STAT 31512 Spring 2024 Robert Koort
May 21, 2024 Lecturer: Frederic Koehler

Final Paper - Survey of Approaches to Parallel Sampling

Abstract

Computing power continues to expand at a rapid rate, but it is expanding differently than it has in the past. Cores aren't much more powerful, but there are a lot more of them. As such, parallel sampling has become a hot topic. It would be useful to be able to utilize that increase in distributed power to sample faster. Unfortunately, many methods have been proposed without empirical proof of their efficacy, their respective papers making big claims about effectiveness that are hard to verify. The goal of this paper is to clarify how these algorithms actually work, then implement them to see if they work well. Implementation is done in Python and can be found at `github.com/rakoort99/parallel_sampling/`.

1 Introduction

Obviously, it is important to be able to draw samples from distributions. Either because it's valuable to get those samples directly, to get an estimate of the shape of a distribution, or to learn some parameter associated with a distribution, standard practice is typically to turn to Markov Chain Monte Carlo (MCMC) methods. Useful theoretical properties, such as "fast" mixing and lack of bias, subject to certain conditions, make this class of algorithms desirable for theoreticians and practitioners.

However, standard sequential MCMC methods have limitations in certain increasingly-common use-cases. When graphical models become large, mixing becomes slow. Sampling one site per step blows up the mixing time considerably, even when the graph otherwise meets the necessary conditions for fast mixing. Similarly, the sequential speed of computing power has not increased nearly as much as its distributed potential. Modern computer architectures are powerful not because of their speed, but because of the number of "independent workers" at their disposal. Sequential sampling algorithms are ill suited to utilize separate workers in this way. Although running sequential chains in parallel will results in more samples, it doesn't make it any quicker to get *good* samples.

The motivations behind parallel sampling methods are to resolve these tensions. If it is possible to find a way to modify standard methods to effectively utilize distributed computing architectures while avoiding race conditions, it may be possible to make sampling from large, complex graphical models substantially more efficient. If it is possible for multiple workers to update multiple sample sites independently, producing a single chain cooperatively without introducing too much bias, then suddenly many tasks become much more efficient. This paper will examine a slew of proposed parallel sampling methods from both practical and theoretical perspectives. I implement all of the discussed methods and examine empirical results on toy distributions.

2 Preliminaries

First, it is necessary to define some preliminary notation and concepts. Sampling will be treated as taking place on a graphical model or undirected graph $G(V, E)$. The n vertices or nodes $v_i \in V$ mark our sample sites and edges mark dependencies. The maximum degree of this graph will be denoted Δ. Second, most of the methods discussed here are only defined in the context of discrete distributions. In order to keep consistent, I will generally discuss the discrete case and denote the set $[q]^V$ to be the set of all possible values at our sample sites, with particular values $X = \{x_1, x_2, ..., x_n\}$. Most algorithms mentioned trivially

extend to the continuous case, but I will use discrete notation for simplicity. The target distribution over $[q]^V$ is denoted π. Last, I will use w_i to denote the individual "workers," or processors, involved in sampling.

Now, it is necessary to establish a baseline algorithm: standard, sequential Gibbs as defined in 1. Further-

Algorithm 1 Sequential Gibbs

Require: initial state $X^{(0)} \in [q]^V$ and target distribution π
  for $t = 1$ to T **do**
    Sample site i uniformly from site indices $\{1, 2, ..., N\}$
    Update $x_i \sim \pi(x_i = \cdot | X_{\sim i}^{(t-1)})$
  end for

more, it is necessary need to establish some metrics for an algorithm's success. Total variation distance is defined

$$||\mu - \nu||_{\text{TV}} = \sup_{A \in \Omega} |\mu(A) - \nu(A)|$$

for two probability measures μ, ν and a metric space Ω. Total variation is thus the maximal difference between the densities of two distributions.

It can be shown that a Gibbs sampler is unbiased with respect to total variation in the limit:

$$\lim_{t \to \infty} ||P^t \mu_0 - \pi||_{\text{TV}} = 0$$

where $P \in \mathcal{R}^{n,n}$ denotes the (Markovian) transition probability matrix of the Gibbs sampler and μ_0 denotes some initial distribution. Importantly, this only holds when the transition probability matrix is ergodic.

In a similar vein, the timestep in which the approximation becomes arbitrarily close to the target distribution is called the ϵ-mixing time.

$$t_{\text{mix}}(\epsilon) = \inf\{t \geq 1 : \forall \mu_0 \;\; ||P^t \mu_0 - \pi||_{\text{TV}} < \epsilon\}$$

Under Dobrushin's condition [3], decently tight theoretical guarantees can be made for speed of mixing. To define this condition, it is necessary to first define Dobrushin's influence matrix:

$$R_{ij} = \sup_{\mathbf{y} \sim j = \mathbf{z} \sim j} |\pi(x_i | \mathbf{y}) - \pi(x_i | \mathbf{z})|$$

This creates a matrix of the maximal differences between the distribution of of x_i conditioned on site values differing only at site j. Then, total influence is defined:

$$\alpha = \max_i \sum_j R_{ij}$$

This is simply the maximum row sum of the influence matrix. When $\alpha < 1 - \epsilon$ for some nonzero epsilon, Dobrushin's condition is met [8]. This is often simplified in the literature to $\alpha < 1$. Under this condition, Gibbs sampling is known to converge rapidly, mixing in $\mathcal{O}(n \log n)$ iterations. This will serve as the measuring stick for mixing times of future methods. However, parallel sampling methods draw many samples concurrently which makes comparisons a little strange. Presuming optimal coordination between workers, it is safe to effectively divide the leading n by the minimum efficiency gained from parallelization in order to compare "fairly."

Spectral methods are often employed in order to analyze sequential samplers. However, parallelization often results in the loss of reversibility and ergodicity. When updates are sent which depend on outdated

states, these conditions are lost. Similarly, the added stochasticity in most parallel methods makes it difficult to evaluate any sort of spectral gap [2]. As such, it is often necessary to look at path couplings instead of linear algebra [1].

