Towards Asynchronous Distributed MCMC Inference for Large Graphical Models

Sameer Singh
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
sameer@cs.umass.edu

Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

1 Introduction

With increasingly cheap availability of computational resources such as storage and bandwidth, access to large amount of data has become commonplace. There is a significant need to organize and extract the *latent* information from such datasets, much of which cannot be achieved by incorporating only local dependencies. Recent work in information extraction, such as [1, 2], represent uncertainty over these large datasets as graphical models. To perform inference over these millions of variables, there is a need to distribute the inference; however the dense, loopy structure with long-range dependencies makes the problem non-trivial.

There has been some recent work in distributed inference for graphical models. Gonzalez et al. [3, 4] distribute Belief Propagation by partitioning the graph. However belief propagation cannot be applied to large dynamic models (such as case factor diagrams), and computing graph-wise computation may be prohibitively expensive. MCMC provides a potential solution, and a few approaches to distributing MCMC have been recently introduced [5, 6]. Along similar lines, considerable work in distributing topic models [7, 8], and recent work has demonstrated impressive scalability [9]. Many of these make strong synchronization assumptions that we do not desire in large-scale models. Gonzalez et al [10] introduce an approach that is partially synchronous that provides linear speedups on a multi-core setup, but requires analysis of global properties of the graph that is not possible in really large, dynamic graphs. Singh et al. [2, 11] propose a Map-Reduce based MCMC that scales, yet can cause performance issues due to synchronization. We are however interested in a distributed, asynchronous approach to MCMC inference.

For really large datasets, the existing literature makes crucial assumptions that restrict scalability. Techniques that rely on multi-core, shared-memory systems are restricted by the local resources and cannot scale with larger number of machines. Methods that do not make these assumptions can easily port between multi-core and multi-node settings. Amongst the distributed approaches, the overheads associated with locking and synchronicity are impractical in very large settings, and asynchronous systems are often simpler and more efficient.

In this work, we explore a number of approaches for distributed MCMC inference for graphical models in an asynchronous manner. The overall architecture consists of inference workers that independently access a data repository to obtain a set of variables and their current values. These workers then perform inference and write back the results to the repository. Although this approach provides more flexibility to the workers, without any "variable ownership" it leads to *conflicts* over values of a variable. We introduce a number of possible resolution strategies, and provide preliminary experiments to study their affect on MCMC convergence.

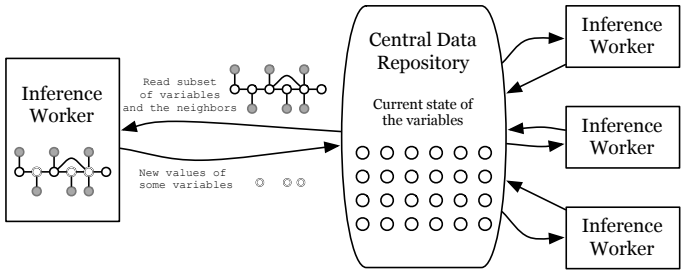


Figure 1: **Distributed Asynchronous MCMC:** The central data repository stores the current state of the variables. The inference workers read the current values of a subset of variables and their markov blanket (shaded circles), and perform inference. The new inferred values (double-stroke circles) are written back to the repository, after conflict resolution.

2 Asynchronous MCMC

We will describe this work in context of *factor graphs* [12], however many of these ideas are applicable to other forms of graphical models. A factor graph represents a probability distribution over variables $\mathbf{Y}$ using a set of factors $\{\psi_i\}$ that define a score over every value assignment to their neighbors $\mathbf{Y}_i$. Computation of the partition function $\mathcal{Z}$ is the main bottleneck for inference, and for large graphs that we are interested in, even computing the model score can be prohibitively expensive.

$$p(\mathbf{Y} = \mathbf{y}) = \frac{1}{\mathcal{Z}} \exp \sum_i \psi_i(\mathbf{Y}_i = \mathbf{y}_i) \tag{1}$$

MCMC inference provides a suitable inference technique for such models. Since the computation of the acceptance score requires the ratio of probabilities, $\mathcal{Z}$ need not be computed. Further, for proposal functions that change a very few variables, only the factors that neighbor the changed variable are needed to evaluate the proposal (see Eq (2) for an example).

$$\frac{p(\mathbf{Y} = \mathbf{y})}{p(\mathbf{Y} = \mathbf{y}')} = \exp \sum_i \psi_i(\mathbf{y}_i) - \psi_i(\mathbf{y}'_i) = \exp \sum_{i, Y \in \mathbf{Y}_i} \psi_i(\mathbf{y}_i) - \psi_i(\mathbf{y}'_i) \tag{2}$$

MCMC inference is trivially parallelized by creating multiple copies of the variables. However this is often not desirable, such as when the number of variables is too large. Further, some tasks use MCMC for MAP inference, for which maintaining multiple configurations is wasteful.

