Parallel Gibbs Sampling for Hierarchical Dirichlet Processes via Gamma Processes Equivalence

Dehua Cheng
University of Southern California
Los Angeles, CA 90089
dehua.cheng@usc.edu

Yan Liu
University of Southern California
Los Angeles, CA 90089
yanliu.cs@usc.edu

ABSTRACT

The hierarchical Dirichlet process (HDP) is an intuitive and elegant technique to model data with latent groups. However, it has not been widely used for practical applications due to the high computational costs associated with inference. In this paper, we propose an effective parallel Gibbs sampling algorithm for HDP by exploring its connections with the gamma-gamma-Poisson process. Specifically, we develop a novel framework that combines bootstrap and Reversible Jump MCMC algorithm to enable parallel variable updates. We also provide theoretical convergence analysis based on Gibbs sampling with asynchronous variable updates. Experiment results on both synthetic datasets and two large-scale text collections show that our algorithm can achieve considerable speedup as well as better inference accuracy for HDP compared with existing parallel sampling algorithms.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics

Keywords

Parallel Inference; Hierarchical Dirichlet Process; Topic Model

1. INTRODUCTION

Modeling large, complex, real-world domains often demands powerful models which can handle rich relational structures. Mixture models, such as those to model groups of data with shared characteristics by enforcing a shared set of mixture components, are one of the most intuitive and also effective solutions. For instance, the latent Dirichlet allocation (LDA) model has been proven successful in modeling a collection of documents [4] and the nonparametric extensions with hierarchical Dirichlet processes (HDP) inherit the advantage of LDA and allow the flexibility to learn the number of mixture components automatically from the data [14].

The HDP has achieved success in modeling many different types of data. However, the biggest challenge towards applications is their inability to scale to large datasets. Recently, some excellent work has been conducted to address this challenging problem, which can be summarized into two directions. One is the general-purpose implementation of parallel inference algorithms [7, 8, 9]. These algorithms are general enough to be applied to any type of graphical models, but it is difficult for them to achieve the desired speedup in specific models. The other direction is the approximation or vanilla parallelism of specific models, such as [10, 2, 16] for parallel inference of LDA models. Furthermore, [17] proposes an exact parallel sampling algorithm for HDP, which is the first nontrivial distributed sampling algorithm for HDP that converges to the true distribution provably. However, there is still room for improvement since the algorithm suffers from unbalanced workload and increasing rejection rate of the Metropolis Hasting step with increasing number of processors.

In this paper, we propose a parallel Gibbs sampling algorithm for HDP by exploring its connection with the gamma-gamma-Poisson process, which has a rich potential for developing various parallel inference algorithms. We propose a parallel sampling algorithm based on the augmented gamma-gamma-Poisson process model and reconstruct the dataset using a bootstrap technique which brings the independence across different mixture components. Therefore we can perform independent variable update for each mixture component. We also provide theoretical convergence analysis for the parallel Gibbs sampling algorithm with asynchronous variable updates, i.e., each variable updates without explicit coordination, which is common in a distributed system. We demonstrate the accuracy of our proposed algorithm on the synthetic datasets and faster convergence rate on large-scale real world datasets.

The rest of the paper is organized as follows: we first review the basic ideas in Section 2. In Section 3, we describe our parallel sampling algorithm based on gamma-gamma-Poisson process and discuss theoretical convergence analysis. Finally, we show the experiment results in Section 4.

KDD'14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623708 .

2. BACKGROUND AND NOTATIONS

A *Dirichlet process mixture model* (DPMM) is a mixture model with an infinite number of mixture components, where the *Dirichlet process* (DP), with the base distribution H and concentration parameter α, serves as the nonparametric prior of the mixing measure over all components. Given a dataset $\{x_i\}_{i=1}^N$ of size N, DPMM assumes that the ith data point $x^{(i)}$ is generated from the mixture component $\eta^{(i)}$ as follows:

$$\begin{aligned} G|\{\alpha, H\} &\sim \mathrm{DP}(\alpha, H), \\ \eta^{(i)}|G &\sim G, \\ x^{(i)}|\eta^{(i)} &\sim p(x^{(i)}|\eta^{(i)}), \qquad i \in \{1, \ldots, N\}. \end{aligned}$$

As proved by Ferguson in [5], $G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$ with the discrete support $\{\theta_k\}_{k=1}^{\infty} \subset H$ where H is the space of all mixture components, δ_{θ_k} is Dirac-delta function of component θ_k and π_k is the associated mixture weight. Each $\theta_k \in H$ represents a mixture component, and each mixture component has its distribution over the data space $p(x|\theta_k)$.

In some applications, we may be interested in modeling groups of data with shared mixture components and prior over mixing measures. Therefore the *hierarchical Dirichlet process* (HDP) mixture model is proposed as a hierarchical structural extension of DPMM [14]. That is, given a dataset with groups $X_d = \{x_d^{(i)}\}_{i=1}^{N_d}$, with $d \in \{1, \ldots, D\}$ as the group index, and N_d as the size of dth group, HDP assumes that they are generated as follows:

$$\begin{aligned} G_0|\{\alpha, H\} &\sim \mathrm{DP}(\alpha, H), \\ G_d|\{\gamma, G_0\} &\sim \mathrm{DP}(\gamma, G_0), \\ \eta_d^{(i)}|G_d &\sim G_d, \\ X_d^{(i)}|\eta_d^{(i)} &\sim p(x_d^{(i)}|\eta_d^{(i)}), \qquad i \in \{1, \ldots, N_d\}. \end{aligned}$$

The upper level DP generates G_0 from the mixture component space H as the common base measure to ensure that each group shares mixture components with a positive probability. Then each group is generated from DPMM based on the mixture component space G_0, with independent concentration parameter γ. Topic model is one nice example for HDP, where each latent topic corresponds to the mixture component, the collection of documents is the groups of data, and each data point is a word in the document.

The HDP has achieved success in modeling observations with latent groups in hierarchical structures [14, 13, 12]. However, the computational cost of inference over HDP makes it infeasible for practical applications. Various inference techniques have been investigated to solve this problem, such as variational inference, sampling techniques and so on. Among them, sampling algorithms have become popular due to their simplicity and high inference quality [14]. However, they are also known to suffer from the slow convergence rate. Therefore several parallel sampling algorithms have been proposed to speed up the convergence via parallel computing paradigm.

