SCALING MCMC INFERENCE AND BELIEF PROPAGATION TO LARGE, DENSE GRAPHICAL MODELS

A Dissertation Presented

by

SAMEER SINGH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2014

School of Computer Science

SCALING MCMC INFERENCE AND BELIEF PROPAGATION TO LARGE, DENSE GRAPHICAL MODELS

A Dissertation Presented

by

SAMEER SINGH

Approved as to style and content by:

Andrew McCallum

Andrew McCallum, Chair

Benjamin Marlin, Member

David Jensen, Member

M. Zink

Michael Zink, Member

Carlos Guestrin, Member

Lori A. Clarke

Lori A. Clarke, Chair
School of Computer Science

To Dadaji, my grandfather.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor and thesis chair, Andrew McCallum. It would not be an exaggeration to say that he has been best advisor I could hope for. His infectious optimism seemingly in the face of all evidence to the contrary has helped me through numerous rough patches, whether it was during the moments of self-doubt that every graduate student faces, or when I was ready to reject ideas based on early negative results. To me, Andrew is a role model for many different aspects of being an academician, including how to be an effective researcher, how to be a skillful hacker and a software engineer, understanding the importance of methods that are both practical and principled[1], how to still be an involved family man while being a successful academic, how to write papers that are precise and a pleasure to read, and so on. My evolution from a confused first-year to a focused researcher would not have been possible without Andrew's guidance.

I am also extremely grateful to the other advisors I have had over the years, especially my dissertation committee. Apart from providing valuable technical feedback on early versions of this work, Ben Marlin helped me with my first grant-writing experience, patiently stepping through the sections and providing suggestions on the many versions. David Jensen's insights on empirical evaluation and research methodology had a significant impact on the quality of this thesis. This work benefited greatly from advice regarding the systems aspects of distributed computing that Michael Zink provided. I am further thankful to Carlos Guestrin, who has been an invaluable mentor and collaborator in my current transition from a graduate student to developing my own research agenda. Finally, I feel incredibly fortunate for having worked with Ben Taskar in his final few months, he will forever remain an inspiration and an ideal.

[1] Even though Andrew really does not like this word.

Over the years, I have learned a lot from my colleagues and friends in the lab. Sebastian Riedel and Greg Druck I consider my mentors and friends; their humility never ceases to amaze me, and their advice and constant encouragement has been invaluable. I would like to thank other senior members of the lab, including Kedar Bellare, Pallika Kanani, Laura Dietz, Khashyar Rohanimanesh, David Mimno, and Hanna Wallach, for their helpful discussions. With my friends and contemporaries in the lab, Michael Wick, Limin Yao, and Karl Schultz, I have shared the highs and the many lows of graduate school. I have also benefited from interactions with others in the lab, including Jason Narad, Ari Kobren, David Belanger, Alexandre Passos, Anton Bakalov, Harshal Pandya, Brian Martin, Luke Vilnis, Tim Vieira, Adam Saunders, David Soergel, and Jinho Choi. I am very grateful to Dan Parker and Andre Gauthier, who provided the systems support whenever and whatever I needed. Last, Kate Moruzzi and Leeanne Leclerc have always efficiently handled all the administrative details, and I thank them for their patience for my mistakes.

During summer internships, I was fortunate to have many fruitful collaborations with the industry. I worked with Dustin Hillard (Yahoo!), Amarnag Subramanya (Google), Fernando Pereira (Google), and Thore Graepel (Microsoft) at very different stages of my PhD, and each provided lessons that informed the thesis in very different ways. Collectively, they helped me realize the importance of broader impact and real-world application of academic research, a lesson I hope to imbibe in my future endeavors.

This thesis would not be possible without the support of family and friends. To my wife, Vibhuti, who has been a steady companion for the majority of my life, I am grateful for always knowing and providing exactly what I need, and I hope to spend the rest of my life returning the favor. I am also thankful to my brother Kartik and my mother for their care and support. Friends in Amherst, Parthsarthi Valluri, Bhooshan Popere, Manjunath Narayan, Manjunatha Jagalur, Moe Mattar, Vimal Mathew, and Gaurav Chandalia, helped me in many different ways, and made my time at UMass a very enjoyable and pleasant one.

Finally, I would like to acknowledge my funding sources. I have been fortunate to receive the Yahoo! Key Scientific Challenges award, Yahoo! Search and Mining award, UMass Graduate

School fellowship for continuing students, and the Facebook PhD fellowship runner-up award. Further, this work was supported in part by the Center for Intelligent Information Retrieval, in part by DARPA under agreement number FA8750-13-2-0020, in part by IARPA via DoI/NBC contract #D11PC20152, in part by IARPA and ARFL contract #FA8650-10-7060, in part by Army prime contract number W911NF-07-1-0216 and the University of Pennsylvania subaward number 103-548106, in part by SRI International subcontract #27-001338 and ARFL prime contract #FA8750-09-C-0181, in part by NSF grant #CNS-0551597, and in part by Lockheed Martin through prime contract #FA8650-06-C-7605 from the Air Force Office of Scientific Research. Any opinions, findings and conclusions or recommendations expressed in this thesis are those of the author and do not necessarily reflect those of the sponsors.

ABSTRACT

SCALING MCMC INFERENCE AND BELIEF PROPAGATION TO LARGE, DENSE GRAPHICAL MODELS

MAY 2014

SAMEER SINGH

B.E., UNIVERSITY OF DELHI

M.Sc., VANDERBILT UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McCallum

With the physical constraints of semiconductor-based electronics becoming increasingly limiting in the past decade, single-core CPUs have given way to multi-core and distributed computing platforms. At the same time, access to large data collections is progressively becoming commonplace due to the lowering cost of storage and bandwidth. Traditional machine learning paradigms that have been designed to operate sequentially on single processor architectures seem destined to become obsolete in this world of multi-core, multi-node systems and massive data sets. Inference for graphical models is one such example for which most existing algorithms are sequential in nature and are difficult to scale using parallel computations. Further, modeling large datasets leads to an escalation in the number of variables, factors, domains, and the density of the models, all of which have a substantial impact on the computational and storage complexity of inference. To achieve scalability, existing techniques impose strict independence assumptions on the model, resulting in tractable inference at the expense of expressiveness, and therefore of accuracy and utility, of the model.

Motivated by the need to scale inference to large, dense graphical models, in this thesis we explore approximations to Markov chain Monte Carlo (MCMC) and belief propagation (BP) that induce dynamic sparsity in the model to utilize parallelism. In particular, since computations over some factors, variables, and values are more important than over others at different stages of inference, proposed approximations that prioritize and parallelize such computations facilitate efficient inference. First, we show that a synchronously distributed MCMC algorithm that uses dynamic partitioning of the model achieves scalable inference. We then identify bottlenecks in the synchronous architecture, and demonstrate that a collection of MCMC techniques that use asynchronous updates are able to address these drawbacks. For large domains and high-order factors, we find that dynamically inducing sparsity in variable domains, results in scalable belief propagation that enables joint inference. We also show that formulating distributed BP and joint inference as generalized BP on cluster graphs, and by using cluster message approximations, provides significantly lower communication cost and running time. With these tools for inference in hand, we are able to tackle entity tagging, relation extraction, entity resolution, cross-document coreference, joint inference, and other information extraction tasks over large text corpora.

CONTENTS

Page

CHAPTER

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Data, data, data! I cannot make bricks without clay!

Sherlock Holmes

With increasingly cheap availability of computational resources such as storage and bandwidth, access to large datasets in many application areas of machine learning has experienced explosive growth. Single processor speed, however, has not followed a similar trend, and instead multi-core and distributed architectures have progressively become the computational platforms of choice. These trends have led to an interesting dilemma; the ever-growing datasets need to be analyzed, yet machine learning algorithms that are traditionally sequential do not translate to the multi-core setting, and thus, are likely to become obsolete.

Inference in large-scale graphical models is one area of machine learning that has been particularly affected. Graphical models are used to compactly represent complex distributions defined over many random variables, and have been widely used in a number of application domains such as natural language processing, computer vision, computational biology, medical diagnosis, and social network analysis. For large datasets of our interest, capturing the complex global dependencies in the probability distribution leads to a dense graph defined over a large number of variables, each of which can take values over a large domain. These characteristics lead to intractable exact inference, and to approximate inference algorithms that do not scale to such models due to their inherent sequential nature. Thus, most existing approaches impose strong restrictions on the model, often limiting themselves to small independent models over subsets of data that cannot capture global dependencies. There is a significant need for inference techniques for complex graphical models that can scale to very large datasets, without sacrificing on the representation capability of dense models.

In this thesis, we explore approaches for approximate inference that scale to such large, corpus-level graphical models using distributed and parallel processing. Our central hypothesis is that by exploiting sparse structures that emerge dynamically in the posterior distribution during the iterative inference, we can obtain significantly improved parallel scalability for Markov chain Monte Carlo (MCMC) and Belief Propagation (BP) approaches. For MCMC, we find that a distributed algorithm that operates on partitions of the model, using the state of inference to discover fruitful partitions, can obtain significant speedups for large models. We show that asynchronous updates along with lock-free distribution address the drawbacks of synchronous communication. For belief propagation (BP), dynamically identifying values of the variables that are relevant for inference results in an accurate, *anytime* variant of message passing. We also find that compressing the probability distribution for communication between the computing nodes allows for a trade-off between accuracy and message size, enabling faster convergence with fewer bits of communication.

1.1 Application to Information Extraction

We are especially motivated by information extraction tasks that have become increasingly relevant with the large amounts of text easily accessible for analysis (see, for example, Sandhaus [2008] and Common Crawl Foundation [2011]). Given these large text corpora, there is a significant need to organize and extract the *latent* information in these datasets. For the newswire corpus, for example, this may involve extracting all the people, organizations and locations that appear in the articles, and identifying the various relations between these entities. For a collection of scientific publications, we would like to deduplicate the papers, construct the citation graph, identify the set of authors, and determine the co-authorship trends. This task of extracting *structured* representations from unstructured or semi-structured text is known as information extraction, and has multiple downstream applications, such as in search, bibliometrics, question answering, and recommendation systems.

Due to widespread application, information extraction has been an active area of research for many years. Significant progress has been made in designing accurate *sentence*- and *document*-

level models of these tasks, for example named-entity recognition [Finkel et al., 2005], coreference [Haghighi and Klein, 2009], and relation extraction [Bunescu and Mooney, 2007]. For constructing a corpus-wide repository of the extracted information, these sentence- and document-level predictions are often followed by an aggregation step, including heuristic record linkage to remove duplicates.

To reduce the error in the predictions further, there is a need to represent the long-range dependencies in the data. For example, identifying whether a noun phrase refers to an entity can be greatly aided by looking at all the other noun phrases that refer to the entity, thereby introducing dependencies between all the noun phrases in the corpus [Haghighi and Klein, 2010, Singh et al., 2011b]. Similarly, for identifying whether a particular relationship is expressed between two entities, it may be useful to examine all the sentences in which the two entities are referred [Yao et al., 2010, Hoffmann et al., 2011]. These examples suggest that to achieve further accuracy gains, it may be necessary to go beyond the local per-sentence or per-document models, and perform inference at the *corpus*-level. Further, recent work has shown the utility of *joint inference* of multiple information extraction tasks [Poon and Domingos, 2007, Finkel and Manning, 2009, Yu and Lam, 2010, Kate and Mooney, 2010, Singh et al., 2009]. Both corpus-level dependencies and joint inference results in complex models that have a high-density of edges, a large number of variables, and variables with large-domains, and thus are an appropriate evaluation of scalable inference.

1.2 Thesis Contributions and Key Findings

Thesis Statement: *A combination of distributed computing and dynamic sparsity-based approximations for MCMC and belief propagation enables inference to scale to complex joint distributions over millions of random variables.*

Parallel and distributed processing for machine learning has recently gained significant interest. In this thesis, we focus on large and complex graphical models that span over many variables forming a single, dense connected component, as opposed to a majority of existing approaches that perform efficient distribution of computations for parameter estimation (*learning*) for a large

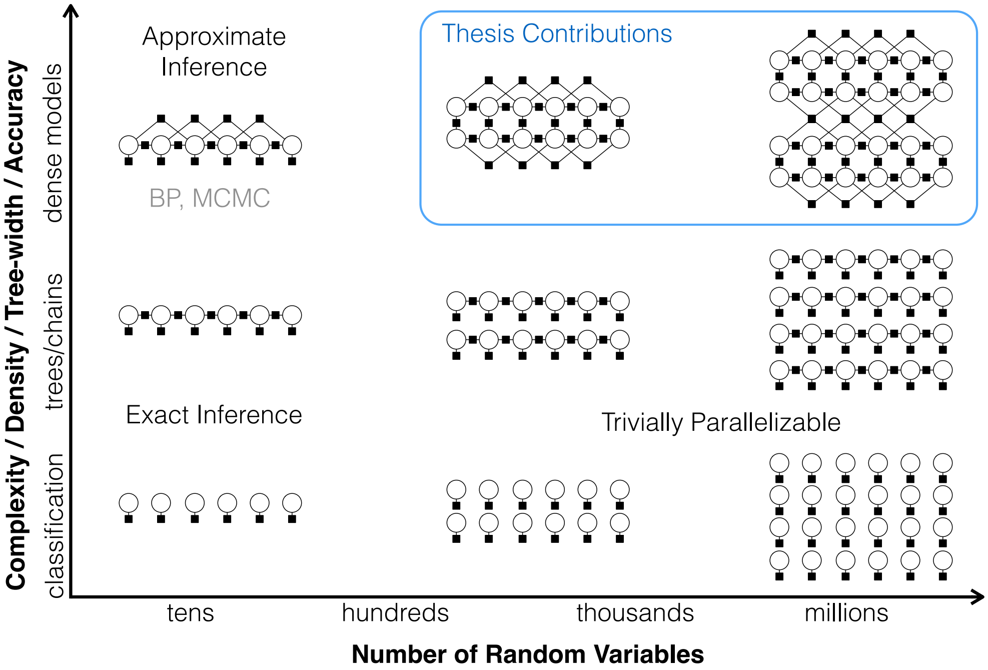


Figure 1.1: **Landscape of Challenges in Scaling Inference:** Tasks may be defined as *large* with respect to the number of variables (X-axis), or with respect to connectivity of the model (Y-axis). For models that are considered large according to only one of these criteria, inference can scale quite efficiently; we can use approximate inference for small, dense models (top-left), or trivially parallelize exact inference for many independent models (bottom-left). Inference problems we focus on, however, deal with large and dense models (top-right).

number of smaller datasets/models. The main motivation for our proposed approximations and distributed processing is the complexity of inference for such models, and we do not claim that our models will not fit in memory, but instead claim that inference for such models will be incredibly inefficient on a single machine; in other words, we are CPU-bound, not memory-bound. Further, we confine our contributions to extensions to MCMC and belief propagation; although these two approaches combined encompass a large majority of approximate inference approaches, other existing techniques may be more efficient than MCMC or belief propagation, or our extensions thereof, for a specific application or model. For the underlying computation architecture, we focus on multi-core and multi-node systems, avoid the shared-memory assumptions as much as possible, and do not address graphics processing units (GPUs), remote direct memory access (RDMA), or other alternatives. Finally, the evaluation in this thesis is primarily through empirical comparison on a large number of applications, and the theoretical analysis is not comprehensive for all our contributions.

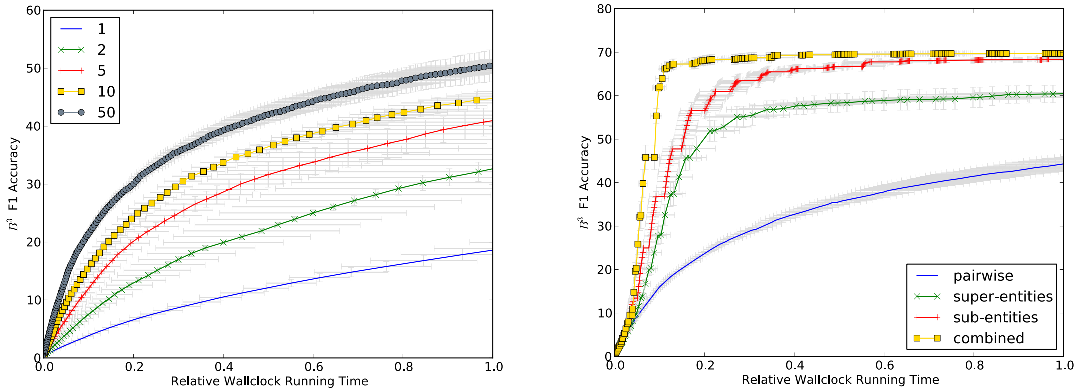
We describe the landscape of challenges for scaling inference in Fig. 1.1, illustrating the difficulty of inference for large, dense graphical models that we focus on. Following, specifically, are the main technical contributions and key results of this work:

1.2.1 Synchronously Distributed MCMC

Motivated by the many advantages of Markov Chain Monte-Carlo (MCMC) inference methods and by the popularity of Map-Reduce, we explore synchronous distribution for MCMC. The graphical model is partitioned into sub-graphs that are sampled independently by inference workers (in a “divide and conquer” manner), followed by combining the model and repeating the steps. We find that naive random partitioning imposes strict restrictions on sampling, leading to sub-optimal scaling. We show that uninformed partitioning of the model yields valid mixing of the resulting MCMC chain, and provides speedups with multiple machines on large-scale models (see Fig. 1.2a). Further, constructing a dynamic, informed partitioning scheme based on a latent variable model leads to fruitful partitions and high-quality samples. In particular, these informed partitions result in a much faster rate of convergence for distributed MCMC, as shown for large-scale entity resolution in Fig. 1.2b.

1.2.2 Asynchronous Updates for MCMC

We observe that the synchronously distributed architecture for sampling contains two major drawbacks: synchronous communication bottleneck and restrictions imposed on the proposal function. We investigate a number of approaches that collectively utilize asynchronous updates to address these concerns. First, we consider approximate sampling that uses stochastic evaluation of the proposals which relaxes the constraints imposed on the proposals. The two strategies for approximating the model, uniform and confidence-based, provide substantial speedups over regular MCMC on a large-scale entity resolution model, as shown in Fig. 1.3a. Second, we find that an asynchronous distributed architecture that uses cheap, central locking on variables guarantees valid



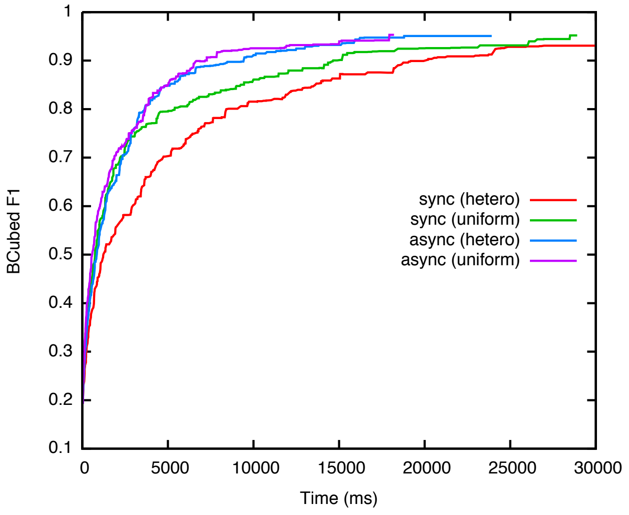
(a) Speedups for Uninformed Partitions

(b) Improvements from Informed Partitions

Figure 1.2: **Synchronous Distributed MCMC:** A synchronously distributed MCMC algorithm used for large-scale cross-document coreference, implemented on a Map-Reduce architecture. *Super-entities* and *sub-entities* refer to dynamically inferred partitions and block moves.

Method	**Speedup**
Baseline	1x
Uniform	
$p = 0.5$	2.02x
$p = 0.2$	4.26x
$p = 0.1$	6.77x
$p = 0.02$	9.78x
Variance	
$i = 0.00001$	0.96x
$i = 0.1$	1.38x
$i = 1$	5.26x
$i = 10$	7.76x
$i = 100$	13.16x

(a) Stochastic MCMC



(b) Comparison to Synchronously Distributed MCMC

Figure 1.3: **Asynchronous MCMC:** On a large-scale entity resolution task, we obtain impressive speedups when stochasticity is introduced in MCMC as shown in (a). In (b), we compare the asynchronous distributed MCMC with the synchronous variant on homogeneous (uniform) and heterogeneous workers.