The path coupling theorem gives a way to bound mixing times on stochastic processes. First, define $\phi(X, Y) = |v \in V : X_v \neq Y_v|$. Then, given $\phi(X, Y) = 1$, if there exists a coupling $(X, Y) \to (X', Y')$ such that $\mathbb{E}[\phi(X', Y')|X, Y] \leq 1 - \delta$ for some $\delta \in (0, 1)$, then $t_{\text{mix}}(\epsilon) = \mathcal{O}(\frac{1}{\delta} \log n)$. As such, many proofs regarding mixing times parallel sampling methods involve bounding this expectation.

Last, I separate parallel sampling methods into two groups: synchronous methods and asynchronous methods. Taking definitions from Terenin et al. [11], I choose to define asynchronous methods to mean those which simply update as soon as compute is available and accept updates immediately after they are submitted. Synchronous methods, on the other hand, propose and process samples in batches. Many proposals may be calculated concurrently, but the entire "round" must terminate before moving onto the next.

3 Synchronous Methods

In the literature, there is a bit less attention paid to synchronous methods than asynchronous methods - they are less of a departure than standard sequential Gibbs. They also lack some of the potential for speedup present in asynchronous methods, as they necessarily require waiting time for updates to "sync up" with others. Still, they serve as a nice jumping-off point as they are less subject to race conditions and tend to be more feasible on a wider class of models.

3.1 Chromatic Gibbs

The Chromatic parallel Gibbs sampler simply cuts up the graph G into independent parts, which are sampled in parallel [7]. The idea is quite intuitive, and has been frequently re-examined since the original paper. New contributions typically involve finding more sophisticated ways of determining node colorings [5] and don't substantively change the core concept.

It is a basic fact that, in a graphical model, conditional distributions at each node are independent of values of far away nodes given the values of their neighbors. In other words, with $a \sim Q(b), b \sim P(c)$, $P(a|b) = P(a|b, c)$. This is part of what makes Gibbs sampling efficient in the first place, as this fact greatly reduces the complexity of sampling from full conditionals. Importantly, it also leads itself to a natural, parallel extension. If a set of sample sites are independent of one another, then they can be updated concurrently without introducing bias to the samples.

It is necessary to introduce the notion of the proper k-coloring of a graph - an assignment of k colors to vertices V such that no vertices who share an edge also share a color. Any proper k-coloring of a graphical model informs which nodes may be sampled in parallel without inducing bias.

Although it should be fairly obvious that this algorithm does not degrade the sequential Gibbs sampler, it is useful to show the proofs from [5]. By showing it is reversible and has stationary distribution μ, it follows that the total variation converges to 0 as $T \to \infty$. With states $X, Y \in [q]^V$ and transition probability matrix P, it is sufficient to show the detailed balance equations:

$$\pi(X)P(X, Y) = \pi(Y)\pi(Y, X)$$

This can be split into cases. First, if X is "infeasible" such that $\pi(X) = 0$, the detailed balance equations are satisfied. The RHS is trivially 0, and the LHS is 0 since our transition matrix never takes us from an acceptable state to an unacceptable one. This applies WLOG to the case where Y is infeasible.

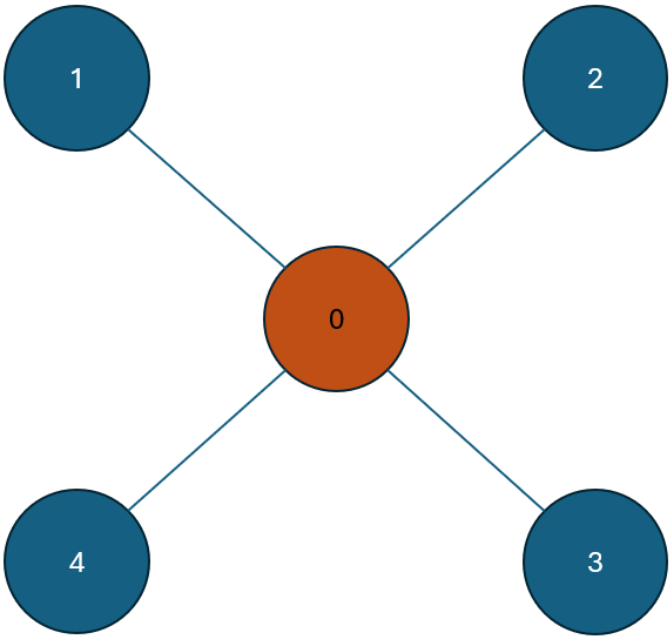

Figure 1: Example minimum-k color graph. Trivially, the conditional distributions of nodes 1 through 4 only depend on node 0. As such, nodes 1 through 4 can be sampled without altering any of the convergence properties of sequential Gibbs. This is analogous to sampling in order $\{0, 1, 2, 3, 4\}$ and thinning such that only the 1st and 5th samples of each 5-cycle are retained.

Algorithm 2 Chromatic Gibbs

Require: initial state $X^{(0)} \in [q]^V$, target distribution π, sets of independent nodes $\mathbf{k}$
  for $t = 1$ to T **do**
    Sample set of independent nodes κ uniformly from $\mathbf{k}$
    for nodes in κ **do in parallel**
      Update $x_i \sim \pi(x_i = \cdot | X_{\sim i}^{(t-1)})$
    end for
  end for

Second, the case where both states X and Y are feasible. Recall that our transitions only apply within colorings, or independent sets of sample sites. Then, it suffices to show $\frac{\pi(X)}{\pi(Y)} = \frac{P(X,Y)}{\pi(Y,X)}$. With an arbitrary set of independent nodes κ, this is shown as follows:

$$\frac{P(X,Y)}{P(Y,X)} = \frac{\sum_{\mathbf{k}} \Pr(\kappa) \Pr(X \to Y|\kappa)}{\sum_{\mathbf{k}} \Pr(\kappa) \Pr(Y \to X|\kappa)} = \frac{\sum_{\mathbf{k}} \Pr(\kappa) \prod_{u \in \kappa} \pi(Y_u|X_{N(u)})}{\sum_{\mathbf{k}} \Pr(\kappa) \prod_{u \in \kappa} \pi(X_u|Y_{N(u)})} = \frac{\pi(Y)}{\pi(X)}$$