Slice sampler [15] is one example of parallel sampling algorithm for DPMM. It reduces the infinite mixture component space to a finite subset at each step, where the mixture component assignment for each data point is conditionally independent of the rest and thus can be updated simultaneously. Parallelizing slice sampler is simple but requires multiple times of synchronization within each update, which could lead to significant communication overhead and makes it impractical for the distributed learning scenario.

In [2], an approximation of the HDP is introduced and a parallel sampling algorithm is developed for the approximate model. It divides the data into subsets and each processor synchronously updates the variable through a Gibbs sampler based on the current state of its local dataset and the previous state of the global dataset. The processors communicate with each other asynchronously to ensure the global convergence. This algorithm is well suited for the parallel environment with distributed storage but the inference results may not be as accurate as the original HDP.

The first real-sense parallel sampling algorithm for HDP has been proposed in [17]. It is based on examining an equivalent generative model of HDP with auxiliary variables. Conditioned on the auxiliary variables, each processor can update its local variables independently. Data points are assigned to processors based on the status of auxiliary variables, and each processor learns the mixture component independently. That is, the algorithm implicitly separates the mixture components space to each processor. One major issue with the algorithm is the potential imbalanced workload across processors. In addition, the rejection rate of the Metropolis Hasting step for updating auxiliary variables could be high and thus resulting in slower convergence rate.

As we can see, most existing work on parallel sampling for HDP has limitations either in inference accuracy or convergence rate. Given the large demand in practical applications, seeking an effective parallel sampling algorithm remains an important and challenging task.

3. PARALLEL GIBBS SAMPLING FOR HDP

In this section, we describe in detail our approach: we first review the gamma-gamma-Poisson process and its equivalence to HDP, and then introduce the parallel Gibbs sampling algorithms on the equivalent model.

3.1 Gamma-Gamma-Poisson Process

Introduction.

A *gamma-Poisson process* is a two-level hierarchy of completely random process defined on base measurable space H[19].It is known that a random process Π' drawn from gamma-Poisson process with parameter $\{m, H\}$ is defined as follows:

$$\begin{aligned} G' &\sim \mathrm{GaP}(H), \\ \Pi' &\sim \mathrm{PoisP}(mG'), \end{aligned}$$

where PoisP refers to Poisson process and GaP refers to Gamma process. If H is discrete and $H = \sum_{k=1}^{\infty} \alpha_k \delta_{\theta_k}$, where α_k is the associated atom weight (which becomes the mixture weight in the equivalent mixture model of HDP defined later), the generation process of Π' can also be described as:

$$G' = \sum_{k=1}^{\infty} \pi'_k \delta_{\theta_k}, \;\; \pi'_k \sim \mathrm{Gamma}(\alpha_k, 1), \tag{1}$$

and

$$\Pi' = \sum_{k=1}^{\infty} n_k \delta_{\theta_k}, \;\; n_k|G' \sim \mathrm{Pois}(m\pi'_k). \tag{2}$$

The *gamma-gamma-Poisson process* is defined by replacing the based measure H in *gamma-Poisson process* with another random measure G' drawn from a gamma process $\mathrm{GaP}(\alpha H)$. The equivalence between HDP and gamma-gamma-Poisson process is well-studied in [19] section 3.1. The generative process of both HDP and gamma-gamma-Poisson process are summarized in Table 1. Note that unlike G_i,

Table 1: Summary of the generative process of HDP and gamma-gamma-Poisson processes

HDP	Gamma-gamma-Poisson Process
$G_0\|\{\alpha, H\} \sim \mathrm{DP}(\alpha, H)$	$G'_0\|\{\alpha, H\} \sim \mathrm{GaP}(\alpha H)$
$G_d\|\{\gamma, G_0\} \sim \mathrm{DP}(\gamma, G_0)$	$G'_d\|\{G'_0\} \sim \mathrm{GaP}(G'_0)$
$\eta_d^{(i)}\|G_d \sim G_d$	$\Pi'_d\|\{m, G'_d\} \sim \mathrm{PoisP}(mG'_d)$
$X_d^{(i)}\|\eta_d^{(i)} \sim p(x_d^{(i)}\|\eta_d^{(i)})$	$X_d\|\Pi'_d \sim p(X\|\Pi'_d)$

the sum of the weight of G'_i is no longer 1, so it does not represent a distribution. But since G'_i is proportional to G_i, we can obtain G_i easily by normalization. Furthermore, instead of normalizing the weights explicitly, we can resort to the property of Poisson process that given the sum of several independent Poisson random variables, the Poisson random variables are conditionally distributed as multinomial distributions with the normalized weights. Thus the normalization is achieved implicitly.

An Example of Topic Models.
We take topic models as an example to illustrate the generative process of gamma-gamma-Poisson process. The upper level DP which generates the global topic space G_0 in HDP is replaced by the gamma process with same parameters $G'_0 = \sum_{k=1}^{\infty} \alpha_k \delta_{\theta_k} \sim \mathrm{GaP}(\alpha H)$, where α_k represents the weight for the kth topic. For each document, we first draw the topic mixing parameter $G'_d|\{G'_0\} \sim \mathrm{GaP}(G'_0)$. Similar to eq(1), we can also have $G'_d = \sum_{k=1}^{\infty} \pi'_{dk}\delta_{\theta_k}$, where π'_{dk} is the weight for the kth topic in the dth document. According to G'_d, we use the Poisson process to generate the count measure $\Pi'_d|\{m, G'_d\} \sim \mathrm{PoisP}(mG'_d)$, which can be represented as $\Pi'_d = \sum_{k=1}^{\infty} n_{dk}\delta_{\theta_k}$. Each n_{dk} has its definitive meaning: the number of words that belongs to topic k in document d. Finally, we generate the words for each document according to n_{dk} and the conditional word distribution given the topic. Notice that [19] provides similar derivations except that the last two steps are combined to directly generate n_{dkx}, i.e., the number of word x that belongs to topic k in document d.

3.2 Parallel Gibbs Sampling Algorithm

Motivation.
Even though we establish the equivalence between HDP and gamma-gamma-Poisson process, these two models, however, behave differently. For simplicity, we use the topic models to illustrate the difference. The advantage of gamma-gamma-Poisson process is that for each different topic, the generative process is completely independent with each other in the entire process, which is the merit of composing completely random processes. Note that when we mention the kth topic, we are referring to all variables α_k, θ_k, π'_{dk} and n_{dk} with the same topic index k. This key property lays the foundation of the parallel Gibbs sampling algorithm.