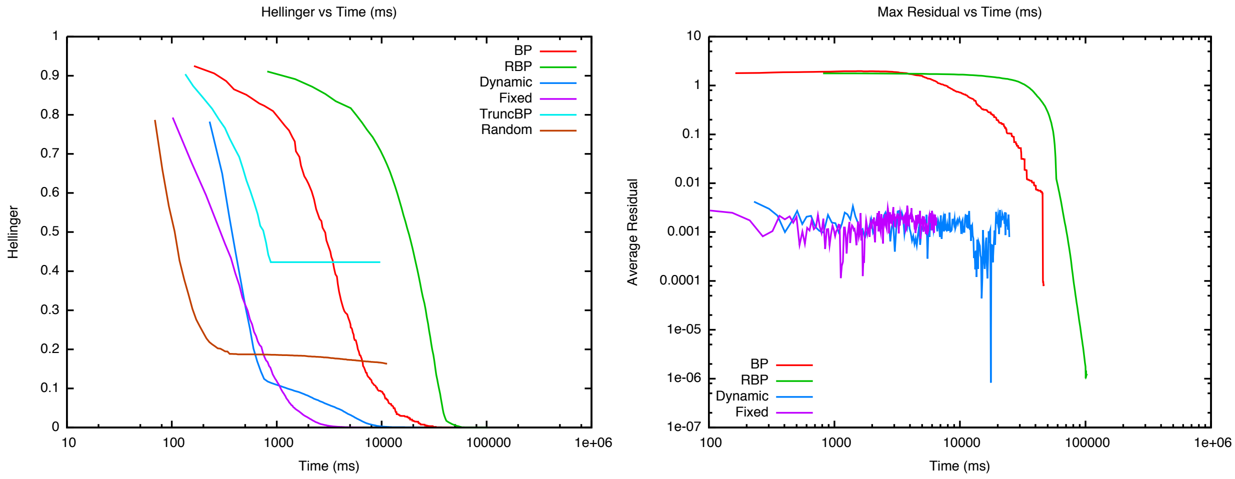
Model	**Tagging**	**Relations**			**Coreference**	
	Accuracy	**Prec**	**Recall**	**F1**	**PW F1**	**B^3 F1**
Tagging → Coreference, Relations	80.23	53.22	54.92	54.05	53.94	76.34
Tagging + Coreference	81.24	-	-	-	57.59	78.06
Tagging + Relations	81.77	54.93	54.02	54.47	-	-
Tagging + Relations + Coreference	**82.69**	56.06	54.74	**55.39**	**58.39**	**78.50**

Table 1.1: **Single-Value Sparsity:** Evaluation on joint inference of entity tagging, relation extraction, and coreference. The baselines models consist of independent models for tagging, and models for relations and coreference condition on the output of tagging (denoted using $\rightarrow$). Our contribution consists of three joint models, denoted by $+$.

MCMC chains, at the same time successfully addressing the bottlenecks of the synchronous communication (see Fig. 1.3b for an example). Finally, the architecture that drops the central locking requirement leads to approximate MCMC chains on overlapping variables that are able to utilize parallelism for faster convergence.

1.2.3 Value Sparsity for Efficient Belief Propagation

Belief propagation is not efficient on models that contain variables with large domains and high-order factors since the message computation is polynomial in the size of the domain and exponential in the size of the factor neighborhood. We first explore an approach to address these bottlenecks by dynamically inducing single-value sparsity on variables during the course of inference by examining their marginals. Reducing domains to a single value induces determinism in the model, which we find facilitates efficient, accurate, and parallel inference for complex models such as joint information extraction (as shown in Table 1.1). We also find that an anytime variation that incrementally grows sparsity achieves consistent marginals during the course of inference, eventually reaching the fixed point of belief propagation with full variable domains. Figure 1.4 presents the runtime properties of the marginals on grid models, demonstrating low error and high-consistency through the course of inference.



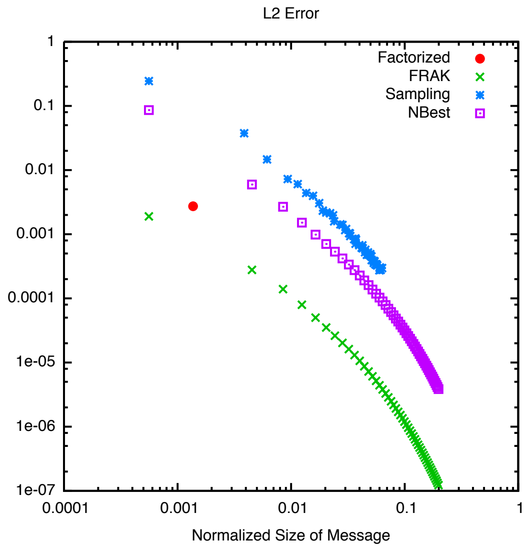
(a) Error in the Marginals (lower is better)

(b) Consistency of the Marginals (lower is better)

Figure 1.4: **Anytime Belief Propagation:** Comparison of our proposed approaches (Fixed and Dynamic) with BP, Residual BP, Random growth of domains and fixed Truncated domain on grid models. (a) Demonstrates low marginal error through the course of inference, and (b) shows average message residual, indicating our marginals have a local consistency throughout inference.

1.2.4 Approximate Cluster Messages for Generalized Belief Propagation

Loopy belief propagation for large, dense models requires a large iterations of message passing to converge, and further, may converge to a poor minima, resulting in an expensive overhead in communication (in distributed inference) and computation (in joint inference). By formulating joint inference and distributed inference as generalized BP on a cluster graph, and by using approximate messages between cluster variables instead of full messages, we can perform inference efficiently on models with large number of variables and domains. We find that the cluster message approximations can trade-off accuracy with communication/computation, can utilize substructures for additional savings, and can be marginalized more efficiently than BP messages. We present the accuracy and size of single cluster message approximations in Fig. 1.5a, demonstrating the gradual trade-off between storage and accuracy, in particular suggesting that the approximation that combines sparse and merged messages, FRAK, provides accurate representation with a small amount of storage. From evaluation on a joint citation segmentation and resolution task presented in Fig. 1.5b, it is also apparent that FRAK outperforms sampling and k-Best approximations.



(a) Error vs Size of Cluster Messages

Method	F1	Time
k-**Best**		
$k = 1$	59.3	0.08
$k = 2$	57.6	0.16
$k = 5$	63.3	0.4
$k = 10$	64.8	0.8
$k = 50$	69.3	4.0
$k = 100$	69.9	8.0
$k = 150$	70.3	12.0
Sampling		
$s = 25$	79.4	3.1
$s = 100$	79.5	10.0
FRAK	**80.7**	0.16

(b) Joint Inference Accuracy

Figure 1.5: **Cluster Message Approximations**: (a) we compute the approximation error and message sizes of a number of proposed approximations to the fully-specified cluster messages (which has 0 error and 1 relative size), and (b) an application to joint citation resolution and extraction demonstrating our approach (FRAK) obtains a higher accuracy than k-Best and sampling, while being much faster (BP is intractable for this task).

1.3 Declaration of Previous Work and Collaborations

Below, the work described in this thesis that have been published or are collaborations with other researchers is enumerated. I am the lead contributer to all of the work presented in the thesis, and all contributions have been under the guidance of, and in collaboration with, **Andrew McCallum**.

- Synchronous distributed MCMC (Chapter 4) was initially conceived as a summer intern at Google Research, under the mentorship of **Amarnag Subramanya** and **Fernando Pereira**, and was presented at the NIPS Workshop on Learning on Cores, Clusters and Clouds [Singh et al., 2010] and published in the Annual Meeting of the Association of Computational Linguists (ACL) [Singh et al., 2011b].

- Stochastic proposal evaluation for MCMC (Section 5.1) is in collaboration with **Michael Wick**, and published in the Conference on Empirical Methods for Natural Language Processing (EMNLP) [Singh et al., 2012b] and presented at the NAACL 2012 Workshop on Automated Knowledge Base Construction (AKBC-WEBEX) [Singh et al., 2012c].

- Asynchronous lock-free, distributed MCMC for marginal inference (Section 5.4) was presented at the NIPS Big Learning workshop [Singh and McCallum, 2011].
- Asynchronous distributed MCMC for MAP inference (Section 5.3) is used for inference by, but is not the focus of, Wick et al. [2013].
- Value sparsity for multi-core belief propagation (part of Chapter 7) is a collaboration with **Brian Martin**, and was presented at the NIPS Workshop on Computational Trade-offs in Statistical Learning (COST) [Singh et al., 2011a].
- Value sparsity for joint inference of relations, types, and coreference (Chapter 7) is joint work with **Jiaping Zheng**, **Brian Martin**, and **Sebastian Riedel**, and was published in the CIKM Workshop on Automated Knowledge Base Construction (AKBC) [Singh et al., 2013a].
- Sparsity in values for anytime belief propagation (part of Section 7.2) is a collaboration with **Sebastian Riedel**, and was presented at the NIPS Workshop on Resource Efficient Learning [Singh et al., 2013b].
- Analysis of approximate cluster messages for joint inference (Chapter 8) is a collaboration with **Sebastian Riedel**, and is currently under preparation for submission.

1.4 Thesis Outline

In the next chapter, we introduce probabilistic graphical models, available architectures for distributed computing, and applications of large graphical models to information extraction.

The remainder of the thesis is organized in two parts. The first part describes our contributions to Markov chain Monte Carlo (MCMC) methods. We provide an overview of the existing work on MCMC in Chapter 3, introducing synchronous distribution in Chapter 4 and asynchronous sampling techniques in Chapter 5. We address belief propagation (BP) in the second part. Chapter 6 introduces belief propagation and its variants, followed by our proposed work on sparse belief propagation in Chapter 7 and generalized belief propagation with approximate cluster messages in Chapter 8.

Chapter 9 concludes this thesis with a review of the limitations of this work, and proposes a number of avenues for future work to address these drawbacks.

CHAPTER 2

BACKGROUND

It is a very sad thing that nowadays there is so little useless information.
Oscar Wilde

In this chapter we provide introductory material on probabilistic graphical models, architectures for distributed processing, and applications of large, dense graphical models to information extraction. We briefly describe the MCMC inference and belief propagation algorithms here, but since much of the thesis focuses on extensions to the two techniques, we provide detailed background and corresponding literature survey in Chapters 3 and 6 respectively.

2.1 Probabilistic Graphical Models

This section provides a brief introduction to the graphical model formulation for representing probability distributions over a large number of variables. We also establish the notation for the rest of this thesis.

2.1.1 Random Variables

Probabilistic graphical models provide a way to represent a joint probability distribution over a fixed set of random variables ($\mathbf{Y}$). In this work, we constrain ourselves to discrete variables. We represent each random variable by $Y_i \in \mathbf{Y}$ and each value of Y_i by $y_i \in \mathcal{Y}_i$, where the set of values a variable can take, $\mathcal{Y}_i$, is also known as the domain of the variable. A subset of variables is represented by $\mathbf{Y}_c \subseteq \mathbf{Y}$ and an assignment to the variables $\mathbf{Y}_c$ by $\mathbf{y}_c \in \boldsymbol{\mathcal{Y}}_c$. Note that the domain $\boldsymbol{\mathcal{Y}}_c$ is the cross-product of the domains $\mathcal{Y}_j, \forall j, Y_j \in \mathbf{Y}_c$. When multiple variables are defined over the same set of values, i.e. they have the same domain, we will omit the subscript when referring to the domain ($\mathcal{Y}$ instead of $\mathcal{Y}_i$).

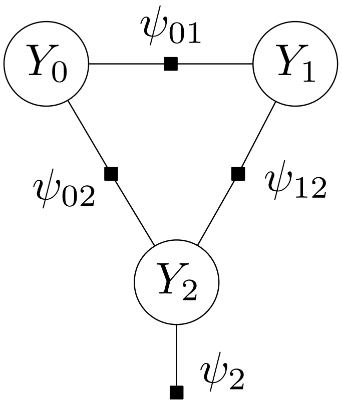


Figure 2.1: **Simple Graphical Model:** with 3 variables and 4 factors.

2.1.2 Undirected Graphical Models

In this work, we will use the undirected factor graph [Kschischang et al., 1998] notation. The factor graph $\mathcal{G}(\mathbf{Y}, \mathcal{F}, G, \psi)$ defines a bipartite graph G between variables $\mathbf{Y}$ and a set of factors $\mathcal{F}$. Let the neighborhood of each factor $f \in \mathcal{F}$ be $\mathbf{Y}_f$ (also referred by the neighborhood function $\mathcal{N}(f)$). Each factor contains an associated *score function* (or log-potential function) $\psi_f : \boldsymbol{\mathcal{Y}}_f \rightarrow \mathcal{R}$, i.e. for any assignment $\mathbf{y}_f \in \boldsymbol{\mathcal{Y}}_f$, $\psi_f(\mathbf{Y}_f = \mathbf{y}_f)$ returns a scalar value. Given the factors, the factor graph $\mathcal{G}$ represents a probability distribution $p_{\mathcal{G}} : \boldsymbol{\mathcal{Y}} \rightarrow [0, 1]$ over the assignments $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ as:

$$p_{\mathcal{G}}(\mathbf{Y} = \mathbf{y}) = \frac{1}{Z} \exp \sum_{f \in \mathcal{F}} \psi_f(\mathbf{Y}_f = \mathbf{y}_f) \tag{2.1}$$

$$Z_{\mathcal{G}} = \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \exp \sum_{f \in \mathcal{F}} \psi_f(\mathbf{Y}_f = \mathbf{y}_f) \tag{2.2}$$

$Z_{\mathcal{G}}$ is known as the normalization constant or the partition function, and sum of the scores of the factors is known as the *log-model score*:

$$\pi(\mathbf{y}) = \sum_{f \in \mathcal{F}} \psi_f(\mathbf{y}_f) \qquad p(\mathbf{y}) = \frac{e^{\pi(\mathbf{y})}}{Z} \tag{2.3}$$

For the sake of brevity, like above, we will often omit "$\mathcal{G}$" and replace "$\mathbf{Y}_c = \mathbf{y}_c$" with "$\mathbf{y}_c$".

We show an example of a simple graphical model in Fig. 2.1. The graphical model is defined over three variables $\mathbf{Y} = \{Y_0, Y_1, Y_2\}$ and contains four factors with their score functions: ψ_{01} :

$\mathcal{Y}_0 \times \mathcal{Y}_1 \rightarrow \mathcal{R}, \psi_{12} : \mathcal{Y}_1 \times \mathcal{Y}_2 \rightarrow \mathcal{R}, \psi_{02} : \mathcal{Y}_0 \times \mathcal{Y}_2 \rightarrow \mathcal{R}$, and $\psi_2 : \mathcal{Y}_2 \rightarrow \mathcal{R}$. Given an assignment $\mathbf{y} = \{Y_0 = y_0, Y_1 = y_1, Y_2 = y_2\}$, the model score is $\pi(\mathbf{y}) = \psi_{01}(y_0, y_1) + \psi_{12}(y_1, y_2) + \psi_{02}(y_0, y_2) + \psi_2(y_2)$. The probability given by the graph G for $\mathbf{y}$ is $p(\mathbf{y}) = \frac{\exp \pi(\mathbf{y})}{Z}$, where Z is defined as above. For some models, it is useful to distinguish fixed (or *observed*) variables, which are random variables that take only a single value, from other variables; we use grey filled circles to denote these, and whenever possible, fold them into the factor potentials they neighbor.

Although we focus only on undirected graphical models in this treatise, factor graphs are a more expressive class of probabilistic graphical models that extend classical Ising models [Ising, 1925] and pairwise Markov Random Fields [Kindermann et al., 1980], and can also represent directed graphical models. We also make use of a generalization of factor graphs with dynamic structure where the set of factors used to compute the model score is dependent on the assignment to the variables:

$$\pi(\mathbf{y}) = \sum_{f \in \mathcal{F}(\mathbf{y})} \psi_f(\mathbf{y}_f) \tag{2.4}$$

This representation is closely related to *imperatively defined factor graphs* [McCallum et al., 2009], and similar to *case-factor diagrams* [McAllester et al., 2004] and *gates* [Minka and Winn, 2008]. Dynamic graphs can be used with algorithms that work with static structure by fully *unrolling* the graph, i.e. $\mathcal{F} = \bigcup_{\mathbf{y} \in \mathcal{Y}} \mathcal{F}(\mathbf{y})$, however this may in general result in exponential number of factors.

2.2 Inference

Probabilistic inference is the main task performed with graphical models. There are two distinct inference paradigms commonly discussed in the literature: *maximum a posteriori* inference and *marginal* inference.

2.2.1 Maximum-a-posteriori (MAP) Inference

For a factor graph, the maximum a posteriori (MAP) problem is to find an instantiation of the variables $\mathbf{y}$ that maximizes $p(\mathbf{y})$ or equivalently $\pi(\mathbf{y})$, i.e. identifying the assignment (or *configu-*

ration) of the variables that has the maximum probability according to the model.

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p(\mathbf{y}) = \arg\max_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} \pi(\mathbf{y}) \tag{2.5}$$

Since a large number of tasks in information extraction, NLP, computer vision, etc. require only the best estimate according to the model (to present to the user, for example), the problem of MAP inference is a very important one. Unfortunately, a naive implementation that iterates over the domain $\boldsymbol{\mathcal{Y}}$ takes exponential time, and is prohibitively expensive even for small models.

2.2.2 Marginal Inference

The marginal inference problem is to compute the complete distribution over a subset of variables $\mathbf{Y}_A$ according to the model. Often the *marginals* of interest are defined over individual variables or factors:

$$p_i(y_i) = \sum_{\mathbf{y}'\in\boldsymbol{\mathcal{Y}}:y'_i=y_i} p(\mathbf{y}') \tag{2.6}$$

$$p_f(\mathbf{y}_f) = \sum_{\mathbf{y}'\in\boldsymbol{\mathcal{Y}}:\mathbf{y}'_f=\mathbf{y}_f} p(\mathbf{y}') \tag{2.7}$$

Similar to MAP inference, marginal inference is non-trivial using a naive approach for all but the smallest of models due to the size of $\boldsymbol{\mathcal{Y}}$. The marginals are an important requirement for a number of applications. First, they provide a measure of confidence or uncertainty in the inferred values. Marginals are also required for maximum-likelihood learning of model parameters. Marginal probabilities are especially important for semi-supervised learning approaches, and for existing approaches for active learning.

2.2.3 Approximate Inference

Exact inference, both MAP and marginal, is tractable for tree-structured models, however is NP-hard in general graphical models [Cooper, 1990, Shimony, 1994]. However, many useful approximations with provable properties make graphical model inference practical in a wide diversity of application areas. The two most common approximate inference frameworks are Monte Carlo sampling methods and message passing.

2.2.3.1 Markov Chain Monte Carlos (MCMC) Methods

A number of methods have been designed to *sample* variable assignments $\mathbf{y}$ according to the probability distribution p as defined by the graphical model. Markov Chain Monte Carlo (MCMC) methods are a family of algorithms that generate samples by iterating a Markov chain to convergence [Smith and Roberts, 1993]. We focus specifically on the Metropolis Hastings (MH), one of the general formulations of MCMC [Metropolis et al., 1953, Hastings, 1970]. MH algorithm uses the current sample $\mathbf{y}$ and a proposal distribution $q(\mathbf{y}'|\mathbf{y})$ to generate the next sample, iterating till the samples are being drawn from p. Evaluating acceptance of a sample is quite inexpensive since it depends only on variables and factors local to the proposed change. Due to this efficient computation, MCMC methods are able to scale to large models with dense structures and large-domain variables.

We can use the set of samples to estimate the expectations of functions under p, for example computing marginals requires aggregating the counts of the values across the samples. MCMC chains can be used for MAP inference by selecting the sample with the highest probability/model score, although in practice a temperature may be used to encourage high-probability samples, similar to simulated annealing [Kirkpatrick et al., 1983]. The details of the MCMC algorithm, and their utility for large, dense graphical models, is presented in Chapter 3.

2.2.3.2 Belief Propagation

In belief propagation [Pearl, 1988], the factors in the factor graph "negotiate" with each other regarding the marginal probabilities of their neighboring variables by passing *messages*. The messages from variable Y_i to factor f and factor f to variable Y_i are given below.

$$\boldsymbol{m}_{i\to f}(y \in \mathcal{Y}_i) = \prod_{\substack{f':Y_i\in\mathbf{Y}_{f'}\\ f'\neq f}} \boldsymbol{m}_{f'\to i}(y), \tag{2.8}$$

$$\boldsymbol{m}_{f\to i}(y \in \mathcal{Y}_i) = \sum_{\mathbf{y}'_f:y'_i=y} \exp\{\psi_f(\mathbf{y}'_f)\} \prod_{\substack{i':Y_{i'}\in\mathbf{Y}_f\\ i'\neq i}} \boldsymbol{m}_{i'\to f}(y) \tag{2.9}$$

Messages are iteratively selected and updated until they converge or a maximum number of iterations is reached. Approximate marginal distributions are then extracted from the messages. The max-product algorithm, an alternate form of BP, can be used to obtain an approximate MAP configuration instead of marginals by replacing the $\sum$ in Eq. (2.9) with a $\max$.