It is also fairly trivial that this sampling scheme should result in faster convergence. Assuming perfect parallelization, i.e. enough workers that each node is processed individually, the expected number of sites sampled per iteration is n/k. Then, this approach should mix in $\mathcal{O}(k \log n)$ iterations. The optimal coloring is hard to find, but the number of colors necessary it generally bounded above by Δ. Then, in the general case, chromatic sampling results in $\mathcal{O}(\Delta \log n)$. Although this is not a speedup that is linear with the number of processors, it's pretty good! It's also relatively easy to implement, not requiring complex scheduling routines.

Unfortunately, there are obvious limitations to this approach. First, the number of sites in each independent set provides a bottleneck it is impossible to meaningfully pass. For instance, highly-connected graphs will not see much speedup compared to sequential Gibbs. With this synchronized approach, only $\max|\kappa|$ updates can ever be taken concurrently. A distributed system with a larger number of processors will necessarily keep them idle. All these updates must be resolved before before the next batch, further increasing processor idle time in the likely case these updates do not resolve at precisely the same time. Alternative approaches to scheduling can result in more worker activity, such as LucyGlauber [5]. Second, finding the optimal $k-$coloring is NP-hard on a general graph [7]. This can be problematic when graphs don't take familiar forms, as k is the main bottleneck in this approach. Luckily, many common structures in ML such as LDA and hidden Markov models have trivial optimal colorings.

3.2 Local Glauber Dynamics

The Local Glauber Dynamics approach to sampling is an attempt to speed up Gibbs sampling from $\mathcal{O}(n \log n)$ to $\mathcal{O}(\log n)$ when Dobrushin's condition is met [6]. In order to accomplish this, they note that only the acceptance of conflicting updates causes the loss of the asymptotic unbiased-ness of sequential Gibbs. Then, if it is possible to update $\approx n$ sites without conflict on every iteration, the mixing time will reduce to $\mathcal{O}(\log n)$. Note that this approach is designed with a specific use case in mind: uniform proper k-colorings. In other words, each sample site takes values uniformly at random with the restriction that no neighboring sites can take the same value. Note that in this setting, there must be $k = \eta\Delta$ colors for some $\eta > 2$ in order for Dobrushin's condition to be met [5].

In this approach, the loop takes two phases. In the first phase, updates for sites are proposed with probability γ. In the second phase, each site compares its proposal to the proposal of its neighbors. If proposals conflict, then the sites retain their previous values.

The condition that sites do not conflict ensures the reversibility and asymptotic convergence of the chain. It can easily be shown that, under this approach, the chain satisfies the detailed balance equation. First, the uniform proper $k-$coloring setting means for any $X, Y \in [q]^V$, $\pi(X) = \pi(Y)$. Second, from the definition of the algorithm, it is apparent that the update rule is symmetric, i.e. $P(X,Y) = P(Y,X)$.

The paper authors find a mixing time of $\mathcal{O}(\log n)$ for Local Glauber Dynamics. To prove this, they use a Path Coupling approach where proposals are shared between chains. The goal is to bound $\mathbb{E}[\phi(X', Y')|X, Y]$ where $\phi(X,Y) = 1$. First, they denote the single non-shared site v_0. Then, it is possible to split the expectation into two parts - one part for the currently-different site v_0, and another for the currently-

Algorithm 3 Local Glauber Dynamics

Require: initial state $X^{(0)} \in [q]^V$, graph $G(V, E)$
  for $t = 1$ to T **do in parallel**
    Draw $m \sim \text{Unif}(0, 1)$
    if $m < \gamma$ **then**
      Propose $x_i' \sim \text{Unif}([q])$
    else
      Propose $x_i' = x_i$
    end if
    if $x_i' \neq x_j' \; \forall \; x_j' \in N(x_i')$ **then**
      Update $x_i = x_i'$
    else
      Keep $x_i = x_i$
    end if
  end for

identical sites $\sim v_0$

$$\mathbb{E}[\phi(X', Y')|X, Y] = \mathbb{E}[\phi(X'_{v_0}, Y'_{v_0})|X, Y] + \mathbb{E}[\phi(X'_{\sim v_0}, Y'_{\sim v_0})|X, Y]$$

Starting with the un-shared case, it is apparent that the site needs to get marked for update with probability γ, update to an acceptable color with worst-case probability $1 - \frac{\Delta}{k}$, and have its neighbors update to acceptable colors with worst-case probability $1 - \frac{3\gamma}{k}$. This results in equation

$$\mathbb{E}[\phi(X'_{v_0}, Y'_{v_0})|X, Y] \leq 1 - \gamma(1 - \frac{\Delta}{k})(1 - \frac{3\gamma}{k})^{\Delta}$$

The first terms are fairly obvious. The last term warrants slightly more explanation. If a neighbor doesn't update, that's fine. If it does, which happens with probability γ, it must not update to block the future "good" value of v_0.