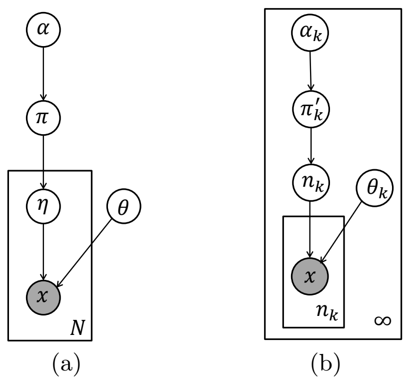


Figure 1: Graphical model representation of topic models by (a) HDP and (b) gamma-gamma-Poisson process for one document.

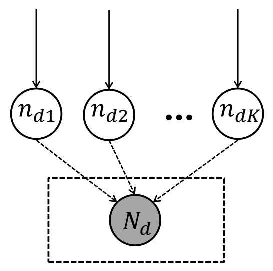


Figure 2: Dependency between n_{dk} and N_d. With N_d observed, $\{n_{dk}\}_{k=1}^{K}$ will form a clique. Otherwise, they are independent with each other.

Another major difference is how we treat the variables n_{dk} and N_d (see in Figure 1). In HDP, n_{dk} does not directly appear in the model, but N_d is given. This agrees better with the real world setting, where the size of the dataset is usually observed. In gamma-gamma-Poisson process, N_d is implicitly determined after we generate n_{dk}, and it cannot be acquired beforehand. This raises one major challenge to parallel inference algorithms because any explicit assumption on N_d will breakdown the cross-topic independence. Therefore we develop a technique that combines bootstrap and Reversible Jump MCMC algorithm to address this particular challenge.

Algorithm.
The inference task of HDP is defined as follows: given the hyperparameters of the model and the observation, how can we infer the parameter θ_k which characterizes the conditional word distribution given the topic as well as the associated topic distribution π_{0k} and $\pi_{dk}, d \in \{1, \ldots, D\}$ for each topic k? The input is a collection of documents. We represent the dth document by X_d and its length by N_d.

Applying the finite approximation on the number of topic K as in [19], we have a much simpler model representation as follows:

$$\alpha_k|\alpha \sim \mathrm{Gamma}(\frac{\alpha}{K}, 1),$$

$$\pi'_{dk}|\{\alpha_k\} \sim \mathrm{Gamma}(\alpha_k, 1),$$

$$\theta_k \sim H_\theta,$$

$$n_{dk}|\{m, \pi'_{dk}\} \sim \mathrm{Pois}(m\pi'_{dk}),$$

$$X_{dk}|\{\theta_k, n_{dk}\} \sim p(X|\theta_k, n_{dk}),$$

$$k \in 1, 2, \ldots, K, \qquad d \in 1, 2, \ldots, D.$$

where H_θ represents the generative model for θ_k, e.g., the Dirichlet distribution.

The joint distribution can be computed as

$$p(n_{dk}, \pi'_{dk}, X_{dk}, \alpha_k, \theta_k) = \prod_{k=1}^{K} \frac{\alpha_k^{\frac{\alpha}{K}-1}}{\Gamma(\frac{\alpha}{K})} e^{-\alpha_k} \tag{3}$$

$$\times \prod_{k=1}^{K} \prod_{d=1}^{D} \frac{\pi_{dk}^{\prime\alpha_k - 1}}{\Gamma(\alpha_k)} e^{-\pi'_{dk}} \frac{(m\pi'_{dk})^{n_{dk}}}{n_{dk}!} e^{-m\pi'_{dk}}$$

$$\times \prod_{k=1}^{K} H_\theta(\theta_k) \times \prod_{k=1}^{K} \prod_{d=1}^{D} \prod_{i=1}^{n_{dk}} p(x_{dk}^{(i)}|\theta_k).$$

Our goal is to design a parallel sampling algorithm which can update each topic and its associated variables asynchronously in parallel. Thus, it is important to analyze variable dependence across topics. From the graphical model in Figure 1(b), we can see that different topics are only connected by their common child nodes x. All θ_k are dependent with each other through x_{dk}, which is necessary for learning topics jointly. But they are independent given the topic assignment for each word, so we can achieve independent update by grouping the word by topic. All n_{dk} for any given d are connected by N_d, as shown in Figure 2. This dependence among n_{dk} is the "side effect" of the new model, which is undesirable and impedes us from developing efficient parallel sampling algorithms. Our solution is to "unobserve" the variable N_d, by constructing a document with flexible length. Details are provided in the sampling step for updating n_{dk}.

We propose the updating rules for α_k, θ_k, π'_{dk} and n_{dk} as follows: (See Algorithm 1 for pseudocode.)

Updating n_{dk}. We update n_{dk} by the Metropolis-Hasting step based on Reversible Jump MCMC with two equally weighted proposed jumps: "$n_{dk} \to n_{dk}+1$"(n_{dk}^{++}) or "$n_{dk} \to n_{dk}-1$"(n_{dk}^{--}). In the likelihood function, the factors regarding n_{dk} (given d and k) are

$$\frac{(m\pi'_{dk})^{n_{dk}}}{n_{dk}!} \prod_{i=1}^{n_{dk}} p(x_{dk}^{(i)}|\theta_k). \tag{4}$$

In addition, with the dataset X_d given, the likelihood function becomes

$$\frac{(m\pi'_{dk})^{n_{dk}}}{n_{dk}!} \prod_{i=1}^{n_{dk}} p_{x_{dk}^{(i)}}(k), \tag{5}$$

where $p_x(k) \propto p(x|\theta_k)$ is the normalized likelihood of topic assignment. Such normalization is equivalent as conditioning on observation X, which is necessary for deriving correct acceptance rate since the length of document is different before and after the jump.

As we can see, the change of n_{dk} also leads to the change of topic assignment in the dth document, which changes n_{d*} in other topics. Given X_d, this operation cannot be conducted within the kth topic. Therefore, as discussed above, we need to construct a new document X'_d of flexible length X'_d based on the original X_d. In this way, we can increase or decrease n_{dk} without affecting other topics directly.