BP is guaranteed to converge to the exact marginals for tree-shaped models. Although BP is also widely used and has achieved considerable empirical success for loopy models, it is not guaranteed to converge, hence a number of convergent variants have been proposed [Yuille, 2002, Wainwright et al., 2003, Hazan and Shashua, 2008]. We describe belief propagation and extensions to it in detail in Chapter 6.

2.3 Architectures for Parallel and Distributed Computing

In this section, we present a brief outline of the different architecture available for parallel computing, and existing machine learning toolkits for distributed processing.

2.3.1 Graphical Processing Units

Graphical Processing Units (GPUs), although designed for fast computations of transformations in computer graphics, have gained recent traction in the machine learning community. GPUs contain a large number of processors (about a hundred in consumer level devices) that can compute floating point vector operations extremely efficiently, providing fine-grained parallelism. For machine learning, a large number of bottlenecks can be traced to vector/matrix products, and GPUs are appropriate for techniques that heavily rely on linear algebra. Unfortunately, inference in graphical models, particularly for information extraction, contains a number of characteristics that make it unsuitable for GPUs. First, due to the large number of factors and variables, we need to control the prioritization of the computations dynamically, which is difficult to perform in GPUs. Second, a number of bottlenecks in the system, such as identifying neighborhoods, extracting features, proposing changes, etc. are not vector/matrix operations. Last, our models may be too large to fit in the memory for GPUs.

2.3.2 Multi-core Systems

With the speeds of individual processors reaching their peak, computing hardware has started moving towards multi-core architectures, with $4-8$ processors on consumer-level machines. These architectures provide a suitable parallelization environment with shared memory access amongst the processors. There are a number of concerns to consider before these architectures can be used directly for our applications. First, even though the memory is shared, splitting of data across the processors has to be done in a way that maximizes reuse in the cache. A number of machine learning methods on multi-core systems faced speedup problems due to this problem, and had to use heuristics for better caching [Gonzalez et al., 2009b, Subramanya and Bilmes, 2009]. Second, methods that are designed for multi-core architectures cannot scale easily if the magnitude of data increases since the number of cores is fixed, and the cost per core for additional cores increases substantially. Although we make considerable use of multi-core architectures in this work, we address the above concerns by implementing generic parallelization algorithms that do not make the shared memory assumption, and can easily be used on multi-node or cloud-computing architectures.

2.3.3 Multi-Node Clusters

Multi-node clusters are the most popular form of addressing scale using parallelization. These systems consist of a large number of computing nodes, with no shared memory, and communication is usually through the network, making it costly. In fact, most impressive results in scalability minimize communication, or only communicate in phases [MapReduce; Dean and Ghemawat, 2004]. Algorithms designed for this architecture can also be used on multi-core architectures, by treating each core as an individual computing node in a cluster. Further, depending on the need, the size of the cluster can be increased easily by adding additional machines, and the cost is proportional to the number of machines. Due to these reasons, for the most part, we focus on these architectures for parallelization.

2.3.4 Cloud Computing

Cloud computing is an upcoming paradigm for providing computing as a service. Although similar to the multi-node architecture described above, and most algorithms can work directly, the differences provide potential for some interesting concerns and trade-offs. Homogeneity amongst the machines is harder to achieve (may require a higher cost), and algorithms that are robust to variation in speed, memory, and communication capacities of the nodes fare better. Furthermore, given a limited budget, performance of the algorithm needs to be characterized precisely to decide the configuration of the computational resources, for example, choosing between a large number of slow machines or a small number of high-performance nodes. Last, often decisions will need to be made between adding more machines or waiting for more time; the *anytime* and speedup properties of the algorithms need to be studied to allow such decisions given a budget. Most of the methods proposed in this work can be used in the cloud computing setting, and we hope we provide adequate evaluation of the methods to inform the decision-making that is unique to cloud computing architectures.

2.3.5 Parallel and Distributed Programming Frameworks

There are a variety of existing frameworks for parallel and distributed programming that can be applied to the problem of probabilistic inference. These frameworks vary in the level of parallelism they support ranging from fine-grained parallelism to very high-level abstractions. The *Message Passing Interface* (MPI) standard and related libraries provides very general parallel programming constructs that can be used in the distributed setting [Gropp et al., 1996]. However, MPI is not explicitly targeted at parallel *data processing* so the programmer must manage the distribution of data to compute nodes. At the other end of this spectrum is *Map-Reduce* [Dean and Ghemawat, 2004], which provides a simple, high-level parallelism abstraction targeted specifically at distributed data processing. Implementing algorithms in Map-Reduce imposes strict constraints on the computation, however, when these constraints can be easily incorporated, Map-Reduce can provide impressive speedups [Chu et al., 2007].

Complex machine learning algorithms that do not meet these constraints are often difficult to parallelize efficiently using Map-Reduce. Along with the open-source implementation Hadoop [White, 2009], there exist a number of extensions to the Map-Reduce framework. *Dryad* [Yu et al., 2008, Isard et al., 2007] support computations that can be represented as an arbitrary *directed, acyclic* graph, instead of the 2-layer bipartite graph imposed by Map-Reduce. The *Spark* framework [Zaharia et al., 2010] augments Map-Reduce with parallel collections and accumulators that gather the results of the distributed computations efficiently. More recently, there has been increasing interest in distributed processing with streaming data, and a number of packages have been introduced [Zaharia et al., 2013, Apache Storm]. However, since none of these approaches are designed specifically for computations over general graphs, they induce artificial restrictions when implementing probabilistic inference algorithms for graphical models.

Recent research has seen the introduction of mid-level graph-based parallel computing abstractions that sit between the complexity and generality of MPI on one hand, and the simplicity and restrictiveness of Map-Reduce on the other. For the specific Cray XMT supercomputer architecture that provides massive-scale shared-memory parallelism, *GraphCT* [Ediger et al., 2012] provides a framework for analysis of billion-scale graphs. *Pregel* was introduced for graph-based computations that can be represented using a bulk-synchronous parallel framework [Malewicz et al., 2009, 2010]. A number of machine learning approaches that cannot be efficiently implemented using Map-Reduce are suitable for bulk-synchronous computation. However, belief propagation and MCMC inference methods are not in this category of algorithms and their correct implementations in Pregel are not efficient. To address the shortcomings of Pregel in the context of machine learning, *GraphLab* [Low et al., 2010] (and its offsprings *GraphChi* [Kyrola et al., 2012] and *PowerGraph* [Low et al., 2012, Gonzalez et al., 2012]) allow partially asynchronous processing of the graph computations. Given a graph consisting of data and computations on nodes and edges, GraphLab schedules the multiple computations asynchronously such that neighborhood constraints are met. By partially removing the synchronization barrier, implementations of Belief Propagation [Gonzalez et al., 2009b] and MCMC [Gonzalez et al., 2011] inference methods in GraphLab can achieve better parallel scalability.

2.4 Applications to Information Extraction

In this section, we explore a number of applications of the proposed work. Although there are a large number of potential applications of inference and learning in the fields of bioinformatics, vision, etc., here we focus primarily on information extraction. Given a large corpus of unstructured text, the goal of information extraction is to extract a structured representation of the information in the corpus. For example, given business newswire articles, information extraction can be used to identify all the companies that are mentioned in the articles, and to also extract the various events and relations between them. As an additional example, consider a large collection of citations extracted from the `References` section of scientific papers. The goal of information extraction, given these citations, may be to identify the sets of citations that are referring to the same paper (useful for computing citation counts), and extracting the authors, title and venues of each of these papers. We describe some tasks for corpus-wide information extraction, and describe some graphical models that may be appropriate for them.

2.4.1 Coreference or Entity Resolution

Coreference or Entity Resolution is the task of de-duplicating a number of records, or, in other words, identifying the set of underlying entities that a number of mentions refer to. A large number of relevant tasks fit under this definition. Within-document coreference is concerned with noun phrase *mentions* in a single document (for example, “she”, “him”, “Clinton”, “Mrs. Clinton”, “Hillary”, “president”), and identifies the sets of mentions that refer to the same underlying *entity* ({“she”, “Mrs. Clinton”, “Hillary”} and {“him”, “Clinton”, “president”}) [Haghighi and Klein, 2009, Culotta et al., 2007]. On the other hand, cross-document coreference refers to resolving the entities where the set of mentions originate from documents across the corpus [Singh et al., 2011b, Rao et al., 2010, Mayfield et al., 2009]. Haghighi and Klein [2010] provides some preliminary work on combining these two types of coreference. Finally, the task of record de-duplication, such as identifying the underlying set of papers from a collection of citations, is known as record-linkage or entity resolution [Singla and Domingos, 2006].

The models of coreference are notoriously challenging when represented as graphical models. Some models of coreference consist of binary decisions between pairs of mentions, indicating whether or not they are coreference. This results in a quadratic number of binary variables, along with cubic number of deterministic factors to enforce transitivity[1]. Instead, we explore models that consist of mention and entity variables [McCallum and Wellner, 2003, Culotta et al., 2007]. Mention variables are random variables that take one of the entities as their values. Entity variables, on the other hand, are *set-valued* variables that can take any subset of the mentions as their values. This representation allows factors both over pairs of mentions (capturing pairwise similarity) and entity-wide statistics (capturing the coherence of the entity). Unfortunately, the pairwise factors and set-valued variables result in exponential sized domains and a fully-connected graph when unrolled, leading to intractable inference.

2.4.2 Relation Extraction

Given a pair of entities that appear in a corpus (such as `Bill_Gates` and `Microsoft`), the task of relation extraction is to identify the relation between them (i.e. `Is_CEO_Of`). Extracting such relations is an integral component of the resulting structured representation, and useful for downstream components (such as coreference) and presentation of information to users.

The task is often decomposed over sentences, and pairs of entity mentions that appear in a sentence (known as a relation mention) are classified as one of the relation types [Zhou et al., 2005]. The predictions over the relation mentions are aggregated over pairs of entities by using a simple coreference system to identify coreferent mentions. This representation of the task cannot propagate information across the corpus, i.e. evidence from a relation prediction in one sentence

[1] Often the transitivity factors are ignored, and applied heuristically as a post-processing step

cannot be used for improving the prediction on the relation mention in a different sentence. We are interested in models of relation extraction that consist of both sentence-level (relation mentions) and corpus-level variables (relations), all of which take a value from a fixed set of relation types [Riedel et al., 2010b, Hoffmann et al., 2011]. Factors exist over relation mentions (evidence from the sentence), between relation mentions and relation variables (coherence between sentence- and corpus-level), and between relations that describe the same entity (to capture similarity of relations for entities). Although we expect this model to provide significant accuracy gains, the inference and learning gets prohibitively expensive due to the number of variables and factors in the model.

2.4.3 Joint Inference

An information extraction pipeline is usually decomposed into a number of individual tasks, such as part of speech tagging, chunking, named-entity recognition, within-doc coreference, relation extraction, and cross-document coreference. These tasks are performed independently, with the best prediction from each task used as an input to the next task. Unfortunately, this leads to cascading errors, since a small error made earlier on in the pipeline can amplify as it propagates through the pipeline. Further, for many tasks the ordering between them in the pipeline is not clear since the tasks may depend on each other, for example both relation extraction and coreference improve if they have access to each others' predictions.

Joint inference approaches have been proposed to address these concerns [McCallum and Jensen, 2003, Poon and Domingos, 2007, Sutton and McCallum, 2005, Wellner et al., 2004]. Many of the earlier approaches propose methods for each task to communicate its uncertainty in the predictions [Finkel et al., 2006, Finkel and Manning, 2009], however significant success is achieved when both the tasks are created as a single model [Poon and Domingos, 2007, Singh et al., 2009]. In this work, we will explore models of joint inference defined both over sentence-level and corpus-level variables. For example, named-entity recognition and relation extraction are strongly tied tasks that can benefit from joint modeling. As mentioned above, coreference and relation extraction are also dependent on each other, and provide potential for joint inference at the corpus-level. Even the coreference task at different levels, such as within-doc and cross-doc, can be represented in the same model.

Modeling multiple tasks in a joint manner introduces considerable complexity to the resulting models. Although the number of variables may not change, the additional cross-task factors make independent components depend on each other. For example, once sentence-level tasks are modeled with corpus-level tasks, inference on each sentence is not independent. Further, joint inference usually adds a large number of strong dependencies that introduce loops in the models, increasing the tree-width. By using the inference and learning approaches in the thesis for joint inference, we hope to address these drawbacks and achieve significant accuracy gains.

2.4.4 Probabilistic Databases

As described above, the output of information extraction is a structured representation of the information extracted from the input corpus. To allow computation of confidences for ranking and filtering of query results, it is often required to store the uncertainty associated with the output of the information extraction pipeline. It is incredibly difficult to incorporate such uncertainties in traditional database systems in a way that allows efficient inference. Motivated by this problem (amongst others), there has been a recent surge in research on *probabilistic databases* [Dalvi et al., 2009]. Some of the most promising approaches to probabilistic databases represent the uncertainty as a graphical model [Wick et al., 2010, Sen et al., 2009, Singh and Graepel, 2013].

Depending on the underlying structure of the uncertainty being represented in the database, the model can become quite large and complex, making traditional forms of inference impractically slow. This provides potential for parallelization based inference techniques to allow efficient inference. Further, a large number of our proposed techniques are *anytime* in nature, and can be used to answer queries under a time constraint, which is especially useful in probabilistic databases.

CHAPTER 3

MCMC OVERVIEW

The sun comes up just about as often as it goes down, in the long run, but this doesn't make its motion random.

Donald Knuth

In this chapter, we provide an introduction to the class of inference techniques known as Markov Chain Monte Carlo (MCMC) methods. MCMC is a Monte Carlo approach (Section 3.1) that constructs a Markov chain used to generate samples from the desired distribution, $p_{\mathcal{G}}$. We focus on Metropolis Hastings (Section 3.2.2), an MCMC algorithm that uses a flexible proposal function to generate samples efficiently. MCMC algorithms can be used both for approximate MAP and marginal inference simply by retaining the correct statistics across samples; we describe the various algorithm parameters for marginal inference in Section 3.2.4 and the application to MAP in Section 3.3. In Section 3.4, we outline the advantages and disadvantages of using MCMC for large, dense graphs, and highlight related work in Section 3.4.3.

3.1 Monte Carlo Methods

Monte Carlo methods include rejection sampling, importance sampling, and Markov chain Monte Carlo (MCMC). These algorithms sample instantiations $\mathbf{y}$ according to a probability distribution $p : \boldsymbol{\mathcal{Y}} \rightarrow [0,1]$, which, in our case, is the distribution $p_{\mathcal{G}}$ induced by a graphical model $\mathcal{G}$. Rejection sampling and importance sampling assume that $p(\mathbf{y})$ can be computed efficiently. Rejection sampling samples from the uniform distribution and then applies a stochastic acceptance step to ensure that samples are drawn from $p(\mathbf{y})$. Many samples can be drawn independently in parallel, but in high dimensions the probability that any sample is accepted can be vanishingly small. Importance sampling improves on rejection sampling by drawing initial samples from a proposal distribution $q(\mathbf{y})$. It retains all samples and accounts for the discrepancy between $q(\mathbf{y})$ and $p(\mathbf{y})$ by weighting the samples. While samples can be drawn independently and in parallel, in high dimensions the chance of generating a sample with high probability under $p(\mathbf{y})$ can again be vanishingly small unless an accurate proposal distribution $q(\mathbf{y})$ is known. More importantly, computing $p(\mathbf{y})$ is intractable for non-trivial models, as it requires computation of the partition function Z.

3.2 MCMC Inference

Markov Chain Monte Carlo (MCMC) is a collection of methods used for sampling from a complex probability distribution with computing the partition function. MCMC approaches construct a *Markov* chain that whose stationary distribution converges to the desired probability distribution at equilibrium. Further, they are *anytime* in nature; running more steps of the algorithm improves the quality of the solution. While a wide variety of MCMC algorithms exist, we focus on the Metropolis-Hastings (MH) algorithm, which is one of the most general approaches.

3.2.1 Markov Chain Monte Carlo

Consider a sequence of random variables $\mathbf{y}_0, \mathbf{y}_1, \ldots$ such that $\forall t = 0, 1, 2, \ldots, \mathbf{y}_t \in \boldsymbol{\mathcal{Y}}$. The sequence is called a Markov chain if the distribution over $\mathbf{y}_t$ depends only on $\mathbf{y}_{t-1}$, i.e.:

$$P(\mathbf{Y}_{t+1} = \mathbf{y} | \mathbf{Y}_t = \mathbf{y}_t, \mathbf{Y}_{t-1} = \mathbf{y}_{t-1}, \ldots, \mathbf{Y}_0 = \mathbf{y}_0) = P(\mathbf{Y}_{t+1} = \mathbf{y} | \mathbf{Y}_t = \mathbf{y}_t) \tag{3.1}$$

We represent $P(\mathbf{Y}_{t+1} = \mathbf{y}_i | \mathbf{Y}_t = \mathbf{y}_j)$ as $P_{ij} : \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \rightarrow [0, 1]$, also known as the *transition probability*. Given a distribution p_0 over initial values $\mathbf{y}_0$, we can compute the distribution over samples $\mathbf{y}_t$ at time t as:

$$p_t(\mathbf{y}_j) = \sum_{\mathbf{y}_i \in \boldsymbol{\mathcal{Y}}} p_{t-1}(\mathbf{y}_i) P_{ij} = (p_{t-1}P)(\mathbf{y}_j) = (p_0 P^t)(\mathbf{y}_j) \tag{3.2}$$

where $p_t(\mathbf{y})$ refers to the probability distribution over $\mathbf{y}$ at step t (with slight abuse of notation we use p_t as the corresponding vector).

We would like the Markov chain to converge to our desired model, i.e. the *stationary distribution* of the chain should be $p_{\mathcal{G}}$. This can be guaranteed if the chain obeys the following *detailed balance property*, $p_{\mathcal{G}}(\mathbf{y}_i)P_{ij} = p_{\mathcal{G}}(\mathbf{y}_j)P_{ji}$, and the transition probability matrix is *irreducible* (non-zero probability for reaching all states from every state) and *aperiodic* (states do not have a finite period, i.e. $\exists N$ s.t. $\forall n > N, P^n_{ii} > 0$). MCMC algorithms thus describe a family of algorithms for specifying a Markov chain that converges to the desired distribution [Smith and Roberts, 1993].

3.2.2 Metropolis-Hastings

Metropolis-Hastings algorithm [Metropolis et al., 1953, Hastings, 1970] constructs a Markov chain to sample from the desired probability distribution by using a flexible proposal function. Specifically, the algorithm takes as input an initial configuration of the variable $\mathbf{y}_0$ and a proposal function $q : \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \rightarrow [0,1]$, s.t.$\forall \mathbf{y} \sum_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} q(\mathbf{y}, \mathbf{y}') = 1$. Thus the proposal function, given any *current* configuration $\mathbf{y}$, provides a distribution over the *next* configurations[1]. To generate each sample, the algorithm first *proposes* a change to the current configuration $\mathbf{y}$ using the distribution induced by $q(\mathbf{y}, \_)$ to obtain the *next* configuration $\mathbf{y}'$. The algorithm then *accepts* $\mathbf{y}'$ as the current sample with the following probability (otherwise retains $\mathbf{y}$ as the current sample):

$$\alpha(\mathbf{y}, \mathbf{y}') = \min\left(1, \frac{p_{\mathcal{G}}(\mathbf{y}')}{p_{\mathcal{G}}(\mathbf{y})} \frac{q(\mathbf{y}', \mathbf{y})}{q(\mathbf{y}, \mathbf{y}')}\right) \tag{3.3}$$

The MCMC samples are thus continually generated, eventually drawn from the desired distribution, as we show next. This algorithm is summarized in Algorithm 3.1.

The transition probability of the algorithm may be computed as follows:

[1] The Metropolis algorithm [Metropolis et al., 1953] assumes that the proposal function is symmetric, Hastings [1970] generalize this condition to asymmetric distributions.

