$, $c = 2000$, and $s = 300$: so each experiment began with 2 uncollapsed samplers and added 4 collapsed samplers after every 2000 iterations. This continued for at most 150 seconds ($s/2$), after samples were drawn for at most an additional 150 seconds.

# Chapter 6

# Experiments

## Data for Evaluation

This approach was evaluated on the UAI 2014 marginal inference competition tasks (Gogate 2014). The true marginal probabilities for these tasks have been provided by the organizers. Several problems were experimented upon and the results are shown here. Specifically, this thesis presents results from sample problems in Alchemy, CSP, Grids, Pedigree and Promedas. As can be seen in Table 6.1, each of these benchmarks have PGM's of different structures.

Table 6.1: UAI Data Sets Used

| Data Set | Models | Median Variable Count | Median Factor Count |
|---|---|---|---|
| Alchemy | 1 | 440 | 860 |
| CSP | 3 | 82 | 462 |
| Grids | 3 | 100 | 300 |
| Pedigree | 3 | 385 | 385 |
| Promedus | 3 | 534 | 534 |

The Alchemy data set specifies statistical relational models. Statistical Relational Learning is in general a subfield of Artificial Intelligence that is concerned with models that model uncertainty with statistical and probabilistic techniques, and take advantage of structure within the model itself (Getoor and Taskar 2007). This has a great deal of overlap with PGM's in general and discrete Markov network in particular.

The CSP data sets specify constraint satisfaction problems in the format of a Markov network. Constraint satisfaction problems are famous and broadly applicable across computer

science (Kumar 1992). For instance, famous problems like Sudoku (Lynce and Ouaknine 2006), the Eight Queens Puzzle (Hoffman, Loessi, and Moore 1969), and Graph Coloring (Bruynooghe 1985) can all be formulated as CSP's. Particularly difficult CSP's can be constructed as PGM's so that approximation and inference can be used to find approximate solutions to the problem; the problem may even be adjusted to include uncertainty in the constraints.

The Grids data sets are two dimensional Ising models. Ising models are from statistical physics and are used to model magnetism. Although exact analytical solutions exist for some cases, it is often (much) easier to find solutions to an Ising model system via Monte Carlo simulation, including Markov Chain techniques. (Gallavotti 2013)

The Pedigree data sets models are used in linkage analysis in biology. Linkage analysis problems are concerned with mapping (or linking) genes onto chromosomes. Given genetic samples from a subject and a sample of the subject's ancestry population (or pedigree), linkage analysis attempts to make inferences about the gene under study, including the most likely chromosomal position. (Allen and Darwiche 2008)

The Promedas data sets are models used for medical diagnosis. The Promedas system is a Medical Expert System that uses diagnoses, doctor findings, and diagnoses-finding connections gathered from internal medicine to construct a probabilistic diagnostic system. In practice, a Bayesian network and a custom version of Belief Propagation (Loop Corrected Belief Propagation) is used to present the most likely diagnosis. (Wemmenhove et al. 2007) [1]

# Setup

The performance of the algorithm is evaluated using the maximum Hellinger distance. Specifically, the Hellinger distance for a set of $\mathbf{X}$ discrete variables is given by,

1. The results table spells the medical system data set "Promedus" because this is the spelling used in the public data repository for the UAI inference competition. However, in the rest of this thesis, the original spelling "Promedas" is used and preferred since that is the spelling used by Wemmenhove et al. (2007), the creators of the system.

$$H(\hat{X}_i, X_i) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{m}(\sqrt{\hat{x_j}} - \sqrt{x_j})^2} \tag{6.1}$$

where $\hat{X}_i = (\hat{x}_1, \cdots, \hat{x}_m)$ is the estimate of the marginal distribution $P(X_i)$ and $X_i = (x_1, \cdots, x_m)$ is the actual, known marginal. (Noting that the outcome is divided by $\sqrt{2}$ so that the error metric is $0 \leq H(\cdot, \cdot) \leq 1$). Then for the $n$ variables in $\mathbf{X}$, our Maximum Hellinger metric $M = \max_{\hat{X}_i, X_i \in \mathbf{X}} H(\hat{X}_i, X_i)$ For ease of comparison, $-log_2(M)$ is used. The negative log of the Maximum Hellinger distance is presented in all our tables and figures. Thus, higher values are better.