The shared case is more involved. First, it is important to reiterate that, in this construction, the two chains share proposals. As such, the only instance where $\phi(X'_{\sim v_0}, Y'_{\sim v_0}) > 0$ is when one chain accepts proposals the other rejects. This is only possible with paths starting at nodes adjacent to v_0 and with proposals of the same color of v_0 in one chain. After the first proposal, there are at most two valid proposals to retain consistency. There are at maximum Δ^l possible paths where l denotes the length of the path. However, these paths can be arbitrarily long. To get a proper upper bound, it is necessary to consider the limit case of an infinite graph. Then, all together, the bound for the shared case is

$$\mathbb{E}[\phi(X'_{\sim v_0}, Y'_{\sim v_0})|X, Y] \leq \sum^{\infty} \Delta^l \frac{\gamma}{k} (\frac{2\gamma}{k})^{l-1} = \frac{1}{2} \sum^{\infty} \Delta^l (\frac{2\gamma}{k})^l \leq \frac{\frac{\gamma\Delta}{k}}{1 - \frac{2\gamma\Delta}{k}}$$

Noting that, in order to ensure Dobrushin's condition is met, $\eta = k/\Delta$, the above bound simplifies:

$$\mathbb{E}[\phi(X', Y')|X, Y] \leq 1 - \gamma(1 - \frac{\Delta}{k})(1 - \frac{3\gamma}{k})^{\Delta} + \frac{\frac{\gamma\Delta}{k}}{1 - \frac{2\gamma\Delta}{k}} \leq 1 - \gamma e^{-\frac{6\gamma}{\eta}} (1 - \frac{1}{\eta}(1 + \frac{e^{\frac{6\gamma}{\eta}}}{1 - \frac{2\gamma}{\eta}}))$$

By the Path Coupling theorem, this shows a mixing time of $\mathcal{O}(\log n)$. Of course, there is a very messy constant dependent on η and γ that is obliterated by $\mathcal{O}$ notation. It is unclear if this constant is generally smaller than the Δ it replaced, as the small update probability and collision blocking make the likelihood of an update at any given node quite low. The original paper does not include experimental results, which I rectify in part 5.

4 Asynchronous Methods

Asynchronous sampling methods involve independent calculation and processing of updates. Unlike in the synchronous approaches, in which updates are processed together in batches, asynchronous updates are processed individually as soon as they are available.

4.1 Hogwild Gibbs

The most obvious asynchronous approach to Gibbs sampling is the Hogwild Gibbs sampler, which is adapted from similar approaches to stochastic gradient descent. The idea is quite simple: process in parallel many Gibbs updates, and send them to a "central" chain. Of all the asynchronous methods, this is by far the most well-studied.

Algorithm 4 Hogwild Gibbs

Require: initial state $X^{(0)} \in [q]^V$ and target distribution $\pi, t = 0$
  while $t < T$ **do in parallel**
    Load current state $X^{(t)}$
    Sample site i uniformly from site indices $\{1, 2, ..., N\}$
    Update $x_i \sim \pi(x_i = \cdot | X_{\sim i}^{(t-1)}), t = t + 1$
  end while

Because updates are occurring asynchronously and in parallel, the reversibility of the Markov process is broken [2]. With more than one worker, sample sites will inevitably be updated while another worker is computing its own update. Therefore, updates are not based on the "true" latest samples. Instead, each update depends on "lagged" samples. Explicitly, this lag is denoted τ. Then, each new sample takes the form:

$$X^{(t)} = (x_1^{(t-1)}, ..., x^*, ..., x_N^{(t-1)})$$
$$x^* \sim p_i(x_i | X_{\sim i}^{(t-\tau)})$$

This is potentially quite problematic. It is clear that this process will rarely be asymptotically unbiased with respect to total variation as it is happy to accept conflicting samples. For example, Hogwild Gibbs samples from the joint distribution $P(0,0) = 0, P(1,1) = P(1,0) = P(0,1) = \frac{1}{3}$ will never be unbiased. If an update depends on the state $(1,1)$, then can send either site to 0. Inevitably, two updates will send contradictory sites to 0 given the nature of the sampling approach.

This does not make the Hogwild Gibbs sampler worthless. It is proven that the sampler will at least converge when sampling multivariate Gaussians parameterized by diagonally dominant precision matrices [9], and empirically it tends to converge to (potentially incorrect) stationary distributions for wide classes of probability distributions. More generally, total variation may not be the best metric to use in all instances. Sometimes, the distance from the true joint distribution is less important than accurately sampling from marginals.

To evaluate convergence-in-part, it is useful to introduce ω-sparse variation.

$$||\mu - \nu||_{\mathrm{SV}(\omega)} = \sup_{|A| \leq \omega} |\mu(A) - \nu(A)|$$

Here, ω bounds the number of sites to simultaneously compare when evaluating the difference between distributions. Then, the $1-$sparse variation distance is equivalent to the greatest distance between marginals, as opposed to the distance between joints described by total variation. From this new sparse distance, it is possible to define sparse estimation time, an the ω-sparse analogue to mixing time.

$$t_{\mathrm{SE}(\omega)}(\epsilon) = \inf\{t \geq 1 : \forall \mu_0 \ \ ||P^t \mu_0 - \pi||_{\mathrm{SV}(\omega)} < \epsilon\}$$

Since it is likely total variation convergence cannot be achieved, it is useful to have surrogate metric. Given a few restrictive conditions, it's possible to show that Hogwild Gibbs compares favorable to sequential Gibbs sampling in terms of sparse estimation time. Although the proofs are omitted from their paper[1], they find that subject to Dobrushin's condition and $\epsilon \geq 2\omega\alpha\tau(1-\alpha)^{-1}n^{-1}$,

$$t^{(\mathrm{hog})}_{\mathrm{SE}(\omega)} \leq \lceil \frac{n}{1-\alpha} \log(\frac{\omega}{\epsilon}) + \frac{2\omega\alpha\tau}{(1-\alpha)^2\epsilon} \rceil$$

where $\lceil \cdot \rceil$ denotes the ceiling operator. The sketch of the proof is as follows. First, derive that $t^{(\mathrm{seq})}_{\mathrm{SE}(\omega)}(\epsilon) \leq \lceil \frac{n}{1-\alpha} \log(\frac{\omega}{\epsilon}) \rceil$ subject to Dobrushin's condition, which follows from the similar bound on mixing time. Second, bound the sparse variation distance between the sequential and Hogwild chains at time t. Third, plug in and clean up by placing restrictions on ϵ. Note that this shows that the sparse estimation time of the Hogwild sampler is only different by a constant term. This means, given proper implementation, the Hogwild sampler has the potential to produce unbiased marginal estimates much faster in real-time than sequential Gibbs.