Algorithm 3.1 MH-MCMC: Metropolis-Hastings Markov Chain Monte Carlo

```
procedure METROPOLISHASTINGS($\mathcal{G}$, $q$, $N$, $\mathbf{y}_0$)
    $\mathbf{y} \leftarrow \mathbf{y}_0$
    $S \leftarrow \{\}$                                                    ▷ Set of samples
    for $N$ samples do
        $\mathbf{y}' \sim q(\mathbf{y}, \cdot)$
        $\alpha \leftarrow \min\left(1, \frac{p_{\mathcal{G}}(\mathbf{y}')}{p_{\mathcal{G}}(\mathbf{y})}\frac{q(\mathbf{y}', \mathbf{y})}{q(\mathbf{y}, \mathbf{y}')}\right)$      ▷ Acceptance Ratio
        if flip($\alpha$) then
            $\mathbf{y} \leftarrow \mathbf{y}'$                                        ▷ Accept the Sample
        end if
        $S \xleftarrow{+} \mathbf{y}$                                        ▷ Accumulate Sample
    end for
    return $S$
end procedure
```

$$P_{ij} = q(\mathbf{y}_i, \mathbf{y}_j) \min\left(1, \frac{p_{\mathcal{G}}(\mathbf{y}_j)}{p_{\mathcal{G}}(\mathbf{y}_i)}\frac{q(\mathbf{y}_j, \mathbf{y}_i)}{q(\mathbf{y}_i, \mathbf{y}_j)}\right) \tag{3.4}$$

Note that $\alpha(\mathbf{y}, \mathbf{y}') < 1 \implies \alpha(\mathbf{y}', \mathbf{y}) = 1$. For such a pair, without loss of generality, it is straightforward to show detailed balance with respect to $p_{\mathcal{G}}$ as:

$$\begin{aligned} p_{\mathcal{G}}(\mathbf{y}_i)P_{ij} &= p_{\mathcal{G}}(\mathbf{y}_j)P_{ji} \\ p_{\mathcal{G}}(\mathbf{y}_i)q(\mathbf{y}_i, \mathbf{y}_j)\alpha(\mathbf{y}_i, \mathbf{y}_j) &= p_{\mathcal{G}}(\mathbf{y}_j)q(\mathbf{y}_j, \mathbf{y}_i)\alpha(\mathbf{y}_j, \mathbf{y}_i) \\ p_{\mathcal{G}}(\mathbf{y}_i)q(\mathbf{y}_i, \mathbf{y}_j)\frac{p_{\mathcal{G}}(\mathbf{y}_j)}{p_{\mathcal{G}}(\mathbf{y}_i)}\frac{q(\mathbf{y}_j, \mathbf{y}_i)}{q(\mathbf{y}_i, \mathbf{y}_j)} &= p_{\mathcal{G}}(\mathbf{y}_j)q(\mathbf{y}_j, \mathbf{y}_i) \\ p_{\mathcal{G}}(\mathbf{y}_j)q(\mathbf{y}_j, \mathbf{y}_i) &= p_{\mathcal{G}}(\mathbf{y}_j)q(\mathbf{y}_j, \mathbf{y}_i) \end{aligned}$$

To ensure that the transition probabilities are irreducible and aperiodic, the proposal function also needs to be irreducible and aperiodic, which is often quite easy to show in practice.

Metropolis-Hastings is a popular algorithm for inference in graphical models for a number of reasons. Since the acceptance probability only depends on the ratio of model scores, it allows the algorithm to ignore the partition function Z, and further, if the proposals make changes to variables *local* in the graph, only the factors neighboring the changed variables are needed to compute the acceptance probability. Similarly, we need to be able to sample conditionally from proposal function, and compute it only up to a constant, allowing significant flexibility in designing fruitful proposal functions. Finally, if an appropriate proposal function is used, the algorithm complexity does not depend on the size of the variable domains, allowing models with large-domain variables and higher-order factors.

3.2.3 Gibbs Sampling

Gibbs sampling is a special case of the Metropolis-Hastings algorithm that does not require a proposal function [Geman and Geman, 1984, Gelfand and Smith, 1990]. To generate each sample, the algorithm iterates through all the variables and samples a value of each variable conditioned on the values of the other variables. This algorithm is usually efficient since sampling a variable conditioned on other model variables is often computationally inexpensive. Unfortunately, for variables with with large domains, sampling from the conditional distribution is not cheap. Further, consecutive samples from Gibbs sampling are often correlated[2], and a number of extension have been proposed in the literature for addressing the correlation [Liu et al., 1994, Jensen et al., 1995, Venugopal and Gogate, 2013].

3.2.4 Algorithm Parameters

For marginal inference, a set of counts for the variable values is maintained. As the samples are generated, the counts are updated using the sampled configuration. As the number of samples grows, the normalized counts approach the true marginal distribution of the variables.

To compute these marginals, a number of parameters need to be specified. The total number of samples directly determines the quality of the marginals, however with an accompanying increase in computation cost. To allow the chains to mix (start sampling from the stationary distribution), samples are only aggregated after a *burn-in* period. Since consecutive samples are correlated,

[2] Metropolis Hastings can avoid correlated samples, to a certain degree, by using sophisticated proposal functions.

samples are often aggregated every i^{th} sample (where i is known as the *thinning period*). Further, parallel MCMC chains may be used to aggregate samples across the chains. Although higher values for burn-in, number of parallel chains, and thinning steps are useful to produce accurate marginals, each results in additional computation cost. These parameters are usually specified heuristically based on the run-time requirements of the inference algorithm.

3.3 MCMC Inference for MAP

MCMC can also be used for MAP inference. The algorithm, in this case, keeps track of the sample with the highest model score π, where MCMC is used to explore the high-probability regions. To facilitate convergence to the mode of the distribution, the model score is tempered with a temperature.

$$\alpha(\mathbf{y},\mathbf{y}') = \min\left(1, \left(\frac{p_{\mathcal{G}}(\mathbf{y}')}{p_{\mathcal{G}}(\mathbf{y})}\right)^{1/t} \frac{q(\mathbf{y}',\mathbf{y})}{q(\mathbf{y},\mathbf{y}')}\right) \tag{3.5}$$

This use of a temperature is closely related to the simulated annealing algorithm [Kirkpatrick et al., 1983], which converges to the MAP configuration if the temperature is lowered slowly enough.

To prove that we can employ MCMC for MAP inference, the Markov chain needs to show irreducibility and aperiodicity, and the acceptance ratio needs to compute the ratio of the model probability correctly. Convergence to MAP configuration is guaranteed even if detailed balance does not hold, and even if $\frac{q(\mathbf{y}',\mathbf{y})}{q(\mathbf{y},\mathbf{y}')}$ is not included in the acceptance probability [Granville et al., 1994]. This allows for usage of customized, domain-specific proposal functions that result in fruitful samples, but for which we cannot compute forward-backward probability ratio correctly.

3.4 MCMC for Large, Dense Models

MCMC algorithms have been a popular choice of inference for large models. We describe the advantages below, and list some of the drawbacks that restrict their use in practice. Existing approaches that scale MCMC to large models are outlined in Section 3.4.3.

3.4.1 Advantages

For large, dense graphical models, MCMC provides an appropriate tool for inference for a number of reasons. First, inference is able to handle models with high-degree factors with little or no impact on time to generate a single sample. Second, generating a single sample is based on the *local structure* around the proposed change, and is independent of the number of variables in the model or the domain of the variables, aiding scalability. Third, by maintaining only a single configuration of the variables, inference requires minimal memory, and is able to handle variables with very large domains (such as set-valued variables). Last, MCMC inference can be made significantly more efficient by changing the proposal distribution; this allows injection of domain knowledge to directly impact the speed of convergence. For these reasons, we feel MCMC is the suitable choice for inference in the large, dense graphical models.

3.4.2 Potential Drawbacks

As we describe above, the time for generating a single sample is (for the most part) independent of the number of variables, the domain size of the variables, and the density of the model. Unfortunately, the number of samples required to capture arbitrarily complex distributions increases exponentially with the number of variables and domain sizes. Further, the number of variables and the structure of the model have an adverse effect on the number of samples required for the chains to mix (burn-in period) and on the correlations between consecutive samples. Parallel MCMC chains are often not practical due to the storage requirements, and parallelizing a single chain of MCMC is non-trivial, since the MCMC algorithm is inherently sequential.Finally, the proposal function has a significant impact on the performance of sampling, and designing a proposal requires considerable manual effort and domain expertise. For the large models we want to perform inference over, these drawbacks lead to a need for more efficient samplers than those currently available.

3.4.3 Related Work

MCMC is a popular method among researchers for inference with large and dense graphical models [Richardson and Domingos, 2006, Poon and Domingos, 2006, Poon et al., 2008, Singh et al., 2009, Wick et al., 2009]. Some probabilistic programming packages popular amongst NLP practitioners also rely on MCMC for inference and learning [Richardson and Domingos, 2006, McCallum et al., 2009]. Although most of these methods apply MCMC directly, the rate of convergence of MCMC has become a concern as larger and more densely-factored models are being considered, motivating the need for more efficient sampling.

A few approaches to distributing MCMC have been recently introduced [Doshi et al., 2009, Yan et al., 2009]. Along similar lines, there has been considerable work in distributing topic models [Newman et al., 2006, Asuncion et al., 2009], and recent work has demonstrated impressive scalability to large document collections [Smola and Narayanamurthy, 2010, Ahmed et al., 2012]. Many of these schemes are designed for custom models, are applicable only to shared-memory multi-core architectures, or make strong synchronization assumptions that we do not desire in large-scale models. Gonzalez et al. [2011] introduce an approach that is partially synchronous and provides linear speedups on a multi-core setup, but requires analysis of global properties of the graph that is not possible for large, dynamic graphs. A popular option for distributing MCMC is to use multiple chains, which is useful for both marginal [Murray, 2010] and MAP [Earl and Deem, 2005] inference. Distributing single chains across multiple machines or cores requires each chain to converge independently, which can be quite inefficient [Rosenthal, 2000], and we would like to utilize all the available computational resources to make the convergence for a single chain more efficient. Further, MCMC with multiple chains requires maintaining copies of the variable assignments, which may require a prohibitive amount of memory.

Recent work on parallelizing multiple chains addresses this concern for certain models by sharing information across chains [Nishihara et al., 2012], however relies on shared-memory parallelism and does not perform informative partitioning. In addition, these approaches are clearly not suitable for massive problems that require distribution of a single chain across multiple compute nodes. Pre-fetching has recently been used to sample multiple possible next configurations in parallel under the assumption that a certain fraction of proposals will be rejected [Brockwell, 2006, Strid, 2010]. However, these approaches can result in a significant amount of wasted computation, and are not suitable for the case where the complete model cannot be placed on a single compute node. There has also been work on Metropolis-Hastings updates that do not depend on the current state of the chain [Jacob et al., 2011]. This method can lead to greater asynchrony, but requires an accurate proposal distribution, which is unlikely to be a reasonable assumption in the case of large-scale graphical models. Most similar to our motivation is an existing MPI-based approach for single chains [Wilkinson, 2004]. However, this approach has heavy inter-process communications costs and synchronization requirements, which make it better suited for the multi-core setting than the distributed setting. It also relies on a fixed partitioning of the graph. Recent work on identifying *super-clusters* for Dirichlet process mixtures [Lovell et al., 2012] is also relevant as it models the partitioning for distributed MCMC as auxiliary variables, however the approach is model-specific, relies on synchronous distribution, and has only been applied to small datasets.

In the next two chapters, we will propose techniques for scaling single MCMC chains, using a combination of distributed processing and asynchronous updates. In Chapter 4 we propose a distributed sampling approach using the MapReduce framework. Our method partitions the model and assigns sub-graphs to computation nodes. While performing local inference, our approach compensate for the variables and factors that are shared between machines. We also propose partitioning and block creation that uses the dynamic sparse structures which arise during inference for efficient sampling. Motivated by the adverse effects of the synchrony bottleneck, we explore asynchronous updates for MCMC in Chapter 5. For single machine sampling, we investigate stochastic evaluation proposals by sampling the factors. We also propose distributed frameworks with asynchronous communication for MCMC, exploring both lock-free and locking based architectures. These contributions efficiently sample from a single chain of MCMC, providing speedups that facilitate inference on large, dense models.

CHAPTER 4

SYNCHRONOUSLY DISTRIBUTED MCMC

Divide each difficulty into as many parts as is feasible & necessary to resolve it.
Rene Descartes

Given the widespread use of MCMC inference for complex graphical models, there is a significant need for scaling MCMC to larger models. Distributing the computations is a challenging task because MCMC sampling is inherently sequential in nature. A number of approaches have proposed techniques to parallelize MCMC [Gonzalez et al., 2011, Smola and Narayanamurthy, 2010, Ahmed et al., 2012, Nishihara et al., 2012, Lovell et al., 2012], but most are either model-specific or make shared memory assumptions. Many of these approaches also introduce custom communication protocols that are not easily deployable on existing distributed computing architectures.

In this chapter, we will explore a MapReduce [Dean and Ghemawat, 2004] based distribution strategy of parallelizing single-chain MCMC. The variables are assigned randomly to machines, leading to some factors that neighbor variables on separate machines. Parallel MCMC-chains are initiated using modified proposal distributions that only propose local changes such that factors that lie across machines are not examined. After a fixed number of samples on each machine, we redistribute the variables amongst the machines to explore proposals across variables that were assigned to different machines. We describe the algorithm and the architecture in Section 4.1. Since dividing the variables randomly is often quite wasteful, we explore more efficient partitioning strategies in Section 4.2. In Section 4.3, we analyze our algorithm's convergence and correctness properties.

To demonstrate the proposed distributed MCMC algorithm, we provide experimental evaluation in Section 4.4. We first explore the utility of parallel sampling for efficient marginal inference by evaluating on synthetic models in Section 4.4.1. Specifically, we study the affect of the number

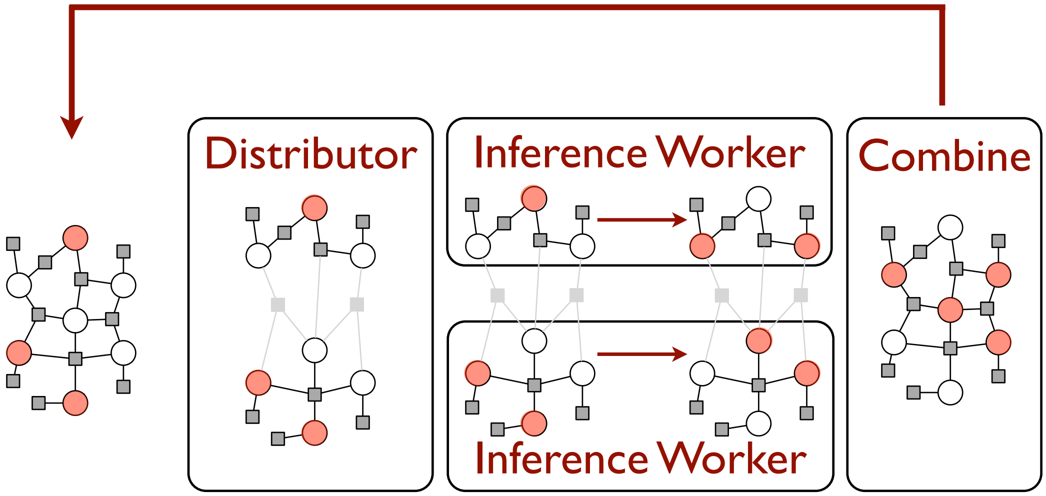


Figure 4.1: **Iterated MapReduce for Synchronously Distributed MCMC:** The algorithm consists of three phases, a *distributor* that partitions the model, *inference workers* that perform independent inference, and a *combine* phase to concatenate the values of the variables.

of inference workers on the quality of the marginals, demonstrating how the restricted proposals effect convergence. We then evaluate our algorithm on real-world information extraction task. First, we evaluate an entity resolution task of disambiguating papers from a large collection of citation texts, demonstrating scalability of MAP inference as the number of workers are increased. For a larger-scale deployment of our approach, we evaluate on the application of cross-document coreference resolution. Given noun phrase mentions from a large document corpus, the problem is to identify clusters of these mentions such that mentions in a cluster refer to the same latent entity. Scalability results using a random redistribution strategy show improved performance when increasing the number of machines. We also explore a hierarchical model that automatically identifies fruitful partitions. Experiments show that the dynamic, hierarchical partitioning converges much faster than random partitioning, even though it contains many more latent random variables. This work has been published as Singh et al. [2010] and Singh et al. [2011b], which can be referred for details.

Algorithm 4.1 Synchronous MCMC: Markov Chain Monte Carlo with Synchronous Distribution

```
procedure SyncMCMC($\mathbf{y}_0$, $\mathcal{G}$, $N$)
    $\mathbf{y} \leftarrow \mathbf{y}_0$                              ▷ Initialize to the initial configuration
    while convergence do
        $\{\mathbf{Y}_c\}_N \leftarrow$ Partition($\mathbf{Y}$, $N$)             ▷ Partition the variables, see Section 4.2
        for $\mathbf{Y}_c$ in parallel do
            $\mathbf{y}_c \leftarrow$ Sample($\mathbf{y}_c$, $\mathcal{G}$)          ▷ Use restricted proposals, see Section 4.1.1
        end for
        $\mathbf{y} \leftarrow$ Combine($\{\mathbf{y}_c\}$)                          ▷ Direct concatenation
    end while
end procedure
```

4.1 Distributed Inference

Our architecture for distributed inference, shown in Fig. 4.1, is built on iterative Map-Reduce [Dean and Ghemawat, 2004], allowing direct deployment on existing systems. The distributor takes the current assignment of all the variables, and partitions the model into as many partitions as workers to create disjoint subsets of variables. This partitioning may be performed randomly, although we study more efficient partitioning later (in Section 4.2). Further, the distributor need not be implemented on a single machine and, depending on the partitioning strategy and the size of the model, can also be parallelized. Each of the sub-models (set of variables) are assigned to inference workers, and the current value of the variables and the factors that lie between them are communicated to the worker. Each worker performs inference on the set of variables that is assigned to it independently (details in the next section). This results in changed values for all the variables without any conflicts since each variable is assigned to only one worker. At the end of the iteration of sampling, resulting configuration of the variables is combined. We repeat the process with the distributor taking this new assignment as input, and initiating another iteration of sampling as described above. We summarize the algorithm in Algorithm 4.1. The distributed synchronous inference can be directly implemented in the MapReduce framework, by combining and distributing in the Map phase, and performing inference in the Reduce phase (or, for sophisticated partitioning schemes, combine and distributor is a single MapReduce, and sampling a separate MapReduce).

4.1.1 Restricting the Proposals

Since MCMC is a sequential process in which each sample may depend on the complete configuration of the previous sample (values assigned to all the variables) as defined by the proposal function, in general it is not possible to parallelize the computation by sampling sub-models independently. However, in practice, proposal functions often make *local* changes to the model, and MCMC inference requires examination of the local neighborhood of the change to evaluate and accept the proposal. If multiple proposals are created such that their local neighborhoods are non-overlapping, then the order in which they are evaluated and accepted are equivalent (i.e. either ordering is a valid Markov chain).

We can utilize this property to evaluate and accept such sets of proposals in parallel. We restrict the proposal function of each inference worker to only propose the changes that require factors that are local to the worker (and do not need to examine the shared factors). By restricting factors to those local to the worker, the sets of factors required to evaluate such proposals across all the workers is disjoint. This removes any ordering constraints between the proposals on different workers, and thus each worker can accept the proposals in parallel without any approximations. Formally, consider a set of variables $\mathbf{Y}_c$ assigned to a worker c on which inference is performed. The set of factors *local* to c is defined by the factors whose neighbors also belong to c, i.e. $\mathcal{F}_c = \{f : \forall Y_i \in \mathcal{N}(f), Y_i \in \mathbf{Y}_c\}$. For a given proposal $q : \mathbf{y} \to \mathbf{y}'$, the factors *local* to q, $\mathcal{F}_q$, are those that participate in the evaluation of q, i.e. appear in $\pi(\mathbf{y}) - \pi(\mathbf{y}')$ after cancellation. The restricted set of proposals for c, therefore, are the proposals that only use the factors local to c for evaluation, i.e. $q_c = \{q : \mathcal{F}_q \subseteq \mathcal{F}_c\}$.

For static factor graphs, i.e. graphs for which the set of factors does not depend on the values of the variables, $\mathcal{F}_q$ is the set of factors neighboring the changed variables in q. Thus a worker c will propose a change q that modifies a single variable Y_i only if the Markov blanket of Y_i is also local to c. This condition can often be quite restrictive in practice. On the other hand, the condition for q_c can be substantially less strict for dynamic factor graphs, in which the set of factors instantiated for a proposal $q : \mathbf{y} \to \mathbf{y}'$ depend not on the changed variables, but on the assignments $\mathbf{y}$ and $\mathbf{y}'$. In this case, the Markov blanket for the modified variables in a given proposal can be much smaller than that of the static graph[1], leading to a smaller $\mathcal{F}_q$ and hence a larger set of valid proposals q_c.