Algorithm 1 Parallel Sampling Algorithm for Gamma-Gamma-Poisson Process

```
 1: Inputs: Group of dataset $X_d$, parameter $\alpha, m$. Number
    of iteration to update $n_{dk}$ in each iteration maxNIter
 2: Outputs: Mixture component $\theta_k$ and corresponding
    normalized weight $\pi_{0k}, \pi_{dk}$
 3: Construct stacks $S_d$ for each dataset $d$
 4: Construct empty collection $X'_{dk}$ and empty buffer $B_{dk}$
    for all dataset $d$ and mixture component $k$, and initialize
    $n_{dk}$ to 0
 5: Initialize $\alpha_k, \theta_k, \pi'_{dk}$ randomly
 6: for each mixture component $k$ asynchronously in parallel do
 7:   repeat
 8:     for each dataset $d$ do
 9:       for nIter from 1 to maxNIter do
10:         Draw $u \sim \mathrm{Uniform}(0, 1)$
11:         if $u < 0.5$ then
12:           Pop $x^*$ from $S_d$ and add it to $X'_{dk}$ with acceptance
              rate $A^{++}_{n_{dk}}$. If failed, add $x^*$ to $B_{dk}$
13:         else
14:           Pop $x^*$ from $X'_{dk}$ randomly and add it to $B_{dk}$
              with acceptance rate $A^{--}_{n_{dk}}$. If failed, add $x^*$
              back to $X'_{dk}$
15:         end if
16:       end for
17:       Push all element from $B_{dk}$ to $S_d$ if $B_{dk}$ exceeds
          certain size.
18:       Update $\pi'_{dk}$ by equation 8
19:     end for
20:     Update $\alpha_k$ by equation 9
21:     Update $\theta_k$ by equation 10, and update partition
        function $Z$.
22:   until convergence
23: end for
24: Normalize $\{\alpha_k\} \to \{\pi_{0k}\}$ and $\{\pi'_{dk}\} \to \{\pi_{dk}\}$
25: Return mixture component $\theta_k$ and corresponding
    weight $\pi_{dk}$
```

First, we build a stack S_d to store X'_d. Each element $x^* \in S_d$ is randomly drawn from X_d with replacement. This ensures that, for all n, the empirical distribution of the first n elements in S_d is an approximation to the empirical distribution of X_d. We also pre-group the elements in X'_d as X'_{dk} by the topic assignment.

When we propose an increase on n_{dk}, we pop a new word x^* from S_d and accept the increase with acceptance rate $A_{n_{dk}^{++}}$ as:

$$A_{n_{dk}^{++}} = min(1, \frac{m\pi'_{dk}}{n_{dk}+1} p_{x^*}(k)). \tag{6}$$

If the proposal is accepted, we will add x^* to X'_{dk} and assign it to the kth topic. Otherwise it returns to S_d.

When we propose a decrease on n_{dk}, we randomly choose one word x^* from X'_{dk}. The acceptance rate $A_{n_{dk}^{--}}$ is

$$A_{n_{dk}^{--}} = min(1, \frac{n_{dk}}{m\pi'_{dk}} \frac{1}{p_{x^*}(k)}). \tag{7}$$

If the proposal is accepted, we will delete x^* from X'_{dk} and return it to the stack S_d.

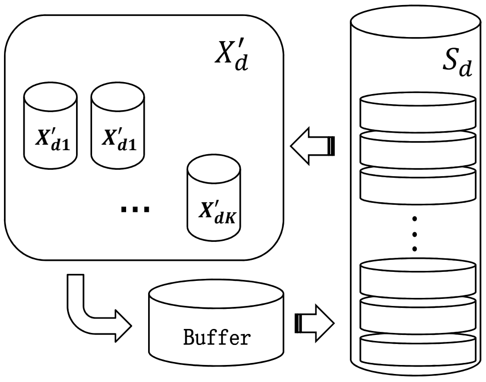


Figure 3: The structure of reconstructed dataset and data flow.

In our implementation, we also add a buffer B_{dk} between X'_{dk} and S_d. The word returning to S_d will be first stored at B_{dk}, and returned to S_d shortly after. This is helpful to avoid consecutive rejections on outliers. m can be empirically set proportional to $1/K$, which increases the acceptance rate. Note that a larger value of m will impede convergence at the initial stage, when n_{dk} is small. The overall flow is illustrated in Figure 3. X'_d serves as an approximation to the original dataset X_d. There might be bias in X'_d due to unassigned but visited elements in stack S_d. The accuracy of this approximation is tested with synthetic datasets in Section 4.1.

This approach appears to be similar to online algorithms, but they are fundamental different: in online settings, we only have one pass of the observations. In our algorithm, although we feed X'_{dk} with the data in stream, the rejected data will return to the stack eventually and similarly for the deleted data from X'_{dk}. This is crucial because it helps to maintain that the empirical distribution of X'_d is close to X_d, otherwise X'_d could be strongly affected by the selection bias during the *add-and-delete* process.

Updating π'_{dk}. In the likelihood function, the factors regarding π'_{dk} are

$$\pi'^{\alpha_k+n_{dk}-1}_{dk} e^{-(m+1)\pi'_{dk}},$$

which means that π'_{dk} follows a gamma distribution with $n_{dk}+\alpha_k$ and $m+1$ as its scale and shape parameter respectively. Therefore, we update π'_{dk} based on n_{dk} and α_k as follows

$$\pi'_{dk} \sim \text{Gamma}(n_{dk}+\alpha_k, m+1). \qquad (8)$$

Updating α_k. In the likelihood function, the factors regarding α_k are

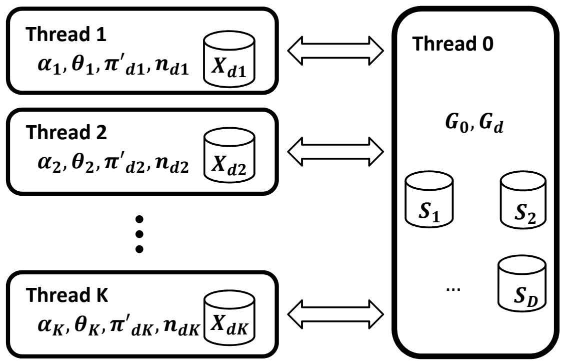
$$\frac{\alpha_k^{\frac{\alpha}{K}-1}}{(\Gamma(\alpha_k))^D} e^{\alpha_k \sum_{d=1}^{D} log(\pi'_{dk})-1}.$$