This was a nice aside for the cases in which total variation is not the key metric, but it would be wrong to discount the importance of accurately sampling from the joint. As shown previously, there are instances where sequential Gibbs sampling will work wonderfully but Hogwild Gibbs sampling will always fail to mix. However, under Dobrushin's condition, one can expect Hogwild sampling to produce comparable results to sequential Gibbs sampling in terms of mixing time.

Under Dobrushin's condition,

$$t^{(\mathrm{hog})}_{\mathrm{mix}}(\epsilon) \approx (1 + \alpha\bar{\tau}n^{-1}) t^{(\mathrm{seq})}_{\mathrm{mix}}(\epsilon) \leq \frac{n + \alpha\bar{\tau}}{1-\alpha} \log(\frac{n}{\epsilon})$$

where $\bar{\tau}$ denotes the expected lag for update dependencies. This proof follows a similar scheme to those in the synchronous methods - it uses path coupling. This framing of the path coupling theorems is slightly different than the previous approach used, but logically follows. First, they note that mixing time is bounded above by coupling time.

$$t_{\mathrm{mix}}(\epsilon) \leq \min_t \{\Pr(T_c > t) \leq \epsilon\}$$

where T_c denotes the coupling time, or time at which two differently-initialized chains X, Y become equivalent when sampled at the same sites. Trivially, one can deduce that $\Pr(T_c > t) \leq n \max_i \Pr(X_{i,t} \neq Y_{i,t})$. In other words, the probability that a chain has mixed at time t is bounded above by the worst-case probability that sites are equivalent at time t. This quantity can be further bounded above by utilizing the total influence, α. Incorporating the lag τ and doing some algebra to get things into a closed form produces the above bound.

The implication is that, under Dobrushin's condition, Hogwild Gibbs is possibly more practically useful than sequential Gibbs. The slowdown in samples-to-converge is lost in big $\mathcal{O}(\cdot)$ notation, and sampling occurs n_w times faster, where n_w denotes the number of available workers. This means ineffeciencies should be easily outweighed by the real-time speedup from using multiple workers concurrently.

This result does not mean that Hogwild Gibbs can be used interchangeably with sequential Gibbs. Although it's true that sequential Gibbs produces best results for distributions that satisfy Dobrushin's

[1] I do as well - it would take up too much real estate for a result that is a relatively minor aside

condition, it still tends to work well enough when the condition is not satisfied. Hogwild Gibbs is much more sensitive in this way, and acts much more unstable when the condition is violated. Still, Terenin et al. note that, so long as π doesn't contain too much dependence between sites and has significantly more dimensions than the sampler has workers, Hogwild Gibbs is a practical alternative to sequential sampling [10].

4.2 Exact Asynchronous Gibbs

Exact asynchronous Gibbs is an attempt to issue a correction to the updates of the Hogwild sampler such that race conditions are mitigated. Rather than a single chain, each worker maintains its own chain but shares proposals. The exact asynchronous Gibbs sampler implements a Metropolis-Hastings acceptance step to prevent race conditions and maintain consistency of the chains, but increases the complexity of the communication structure between workers as a byproduct [10].

Algorithm 5 Exact Asynchronous Gibbs

Require: initial state $X^{(0)} \in [q]^V$ and target distribution $\pi, t = 0$
  while $t < T$ **do in parallel**
    Accept proposals from other workers with Metropolis-Hastings probability
    Sample site i uniformly from site indices $\{1, 2, ..., N\}$
    Update $x_i \sim \pi(x_i = \cdot | X_{\sim i}^{(t-1)})$, $t = t + 1$
    Send proposal x_i, state X^t to other workers
  end while

For a basic intuition on this approach, it is useful to base a mental model off the parallel-sequential approach, in which several independent sequential chains are sampled in parallel. In this setting, updates within individual chains follow normal Gibbs update steps just as in the parallel-sequential case. The place the exact asynchronous approach differs is in what occurs between updates. Here, each worker w_s sends both its current state and most recent update to every other worker w_i. Before resampling, each worker processes the updates it was sent in the time elapsed during while calculating its previous proposal.

To clarify the process, I will trace the example in which w_s has updated while w_r was calculating its last update. Before its next Gibbs update, w_r must process the proposal from w_s. For simplicity, w_s can be thought of as the "sender worker" and w_r the "receiver worker." Next, denote the state at the sender worker X_s and the state at the receiver worker X_r. Then, the the recent proposal from w_s, which occurred at index j, is x_{sj}. Last, the state in which w_r accepts the proposal x_{sj} is denoted X_r'. Finally, the proposed update is accepted according to the following Metropolis Hastings probability:

$$\alpha = \min(1, \frac{\pi(X_r')q(X_s \to X_r)}{\pi(X_r)q(X_s \to X_r')}$$

where $q(x \to z)$ denotes the Markov transition probability from state x to state z.

It is not difficult to see intuitively why, given sequential Gibbs sampling of π is asymptotically unbiased, this procedure results in unbiased estimates across all chains. The Metropolis-Hastings acceptance step means that proposals from other workers will be accepted consistently with the true distribution. The argument for speedup is similarly intuitive. As proposals are shared between chains, each chain simply receives more expected updates per iteration than a purely sequential approach.

The computational and resource cost of communication between workers is substantial. At minimum, each worker must store its own chain, the queue of proposals it needs to process. The addition of the Metropolis

step also greatly increases the computational complexity of each iteration at each worker, especially when the cache of updates to process is large. This greatly limits the practical use of this algorithm. Although the exact asynchronous Gibbs sampler certainly mixes quicker, the cost of each sample is quite substantial.