Consider, for example, a model that is fully-factorized for a part of its domain $\boldsymbol{\mathcal{Y}}^C$, i.e.

$$\pi(\mathbf{Y} = \mathbf{y}) = \begin{cases} \psi_f(\mathbf{y}) & \text{if } \mathbf{y} \in \boldsymbol{\mathcal{Y}}^C \\ \sum_i \psi_i(y_i) & \text{otherwise} \end{cases} \tag{4.1}$$

For a proposal $q : \mathbf{y} \to \mathbf{y}'$ such that a single variable Y_j is changed $\mathbf{y}' = (\mathbf{y}|Y_j = y'_j)$, if $\mathbf{y}, \mathbf{y}' \notin \boldsymbol{\mathcal{Y}}^C$, we can evaluate q by only examining the single factor ψ_j. The Markov blanket for Y_j for the equivalent static graph, however, consists of all the variables due to the joint factor ψ_f, and thus Y_j cannot be sampled by a worker that only has a subset of the variables assigned to it.

The above structure can appear as part of a model containing other such structures, and hence this example is more generally applicable. In particular, such factors are common in entity resolution models that contain pairwise factors that specify a preference only if the mentions are in the same cluster, without taking into account which exact clusters the mentions are in. Similarly, skip-chain CRFs [Sutton and McCallum, 2011] contain factors that encourage the labels of their neighbors to be the same. Tree factors prevalent in dependency parsing [Smith and Eisner, 2008] are also examples of such models. Dynamic factors are a formulation of a number of existing representations, such as gates [Minka and Winn, 2008], case-factor diagrams [McAllester et al., 2004], and imperatively-defined factor graphs [McCallum et al., 2009].

Compared to a single machine chain that uses the same *restricted* proposal function (i.e. $\bigcup_c q_c$), this approach provides linear speedup. However, note that a single machine implementation need not use the restricted proposal distribution, and thus may converge using a smaller number of samples. It is difficult to analyze the difference in quality of these samples, since the restrictions placed on the proposal function depends considerably on the graph properties of the neighborhood, number of machines, exploitable structure within the factors, and the partitioning strategy.

[1] A Markov blanket of a variable in a dynamic graph is a subset of the Markov blanket of the same variable in the equivalent static graph (generated by *unrolling*).

As we describe here, the proposals are evaluated correctly in terms of computing the model ratio. The other component of the acceptance score, the backward-forward probability ratio of the proposal function (see Eq. (3.3)), may be difficult to compute in general since it depends on the partitioning strategy and the restrictions imposed on the proposal function. We will provide analysis later in the chapter for a number of partitioning strategies and assumptions on the model structure. Further, fortunately, when performing MAP inference with MCMC, computing backward-forward ratio probability is not required to be exact, and we can use our distributed inference approach directly to compute the MAP configuration for really large, dense models in a scalable fashion.

4.2 Partitioning

The above method of distributed inference assigns each partition of the variables to the workers, i.e. given the set of variables $\mathbf{Y}$, the objective is to create N subsets $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_c, \ldots, \mathbf{Y}_N$ such that $\bigcup_c \mathbf{Y}_c = \mathbf{Y}$ and $\forall c, d, \mathbf{Y}_c \cap \mathbf{Y}_d = \emptyset$. As described in the previous section, the set of variables assigned to a worker defines the set of proposals that the worker can explore, and hence the partitioning can have a significant impact on the performance. In this section, we propose a number of different strategies, and present a case study with entity resolution.

4.2.1 Random Variable Partitions

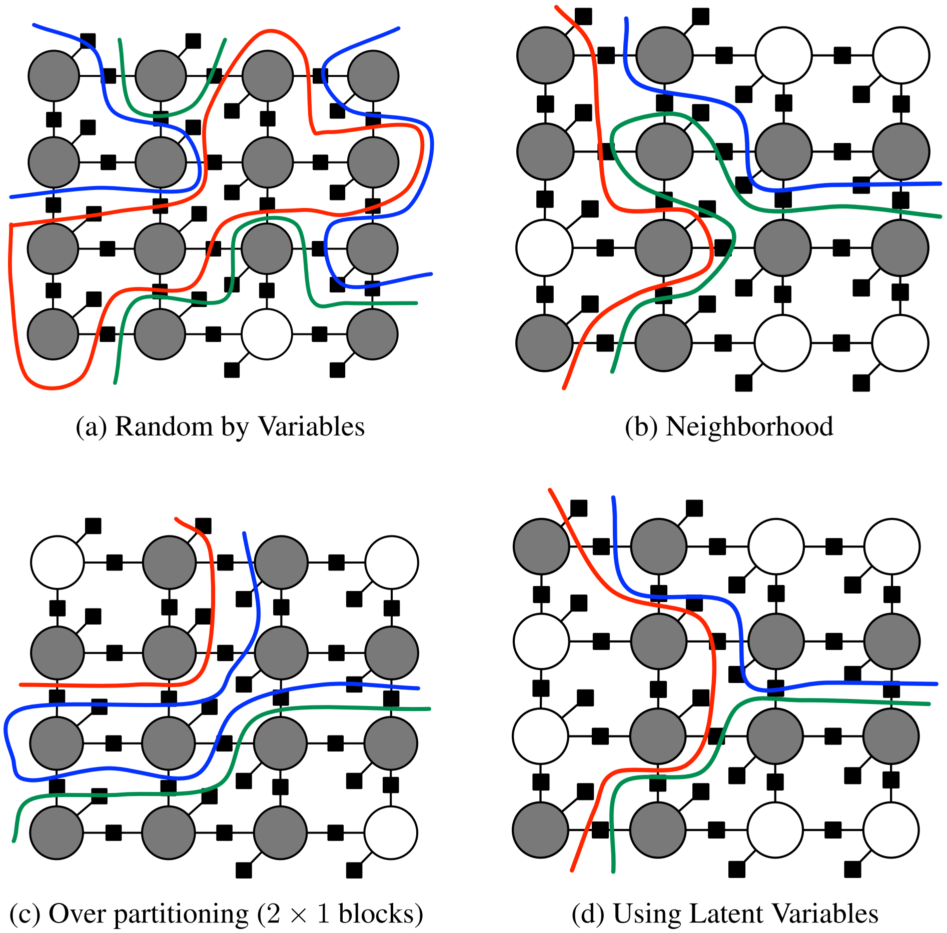
The most direct approach to partitioning a set of variables is to create a random split, i.e. each variable is randomly assigned to a worker. Such partitions create a significant restriction on the proposals that can be explored on each worker. If we assume a simple proposal function that modifies a single variable in a static factor graph, the restrictions arise from the Markov blanket, and the only way a variable will be sampled is if all of its neighbors are assigned the same worker. For a variable that has d_i^v neighbors (set of variables that neighbor the factors, i.e. $d_i^v = |\bigcup_{f \in \mathcal{N}(i)} \mathcal{N}(f)|$), the probability that all of the neighbors will be assigned to the same partition is $N^{-d_i^v}$. This is quite unfortunate as this probability is quite low for non-trivial number of machines, or for models that have high-degree variables. From a different perspective, out of the $\dfrac{|\mathbf{Y}|}{N}$ variables that are assigned to a machine, the expected number of variables that will be actually be sampled is $\frac{|\mathbf{Y}|}{N^{d^v+1}}$, which indicates that sampling will be ineffective unless $|\mathbf{Y}| >> N^{d^v+1}$. Thus we expect distributed inference with random partitioning to fare poorly on non-trivial sized models and high-density variables and factors, however it might be useful for models with low degree components and very few number of machines. We see an example of random partitioning of a 4×4 grid in Fig. 4.2a for which only a single variable can be sampled (in white).

4.2.2 Random Factor Partitions

The restrictions imposed on the proposals arise due to the factors that lie across the workers. Random partitioning is impractical since it does not take the model structure into account, resulting in a large number of factors that lie across workers. Alternatively, one can create a round-robin partitioning that randomly assigns each factor and its neighbors to the workers. Since variables cannot be assigned to multiple machines, we can only select factors for which none of the neighbors have been assigned. The probability that a variable can be sampled by a worker depends on whether all its neighboring factors are assigned to the same machine, which in turn depends on the exact structure of the variable neighborhood, and is upper-bounded by $N^{-d_i^f}$, where $d_i^f = |\mathcal{N}(i)|$. Although this strategy may improve upon the random partitioning of variables on models with high-order factors, it is not substantially different.

4.2.3 Neighborhood-based Partitioning

In this section, we focus on a partition strategy that specifically targets locality. In particular, we pick a random variable, and perform a bread-first search around its neighborhood till the adequate sized partition is created. We perform such a partition in a round-robin fashion, removing the variables that have been assigned to a worker for subsequent searches. Note that this is similar to the splitting strategy used by Gonzalez et al. [2011]. We expect this partition strategy to create partitions where many of the variables have their Markov blanket assigned to the same worker (disconnected components may be created for the last few workers, leading to multiple connected


(a) Random by Variables

(b) Neighborhood

(c) Over partitioning (2 × 1 blocks)

(d) Using Latent Variables

Figure 4.2: **Partitioning of the Grid Model:** Examples of 4×4 grids over 3 machines using the various strategies for partitioning, where colored lines indicate the partitions, and filled-in variable are ones that cannot be sampled (assuming Markov blanket restriction). This examples illustrates the advantages and drawbacks of the proposed approaches.

components assigned to the same worker). One drawback with such a partitioning strategy is that it is not easy to parallelize the partitioning, since assignments for subsequent workers depend on the variables that have already been assigned. Figure 4.2b shows the clear benefits over alternate strategies for a sample 4×4 model since it results in 5 variables that can be sampled.

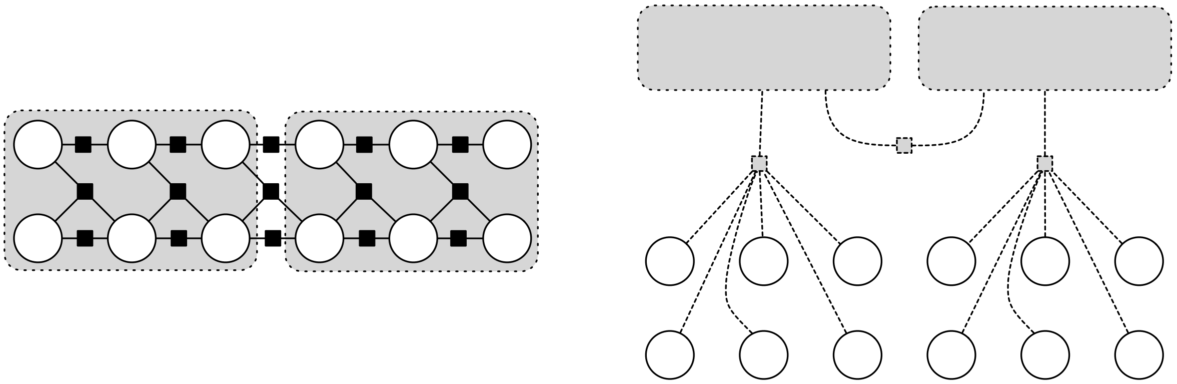
4.2.4 Over-partitioning

Here we propose an alternative to neighborhood partitioning that retains the locality benefits but is distributable. We pre-compute an over-partitioning of the model into M partitions based on locality ($M >> N$), using similar process as neighborhood-based partitioning. Then, at each iteration, M/N of these partitions are randomly assigned to each worker. This partitioning strategy can be easily parallelized, we are selecting a random worker for each of the *over*-partitions. Although we lose some of the locality benefits as partitions that are not overlapping may get assigned to the same worker, this approach provides additional flexibility in using domain-expertise for the pre-computed graph partitioning schemes, as it is performed only once. In Fig. 4.2b, over-partitioning results in fewer variables that can be sampled than neighborhood-based partitioning (3 versus 5), however it may be preferred due to the benefits in scalability and flexibility in initial partitioning.

4.2.5 Using Latent Variables

One of the drawbacks of the above strategies is that it not easy to take into account the values of the variables, the dynamic structure of the model, or the state of inference, in a scalable manner. In particular, the pre-computed set of partitions introduced in the previous section may not always lead to fruitful proposals for the duration of the inference. As inference progresses, the fraction of proposals that are fruitful diminishes considerably, and a large number of samples are required to reach convergence. The goal of partitioning the variables should be to encourage the fruitful proposals, often requiring information not available at pre-computation. Further partitioning with dynamic models is extremely difficult, since it may be too expensive to fully unroll the graph, and local variables as per the partitions constructed using the initial configuration may not remain local during inference (due to changing structure of the model). Finally, it may be difficult to encode sophisticated forms of domain knowledge for dynamic models in the pre-partitioning scheme.

In this section we propose a novel strategy that represents the partitioning task as a graphical model. We represent the partitions as a group of additional N random variables $\mathbf{S} = \{S_c\}$, one for each worker. Each of the random variables is a *set-valued* variable that takes a subset of $\mathbf{Y}$ as



(a) Model used during inference of $\mathbf{Y}$ using $\mathbf{S}$ for partitioning

(b) Model for inferring $\mathbf{S}$ to identify fruitful partitions, conditioned on $\mathbf{Y}$

Figure 4.3: **Latent Variables for Partitioning:** The regular model is represented as usual by using circles for $\mathbf{Y}$ and solid lines/squares for $\mathcal{F}$, while the partitioning model is represented by rounded, grey boxes for $\mathbf{S}$ and dotted lines/squares for Φ. Distributed inference of $\mathbf{Y}$ uses $\mathbf{S}$ for partitioning, while inference of $\mathbf{S}$ uses values of $\mathbf{Y}$, or other statistics.

its value, i.e. $S_c \in \mathcal{P}(\mathbf{Y})$, where $\mathcal{P}$ is the power set function. To represent efficient partitions, we can add factors Ψ between the set-valued variables $\mathbf{S}$ such that these factors encourage fruitful partitions by assigning a higher score to them. The factors in the partition model can observe the dynamic structure of the model, the current values of the variables, and other statistics that reflect the state during inference, providing flexibility of using arbitrary preferences over which partitions are useful. The task of partitioning is now reduced to that of inference of these set-valued variables.

In general the structure of the partition model itself may be quite complicated, and due to set-valued variables, inference is intractable. Although it seems that we have just introduced yet another model to perform inference over, note that our discussion so far is a scalable approach to distributing MCMC inference. Thus, to perform inference for the partition variables, we use our proposed distributed approach as described in the previous sections, however we do not sample the partition variables till convergence. Instead, we switch between inference over regular variables $\mathbf{Y}$ (using the current value of the partition variables to define the assignment of the variables to the workers) and inference over the partition variables $\mathbf{S}$ (i.e. explore different partitions using distributed MCMC to discover more efficient partitions). The models used in these two phases is shown in Fig. 4.3. Unfortunately, we have increased the space over which sampling is defined, and in general this requires more samples to converge (as will be the case on a single machine). However, if the partitioning model is defined such that these additional variables and factors facilitate efficient partitioning, distributed inference can lead to much faster convergence.

Finding the optimal partitioning is a difficult task, and even heuristic/greedy algorithms can take a long time for large graphs. By including it as a graphical model, the task of finding partitions for distributed inference is left to MCMC, which is a good anytime algorithm for optimization. The partitions that are inferred depend on the factor scores, and the model designer can thus declaratively define the properties of a good partitioning. Unfortunately these factors have to be task-specific, and we explore the design for the task of cross-document coreference in the next section.

4.2.6 Case Study: Cross-Document Coreference

In this section we describe the partitioning strategies in context of the model used for large-scale entity resolution, introduced earlier in Section 2.4.1. Given a number of mentions, the task of coreference is to *cluster* the mentions such that mentions in the same cluster refer to the same latent entity (the identity and the number of entities is unknown). The model consists of mention variables that take one of the entities as their values, and entities are set-valued variables over the mentions. The model is defined using a pairwise dynamic factor graph, with *affinity* factors between mentions that are assigned the same entity, and *repulsion* factors between mentions that are not assigned to the same clusters. We describe the model and inference in further detail later in the chapter, in Section 4.4.2.

This model contains a number of properties that make inference challenging. First, the entity variables contain set-valued variables for which the size of the domain is exponential in the number of mentions. Second, the structure of the model is dense and extremely dynamic; number of factors for each configuration is quadratic in the number of mentions, and *unrolling* the model is intractable. Finally, the number of mentions for entity resolution is often quite high, often numbering in the millions. MCMC inference is able to address a number of these concerns, and is particularly efficient in evaluating proposals. For example, if a single mention m is moved from an entity e to another entity e' as a proposal, the set of factors that are used in evaluation of such a proposal are ones that neighbor m and mentions in e and e', and are independent of the rest of the entities and mentions.

We would like to employ our distributed inference framework for large-scale entity resolution, and below, we describe the various partitioning schemes as they apply to this model.

4.2.6.1 Random Partitioning

As mentioned above, to be able to sample a mention variable, we need all the mentions that belong to its entity. If we perform a random partitioning of these variables, the probability that all the mentions of the mention's entity will be assigned to the same machine is incredibly small, since d_i^v as used in Section 4.2.1 is the size of the entity. Similarly, random partitions of factors as described in Section 4.2.2 is likely to fare quite poorly as well.

4.2.6.2 Neighborhood

The local neighborhood selects random variables, and performs breadth-first search with the selected variable as the root. For the entity-resolution model, selecting a random mention and performing naive BFS would result in *all* variables, as all the mentions are connected to each other. However, as described earlier, the model is structured such that mentions belonging to the same entity as the mention that is selected are required for evaluating proposals, and other mentions are not. Although we can easily modify this scheme to select other mentions in the entity first, it is not clear how to augment the set of variables if the number of mentions is smaller than the required size of the partition. To expand the neighborhood to other mentions or entities, one has to identify *similar* mentions and entities, in general requiring a quadratic number of similarity values. Instead, our neighborhood partitioning takes all the entities in the model, and assigns them randomly to the partitions (each mention is assigned to the worker that its entity is assigned to).

4.2.6.3 Over partitioning

In the context of entity resolution, a precomputed over-partitioning corresponds to finding small sets of similar mentions that are constrained to be assigned to the same machine for the purpose of inference. Unfortunately, this requires a quadratic number of comparisons in general, and more importantly, a random assignment of these partitions to workers may not result in all mentions of an entity assigned to the same worker, leading to ineffectual sampling. On the other hand, one can imagine the neighborhood model above as an instance of dynamic over-partitioning, whereby each entity acts as a partition, and are randomly assigned to the workers.

Identifying sets of similar mentions, however, can be extremely useful for sampling. Consider, for example, a successful proposal that moves a mention m from entity e to e'. The acceptance of such a proposal indicates that the mention did not belong in e, and it is likely that other mentions in e that are very similar to m also belong to e'. However, the probability of the sampler proposing such moves is incredibly small, especially with a large number of entities and mentions. To address this problem, we propose a model that identifies such clusters of mentions (*sub-entities*) that are very similar to each other. The model is similar to the latent partitioning model described in Section 4.2.5 and in the next section, and differs in that the factors encourage a very large number of small clusters. As with the latent variables for partitioning, we use the distributed inference to both sample the regular mentions (the proposals move sub-entities of mentions instead of single mentions) and the sub-entities (the proposals move mentions between sub-entities). Although over-partitioning scheme is not appropriate for partitioning for distributed MCMC, the dynamic variation proposed here improves per-machine proposals.

4.2.6.4 Latent Variables

To use latent variables for partitioning the model, we need to define the partitions that we prefer. As we near convergence, many of the entities tend to be quite accurate, with a small fraction of mentions from the wrong entities. Proposals that move the mentions to the correct entities are only possible if the correct destination entities is also assigned to the same worker. Random partitioning of entities, as described above in the neighborhood scheme, would likely not assign such entities to the same worker, especially for a large number of workers. To perform fruitful proposals, we would like to assign entities to the same worker that are similar to each other, since the mentions between them are more likely to be mixed up.

This suggests that the partition model that encourages similar entities to be assigned to the same worker may be more efficient. To facilitate such a *clustering* over the entities, the partition model contains set-variables that represent groups of similar entities (we call these additional variables *super-entities*). The factors between super-entities are similar to the pairwise affinity factors, the former computes the similarity of two entities, while the latter between two mentions. As we describe in Section 4.2.5, inference switches between inference of entities by using super-entities as the partition, and inference of super-entities by using the current assignment of the entities.