Our proposed architecture for asynchronous MCMC is shown in Fig 1. The variables and their current state is maintained by the data repository[1]. Inference is performed by independent workers that: (1) Request a set of variables, their current value, and the current value of the neighbors, (2) Perform independent inference by proposing changes and evaluating them using the information stored in the worker itself, (3) Use the accepted proposals to write a value back to the repository, and repeat. There is no communication between the workers, allowing the architecture to scale to large number of inference workers.

As long as the variables and the neighborhood that each worker reads is exclusive to the worker for the duration of the inference, the proposals have been evaluated correctly. However, without communication, multiple workers may write to the same variable, invalidating the evaluations made by other workers that are using the obsolete value. The other workers can only detect this when writing to the repository after inference.

3 Dealing with Conflicts

We study several resolution policies in an attempt to better understand these conflicts and their affects on inference. Some of these are more accurate (but slower) than the others; the main question we want to ask is whether, in terms of wall clock running time, is it better to do something approximate quickly ("quantity") or to wait for better "quality" samples.

[1] Note that the central repository is not a *single point of failure*; it may be a distributed file system or database.

3.1 Overwrite

The simplest and the quickest strategy is to simply ignore the changed variables with the assumption that proposals are still informative in the new context. This technique will often write values to the repository that have a low score according to the model.

3.2 Combined Proposal

One simple solution to the problem of new values being low scoring is to evaluate the aggregate change that the worker is proposing. This is equivalent to treating all the changes that the worker accepted as a single combined proposal, and to evaluate it using the new configuration.

3.3 Separate Evaluations

Consider the case where, in the new context, many of the proposals are low-scoring along with a few high quality ones. In the combined proposal method, all of these changes may be rejected, with inference losing out on the high quality samples. Instead, we do a finer-grained search for good proposals. This method will reset the values of the variables to their new values, and iterate through all the proposals to accept/reject them separately. Although this method may find good proposals, it will be substantially slower than the previous ones.

3.4 Restricted to Proposals Neighboring the Conflict

Instead of evaluating all the proposals, the only proposals that will have their scores changed are the ones that either change one of the changed variables, or share a factor with a changed variable. To improve efficiency of the previous approach, we iterate through the all proposals, but only evaluate ones that touch a changed variables. The set of changed variables is updated every time the worker accepts a proposal.

3.5 Resampling the Variables

All of the above approaches take a subset from the proposals, potentially rejecting a few. If all of the proposals are low scoring, then of the worker's accepted changes will be accepted. In order to allow the worker to use this new information, this method takes the changed variables, and its neighbors, and performs Gibbs sampling.

4 Experiments

To evaluate the convergence properties of the various approaches outlined above, we run experiments on synthetic models. Our synthetic model is inspired by the skip-chain models used in information extraction [13]. The model consists of multiple chains of discrete variables, with factors on every variable, and a factor on pairs of neighboring variables in the chain. Additionally we include a number of "skip" factors that lie between pairs of variables across chains (see Fig 2), creating a loopy dense graph for which inference is incredibly expensive. For this paper, we set number of labels to 25, and create 100 chains of 25 variables each. We randomly add skip factors such that on average half of the variables have a skip factor, and set its $\log$ potential to 5. The local $\log$ potentials are uniformly sampled from $(3, 5)$ (with random sign switching), and the transition potentials from $(-5, 5)$. The model and inference algorithms are implemented in FACTORIE [14].

For evaluation, we run 3 single worker Gibbs chains independently till they have converged (by comparing within chain variance to cross chain). We treat the empirical distribution of the samples as the "true" marginals to compare convergence against. We run 3 random initialization chains for each choice of conflict resolution strategy and number of parallel workers. The marginals obtained from each MCMC chain is compared to the true marginals using the L1 distance between the probabilities (total variation loss) and by counting the number of times the max marginal values match.

Fig 3a shows the quality of the marginals versus running time for different numbers of workers. The plot clearly shows that a larger number of workers result in faster convergence to better marginals. This speedup is not linear, and 16 workers do not provide much advantage over 8. To compare the