It is worth noting three things. First, it is useful note that this approach should work in the same instances where a Gibbs sampler is acceptable. There are no additional limitations, other than the ability to calculate Metropolis-Hastings acceptance probabilities. Second, this is the only method discussed in this paper which necessarily creates "separate" chains for each worker. Third, there are no theoretical guarantees of a tighter asymptotic bound on real-time mixing times - one must trust that benefits of parallelism outweigh implementation costs. The main improvement made in this method is that it allows for correction of the bias introduced by the Hogwild sampler, but it is unclear that the medicine isn't more harmful than the disease.

4.3 Distributed Metropolis Sampler

The last method I will introduce in this paper is the Distributed Metropolis sampler [4]. I had a lot of difficulty with this paper due to over-complicated, inconsistent notation and unclear explanation. However, it is ethically quite similar to the previous approach.

This approach can be considered a simplification of the exact asynchronous Gibbs sampler. Rather than generating proposals according to Gibbs updates across numerous chains, this approach makes proposals with arbitrary distribution ν and includes a Metropolis-Hastings acceptance step. Rather than having separate chains updated by each worker, workers are assigned to individual nodes and share a single chain. Last, rather than having workers alternate between updating the current state and proposing new updates, this approach takes place in two phases: the first where *all* proposals are drawn, and the second where *all* proposals are processed.

This process is outlined in 6. In phase one, at each node, update times are drawn according to a Poisson process and the corresponding proposals are drawn from some arbitrary distribution. For example, node i produces a list of update times $0 < t_i^1 < t_i^2 < ... < t_i^m < T$, where m denotes the number of updates and is drawn $m \sim \text{Poisson}(T)$, and a list of proposals $x_i^{1'}, x_i^{2'}, ..., x_i^{m'}$ drawn i.i.d. from arbitrary distribution ν_i. Each node shares both its list of updates and list of proposals with all its neighbors. Then, phase two begins.

In phase two, the updates are processed. Given the information distributed in phase one, each node can know when it is safe to update. Obviously, it is safe to propose update $x_i^(t)$ when all values in the neighborhood of x_i at time $t-1$ are known. This results in sufficient information to evaluate the Metropolis-Hastings acceptance probability. However updates can be processed even with limited information. For example, with proper graph colorings, let proposal $x_i^{t'} = \text{red}$. Then, once it is known that any neighbor is red at time t, it is safe to discard this proposal.

This trick is actually possible in the general case, too. By pre-drawing $z^t \sim \text{Unif}(0, 1)$, it will sometimes be possible to make an update before it is possible to calculate the Metropolis acceptance probability α. Because each node knows the full proposal schedule of its neighbors after phase one, it's possible to calculate the minimum acceptance probability $\alpha_{\min}$ and maximum acceptance probability $\alpha_{\max}$. As information information from neighbors trickles in, and the space of possible neighboring values constricts, these bounds approach the true α. If, at any point, $z < \alpha_{\min}$ or $z \geq \alpha_{\max}$, the next proposal can be safely accepted or rejected without full information. This is the motivation for departing from the Gibbs sampler in favor of the more primitive Metropolis.

The ultimate result is that, since n worker contributing to the chain without inducing race conditions,

Algorithm 6 Distributed Metropolis

Require: initial state $X^{(0)} \in [q]^V$ and target distribution $\pi, t = 0$,
  Phase One:
  for $v \in V$ **do in parallel**
    Draw strike times $0 < t_v^1 < t_v^2 < ... < t_v^m < T$ according to Poisson process
    Draw proposals $x_v^{1'}, x_v^{2'}, ..., x_v^{m'}$ i.i.d. from ν_v
    Send strike times, proposals to $u \in N(v)$
  end for
  Phase Two:
  for $v \in V$ **do in parallel**
    for t_v^i, $x_v^{i'}$ in strike times, proposals **do**
      Sample $z \sim \text{Uniform}(0, 1)$
      Calculate $\alpha_{\min}, \alpha_{\max}$
      while $x_v^{i'}$ unresolved **do**
        if $z < \min \alpha$ **then**
          Accept $x_v^{i'}$, set $X_v^{(t)} = x_v^{i'}$
        end if
        if $z \geq \max \alpha$ **then**
          Reject $x_v^{i'}$, , set $X_v^{(t)} = X_v^{(t-1)}$
        end if
        if update occurs **then**
          Recalculate $\alpha_{\min}, \alpha_{\max}$
        end if
      end while
    end for
  end for

mixing by is sped up by a factor of $\approx n$. Then, if there exists a Metropolis sampler that achieves the optimal mixing time in the order $O(n \log n)$ in the sequential case, the paper's claim is that this sampling scheme takes $O(N/n + \log n)$ time. The first term accounts for communication overhead.

This approach is actually more lightweight than the Exact Asynchronous sampler, but substantially more complex to implement and suffers from all the drawbacks of the Metropolis sampler. Notably, it is incredibly difficult to find a sequential Metropolis sampler that actually achieves optimal mixing time. The first reason is that Metropolis involves discarding proposals. One of the main motivations for Gibbs it ensures an acceptance probability of 1, which generally greatly increases sampling efficiency. Even then, Gibbs still requires Dobrushin's condition in order to ensure optimal mixing. The class of Metropolis samplers that result in optimal mixing is incredibly small. This method also requires a restricted class of Metropolis samplers, as all proposals must be generated *a priori*. This means common proposal strategies, like Gibbs or the random walk Metropolis-Hastings sampler, cannot be used. This makes it unclear in what circumstances this approach is actually an improvement over previous methods.

As such, there are no guarantees for what types of distributions this approach will work on. The only guarantee[2], presuming flawless implementation, it will provide roughly linear speedup compared to the identically-parameterized sequential Metropolis sampler.

5 Comparison

In order to compare how these methods fair in practice, I implemented them all in Python. This has a couple limitations. First, parallelization in Python is generally not a good idea. It's really slow, especially when it is necessary to pass information between workers. The exact asynchronous Gibbs and distributed Metropolis algorithms, which have a pre-determined schedule for resolution, were "sequentialized" by simulating all workers on a single thread. Because of this, the meaning of "time" on the x-axis of charts is a little fuzzy.