4.3 Analysis

In this section, we analyze the validity of our approach for synchronous distribution in context of different partitioning schemes. We make the following assumptions for the rest of this section. First, we assume that the unrestricted proposal function is a valid proposal function for the single-machine implementation, i.e. it is irreducible and aperiodic. Second, we assume each proposal is a change to a single variable at a time, and further, that the proposals are restricted due to the Markov blanket property (we only sample variables that have all the variables in their Markov blanket also assigned to the same worker). Third, although we perform a fixed number of iterations per worker before re-partitioning, the analysis is described for a single sample per worker (same result applies to multiple samples per worker). Fourth, we assume that for each variable in our model, there exists at least one partitioning of the model that allows the variable to be sampled, and further, at least one such partitioning for each variable can be generated by our partitioning schemes from Section 4.2.

4.3.1 MAP Inference

We first consider maximum a posteriori (MAP) inference. Since the objective of MAP inference is to identify the assignment to the variables with the highest model score, we can perform the MCMC chain and keep track of the highest-scoring sample. This is not trivial in the distributed setting, since the variables are distributed over multiple machines. However, since the variables are not conflicting, we can keep track of the local assignment that has the highest score as defined by the local factors. In the combine stage, we compute the global assignment by combining the local highest scoring assignments. To see that the combined assignment will be the highest scoring assignment even when the cross-machine factors are included, note that the proposals on each machines do not use the cross-machine factors during sampling. Thus the score according to the cross-machine factors is the same at the beginning of iteration as it is at the end, and hence the combined assignment is the highest scoring assignment. When partitioning the variables and assigning them to the workers, we also communicate the values of the variables from the best-scoring assignment, and each worker computes an initial best local score by using the factors that are local to the worker.

The model score is computed exactly for the restricted set of proposals by design, as we show in Section 4.1.1. For many of the partitioning schemes, we can compute the forward-backward ratio exactly, as we describe in Section 4.3.2. However, since the forward-backward ratio condition can be relaxed for MAP inference, we have a lot more flexibility in the choice of partitioning strategy. Thus we can easily employ latent variable partitioning or manually-specified partitions such as blocking and canopies [McCallum et al., 2000], or any arbitrary combinations of partition schemes. To encourage MAP inference to explore high-probability regions of the distribution, we include an additional temperature as described in Section 3.3. The temperature is lowered after every partitioning stage, however can be just as easily included in the workers if necessary.

4.3.2 Marginal Inference

To show that our distributed inference converges to a valid Markov chain with the desired stationary distribution, we have to show the following properties:

1. **Model ratio:** The ratio of model scores is computed exactly for each proposal

2. **F/B ratio:** The forward-backward ratio is computed exactly for each proposal

3. **Proposals:** The restricted proposal function is irreducible and aperiodic

By the definition, the restricted proposal function only proposes the changes to local variables that do not use factors across the machines for computing the ratio of the model scores, i.e. (1) is trivially true for all of the partitioning schemes.

For the purpose of this analysis we assume our proposal function modifies a single variable at a time. The forward-backward ratio for such a proposal depends on the forward-backward ratio of the original proposal function, and the probability that the variable will be in the sample pool again (will have its Markov blanket on the same machine). If the partitioning does not depend on the values of the variables, the probability of the variable being selected for sampling does not change, and hence the forward-backward ratio of the proposal function can be used directly. Thus we see that (2) is true for random, neighborhood, and over-partitioning.

With our random, neighborhood, and over-partitioned schemes, there is a non-zero (but small) probability that a variable and its Markov blanket will be assigned to the same worker. Thus every variable will be sampled with a non-zero probability, demonstrating irreducibility. Further, since the partitioning and the sampling per worker is random, the proposal distribution is also aperiodic. Hence (3) is also true.

4.3.2.1 Latent Variables Partitioning

Analysis of (2) and (3) do not hold for the partition scheme with the latent variable models. If inference for latent variables converges, the partitioning might be near deterministic. i.e. some variables may not be sampled as their Markov blanket is split across partitions. This breaks the irreducibility condition, however, it can be easily fixed by using a single round of random partitioning after every k rounds of latent variable based partitioning.

The more crucial issue arises from the fact that for latent variable partitions, the partitioning depends on the values of the variables (potentially along with other sources of information). This changes the probability that a variable will be selected for sampling, hence changing the backward probability. To compute the exact forward-backward probability, one would need to combine the forward-backward ratio of the latent variable partition model with the forward-backward ratio of the original proposal function. This is difficult to compute in general; we thus use latent variable partitions only for MAP inference.

4.4 Experiments

We now turn to empirical evaluation of our synchronous distributed MCMC architecture. In Section 4.4.1, we provide experiments on synthetic models to assess the utility of the architecture for marginal inference. In order to demonstrate applications to real-world tasks, we perform MAP inference for citation deduplication in Section 4.4.2, and large-scale cross-document coreference task containing millions of variables in Section 4.4.3.

4.4.1 Marginal Inference on Synthetic Models

In this section, we evaluate the utility of our approach on marginal inference for synthetic models for which we can compute the exact marginals. We generate synthetic models with 24 binary variables, each with a local factor s.t. $\psi_i(y) = (-1)^y e_i$ where $e_i \sim \mathcal{N}(0,1)$. For every variable, we select 2 other random variables as neighbors, and create the following pairwise factor $\psi_{ij} = \begin{cases} e_{ij} & y_i = y_j \\ -e_{ij} & o.w. \end{cases}$, where $e_{ij} \sim \mathcal{N}(0,1)$. We simulate our distributed inference scheme with random, over-partitions and neighborhood-based partitions on $N = 1,2,3,4$ and evaluate the marginal error over time (measured as *effective* number of samples). The plots are shown in

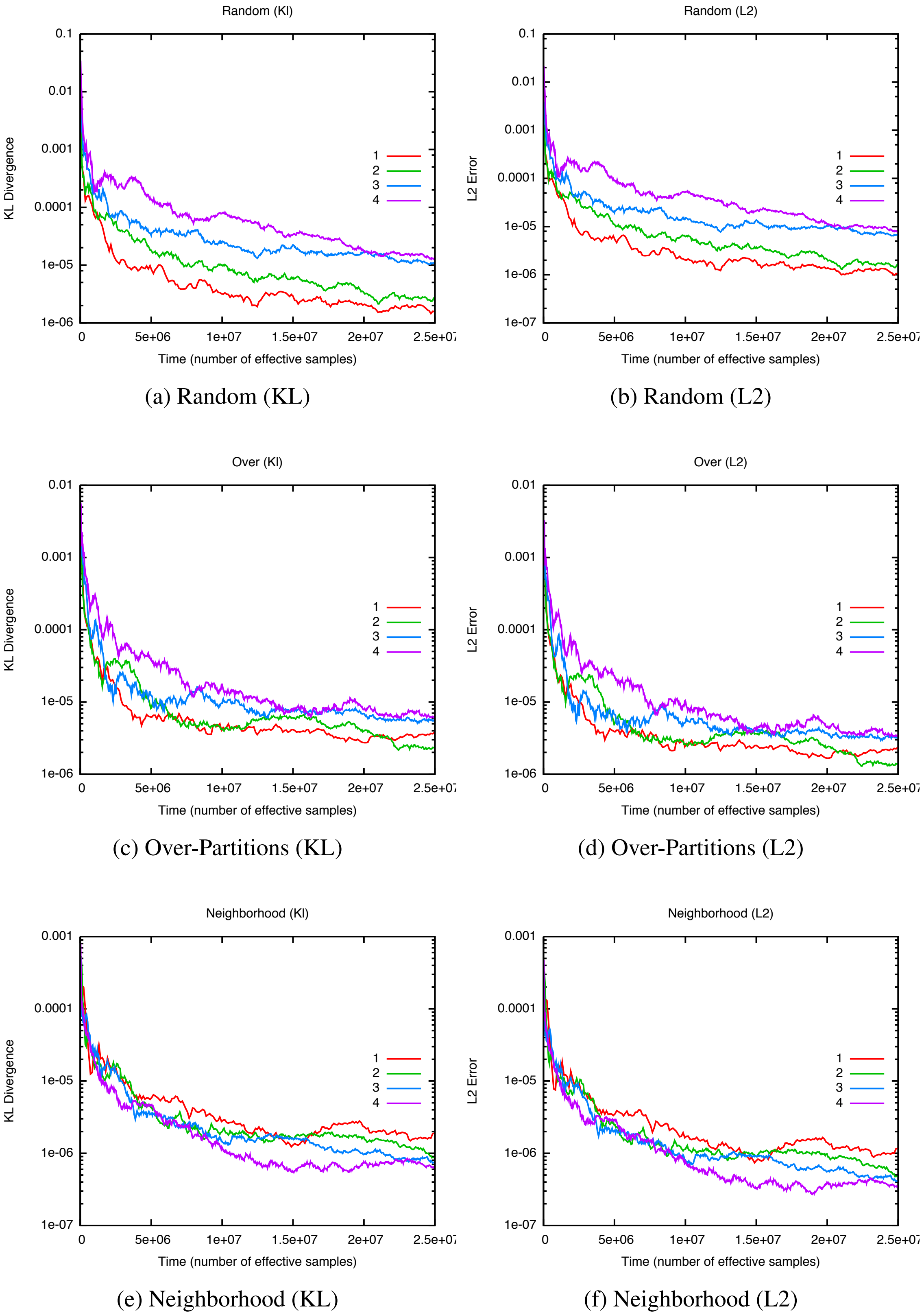
Fig. 4.4. Note that this experiment assumes the time for partitioning and communication over the network is negligible in comparison to sampling.

We can observe a number of interesting aspects from these figures. First note that the random partitioning performs poorly with higher number of workers. Since the model is quite small, creating random partitions often allows only a small number of variables that can be sampled, e.g. for $N = 3$ the average number of variables out of 24 that can be sampled in each round is only 2.67. Over-partitions do take the locality of the graph into account, however assign the partitions randomly, leading to a significant improvement over random partitioning in Figs. 4.4c and 4.4d yet still fail to demonstrate faster convergence for more workers. Finally, in Figs. 4.4e and 4.4f we see that for the neighborhood partitioning, performing a breadth-first search on the graph results in effective use of more cores, leading to a faster convergence to lower error marginals.

4.4.2 Citation Resolution

As an evaluation of synchronously-distributed MCMC on a real-world application, we select the task of citation resolution. Given a large number of citations (that appear at the end of research papers, for example), the task is to group together the citations that refer to the same paper. The citation matching problem is an instance of entity resolution (introduced in Section 2.4.1 and studied in Section 4.2.6), in which observed mentions need to be partitioned such that mentions in a set refer to the same underlying entity. Note that neither the identities, or the number of underlying entities, is known.

The graphical model of entity resolution consists of observed mentions $\mathbf{m}$ (citations in this case), and set-valued entities $\mathbf{e}$ that take sets of mentions as their values and represent whether the mentions refer to the same latent entity (in our case, whether all the citations refer to the same paper). The model consists of two sets of pairwise factors: (1) *affinity* factors between mentions that belong to the same entity (their score is high if the mentions are similar), and (2) *repulsion* factors between mentions that belong to different entities (the score is high if the mentions are dissimilar). The model is therefore an instantiation of dynamic factor graphs since the set of



(a) Random (KL)

(b) Random (L2)

(c) Over-Partitions (KL)

(d) Over-Partitions (L2)

(e) Neighborhood (KL)

(f) Neighborhood (L2)

Figure 4.4: **Synchronous MCMC for Marginals:** Error in marginals using KL divergence and L2 metrics, when varying the number of cores, and for different partitioning schemes.

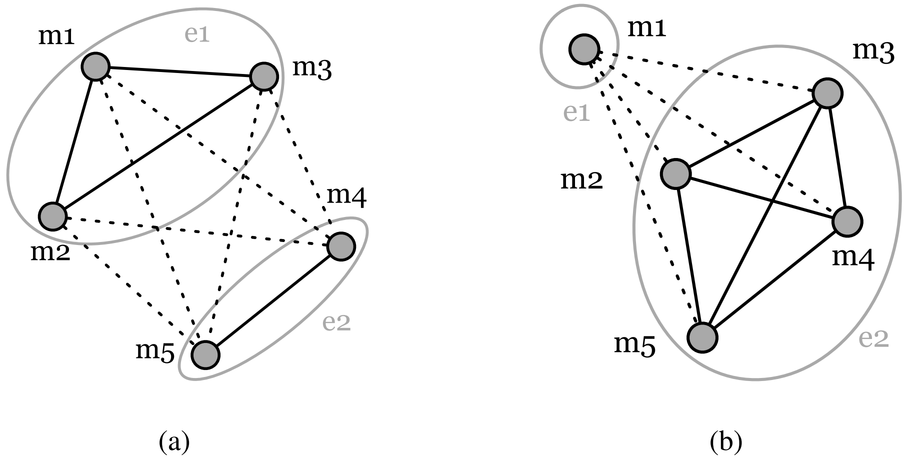


Figure 4.5: **Graphical Model for Entity Resolution:** Two entity resolution configurations defined over 5 mentions (solid, grey circles), with the setting of the variables resulting in 2 entities (light-gray circles). The factors between mentions in the same entity represent affinity (solid lines), and the factors between mentions in different entities represent repulsion (dashed lines). For clarity, we omit the squares from the factors.

factors depends on the current assignment of the entities. Thus,

$$\pi(\mathbf{e}) = \sum_{m_i : m_i \in e} \sum_{m_j \in e} \psi^a_{ij} + \sum_{m_j \notin e} \psi^r_{ij}. \tag{4.2}$$

The set of possible worlds consists of all settings of the $\mathbf{e}$ variables, i.e. all possible clusterings over the mentions. Examples of the model defined over 5 mentions is given in Fig. 4.5. This representation is equivalent to Model 2 as introduced in McCallum and Wellner [2004]. As opposed to belief propagation and other approximate inference techniques, MCMC is especially appropriate for the task as it can address set-valued variables and directly enforces transitivity. This model can be easily extended to using entity-based factors [Culotta et al., 2007].

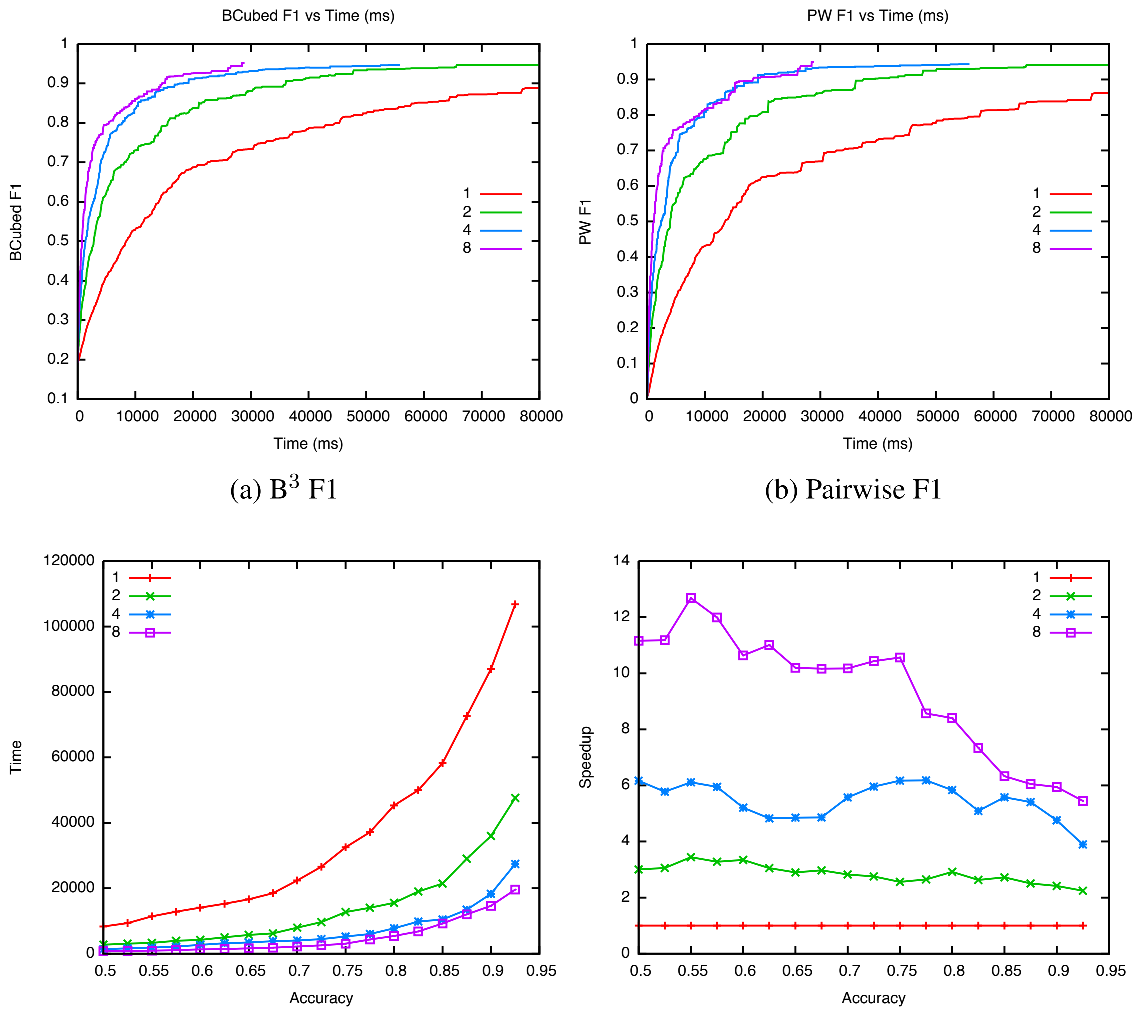
When performing MCMC, we initialize to *singleton entities*. Our proposal function for Metropolis Hastings selects a random mention m and moves it form its current entity e to a random entity e'. It is easy to see that to evaluate such a proposal, all the factors but for the ones between m

and mentions in e and e' cancel out in the model ratio. Evaluation of such a proposal thus requires scoring a number of factors linear in the size of the entities involved in the change.

Returning to our problem of citation resolution, our mentions consist of citation strings from the `References` section of a collection of scientific articles. In particular, we evaluate on the Cora dataset [McCallum et al., 1999], used previously to evaluate a number of information extraction approaches [Pasula et al., 2003], including MCMC based inference [Poon and Domingos, 2007, Singh et al., 2009]. The dataset consists of 1295 mentions, that refer to 134 true underlying entities. We use the same features for our model as Poon and Domingos [2007], using true *author*, *title*, and *venue* segmentation for features. Since our focus is on evaluating scalability of inference, we combine all the three folds of the data, and train the model using Samplerank [Wick et al., 2011]. Since the size of this dataset is quite small, we use our own multi-core implementation of the distribution architecture, instead of using an off-the-shelf Map-Reduce implementation.

As we describe in Section 4.2.6, all the mentions of each entity need to be assigned to the same partition to be able to sample the entity. We focus only on random partitioning of the entities in this section, and perform $10,000$ steps of sampling within each worker. Since we are interested in the MAP configuration in this section, we use a low temperature ($t = 0.001$) during sampling. The speedups as the number of machines are varied is presented in Fig. 4.6, showing the accuracy (measured using two standard entity resolution metrics[2]) against wall-clock running time. We present an alternate representation of the same data in Figs. 4.6c and 4.6d, showing the time taken and the speedup obtained to achieve a fixed level of B^3 accuracy. From the results, we find that an increase in the number of inference worker leads to faster convergence. This speedup is almost linear for up to 4 workers (single machine implementation is 4 times slower to reach an accuracy of 95% B^3 F1). However, using additional processors do not provide substantial gains, for example $N = 8$ only slightly outperforms $N = 4$ workers on B^3, and for the pairwise (PW) metric even that benefit is not evident. For the initial rounds of sampling, we find that the speedup is in fact

[2] Pairwise metric uses standard precision and recall using decisions between pairs of mentions as a binary task, while B^3 is a coreference evaluation metric introduced by Bagga and Baldwin [1998].



(a) B^3 F1

(b) Pairwise F1

(c) Time to reach fixed B^3

(d) Speedup to reach fixed B^3

Figure 4.6: **Citation Resolution with Synchronous MCMC:** Evaluation of scalability of Synchronous MCMC by varying the number of workers for citation resolution on the Cora dataset.

super-linear, providing up to $12\times$ speedup for $N = 8$ and up to $6\times$ speedup for $N = 4$. This can be attributed to the fact that any partitioning of the mentions in the beginning is useful by itself (without taking parallelism into account), and the acceptance rate is higher at this stage for sampling under the partitioning constraints than over all the mentions as in the single machine scenario. This suggests that our approach would benefit single-machine implementations as well.

And we can see **Barack Obama** walking slowly, behind Speaker Nancy Pelosi and Senator Dianne Feinstein...

... the economy by unleashing another frustrated tirade against **Barack Obama**," Tommy Vietor, an Obama campaign spokesman...