The results are compared to Merlin (Marinescu 2016) which implements Iterative Join-Graph Propagation (IJGP) (Mateescu et al. 2010). It should be noted that IJGP is a state-of-the-art system for marginal inference and has in fact won the UAI inference competition in several categories. Merlin's default settings were used for IJGP. In all cases, Merlin converged extremely quickly, and therefore, their results do not show changes over time.

We calculated Maximum Hellinger error from the solution file written by Merlin to use a baseline to compare our approach. The results are in table 6.2. For comparison the Merlin score is included as a dashed threshold in the graphs in figures 6.1, 6.2, 6.3, 6.4.

To test how well using adaptation performed, a non-adaptive Rao-Blackwellised Gibbs sampler (RC) that tractably collapses a subset of variables beforehand was also evaluated. This approach is similar to the one taken by several existing methods for Rao-Blackwellisation (Hamze and de Freitas 2004; Bidyuk and Dechter 2007). Once again, in order to ensure that the system did not collapse all variables, and for a fair comparison, the same number of variables were collapsed in these tests. Furthermore, the same number of parallel chains were used as in ARB (but without adaptation) for a fair comparison. The samplers were allowed to run for a maximum of 600 seconds or until convergence.

The experiments were run in an Ubuntu 18.04 environment on a physical machine with 16GB of RAM and 6 CPU's. The CPU's each have 2 hardware threads, for a total of 12 hardware

threads available to the implementation.

Table 6.2: UAI Benchmark Results, negative log of Maximum Hellinger (higher is better). RC is Random Collapsed Gibbs Sampling, and ARB is our Adaptive Rao-Blackwellisation technique. Best result is in bold.

| **Model** | **Merlin** | **RC** | **ARB** |
|---|---|---|---|
| Alchemy 11 | 2.683 | 4.018 | **4.018** |
| CSP 11 | 1.128 | 1.870 | **1.870** |
| CSP 12 | 0.746 | **1.910** | 1.860 |
| CSP 13 | 1.009 | **1.883** | 1.785 |
| Grids 11 | 0.039 | 0.583 | **1.253** |
| Grids 12 | 0.043 | 0.707 | **1.211** |
| Grids 13 | 0.010 | 0.596 | **0.879** |
| Pedigree 11 | 0.128 | 0.680 | **0.795** |
| Pedigree 12 | 0.459 | 0.742 | **0.984** |
| Pedigree 13 | 0.001 | 0.278 | **0.711** |
| Promedus 11 | 0.093 | 1.132 | **1.421** |
| Promedus 12 | 0.185 | 0.643 | **1.483** |
| Promedus 13 | 0.230 | 1.087 | **1.211** |

# Results

The results are shown in figures 6.1, 6.2, 6.3, 6.4. As shown here, in a majority of the cases, ARB is the best performing method. The results are also summarized in Table 6.2 that specifies the numerical value for the maximum Hellinger error. The benchmark problems in Grids, CSP, and Alchemy converged quickly for all algorithms. On CSP the performance of ARB and RC were more or less similar, while on the others ARB outperformed the other methods. On the Promedas and Pedigree benchmarks, ARB was the best performer followed by RC. Merlin converged much faster but could not improve on its results, and therefore had lower accuracy. On Promedas benchmarks, once again, ARB produced the most accurate solution.

In addition, the relative difficulty of the larger benchmarks (Promedas) allowed evaluation of the use of mean-field approximation for chain initialization. As mentioned above, during the

adaptive phase, new chains are created by collapsed variables chosen with a convergence diagnostic score. However, the burn-in is reduced since the initialization is performed with a mean-field approximation using the marginal estimates. The Promedas results shows that on larger benchmarks, error was continuously decreasing as more chains were added even as the burn-in period was dropping. If the new chains had started in poorly chosen areas of the sample space, then on decreasing burn-in as iterations progressed, there should have been poorer samples which would have caused consistent dips in accuracy whenever a parallel chain was added, followed by a recovery period where the accuracy would improve as the sample quality improves.