Because α_k is usually quite small, the first order Laurent expansion provides a simple and accurate approximation, which is $\Gamma(z) \approx 1/z$ when $|z| < 1$. α_k is approximately distributed as

$$\alpha_k \sim \text{Gamma}(\frac{\alpha}{K}+D, \sum_{d=1}^{D} -log(\pi'_{dk})). \qquad (9)$$

Table 2: Rules of update for all variables

Variable	Rule of update
n_{dk}	n_{dk}^{++} with $A_{n_{dk}^{++}}$ n_{dk}^{--} with $A_{n_{dk}^{--}}$
π'_{dk}	$\text{Gamma}(n_{dk}+\alpha_k, m+1)$
θ_k	$H_\theta(\theta_k)\prod_{d=1}^{D} p(X_{dk}\|\theta_k)$
α_k	$\text{Gamma}(\frac{\alpha}{K}+D, \sum_{d=1}^{D} -log(\pi'_{dk}))$



Thread 1
$\alpha_1, \theta_1, \pi'_{d1}, n_{d1}$
X_{d1}
Thread 2
$\alpha_2, \theta_2, \pi'_{d2}, n_{d2}$
X_{d2}
Thread K
$\alpha_K, \theta_K, \pi'_{dK}, n_{dK}$
X_{dK}
Thread 0
G_0, G_d
S_1
S_2
...
S_D

Figure 4: The thread architecture for the proposed parallel sampling algorithm

We can either use this approximation, or treat it as the proposed distribution in a Metropolis-Hasting step.

Updating θ_k. We update θ_k based on its posterior distribution.

$$\theta_k \propto H(\theta_k) \prod_{i=1}^{n_i} p(x_k^{(i)}|\theta_k). \qquad (10)$$

The updating rules for all variables are summarized in Table 2.

3.3 Parallel Structure

As mentioned before, our algorithm updates each topic and its associated variables asynchronously in parallel. Each topic is assigned to a thread. The architecture is showed is Figure 4. Moreover, to minimize the possible conflicts when the same document is accessed by different topics at the same time, we separate the documents to several disjoint subsets at step 8 in Algorithm 1. In each iteration, we update the topic only based on a subset of documents and rotate through all subsets, so the conflict can be avoided completely. Similar techniques have been used by [18] in a parallel GPU sampling algorithm for LDA to reduce parameter storage redundancy and avoid access conflicts.

The only connection across topics is through computing term $p_{x^*}(k)$ when updating n_{dk}. We store the partition function $Z(x^*) = \sum_{k=1}^{K} p(x^*|\theta_k)$ locally, so we can obtain $p_{x^*}(k) = p(x^*|\theta_k)/Z(x^*)$. We update Z periodically to ensure the accuracy.

Because the updates are made asynchronously for each topic, we store each topic update with its time stamp. So we can sort the updates according to its time stamp and reconstruct the parameters for the whole model.

3.4 Convergence Analysis

For Gibbs sampling based parallel sampling algorithm, the variables are updated asynchronously in parallel, which is different from that in sequential Gibbs sampling algorithm, where the order of updates is fixed. The proof of convergence for Gibbs sampler does not generalize to all parallel Gibbs sampling algorithms. Fortunately, for some of them, the order of updates has been proven to be irrelevant to a certain extent. For instance, in chromatic Gibbs sampler [6], all possible orders of updates within the same color are indeed equivalent. In this section, we study the convergence for general parallel sampling algorithm based on Gibbs sampling. There are two differences:

- In a global iteration, some variables have been updated more than once.
- The order of updates can be different in each global iteration.

The term "global iteration" is not well defined in asynchronous settings. Here we use it loosely: we separate the clock time to disjoint intervals. If in each time interval, every random variable has been updated at least once, we call such interval as a global iteration. We also define the order of update as follows:

Definition 3.1 (Order of update)**.** *An order of update is a sequence of index from the index set of random variables, where each index must occur at least once.*

For example, $(1, 2, 1, 3)$ and $(3, 2, 1)$ are legitimate orders of update for the index set $\{1, 2, 3\}$, while $(1, 2, 1)$ is not. We model the order of update by a random distribution $P_{\mathcal{O}}$, which depends on various factors including the algorithm itself, the computation hardware, the probabilistic model (i.e. the graph structure of the graphical model), and current state of the sample. We prove in the following theorem that if the distribution $P_{\mathcal{O}}$ is independent with the current state of the sample, the convergence is guaranteed.

Theorem 3.1. *If a probabilistic model $P(\cdot)$ has positive support and the distribution $P_{\mathcal{O}}$ on order of update is independent with the current state of the sample, the convergence of Gibbs sampler is guaranteed, and it converges to the true distribution $P(\cdot)$.*

Proof. First, we exam the stationary measure of the Markov chain defined by Gibbs sampler with an arbitrary order of updates $\mathcal{O}$, we first prove that, if the probabilistic model $P(\cdot)$ has positive support, the stationary measure is the true distribution $P(\cdot)$. The proof is very similar to that for Gibbs sampler. Assume that we have $x = (x_1, x_2, \ldots, x_n) \sim P(\cdot)$, we update the variables according to the order of updates $\mathcal{O} = (i_1, i_2, \ldots, i_{n_{\mathcal{O}}})$. For the first step, we have

$$\sum_{x_{i_1}} p(x_1, x_2, \ldots, x_n) p(x^*_{i_1} | x_{-i_1}) = p(x_1, \ldots, x^*_{i_1}, \ldots, x_n),$$

where x_{-i} represents the set $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. So we have that $x^{(1)} = (x_1, \ldots, x^*_{i_1}, \ldots, x_n) \sim P(\cdot)$. Similarly, after second step, the result $x^{(2)}$ also has the distribution $P(\cdot)$. By simple induction, we prove that after updating $x_{n_{\mathcal{O}}}$, we have $x^* = (x^*_1, x^*_2, \ldots, x^*_n) \sim P(\cdot)$, which proves the statement. Note that it indicates that the stationary measure is invariant to different order of updates, and it is indeed the true distribution $P(\cdot)$.