To be consistent with most of the literature, I generalize time to "time units," referring to the rate of sequential sampling. For instance, the sequential sampler updates the chain once per time unit, the Hogwild Gibbs sampler with four workers updates the chain four times per time unit, and the chromatic Gibbs sampler updates the chain once per time unit but updates multiple sites.

model	applicable distributions	TV convergence	Mixing Time
Sequential Gibbs	any	Dobrushin's	$\mathcal{O}(n \log n)$
Hogwild Gibbs	any	Dobrushin's	$\mathcal{O}(n \log n)^*$
Exact Asynchronous Gibbs	any	Dobrushin's	$\mathcal{O}(n \log n)^*$
Distributed Metropolis	any	Optimal a priori Metropolis	$\mathcal{O}(\log n)^{**}$
Chromatic	any	Dobrushin's	$\mathcal{O}(\Delta \log n)$
Local Glauber	graph colorings	$k > 2\Delta$	$\mathcal{O}(\log n)^{**}$

Table 1: Use conditions and mixing times for each model discussed in this paper. Mixing times with * are hurt by big $\mathcal{O}(\cdot)$ notation, which hides linear speedup based on number of workers. Mixing times with ** are helped, as incredibly messy constants or restrictive conditions are obscured.

As such, I cannot accurately compare real-time speed of these algorithms. However, it is possible to compare their feasibility on different toy distributions. To see how quickly mixing occurs and how well

[2]It's not actually a guarantee due to the randomness in the Poisson process scheduling, but it's close.

Figure 2: Histogram of the race detection distribution after 10^5 samples. As shown, the Distributed Metropolis and Exact Asynchronous samplers converge to the correct stationary distribution. However, the Hogwild sampler can't avoid race conditions and places nonzero density on (0,0)

sampling methods converge to the joint, I plot the joint distributions and estimates of marginals produced when sampling from three distributions. I omit the chromatic sampler from several charts to save space, as it obviously shares a stationary distribution and convergence rate with Sequential Gibbs for two of the distributions I test.

In all tests, the Hogwild Gibbs sampler uses four workers, the exact asynchronous Gibbs sampler uses three workers, and the distributed Metropolis, Chromatic, and Local Glauber samplers use as many workers as are required.

First, the "race detection" distrubution described in the section on Hogwild Gibbs: $P(0,0) = 0, P(1,1) = P(1,0) = P(0,1) = \frac{1}{3}$. Any sampler than cannot prevent race conditions will place non-zero mass on $(0,0)$.

Second, I attempt to sample from the bivariate normal with mean $\mu == \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. This is chosen to be a simple, easily recognized distribution that clearly violates Dobrushin's condition.

Last, I attempt to sample uniform 5-colorings from a ring with ten nodes. As $5 = 2\Delta + 1$, this meets Dobrushin's condition, which should make it the "easy" distribution to sample from. Every method easily produced a credible joint distribution, so I omit that figure here.

6 Conclusion

So, what method works best?

Ultimately, I find that the local Glauber dynamics and distributed Metropolis sampler both fail to live up to the linear speedup promised in their respective papers. The local Glauber dynamics fails to improve performance in any way when in the setting it was designed for. Finding an optimal Metropolis sampler with *a priori* proposals seems like an incredibly difficult task for all but the most basic target distributions. The Hogwild distribution has obvious problems with race conditions in many instances, preventing it from

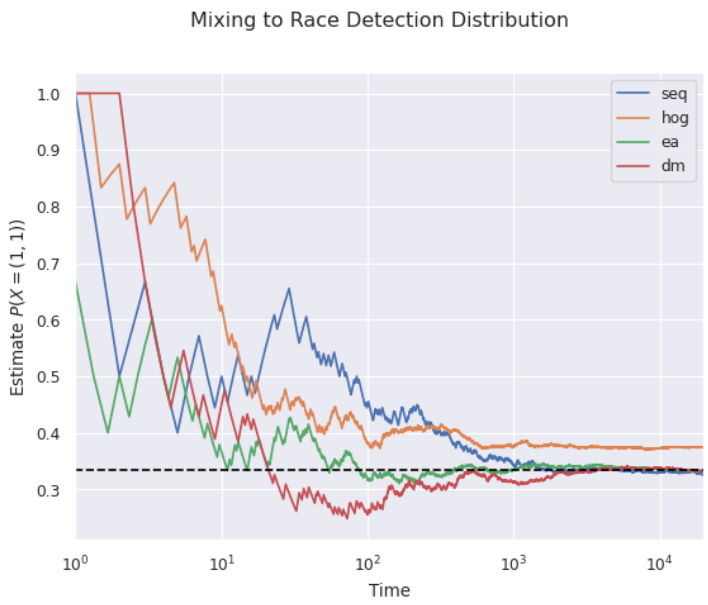

Figure 3: Convergence to race detection distribution. The distributed Metropolis and exact asynchronous Gibbs samplers converge to the true marginal distribution faster than the other approaches. In this setting, they are successfully able to leverage the increased number of samples they generate per timestep. The Hogwild Gibbs sampler fails to converge to the correct marginal.

Figure 4: Histogram of multivariate normal after 10^5 samples. Similar results to the race detection distribution are observed, except now the distributed Metropolis sampler fails to adequately explore the space. This is because it relies on *a priori* sampling, which struggles to provide good proposals.