People who heard President **Barack Obama** out on the subject here last month left a meeting divided...

Secretary of State **Hillary Rodham Clinton**, just last week, proclaimed that the world was entering...

Hillary Rodham Clinton speaks no foreign languages, but has visited 90 countries...

... played a central role, with Secretary of State **Hillary Rodham Clinton** and Mr. Mitchell taking part in the meetings.

(a) Original set of mentions

And we can see XXX walking slowly, behind Speaker Nancy Pelosi and Senator Dianne Feinstein...

... the economy by unleashing another frustrated tirade against XXX," Tommy Vietor, an Obama campaign spokesman...

People who heard President XXX out on the subject here last month left a meeting divided...

Secretary of State XXX, just last week, proclaimed that the world was entering...

XXX speaks no foreign languages, but has visited 90 countries...

... played a central role, with Secretary of State XXX and Mr. Mitchell taking part in the meetings.

(b) Mentions when provided to the coreference system

Figure 4.7: **Person-X Evaluation:** We illustrate the evaluation by taking 3 mentions each of Hillary Clinton and Barack Obama, and *hide* the string representations before sending them to the coreference system. Output is evaluated using the original entity names. Note that the disambiguation is quite a difficult task for this evaluation.

4.4.3 Large-scale Cross-document Coreference

In this section, we describe results on a large-scale coreference task. In particular, we study the task of cross-document coreference, i.e. given a large number of mentions from a text corpus, the task of identifying the latent entities and assigning the mentions to the entities. The model is same as described earlier in Section 4.2.6, consisting of set-valued entity variables, and pairwise *affinity* and *repulsion* factors. For synchronous distribution, we randomly assigns entities to workers (assigning the mentions of an entity to the same worker). The restricted proposal requires moving the mentions amongst the entities assigned to the same worker. See Singh et al. [2011b] for a description of the model and the restricted proposal function.

Due to the lack of labeled cross-document coreference data, we use Person-X evaluation [Gooi and Allan, 2004] for benchmarking our approach. Person-X evaluation identifies a number of

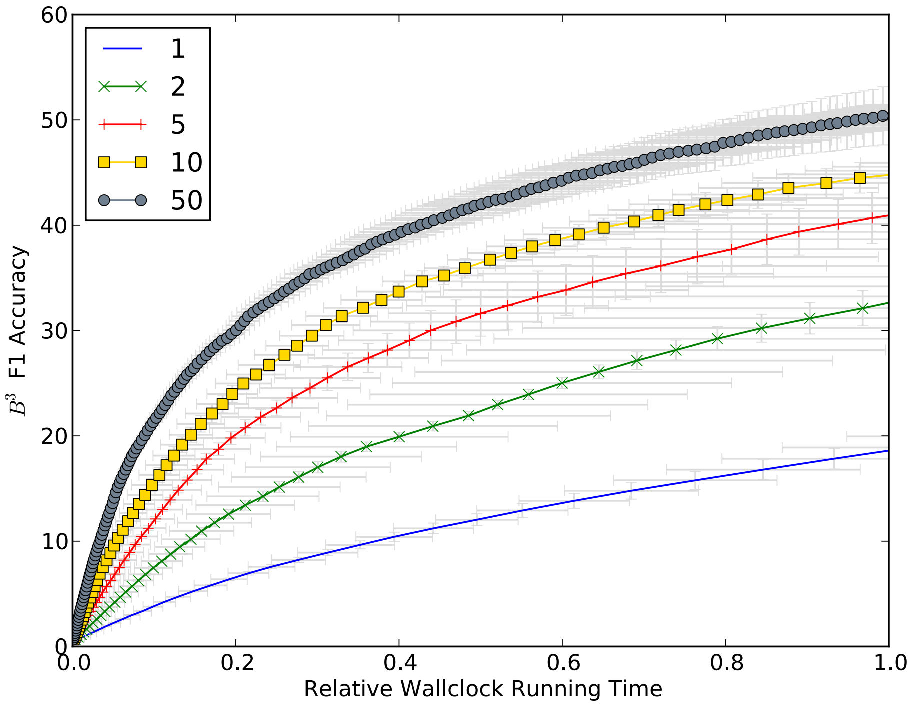


Figure 4.8: Speedups on PersonX Evaluation from Synchronous Distribution

entities and their mentions that are unambiguous based on the string names, however *hides* the string names (replacing them by X) so that the mentions need to be disambiguated solely based on the context. This is an artificially difficult task, as we can see in Fig. 4.10a. We extract $25,000$ mentions for 50 entities from the NYT dataset [Sandhaus, 2008]. The similarity of mentions is computed using a shifted cosine distance of the bag of words of the context around the mentions. We perform rounds of 1 million inference steps between re-partitioning of the model. The results are averaged over five runs.

Figure 4.8 shows accuracy obtained by our proposed inference technique as the number of machines is varied. As we increase number of machines, the rate of convergence increases dramatically. For the lower end of the plot, we can see at least a linear speedup (super-linear in the very early stages, as we observed in the previous section). This gain drops off as we increase the number of machines, since the proposal function gets more restricted as the number of partitions of the model increase, and thus requires more samples to obtain the same accuracy. For example,

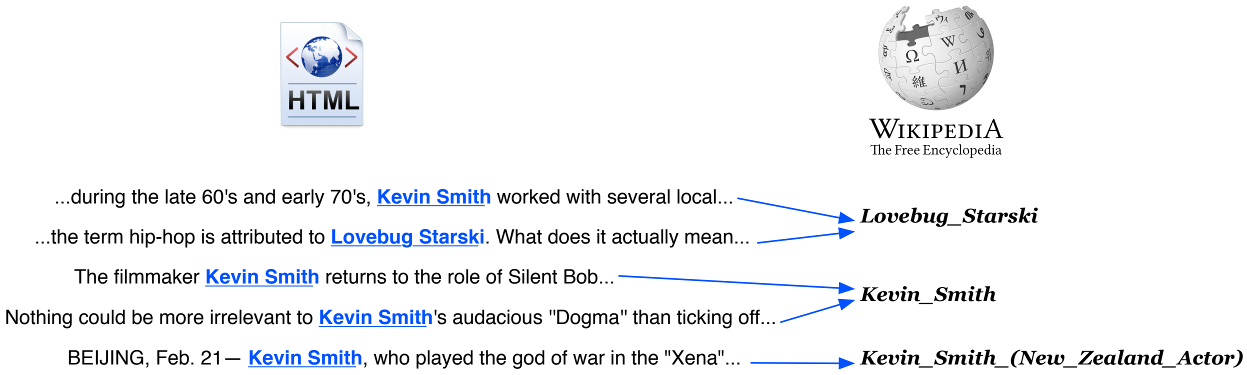

HTML
WIKIPEDIA
The Free Encyclopedia
...during the late 60's and early 70's, Kevin Smith worked with several local...
...the term hip-hop is attributed to Lovebug Starski. What does it actually mean...
The filmmaker Kevin Smith returns to the role of Silent Bob...
Nothing could be more irrelevant to Kevin Smith's audacious "Dogma" than ticking off...
BEIJING, Feb. 21— Kevin Smith, who played the god of war in the "Xena"...
Lovebug_Starski
Kevin_Smith
Kevin_Smith_(New_Zealand_Actor)

Figure 4.9: **WikiLinks:** Automatically labeled dataset for large-scale coreference by extracting links to Wikipedia from the web.

with 50 machines, we are dividing all the entities amongst the workers, resulting in only 1 entity per machine for the proposal function to change towards the end of inference.

Next we evaluate our latent variable models for partitioning and blocked sampling, as described in Section 4.2.6 as super-entities and sub-entities respectively. The latent models for super-entities and sub-entities are similar to the coreference model, as they also express distributions over clusters such that similar points are assigned to the same cluster. These models, however, differ in the similarity they expect clustered mentions to use; the super-entities use a lenient threshold than regular coreference, while sub-entities uses a stricter threshold (both manually set). We use these additional latent variables to run inference on the New York Times dataset over 50 machines, and show results in Fig. 4.10a. The addition of super-entities to the random partitioning (referred to as “pairwise”) results in a significant improvement. Similarly the addition of sub-entities to the pairwise model provides a considerable boost, however the combination of both the latent variable models with the pairwise model results in the best performing sampler that converges quickly.

We also apply this inference technique to a much larger coreference dataset containing 1.5 million mentions called WikiLinks. We create this dataset by extracting links from the web to Wikipedia pages as mentions, and using the target of the links (the Wikpedia page) as the entity label of the mention. See Fig. 4.9 for an example extraction. In this experiment we use a subset, the

Method	**Pairwise**		**B^3 Score**	
	P/ R	F1	P/ R	F1
String-Match	30.0 / 66.7	41.5	82.7 / 43.8	57.3
Subsquare	38.2 / 49.1	43.0	87.6 / 51.4	64.8
Our Model	44.2 / 61.4	**51.4**	89.4 / 62.5	**73.7**

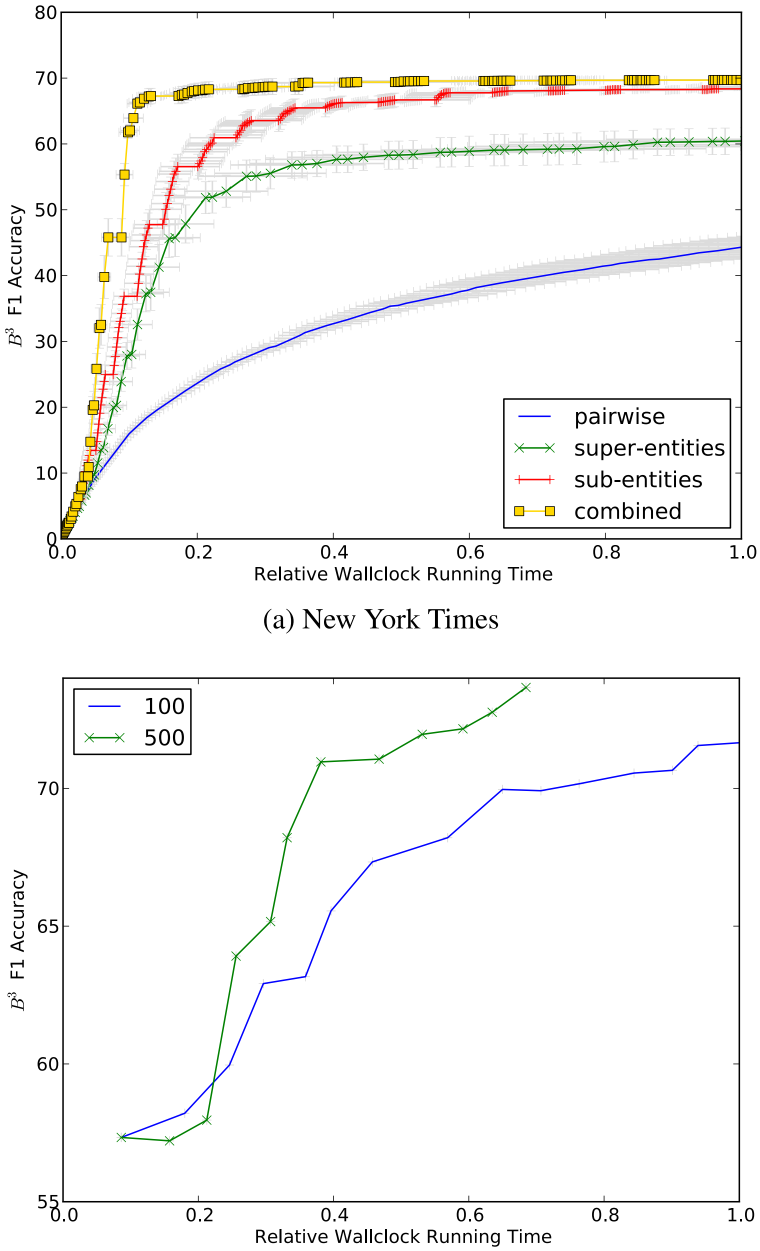
Table 4.1: **F1 at Convergence on WikiLinks Data:** . The results are significant at the 0.0001 level over Subsquare according to the difference of proportions significance test.

complete dataset has since been released for public use [Singh et al., 2012a]. Inference is run for 20 rounds of 10 million samples each, distributed over N machines. We use $N = 100, 500$ and the B^3 F1 score results obtained for each case are shown in Figure 4.10b. It can be seen that $N = 500$ converges to a better solution faster, showing effective use of parallelism. Note that for a corpus of this size, evaluating the pairwise scores at the end of each round is computationally expensive. As a result, we compute the B^3 scores at each iteration and compute the pairwise scores only for the final converged model. Table 4.1 compares the results of our approach (at convergence for $N = 500$), the baseline mention-string match and a state of the art distributed clustering algorithm, Subsquare [Bshouty and Long, 2010]. Our approach significantly outperforms the competitors.

4.5 Related Work

In this section we contrast our approach directly with some of the related approaches. For a general overview of scaling MCMC, we refer the reader to Section 3.4.3.

Asuncion et al. [2009] and Smola and Narayanamurthy [2010] distribute MCMC-based inference for topic models. Our approach is similar, but we do not calculate probabilities (marginals) for the entity resolution applications. This allows us to specify non-random redistribution and customized proposal functions, which can lead to faster convergence. Further, these techniques do not apply to general undirected graphical models, although the class of models they address have been expanded recently by Ahmed et al. [2012]. Low et al. [2010] introduce the *GraphLab* library that distributes computation when specified on nodes and edges. It is not straightforward to express our model as computations on nodes and edges because of set-valued variables and hierarchical mod-



(a) New York Times

(b) WikiLinks data

Figure 4.10: Accuracy Improvements when using Latent Variables for Partitioning

els. In comparison to the above approaches, we allow the most flexibility in specifying structure, partitioning strategy, and the proposal function.

Our latent variable model for partitions and blocked sampling is similar to some approaches proposed since we published this work. Work on identifying *super-clusters* for Dirichlet process mixtures [Lovell et al., 2012] is relevant as it models the partitioning for distributed MCMC as auxiliary variables, however the approach is model-specific and has been applied only to small datasets. It is also one of the few examples of distributed MCMC on a synchronous distribution framework. Recent work by Venugopal and Gogate [2013] propose an approach similar to sub-entities for identifying which variables to sample in combination. Their approach is also dynamic, however they solve the optimization for identifying sub-entities in a greedy manner, as opposed to our use of MCMC.

Our work on informed partitions is also related to Gonzalez et al. [2009b], where message passing is distributed by efficiently splitting the graph to minimize communication. This method is not applicable when the graph structure changes with every configuration, or the ground graph[3] is fully connected. Furthermore, set-valued variables are difficult to incorporate into message passing.

4.6 Conclusion and Future Work

Motivated by the need for scalable, distributed inference for large, dense graphical models that can be deployed on existing architectures, we proposed synchronous distributed MCMC. In each iteration, the algorithm partitions the model amongst the workers, which then sample their local variables independently using a restricted sampler. We proposed a number of partitioning strategies, including a novel model-based partitioning that is dynamic, adaptive to the state of inference, and can employ domain knowledge for fruitful partitions. We compared the convergence

[3] In our work, as in much of the previous work [Culotta, 2008, McCallum et al., 2009], the dynamic model is specified implicitly. The *ground* graph is the result of instantiating all the factors into a possibly very large graph-structured model.

correctness of the resulting MCMC chain, and demonstrated speedups on citation resolution and large-scale cross-document coreference.

A number of avenues exist for future efforts. Our architecture assumes that the model is sufficiently compact such that the transfer of all the variables and their assignments over the network is not a bottleneck, which may not be reasonable for many models. Further, the partition scheme may not be load balanced well, and may assign a large number of variables to a single machine. Although our asynchronous MCMC approaches introduced in the next chapter partly address this concern, it is possible to design a partitioning scheme that is load-balanced yet efficient and converges to the desired posterior. Finally, although we show that our samplers converge, we do not have a theoretical understanding of how the number of machines influences the convergence speed. Although more machines leads to more samples per minute, each sample is of a poorer quality as the proposals are restricted, and we do not expect a linear speedup.

CHAPTER 5

ASYNCHRONOUS MCMC SAMPLING

Many shall run back and forth, and knowledge shall be increased.

Daniel 12:4

Although we obtain impressive results when using synchronously distributed MCMC inference in the last chapter, there exist a number of concerns that may restrict scalability. One of the biggest drawbacks of the system is the synchronization *barrier* imposed by the distributor, i.e. we require all workers to finish inference before re-partitioning can begin, and for partitioning to finish before any inference can begin. This causes a significant hit on performance in a heterogeneous cluster of nodes, since the approach is only as fast as the slowest machine (also known as the curse of the last reducer [Suri and Vassilvitskii, 2011]). Second, even though some of our partitioning schemes can be distributed, it still provides a significant bottleneck to inference, especially for large models. Third, the restrictions imposed on the proposal function can sometimes be extremely severe and lead to lower quality samples. Last, the synchronous inference architecture is designed around a fixed number of machines, and it is not trivial to add/remove computing nodes as inference is progressing.

In this chapter we explore a number of techniques that provide efficient MCMC inference whilst addressing some of the concerns above. Since the restrictions on proposal function arise from exact computation of the model scores, in Section 5.1 we propose a novel sampling technique that approximates the model score evaluation by using only a subset of the factors for every sample. Although this results in inexact sampling, experiments show significant improvements on a single machine implementation, suggesting approaches that break some of the restrictions on the proposal functions imposed in the previous chapter may still provide efficient inference. In Section 5.2 we demonstrate that the synchronous bottleneck can have a substantial impact on the scalability. We propose a distributed MCMC inference framework in Section 5.3 that supports the advantages of the using asynchronous communication however retains the restrictions on the proposals to allow exact model score computation. We evaluate the approach on two real-world entity resolution tasks, citation resolution and author disambiguation, demonstrating improvements over synchronous distribution, especially for heterogeneous workers. Finally, in Section 5.4, we investigate a completely lock-free distribution framework for MCMC inference, that combines asynchronous communication with no restrictions on the proposals (leading to inexact sampling). We provide a number of strategies for resolving conflicts amongst the inference workers. The work presented in this chapter has previously appeared as Singh and McCallum [2011], Singh et al. [2012b], and Singh et al. [2012c], and conflict-free asynchronous distributed MCMC was used for inference in Wick et al. [2013].

5.1 MCMC with Stochastic Proposal Evaluation

In the previous chapter, we proposed a synchronously distributed MCMC inference algorithm that partitions the model in each iteration, and performs sampling independently on each subgraph using a restricted proposal function. The proposal function is restricted since the factors that cross workers cannot be used for exact computation of the model score ratio. These restrictions imposed on the proposal function, due to sub-optimal partitioning or high density of the model, lead to poor quality of samples on each worker, and can result in poor scalability as the number of workers is increased. Here, we study the effect of not imposing an exact model score constraint, and investigate its utility on single-machine, single-chain inference.

Computing acceptance exactly for Metropolis Hastings can be slow in situations where (a) the model exhibits variables that have a high-degree (neighbor many factors), (b) proposals modify a substantial subset of the variables to satisfy domain constraints (such as transitivity in entity resolution), or (c) evaluating a single factor is expensive, such as when features are based on string-similarity. For example, the seemingly innocuous proposal changing the entity type of a single entity requires examining all its mentions, i.e. scoring a linear number of factors (in the number of mentions of that entity). Similarly, evaluating coreference of a mention to an entity also requires scoring factors to all the mentions of the entity, as we saw in the previous chapter. Often, however, the factors are somewhat *redundant*, for example, not all mentions of the "USA" entity need to be examined to confidently conclude that it is a COUNTRY, or that it is coreferent with a "US of A" mention.

In this section we propose an approximate MCMC framework that facilitates efficient inference in high-degree graphical models. In particular, we approximate the acceptance ratio in the Metropolis Hastings algorithm by replacing the exact model score with a stochastic approximation that samples from the set of relevant factors. We explore two sampling strategies, a fixed proportion approach that samples the factors uniformly, and a dynamic alternative that samples factors until the method is confident about its estimate of the model score. We evaluate our method empirically on both synthetic and real-world data. On synthetic classification data, our approximate MCMC procedure obtains the true marginals faster than a traditional MCMC sampler. On real-world tasks, our method achieves 7 times speedup on citation matching, and 13 times speedup on large-scale author disambiguation.