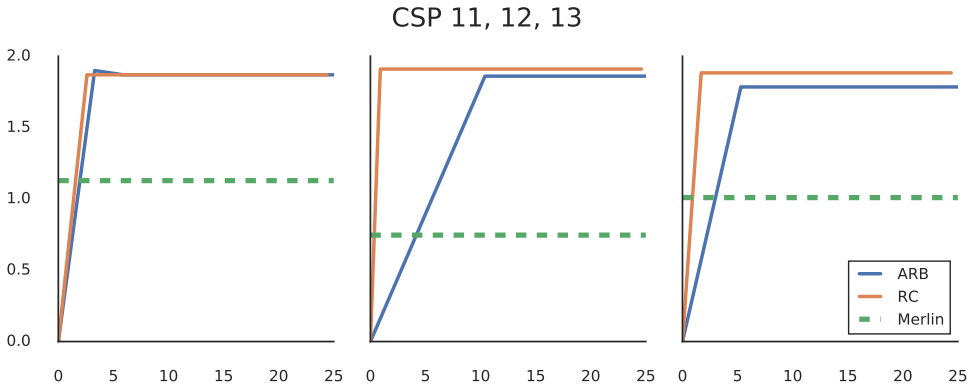



Figure 6.1: Performance of ARB algorithm on CSP vs Random Collapsed Gibbs Sampling. Dashed line is Merlin baseline result. X axis is time; Y axis is negative log of Maximum Hellinger. Note that X axes vary by model.

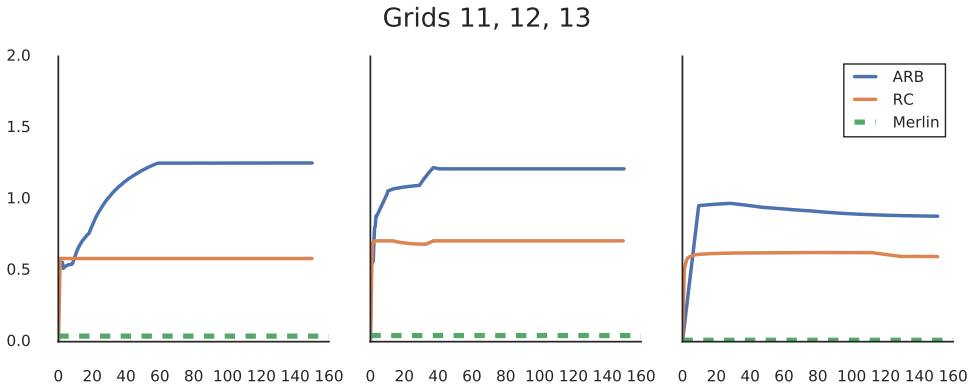



Figure 6.2: Performance of ARB algorithm on Grids vs Random Collapsed Gibbs Sampling. Dashed line is Merlin baseline result. X axis is time; Y axis is negative log of Maximum Hellinger. Note that X axes vary by model.

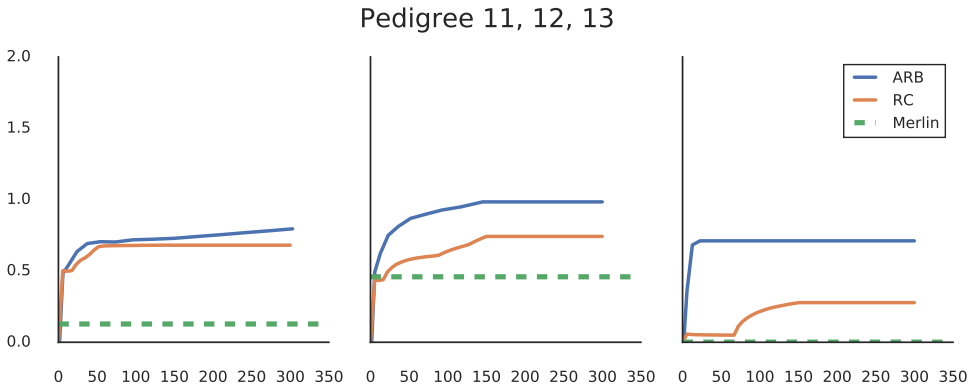



Figure 6.3: Performance of ARB algorithm on Pedigree vs Random Collapsed Gibbs Sampling. Dashed line is Merlin baseline result. X axis is time; Y axis is negative log of Maximum Hellinger. Note that X axes vary by model.