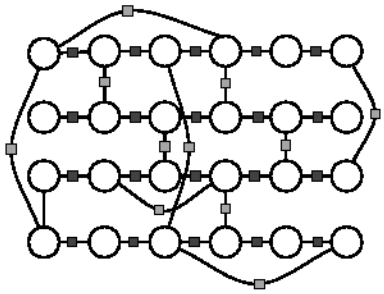
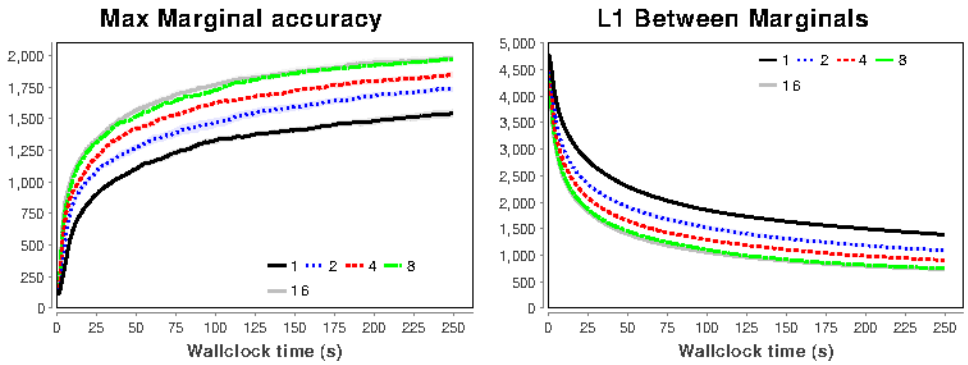
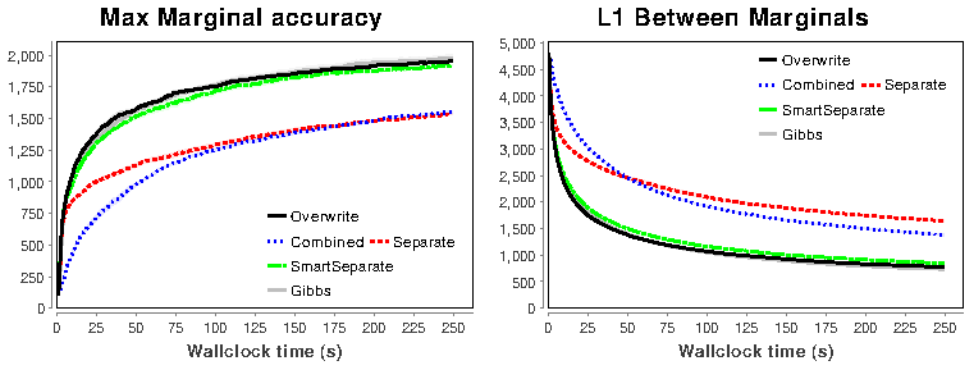


Figure 2: **Linear Chains with Skip factors:** The model that we use consists of *chains* of variable-sand three types of factors. Per-variable local factors (not shown), transition factors (black boxes) that capture dependencies between neighboring variables in the chain, and *skip* factors (gray boxes) that exist between pairs of variables across corpus and act as affinity factors.



(a) Varying the Number of Inference Workers



(b) Varying the Conflict Resolution Policy

Figure 3: **Marginal Accuracy and L1 distance**

various ways to handle conflicts, we examine their convergence for a fixed number of workers in Fig 3b. Combined and Separate perform substantially worse than others on both accuracy and L1, but most surprisingly the simple strategy of Overwrite works as well Gibbs sampling.

5 Discussion

Experiments on synthetic experiments demonstrated several interesting aspects. The most accurate and time consuming method (Gibbs) produces fewer samples yet achieves fast convergence. Overwrite, on the other hand, is very fast and produces lots of presumably low quality samples, yet is able

to reach the same level of performance as Gibbs. Other approaches that lie between the are slow to converge, with the exception of SmartSeparate. These experiments are far from comprehensive, and we will run more experiments for finer analysis of this approach. We also wish to understand the system theoretically to study the convergence bounds.

A number of related avenues remain before this work is applicable to a large-scale system. It is possible to introduce minimal communication between the workers to hasten the conflict resolution. The conflict resolution policy may remain unchanged, however since the worker will be receiving the information faster, fewer proposals will become obsolete, aiding convergence. Similar to [2, 4], we will explore approaches for smarter distribution of the variables that uses the current state of inference to facilitate inference itself.

With the promising convergence speeds achieved when using distributed workers, we feel we are on our way to develop a truly large-scale, asynchronous MCMC inference for graphical models.

References

[1] Michael Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and mcmc. *International Conference on Very Large Databases (VLDB)*, 3:794–804, September 2010.

[2] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Association for Computational Linguistics: Human Language Technologies (ACL HLT)*, 2011.

[3] Joseph Gonzalez, Yucheng Low, Carlos Guestrin, and David OHallaron. Distributed parallel inference on large factor graphs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[4] Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*, 2009.

[5] Finale Doshi, David Knowles, Shakir Mohamed, and Zoubin Ghahramani. Large scale nonparametric bayesian inference: Data parallelisation in the indian buffet process. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 2009.

[6] Feng Yan, Ningyi Xu, and Yuan Qi. Parallel inference for latent dirichlet allocation on graphics processing units. In *Neural Information Processing Systems (NIPS)*, volume 22, pages 1–9, 2009.

[7] D. Newman, P. Smyth, and M. Steyvers. Scalable parallel topic models. *Journal of Intelligence Community Research and Development*, 2006.

[8] Arthur Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Neural Information Processing Systems (NIPS)*, pages 81–88, 2009.

[9] Alexander J. Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *International Conference on Very Large Data Bases (VLDB)*, 3(1):703–710, 2010.

[10] Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *Artificial Intelligence and Statistics (AISTATS)*, Ft. Lauderdale, FL, May 2011.

[11] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Distributed map inference for undirected graphical models. In *Neural Information Processing Systems (NIPS), Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[12] Frank R. Kschischang, Brendan J. Frey, and Hans Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions of Information Theory*, 47:498–519, 1998.

[13] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. 2007.

[14] Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.