With this statement, we then prove our main theorem with help of the main result in [11], which indicates that for an inhomogeneous Markov chain, when there exists a probability measure $P(\cdot)$ such that each of the different steps corresponds to a nice ergodic Markov kernel with stationary measure $P(\cdot)$, it will converge to $P(\cdot)$. We have already proved that $P(\cdot)$ is the stationary measure of any order of updates, so the Markov kernel at each step has the same stationary measure $P(\cdot)$, which proves the theorem. $\square$

The assumption of the independence with the current state of the sample is essential to the conclusion, and similar issue has been discussed in [1] for *adaptive MCMC*. The validity of such assumption relies on the algorithm and its implementation. For example, in LDA, the speed of updating the word distribution for a topic depends on the current number of words assigned to the topic. However, such dependence is not strong enough to affect the convergence of our algorithm, which is confirmed by later experimental results.

4. EXPERIMENTAL RESULTS

In this section, we first study the accuracy of our parallel sampling algorithm (G2PP) on the synthetic dataset, then we test the interpretability of the G2PP on the *Bitcoin Blog* dataset, and finally we compare with with other baseline algorithms on large real world dataset in terms of convergence rate.

We implement the *synchronous Gibbs sampler* (Synch) [2] based on the posterior sampling with auxiliary variable algorithm described in [14], the *AVparallel* algorithm (AVP) based on the *Chinese restaurant franchise* (CRF) with global update scheme proposed in [17] and the *slice sampler* based on [15]. All algorithms are implemented in C++. We test all algorithms under the same hyperparameter setting. Note that the iteration for G2PP and other algorithms are not equivalent, so it is unreasonable to compare the performance based on number of iteration. Therefore, we present our results based on actual runtime.

In our experiments, the convergence rate of AVP is significantly slower than the rest of candidates. The reported convergence time for CRF of a java implementation in the original paper is around 4000 minutes on the same NIPS dataset, which is far longer than the convergence time of other algorithms. The slice sampler also suffers from slow convergence in the preliminary experiments. Note that AVP and slice sampler are theoretically accurate. In contrast, G2PP and Synch are approximate sampling algorithms. Therefore, we only present the results of G2PP, Synch and Gibbs, which is Synch with one processor.

4.1 Inference Accuracy

In order to demonstrate the inference accuracy of G2PP, we design two experiments on the synthetic datasets. The algorithm performance is measured by the prediction accuracy and model perplexity.

Table 3: F1 Scores on mixture of Gaussian dataset by G2PP and Synch

ALGORITHM	F1 SCORE
G2PP	0.94
GIBBS	0.91

Table 4: F1 Scores on LDA dataset by G2PP and Synch

ALGORITHM	F1 SCORE
G2PP	0.65
GIBBS	0.60
SYNCH-2	0.57
SYNCH-4	0.54
SYNCH-8	0.48

4.1.1 *Mixture of Gaussians*

We first generate a synthetic dataset with mixture of Gaussian, which consists of 50 bivariate Gaussian distributions, each with mean distributed according to Norm$(0, 10)$ and variance of 0.01. The dataset contains one million data points in total.

We evaluate the results according to the F1 score between clusters obtained by each algorithm and the ground truth. We adopt the definition of F1 score in [17], which is defined on the pairwise observations. We define the *precision* and *recall* as $|P^{(g)} \cap P^{(p)}|/|P^{(p)}|$ and $|P^{(g)} \cap P^{(p)}|/|P^{(g)}|$ respectively, where $P^{(g)}$ is the set of pairs of data point of the same cluster under the ground truth, and $P^{(p)}$ is all pairs of data points of the same cluster under the prediction. The result is shown in Table 3. The F1 scores shown in the table represent the average prediction results on the Markov chain, excluding the burn-in period. We can see that G2PP yields comparable and even better performance in terms of prediction accuracy than Gibbs sampler.

4.1.2 *Latent Dirichlet Allocation*

We also test our algorithm on a second synthetic dataset generated by latent Dirichlet allocation (LDA). The synthetic corpus contains 1000 documents and 10 latent topics. Each document is of the same length of 1000. The vocabulary size is set to 1000. The concentration parameters of all Dirichlet distributions are set to 1. We compare the performance of G2PP with Gibbs sampler and synchronous Gibbs sampler (Synch) [2] on F1 score and model perplexity.

The F1 scores for each method are listed in Table 4 and the perplexity curves over time is shown in Figure 5. As we can see, G2PP performs better in both F1 score and perplexity. In addition, Synch shows decreasing accuracy when the number of processors increases, because each processor owns smaller subset of data.

In summary, we empirically show that our parallel sampling algorithm not only does not compromise on accuracy, but also may achieve slight improvement in part due to the bias induced by X'_d (which makes it more robust on outliers).

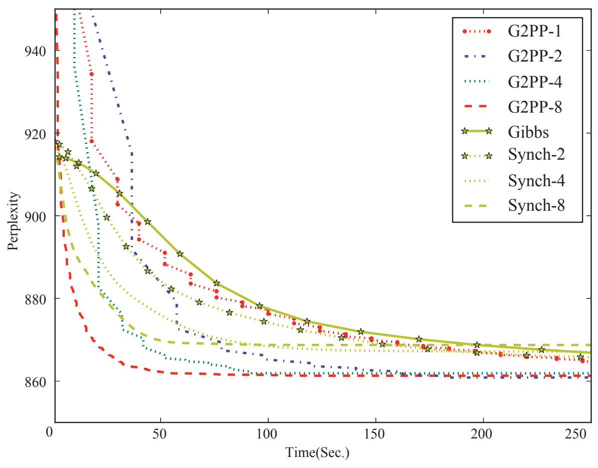


Figure 5: The convergence plot for G2PP and Synch with 1, 2, 4, 8 processors on LDA dataset.

Table 5: Latent topics Inferred by G2PP on Bitcoin dataset

TOPIC 1	TOPIC 2	TOPIC 3	TOPIC 4	TOPIC 5
BITCOIN	LIKE	BITCOIN	GOLD	SECURITY
CURRENCY	PEOPLE	CURRENCY	MARKET	MALWARE
ONLINE	BITCOIN	CHINA	PRICE	DATA
PAYMENT	ONE	BANK	FED	NSA
SAID	SEE	VIRTUAL	YEAR	COMPUTER
DIGITAL	WOULD	CENTRAL	ECONOMY	USERS
FIRST	WAY	EXCHANGE	US	ATTACKS
COMPANY	COULD	PRICE	INVESTORS	INTERNET
TRANSACTIONS	EVEN	CHINESE	BUBBLE	INFORMATION
ONE	RE	SYSTEM	RATE	WASHINGTON

4.2 Interpretability on Real World Dataset

We test our G2PP algorithm on a real world dataset for topic modeling. We collected the online blogs related to the *Bitcoin* posted between Nov. 19th, 2013 and Dec. 20th, 2013 based on Google News search results. *Bitcoin* is a peer-to-peer payment system and digital cryptocurrency introduced as open source software in 2009 by pseudonymous developer Satoshi Nakamoto, which becomes extremely popular in the last few months of 2013.