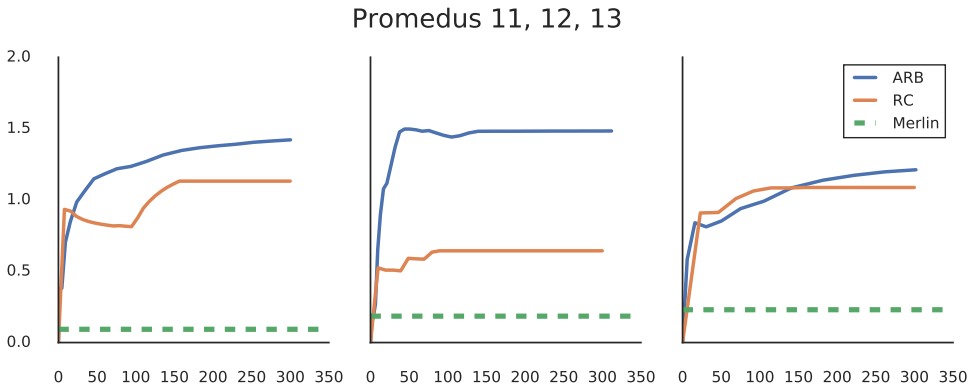



Figure 6.4: Performance of ARB algorithm on Promedas vs Random Collapsed Gibbs Sampling. Dashed line is Merlin baseline result. X axis is time; Y axis is negative log of Maximum Hellinger. Note that X axes vary by model.

# Chapter 7

# Future Work

Future work includes applying this approach to continuous PGM's and applying this technique to difference inference tasks. There is also much additional work that could be done with regards to the diagnostic used for adaptation. Finally, there are useful improvements that could be applied to the open source implementation published as part of this work.

Although marginal estimation was a good first target for this work, Gibbs sampling can also be applied in other situations. For instance, Gibbs sampling has been to used to approximate the partition function for PGM's (Evans and Fisher 1994). In addition, Gibbs sampling has a long history of being used to approximate posterior distributions, and so this technique could also be applied to tasks over the joint distribution of PGM's. For instance, estimating the most likely state of the network variables given evidence is a popular and useful task (Tierney 1994).

In addition, this work has focused on discrete networks. Updating this work for continuous and hybrid continuous/discrete networks would allow for more general applicability across all possible PGM's. Although gradient-based sampling techniques like NUTS (Hoffman and Gelman 2014) are powerful, there are continuous models that benefit from Gibbs sampling. For instance, collapsed Gibbs sampling has already been used for Latent Dirichlet Allocation (LDA) models (Porteous et al. 2008).

The current adaptive strategy is sufficiently general that there is a great deal of room for research. One could try using an entirely different convergence diagnostic: many are listed in Deonovic and Smith (2017). Alternatively, the same diagnostic calculation could be used with an error measure other than the Hellinger distance. For instance, the Jensen-Shannon divergence (Fuglede and Topsoe 2004) is symmetric and is a good candidate for research. Given enough distance metrics, an ensembled diagnostic might provide even better empirical performance.

Finally, all of the above could be implemented in the open source software developed for this thesis. Other approximate inference algorithms could be added for comparison. Any chain-based sampling strategy could theoretically be implemented. Once done, this would enable techniques with independent chains to also benefit from adaptive parallelization.

# Chapter 8

# Conclusion

Rao-Blackwellisation improves convergence in Gibbs sampling based inference by marginalizing out variables from the PGM. However, selecting the variables to sum-out is a key problem in Rao-Blackwellised Gibbs sampling. While several previous approaches have utilized the structure of the PGM to perform Rao-Blackwellisation tractably, few approaches have exploited the sampler history to choose the optimal variables to collapse.

This thesis presents an adaptive Rao-Blackwellised sampler built on the premise that, when it is intractable to jointly collapse several variables, adding parallel chains that sample from tractably collapsed structures improves marginal inference. Specifically, the new algorithm utilizes convergence diagnostics to identify the optimal variables to collapse, and starts parallel Markov chains that sample from the collapsed distribution. However, typically, adding parallel chains also wastes samples since the samplers need to get burned-in. New samplers from a mean-field approximation obtained using the marginal estimates, and showed that this approximation converges to the optimal approximation that can be obtained using the mean-field family. An open source software package has been implemented providing this functionality. This package was used to perform experiments on several UAI benchmarks which clearly showed that this approach is more accurate than state-of-the-art marginal inference solvers.

# References


Allen, D., and A. Darwiche. 2008. “RC_Link: Genetic linkage analysis using Bayesian networks.” *International Journal of Approximate Reasoning* 48 (2): 499–525.