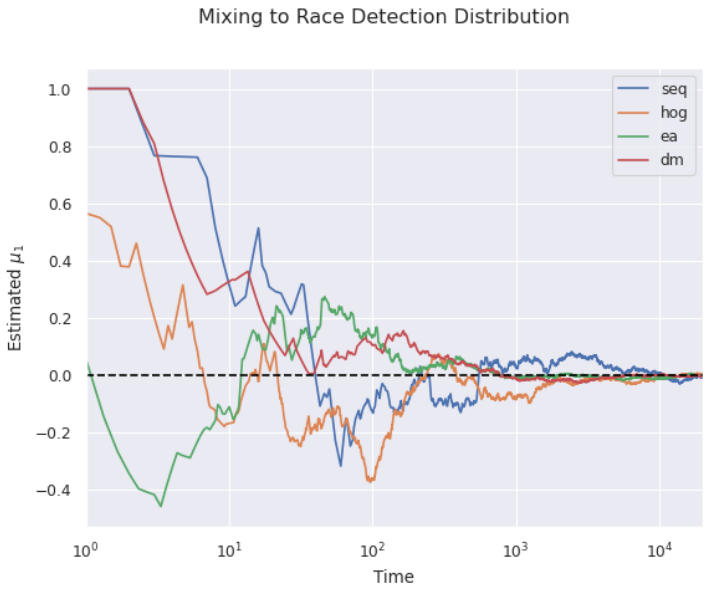

Figure 5: Convergence to mean in multivariate normal. As expected, everything seems to converge faster than sequential Gibbs sampler. The big winner appears to be the exact asynchronous sampler, as it converges fastest to the proper mean while not introducing bias to the joint.

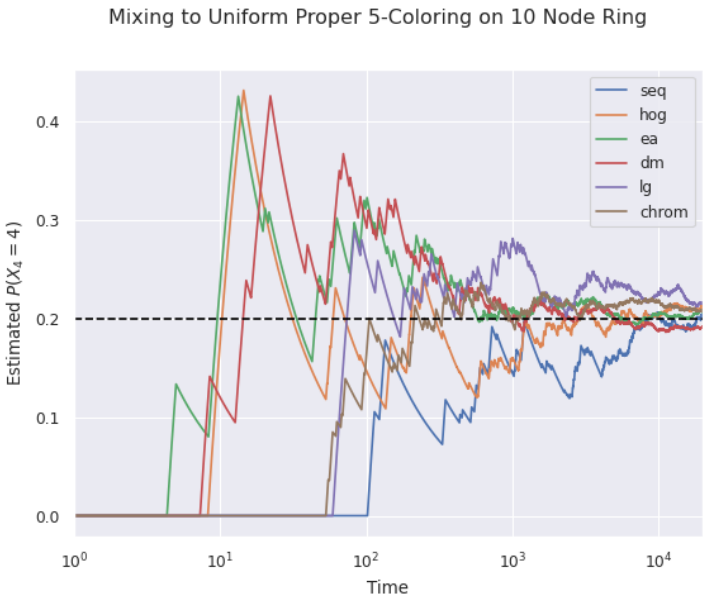

Figure 6: Mixing to uniform proper 5-coloring on ring of 10 nodes. Interestingly, the local Glauber dynamics approach seems to mix at a similar rate to the sequential Gibbs sampler, despite the claims of $\mathcal{O}(\log n)$ mixing. It seems that, again, the exact asynchronous and distributed Metropolis samplers perform best. However, this time they are joined by the chromatic Gibbs sampler, which functions very well as the number of nodes in the factor graph increase.

converging to the target distribution.

Although commenting on speed is difficult given my Python implementations, I am skeptical that the exact asynchronous Gibbs sampler is particularly feasible. The amount of information that needs to be passed between nodes and the hugely increased computational load at each update make it hard to believe that it will often lead to real-time speedups.

That leaves the sequential and Chromatic Gibbs samplers, which I believe are often the best choices. They are computationally efficient, and the chromatic Gibbs sampler scales well with large graphical models without losing any of the nice properties of the sequential Gibbs sampler. The main drawback of Chromatic Gibbs is that it is synchronous and that general optimal coloring is NP-hard. However, in all but the most extreme scenarios these are not serious problems. It's easy to get a general pretty-good coloring, and it's already figured out for common cases.

As of right now, I don't think the wheel has been reinvented yet for the Gibbs sampler. The basic methods practically compare really favorably to fancier methods with fancier theoretical guarantees. To me, it still seems simple is better.

References

[1] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 223–231, Oct. 1997. ISSN: 0272-5428.

[2] C. De Sa, K. Olukotun, and C. Ré. Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4811–4815, Melbourne, Australia, Aug. 2017. International Joint Conferences on Artificial Intelligence Organization.

[3] P. L. Dobruschin. The Description of a Random Field by Means of Conditional Probabilities and Conditions of Its Regularity. *Theory of Probability & Its Applications*, 13(2):197–224, Jan. 1968. Publisher: Society for Industrial and Applied Mathematics.

[4] W. Feng, T. P. Hayes, and Y. Yin. Distributed Metropolis Sampler with Optimal Parallelism, July 2019. arXiv:1904.00943 [cs].

[5] W. Feng, Y. Sun, and Y. Yin. What can be sampled locally? *Distributed Computing*, 33(3-4):227–253, June 2020. arXiv:1702.00142 [cs].

[6] M. Fischer and M. Ghaffari. A Simple Parallel and Distributed Sampling Technique: Local Glauber Dynamics, May 2018. arXiv:1802.06676 [cs].

[7] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin. Parallel Gibbs Sampling: From Colored Fields to Thin Junction Trees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 324–332. JMLR Workshop and Conference Proceedings, June 2011. ISSN: 1938-7228.

[8] T. P. Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 39–46, Oct. 2006. ISSN: 0272-5428.

[9] M. J. Johnson, J. Saunderson, and A. Willsky. Analyzing Hogwild Parallel Gaussian Gibbs Sampling. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[10] A. Terenin, D. Simpson, and D. Draper. Asynchronous Gibbs Sampling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 144–154. PMLR, June 2020. ISSN: 2640-3498.

[11] A. Terenin and E. P. Xing. Techniques for proving Asynchronous Convergence results for Markov Chain Monte Carlo methods, June 2018. arXiv:1711.06719 [cs, stat].