We preprocessed the dataset in a standard manner. We only consider the blog posts in English, which leads to a collection of 1899 documents with averaged length of 292 and 7379 unique words. The top 5 latent topics and the top 10 words associated with each topic by G2PP is listed in Table 5. We also present the results of Gibbs sampler in Table 6. Both results are intuitive and reasonable. For example, in Table 5, Topic 1 is the introduction of *Bitcoin*, Topic 3 is the events related to China, which is the main cause for both soar and drop in *Bitcoin* price. Moreover, the resemblance between the results from Gibbs and G2PP is remarkable, which indicates that the results of G2PP are as reliable as the Gibbs sampler. Moderate differences in the word and topic ranking are due to the randomness in the inference process.

Table 6: Latent topics inferred by Gibbs on the Bitcoin dataset

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
BITCOIN	ONE	CHINA	YEAR	DATA
CURRENCY	PEOPLE	BITCOIN	MARKET	NSA
EXCHANGE	LIKE	CHINESE	FED	COMPANY
DIGITAL	WOULD	CURRENCY	ECONOMY	INTERNET
VALUE	TIME	BTC	US	SECURITY
VIRTUAL	EVEN	BANK	LAST	GOVERNMENT
SAID	COULD	SAID	GROWTH	SNOWDEN
MONEY	RE	TRADING	MONTH	PRIVACY
ONE	GET	YUAN	WEEK	SOCIAL
WAY	TRANSACTIONS	CENTRAL	INVESTORS	ONLINE

Table 7: Statistics of NIPS and NYT datasets

Dataset	# Documents	Vocabulary Size
NIPS1-17	2484	14036
NYTimes news	300000	102660

With the results on the synthetic datasets and this real-world application dataset, we can safely confirm that the inference results by G2PP is accurate in terms of both statistical criteria and topic interpretation. Next, we investigate the performance of convergence rate.

4.3 Convergence Rate

We choose two benchmark datasets, i.e., NIPS1-17 (NIPS) dataset[1] and NYTimes news (NYT) dataset[2] for convergence analysis. The statistics of the two datasets are listed in Table 7.

4.3.1 NIPS 1-17 Dataset

We split the dataset into a training set with 2284 documents and a test set with 200 documents. We evaluate the algorithm by the perplexity on the test set. The convergence time and final perplexity are listed in Table 8. The typical convergence curves are shown in Figure 6. The result shows that G2PP performs well in terms of convergence speed, accuracy and scalability with respect to the number of processors.

Note that as the number of processors increases, the perplexity value by Synch increases, which means that the accuracy decreases. Moreover, Synch becomes more likely to be trapped in local optimal. By "local optimal", we refer to the phenomenon that the perplexity stably stays at a higher level than the known optimal value before reaching it. For Synch on 16 processors, it is possible that all runs are trapped in local optimal.

4.3.2 NYTimes news Dataset

NYTimes news dataset is a large dataset contains over 100 million words. The inference of HDP on such a dataset is extreme time-consuming and impractical. Efficient inference algorithm on such scale will significantly widen the range of application for the powerful HDP.

We selected a test set of 3000 articles from the dataset and used the rest as the training set. We evaluate the algorithm by the perplexity on the test set. The convergence curves are illustrated in Figure 7. All the results are obtained on 16 processors.

On NYTimes dataset, G2PP can converge in a short time. While there is no good way (except for running the experiment for infinite time) to tell whether Synch finally converged or trapped in "local optimal", it is safe to conclude that G2PP can provide better parameter estimation than Synch within a reasonable time limit.

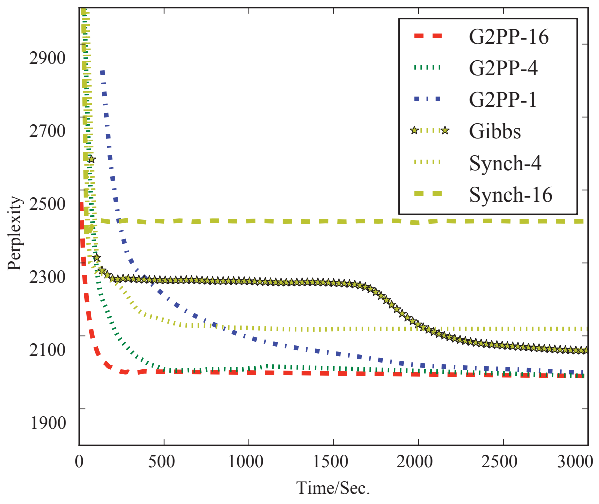


Figure 6: The convergence plot for G2PP and Synch with 1, 4, 16 processors on NIPS dataset.

Table 8: The experiment results of different sampling algorithms on NIPS dataset. Note that the results marked by $*$ represent those didn't converge within limited time.

Algorithm	# Proc.	Conv. time(sec.)	Perp.
G2PP	1	$1467 \pm 345(1164)$	2020
G2PP	4	$481 \pm 101(306)$	2016
G2PP	16	$102 \pm 45(63)$	2004
Gibbs	1	$1920 \pm 880(701)$	2060
Synch	2	$922 \pm 419(368)*$	2080
Synch	4	$519 \pm 331(188)*$	2117
Synch	16	$85.5 \pm 42(49)*$	2413

4.4 Comparison with Subsampling

In section 4.3, the G2PP algorithm performs consistently better than other baselines. But the underlying reason is unclear at this point. The major question is that because the G2PP performs bootstrap on the original dataset, is the speedup merely an outcome of subsampling on the original dataset? As the computation complexity is approximately proportional to the total length of documents, inference on the subsampled dataset should yield competitive speedup at the expenses of inference accuracy. Subsampling on the original dataset also places an unknown effect on the "local

[1]`http://ai.stanford.edu/~gal/data.html`
[2]`http://archive.ics.uci.edu/ml/datasets/Bag+of+Words`[3]

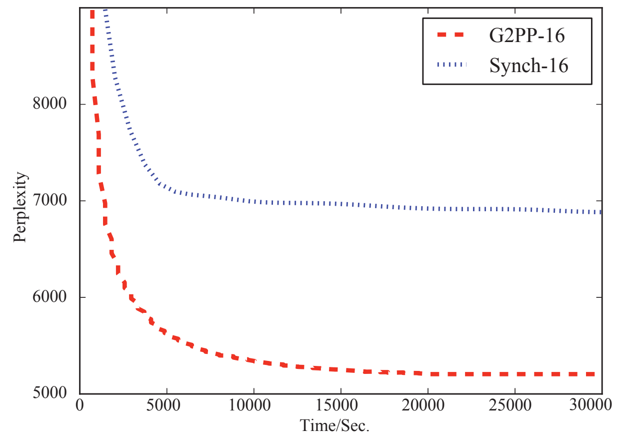


Figure 7: The convergence plot on NYT dataset for G2PP and Synch with 16 processors.