Andrieu, Christophe, and Johannes Thoms. 2008. “A Tutorial on Adaptive MCMC.” *Statistics and Computing* 18, no. 4 (December): 343–373.

Arnborg, S., D. G. Corneil, and A. Proskurowski. 1987. “Complexity of finding embeddings in a k-tree.” *SIAM Journal of Algebraic Discrete Methods* (Philadelphia, PA, USA) 8, no. 2 (April): 277–284. ISSN: 0196-5212.

Authors, The Go. 2009. “Go Programming Language Source Code.” Accessed October 1, 2018. `https://golang.org/src/`.

Bidyuk, B., and R. Dechter. 2007. “Cutset Sampling for Bayesian Networks.” *Journal of Artificial Intelligence Research* 28:1–48.

Bruynooghe, Maurice. 1985. *Graph coloring and constraint satisfaction.* Technical report.

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. “Stan: A probabilistic programming language.” *Journal of Statistical Software,* no. 76.

Chellappa, R., and A. K. Jain, eds. 1993. *Markov Random Fields: Theory and Application.* Boston, MA: Academic Press.

Cowles, Mary Kathryn, and Bradley P. Carlin. 1996. “Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review.” *Journal of the American Statistical Association* 91:883–904.

Cox, Russ. 2011. “RNGs, algos, and concurrency.” Accessed October 1, 2018. `https://groups.google.com/forum/%5C#!msg/golang-nuts/hnG34EuJtU0/ejzvYfC3bfUJ`.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks.* Cambridge University Press.

Deonovic, Benjamin, and Brian Smith. 2017. “Convergence diagnostics for MCMC draws of a categorical variable.” *CoRR.*

Deshpande, Neil, Erica Sponsler, and Nathaniel Weiss. 2012. “Analysis of the Go runtime scheduler.” *http://www.cs.columbia.edu/ãho/cs6998/reports/12-12-11_DeshpandeSponslerWeiss_GO.pdf.*

Evans, B., and D. Fisher. 1994. “Process Delay Analysis Using Decision Tree Induction.” *IEEE Expert* 9 (1): 60–66.

Fishelson, M., and D. Geiger. 2004. “Optimizing Exact Genetic Linkage Computations.” *Journal of Computational Biology* 11 (2/3): 263–275.

Fuglede, Bent, and Flemming Topsoe. 2004. “Jensen-Shannon divergence and Hilbert space embedding.” In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on,* 31. IEEE.

Gallavotti, Giovanni. 2013. *Statistical mechanics: A short treatise.* Springer Science & Business Media.

Gamma, Erich, Richard Helm, Ralph Johnson, John Vlissides, and Grady Booch. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, Mass: Addison-Wesley Professional.

Gelman, A., and D. B. Rubin. 1992. “Inference from Iterative Simulation Using Multiple Sequences.” *Statistical Science* 7 (4): 457–472.

Geman, S., and D. Geman. 1984. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.

Getoor, L., and B. Taskar, eds. 2007. *Introduction to Statistical Relational Learning.* MIT Press.

Gogate, V., and R. Dechter. 2004. “A Complete Anytime Algorithm for Treewidth.” In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence,* edited by David Maxwell Chickering and Joseph Y. Halpern, 201–208. AUAI Press.

Gogate, Vibhav. 2014. *UAI Inference Competetion.* Technical report. Http://www.hlt.utdallas.edu/ vgogate/uai14-competition/index.html. UT-Dallas.

Gonzalez, J., Y. Low, A. Gretton, and C. Guestrin. 2011. “Parallel Gibbs Sampling: From Colored Fields to Thin Junction Trees.” In *In Artificial Intelligence and Statistics.* May.

Griesemer, Robert, Rob Pike, and Ken Thompson. 2012. *The Go Programming Language Specification.* Technical report. Google. Accessed October 1, 2018. `https://golang.org/ref/spec`.

Hamze, F., and N. de Freitas. 2004. “From Fields to Trees.” In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence,* 243–250.

Hoare, C. A. R. 1985. *Communicating Sequential Processes.* Prentice Hall.

Hoffman, EJ, JC Loessi, and RC Moore. 1969. “Constructions for the solution of the m queens problem.” *Mathematics Magazine* 42 (2): 66–72.

Hoffman, Matthew D., and Andrew Gelman. 2014. “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *Journal of Machine Learning Research* 15:1593–1623.

Ihler, A., J. Fisher, and A. Willsky. 2005. “Loopy Belief Propagation: Convergence and Effects of Message Errors.” *Journal of Machine Learning Research* 6:905–936.

Jordan, M. I., and Francis Bach, eds. 1998. *Learning in Graphical Models.* Boston, MA: A Bradford Book.

Jordan, M. I., Z. Ghahramani, T. Jaakkola, and L. K. Saul. 1999. "An Introduction to Variational Methods for Graphical Models." *Machine Learning* 37 (2): 183–233.

Kelly, Craig. 2018. "Grample: Sampling for Probabilistic Models." Accessed October 31, 2018. `https://github.com/CraigKelly/grample/`.

Koller, D., and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press.

Kullback, Solomon, and Richard A Leibler. 1951. "On information and sufficiency." *The annals of mathematical statistics* 22 (1): 79–86.

Kumar, Vipin. 1992. "Algorithms for constraint-satisfaction problems: A survey." *AI magazine* 13 (1): 32.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing.* Springer Publishing Company, Incorporated.

Liu, J. S., W. H. Wong, and A. Kong. 1994. "Covariance structure of the Gibbs sampler with applications to the comparison of estimators and augmentation schemes." *Biometrika* 81 (1): 27–40.

Lynce, Inês, and Joël Ouaknine. 2006. "Sudoku as a SAT Problem." In *ISAIM.*

Marinescu, Radu. 2016. "Merlin:An extensible C++ library for probabilistic inference in graphical models." Accessed October 1, 2018. `https://github.com/radum2275/merlin`.

Mateescu, R., K. Kask, V. Gogate, and R. Dechter. 2010. "Iterative Join Graph Propagation algorithms." *Journal of Artificial Intelligence Research* 37:279–328.

Matsumoto, Makoto, and Takuji Nishimura. 1998. “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.” *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8 (1): 3–30.

McCallum, A., and B. Wellner. 2004. “Conditional Models of Identity Uncertainty with Application to Noun Coreference.” In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS).*

Neal, R. M. 1993. *Probabilistic Inference Using Markov chain Monte Carlo Methods.* Technical report CRG-TR-93-1. Toronto, Canada: Department of Computer Science, University of Toronto.

Paskin, M. A. 2003. “Sample Propagation.” In *Advances in Neural Information Processing Systems,* 425–432.

Pearl, Judea. 1982. “Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach.” In *Proceedings of the Second National Conference on Artificial Intelligence: AAAI-82,* 133–136.

———. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Francisco, CA: Morgan Kaufmann.

Porteous, Ian, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. “Fast collapsed gibbs sampling for latent dirichlet allocation.” In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining,* 569–577. ACM.

Scharstein, D., and R. Szeliski. 2002. “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms.” *International Journal on Computer Vision* 47, nos. 1-3 (April): 7–42. ISSN: 0920-5691.

Shwe, M.A., B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. 1991. "Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms." *Methods of Information in Medicine* 30 (4): 241–55.

Tierney, Luke. 1994. "Markov chains for exploring posterior distributions." *Annals of Statistics* 22:1701–1762.

Venugopal, Deepak, and Vibhav Gogate. 2013. "Dynamic Blocking and Collapsing for Gibbs Sampling." In *Uncertainty In Artificial Intelligence.*

Voss, Jochen. 2013. "The Mersenne Twister in Go." Accessed October 1, 2018. `https://github.com/seehuhn/mt19937/`.

Wemmenhove, B., J. M. Mooij, W.Wiegerinck, M. A. R. Leisink, H. J. Kappen, and J. P. Neijt. 2007. "Inference in the Promedas Medical Expert System." In *Eleventh Conference on Artificial Intelligence in Medicine,* 456–460.

Williamson, David P., and David Bernard Shmoys. 2011. *The Design of Approximation Algorithms.* New York: Cambridge University Press.

Yedidia, J. S., W. T. Freeman, and Y. Weiss. 2001. "Generalized Belief Propagation." In *Advances in Neural Information Processing Systems 13,* edited by T. Leen, T. Dietterich, and V. Tresp, 689–695. Cambridge, MA: MIT Press.

Zhang, N., and D. Poole. 1994. "A simple approach to Bayesian network computations." In *Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference.*