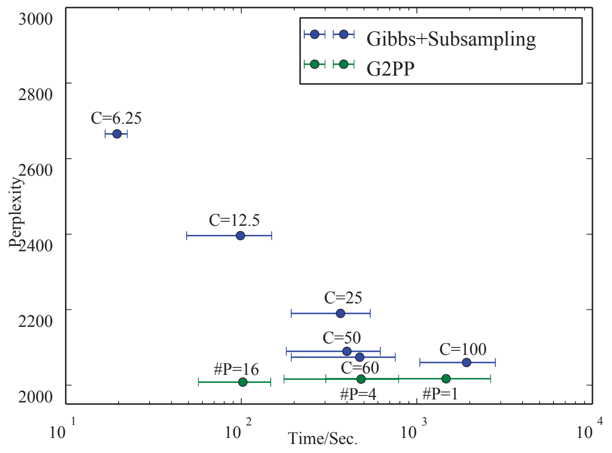


Figure 8: The convergence plot for Gibbs with subsampling on NIPS. $\#P$ denotes the number of processors, and C is the percentage of subsampling.

optimal" phenomenon. We design a series of experiments to answer this question.

In one experiment, we test the speedup and the accuracy of Gibbs algorithm with dataset subsampling. For each run, we first subsample the training set of NIPS dataset to the $C\%$ of its original length, then we conduct the inference only based on the subsampled dataset. The perplexity is tested on same test set as in section 4.3.1. We choose C from $\{6.75, 12.5, 50, 60\}$. The results are illustrated in Figure 8. The results by G2PP on the original dataset are also included for comparison.

As shown in Figure 8, subsampling provides a flexible tradeoff between speed and accuracy. Note that the speedup is not proportional to $1/C$, because subsampling affects both the computational complexity of each iteration and the overall convergence rate. By varying the subsampling rate, we can achieve comparable speedup similar to parallel algorithms with 4 processors on the NIPS dataset. But the accuracy decreases quickly when we further lower the subsampling rate. More importantly, we observe that the G2PP algorithm provides better trade off between accuracy and convergence rate comparing to Gibbs sampling with subsampling. This demonstrates that unique factors other than bootstrapping contribute to the gain in performance by G2PP.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a parallel Gibbs sampling algorithm for HDP based on gamma-gamma-Poisson process. By actively bootstrapping the dataset, we constructed a *new* dataset with flexible size. Together with Reversible Jump MCMC, we proposed a parallel Gibbs sampling algorithm where each mixture component can update in parallel. The proposed algorithm is also suitable for distributed system. We showed its accuracy on synthetic datasets and remarkable speedup comparing with other HDP sampling algorithms on large scale real world dataset. We also examined convergence for parallel Gibbs sampling algorithm with asynchronous variable updates, and we provided the necessary condition to ensure convergence.

For future work, we will further improve the proposed algorithm from the following perspectives: First, we will study the systematic approach for setting parameter m; Second, we will investigate how to choose a dynamic upper bound on the number of mixture components; Lastly, we will examine the generality of the gamma-gamma-Poisson process for parallel inference over other nonparametric mixture models.

6. ACKNOWLEDGMENT

We thank Shang-Hua Teng, Eric Xing, Jun Zhu for discussions, and Xinran He for proof-reading. The research was sponsored by the NSF research grants IIS-1134990, NSF research grants IIS-1254206, and Okawa Foundation Research Award. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agency, or the U.S. Government.

References

[1] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[2] A. Asuncion, P. Smyth, and M. Welling. Asynchronous distributed learning of topic models. *Advances in Neural Information Processing Systems*, 21(81-88):17, 2008.

[3] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.

[6] J. Gonzalez, Y. Low, A. Gretton, C. Guestrin, and U. Gatsby Unit. Parallel gibbs sampling: From colored fields to thin junction trees. *Journal of Machine Learning Research*, 2011.

[7] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *In Artificial Intelligence and Statistics*, Clearwater Beach, Florida, April 2009.

[8] J. Gonzalez, Y. Low, C. Guestrin, and D. O'Hallaron. Distributed parallel inference on large factor graphs.

In *Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, July 2009.

[9] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence*, Catalina Island, California, July 2010.

[10] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 20(1081-1088):17–24, 2007.

[11] L. Saloff-Coste and J. Zúñiga. Convergence of some time inhomogeneous markov chains via spectral techniques. *Stochastic processes and their applications*, 117(8):961–979, 2007.

[12] K.-A. Sohn and E. P. Xing. A hierarchical dirichlet process mixture model for haplotype reconstruction from multi-population data. *The Annals of Applied Statistics*, pages 791–821, 2009.

[13] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing visual scenes using transformed dirichlet processes. *Advances in Neural Information Processing Systems*, 18:1297, 2006.

[14] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476), 2006.

[15] S. G. Walker. Sampling the dirichlet mixture model with slices. *Communications in Statistics-Simulation and Computation*, 36(1):45–54, 2007.

[16] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, pages 301–314. Springer, 2009.

[17] S. Williamson, A. Dubey, and E. P. Xing. Parallel markov chain monte carlo for nonparametric mixture models. In *Proceedings of the 30th International Conference on Machine Learning*, pages 98–106, 2013.

[18] F. Yan, N. Xu, and Y. Qi. Parallel inference for latent dirichlet allocation on graphics processing units. In *Advances in Neural Information Processing Systems*, pages 2134–2142, 2009.

[19] M. Zhou and L. Carin. Augment-and-conquer negative binomial processes. In *Advances in Neural Information Processing Systems*, pages 2555–2563, 2012.