5.1.1 Monte Carlo MCMC

Instead of evaluating the log-score π of the model exactly, we propose a *Monte-Carlo* estimation of the log-score. In particular, if the set of factors for a given proposal $\mathbf{y} \to \mathbf{y}'$ is $\mathcal{F}(\mathbf{y}, \mathbf{y}')$, we use a sampled subset of the factors $\mathcal{S} \subseteq \mathcal{F}(\mathbf{y}, \mathbf{y}')$ as an approximation of the model score. In the following we use $\mathcal{F}$ as an abbreviation for $\mathcal{F}(\mathbf{y}, \mathbf{y}')$. Formally,

$$\pi(\mathbf{y}, \mathbf{y}') = \sum_{f \in \mathcal{F}} \psi_f(\mathbf{y}'_f) - \psi_f(\mathbf{y}_f) = |\mathcal{F}| \cdot \left(\mathbb{E}_{\mathcal{F}}\left[\psi_f(\mathbf{y}'_f)\right] - \mathbb{E}_{\mathcal{F}}\left[\psi_f(\mathbf{y}_f)\right]\right) \tag{5.1}$$

$$\pi_{\mathcal{S}}(\mathbf{y}, \mathbf{y}') = |\mathcal{F}| \cdot \left(\mathbb{E}_{\mathcal{S}}\left[\psi_f(\mathbf{y}'_f)\right] - \mathbb{E}_{\mathcal{S}}\left[\psi_f(\mathbf{y}_f)\right]\right) \tag{5.2}$$

We use the sample log-score ($\pi_{\mathcal{S}}$) in the acceptance probability α to evaluate the samples. Since we are using a stochastic approximation to the model score, in general we need to take more MCMC

Algorithm 5.1 MCMCMC: Monte Carlo Markov Chain Monte Carlo

```
procedure MCMCMC($\mathcal{G}$, $q$, $N$, $\mathbf{y}_0$)
    $\mathbf{y} \leftarrow \mathbf{y}_0$
    $S_y \leftarrow \{\}$
    for $N$ samples do
        $\mathbf{y}' \sim q(\mathbf{y}, \cdot)$
        $\mathcal{S} \leftarrow$ SAMPLE($\mathcal{F}(\mathbf{y}, \mathbf{y}')$)        ▷ Uniform (5.1.1.1) or Confidence-Weighted (5.1.1.2)
        $\alpha \leftarrow \min\left(1, \frac{q(\mathbf{y}', \mathbf{y})}{q(\mathbf{y}, \mathbf{y}')} \exp \pi_{\mathcal{S}}(\mathbf{y}, \mathbf{y}')\right)$        ▷ Approximate acceptance probability
        if $\texttt{flip}(\alpha)$ then
            $\mathbf{y} \leftarrow \mathbf{y}'$        ▷ Accept the Sample
        end if
        $S \overset{+}{\leftarrow} \mathbf{y}$        ▷ Accumulate Sample
    end for
    return $S_y$
end procedure
```

samples before we converge, however, since evaluating each sample will be *much* faster ($O(|\mathcal{S}|)$ as opposed to $O(|\mathcal{F}|)$), we expect overall sampling to be faster. The algorithm is summarized in Algorithm 5.1.

In the next sections we describe two strategies for sampling the set of factors $\mathcal{S}$. The primary restriction on the set of samples $\mathcal{S}$ is that their mean should be an unbiased estimator of $\mathbb{E}_{\mathcal{F}}[\psi]$. Further, time taken to obtain the set of samples should be negligible when compared to scoring all the factors in $\mathcal{F}$. Note that there is an implicit minimum of 1 to the number of the sampled factors.

5.1.1.1 Uniform Sampling

The most direct approach for subsampling the set of $\mathcal{F}$ is to perform uniform sampling. In particular, given a proportion parameter $0 < p \leq 1$, we select a random subset $\mathcal{S}_p \subseteq \mathcal{F}$ such that $|\mathcal{S}_p| = p \cdot |\mathcal{F}|$. Since this approach is agnostic as to the actual factors scores, $\mathbb{E}_{\mathcal{S}}[f] \equiv \mathbb{E}_{\mathcal{F}}[f]$. A low p leads to fast evaluation, however it may require a large number of samples due to the substantial approximation. On the other hand, although a higher p will converge with fewer samples, evaluating each sample is slower.

5.1.1.2 Confidence-Based Sampling

Selecting the best value for p is difficult, requiring analysis of the graph structure, and statistics on the distribution of the factors scores; often a difficult task in real-world applications. Further, the same value for p can result in different levels of approximation for different proposals, either unnecessarily accurate or problematically noisy. We would prefer a strategy that adapts to the distribution of the scores in $\mathcal{F}$.

Instead of sampling a fixed proportion of factors, we propose to sample until we are confident that the current set of samples $\mathcal{S}_c$ is an accurate estimate of the true mean of $\mathcal{F}$. In particular, we maintain a running count of the sample mean $\mathbb{E}_{\mathcal{S}_c}[\psi]$ and variance $\sigma_{\mathcal{S}_c}$, using them to compute a confidence interval $I_{\mathcal{S}}$ around our estimate of the mean. Since the number of sampled factors $\mathcal{S}$ could be a substantial fraction of the set of factors $\mathcal{F}$[1], we also incorporate *finite population control (fpc)* in our sample variance computation. We compute the confidence interval as follows:

$$\sigma_{\mathcal{S}}^2 = \frac{1}{|\mathcal{S}|-1} \sum_{\psi_f \in \mathcal{S}} \left(f - \mathbb{E}_{\mathcal{S}}\left[f\right]\right)^2 \tag{5.3}$$

$$I_{\mathcal{S}} = 2z \frac{\sigma_{\mathcal{S}}}{\sqrt{|\mathcal{S}|}} \sqrt{\frac{|\mathcal{F}| - |\mathcal{S}|}{|\mathcal{F}| - 1}} \tag{5.4}$$

where we set the z to 1.96, i.e. the 95% confidence interval. This approach starts with an empty set of samples, $\mathcal{S} = \{\}$, and iteratively samples factors without replacement to add to $\mathcal{S}$, until the confidence interval around the estimated mean falls below a user specified maximum interval width threshold i. As a result, for proposals that contain high-variance factors, this strategy examines a large number of factors, while proposals that involve similar factors will result in fewer samples. Note that this user-specified threshold is agnostic to the graph structure and the number of factors, and instead directly reflects the score distribution of the relevant factors.

[1] Specifically, the fraction may be higher than $> 5\%$

5.1.2 Negative Proof of Convergence

By using a stochastic evaluation of the proposal, we are in effect include stochastic noise to the log ratio of the model scores. In particular, since $\pi(\mathbf{y},\mathbf{y}') = \pi(\mathbf{y}') - \pi(\mathbf{y})$, then

$$\pi_{\mathcal{S}}(\mathbf{y}',\mathbf{y}) = \pi(\mathbf{y}',\mathbf{y}) + \xi_{\mathcal{S}} \tag{5.5}$$

where $\xi_{\mathcal{S}}$ is the additive noise due to the stochastic evaluation, and as such, $\mathbb{E}\left[\xi_{\mathcal{S}}\right] = 0$.

Let us consider the transition probability as defined by the set of samples $\mathcal{S}$

$$\begin{aligned}
P^{\mathcal{S}}_{ij} &= q(\mathbf{y}_i,\mathbf{y}_j)\min\left(1, \frac{q(\mathbf{y}_j,\mathbf{y}_i)}{q(\mathbf{y}_i,\mathbf{y}_j)}\exp\left(\pi_{\mathcal{S}}(\mathbf{y}_j,\mathbf{y}_i)\right)\right) && (5.6)\\
&= q(\mathbf{y}_i,\mathbf{y}_j)\min\left(1, \frac{q(\mathbf{y}_j,\mathbf{y}_i)}{q(\mathbf{y}_i,\mathbf{y}_j)}\exp\left(\pi(\mathbf{y}_j,\mathbf{y}_i) + \xi_{\mathcal{S}}\right)\right) && (5.7)\\
&= q(\mathbf{y}_i,\mathbf{y}_j)\min\left(1, \frac{q(\mathbf{y}_j,\mathbf{y}_i)}{q(\mathbf{y}_i,\mathbf{y}_j)}\exp\left(\pi(\mathbf{y}_j,\mathbf{y}_i)\right)\exp\xi_{\mathcal{S}}\right) && (5.8)
\end{aligned}$$

Assume that the same set of samples $\mathcal{S}$ is used for $\mathbf{y}_i \rightarrow \mathbf{y}_j$ as for $\mathbf{y}_j \rightarrow \mathbf{y}_i$. Without loss of generality we can let $P^{\mathcal{S}}_{ij} = q(\mathbf{y}_i,\mathbf{y}_j)\frac{q(\mathbf{y}_j,\mathbf{y}_i)}{q(\mathbf{y}_i,\mathbf{y}_j)}\exp\left(\pi(\mathbf{y}',\mathbf{y})\right)\exp\xi_{\mathcal{S}}$, which implies $P^{\mathcal{S}}_{ji} = q(\mathbf{y}_j,\mathbf{y}_i)$. Therefore,

$$\begin{aligned}
p(\mathbf{y}_i)P^{\mathcal{S}}_{ij} &= \frac{\exp\pi(\mathbf{y}_i)}{Z}q(\mathbf{y}_i,\mathbf{y}_j)\frac{q(\mathbf{y}_j,\mathbf{y}_i)}{q(\mathbf{y}_i,\mathbf{y}_j)}\exp\left(\pi(\mathbf{y}_j,\mathbf{y}_i)\right)\exp\xi_{\mathcal{S}} && (5.9)\\
&= \frac{\exp\pi(\mathbf{y}_i)}{Z}q(\mathbf{y}_j,\mathbf{y}_i)\frac{\exp\pi(\mathbf{y}_j)}{\exp\pi(\mathbf{y}_i)}\exp\xi_{\mathcal{S}} && (5.10)\\
&= p(\mathbf{y}_j)P^{\mathcal{S}}_{ji}\exp\xi_{\mathcal{S}} \neq p(\mathbf{y}_j)P^{\mathcal{S}}_{ji} && (5.11)
\end{aligned}$$

Thus detailed balance does not hold with stochastic evaluation. Further, it does not hold in expectation either, since $\mathbb{E}\left[\exp\xi_{\mathcal{S}}\right] \neq 1$. The main issue arises from the fact that $P^{\mathcal{S}}_{ij} \neq P_{ij}\exp\xi_{\mathcal{S}}$ due to the asymmetric nature of $\min$ in the acceptance probability.

Although MAP inference does not require detailed balance to hold, it does require the model probability ratio to be computed exactly. With approximate sampling, we do not compute the model ratio exactly, and thus even for MAP inference approximate sampling does not converge.

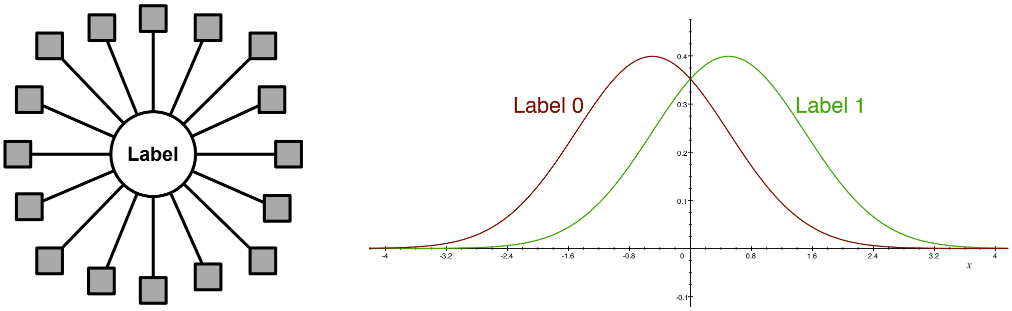
5.1.3 Experiments

Although the chain may not converge to the exact marginals, it can still be used for speeding up inference at the earlier stages of sampling, after which we can switch to non-stochastic MCMC evaluation to produce accurate samples. This is akin to the literature in adaptive sampling that show convergence by assuming the adaptive nature vanishes asymptotically [Andrieu and Thoms, 2008, Atchadé et al., 2011]. In this section we provide empirical evidence of the utility of the stochastic evaluation of the proposals for marginal and MAP inference.

5.1.3.1 Marginal Inference on Synthetic Data

Consider the task of classifying entities into a set of types, for example, POLITICIAN, VEHICLE, CITY, GOVERMENT-ORG, etc. For knowledge base construction, this prediction often takes place on the entity-level, as opposed to the mention-level common in traditional information extraction. To evaluate the type at the entity-level, the scored factors examine features of all the mentions of the entity, along with the labels of all relation mentions for which it is an argument. See Yao et al. [2010] and Hoffmann et al. [2011] for examples of such models. Since a subset of the mentions can be sufficiently informative for the model, we expect our stochastic MCMC approach to work well.

We use synthetic data for such a model to evaluate the quality of marginals returned by the Gibbs sampling MCMC. Since the Gibbs algorithm samples each variable using a fixed assignment of its neighborhood, we represent generating a single sample as classification. We create star-shaped models with a single unobserved variable (entity type) that neighbors many unary factors, each representing a single entity- or a relation-mention factor (See Fig. 5.1a for an example). We generate a synthetic dataset for this model, creating 100 variables consisting of 100 factors each. The scores of the factors are generated from gaussians, $\mathcal{N}(0.5, 1)$ for the positive label, and



(a) Classification

(b) Distribution of Factor scores

Figure 5.1: **Synthetic Model for Binary Classification:** containing a single variable that neighbors 100 factors, forming a star graph, shown in (a). Each factor score is sampled from $\mathcal{N}(\pm 0.5, 1)$, depending on the value of the variable. The similarity of the potentials is shown in (b).

$\mathcal{N}(-0.5, 1)$ for the negative label (note the overlap between the weights in Fig. 5.1b). Although each structure contains only a single variable, and no cycles, it is a valid benchmark to test our sampling approach since the effects of the setting of burn-in period and the thinning samples are not a concern.

We perform standard Gibbs sampling, and compare the marginals obtained during sampling with the true marginals, computed exactly. We evaluate the previously described uniform sampling and confidence-based sampling, with several parameter values, and plot the L_1 error to the true marginals as more factors are examined. Note that here, and in the rest of the evaluation, we shall use the number of factors scored as a proxy for running time, since the effects of the rest of the steps of sampling are relatively negligible. The error in comparison to regular MCMC ($p = 1$) is shown in Fig. 5.2, with standard error bars averaging over 100 models. Initially, as the sampling approach is made more stochastic (lowering p or increasing i), we see a steady improvement in the running time needed to obtain the same error tolerance. However, the amount of relative improvements slows as stochasticity is increased further; in fact for extreme values ($i = 0.05, p = 0.1$) the chains perform worse than regular MCMC.

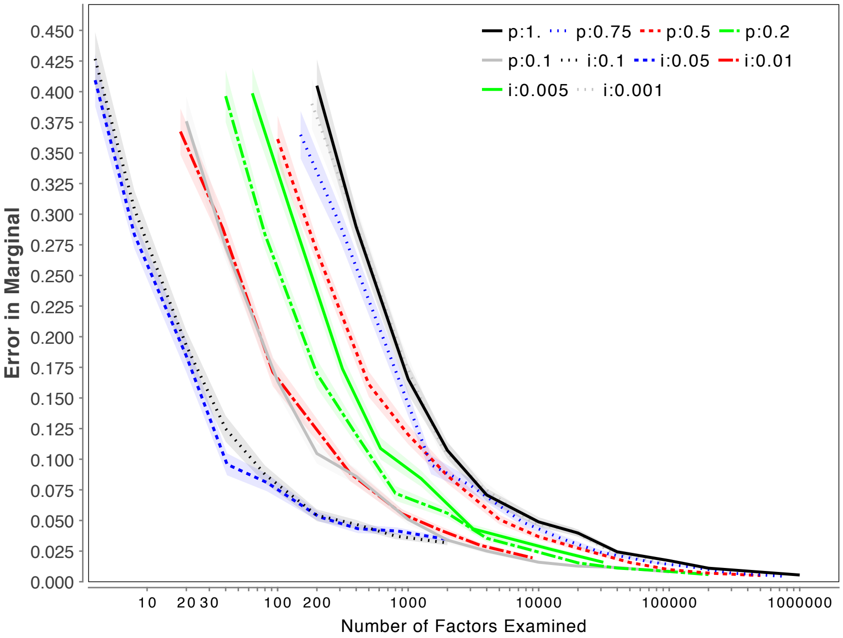


Figure 5.2: **Marginal Inference with MCMCMC:** L1 Error on the Marginals for Classification on Synthetic Data. Black solid line ($p = 1$) corresponds to regular MCMC. Lower p and Higher i correspond to more stochastic versions. Note the log-scale on the x-axis.

5.1.3.2 Citation Resolution

To evaluate our approach on a real world dataset, we apply stochastic MCMC for MAP inference on the task of citation matching. The model, described earlier in Section 4.4.2, consists of factors from mentions to all the other mentions that belong to the same entity, often leading to a very large number of factors. In practice, however, these set of factors are often highly redundant, as many of the mentions that refer to the same entity contain redundant information and features, and entity membership may be efficiently determined by observing a subset of its mentions.

We run MCMC on the entity resolution model using the proposal function described in Section 4.4.2, running our approach with different parameter values. Since we are interested in the MAP configuration, we use a temperature term for annealing. As inference progresses, we compute B^3 F1 of the current sample, and plot it against the number of scored factors in Fig. 5.3. We observe consistent speed improvements as stochasticity is improved, with uniform sampling and confidence-based sampling performing competitively. To compute the speedup, we measure the

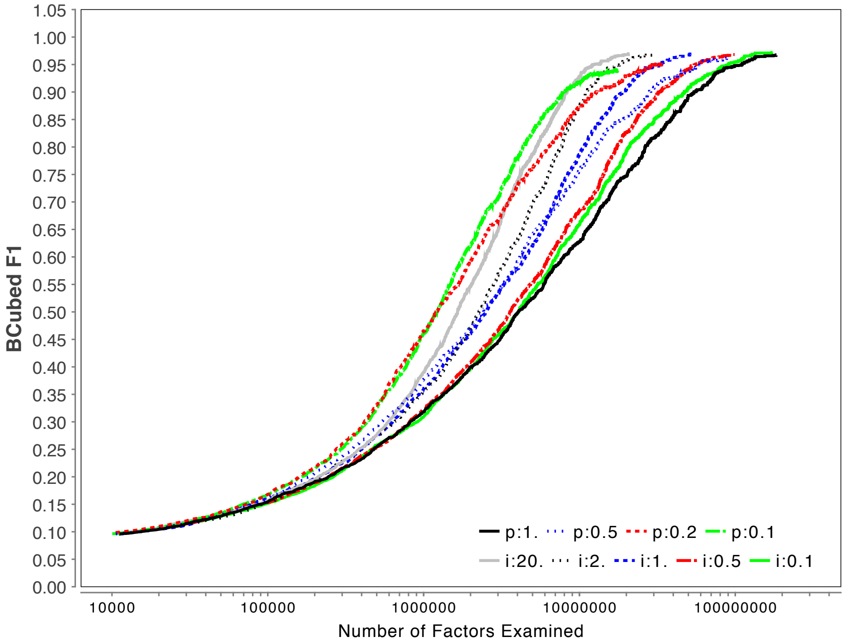


Figure 5.3: **MCMCMC for Citation Resolution:** B^3 F1 of the MAP configuration plotted against running time (in log-scale) for uniform and variance-based sampling compared to regular MCMC (black, solid line, $p = 1$).

number of factors scored to obtain a desired level of accuracy (90% F1), shown for a diverse set of parameters in Table 5.1. With a very large confidence interval threshold ($i = 20$) and small proportion ($p = 0.1$), we obtain up to 7 times speedup over regular MCMC. Since the average entity size in this data set is < 10, using a small proportion (and a wide interval) is equivalent to picking only a single mention to compare against.

5.1.3.3 Large-Scale Author Coreference

As the body of published scientific work continues to grow, author coreference, the problem of clustering mentions of research paper authors into the real-world authors to which they refer, is becoming an increasingly important step for performing meaningful bibliometric analysis. However, scaling typical pairwise models of coreference (e.g., McCallum and Wellner [2004]) is difficult because the number of factors in the model grows quadratically with the number of mentions (research papers) and the number of factors evaluated for every MCMC proposal scales linearly in

Method	**Factors Examined**	**Speedup**
Baseline	57,292,700	1x
Uniform Sampling		
$p = 0.75$	34,803,972	1.64x
$p = 0.5$	28,143,323	2.04x
$p = 0.3$	17,778,891	3.22x
$p = 0.2$	12,892,079	4.44x
$p = 0.1$	7,855,686	7.29x
Variance-Based Sampling		
$i = 0.001$	52,522,728	1.09x
$i = 0.01$	51,547,000	1.11x
$i = 0.1$	47,165,038	1.21x
$i = 0.5$	32,828,823	1.74x
$i = 1$	18,938,791	3.02x
$i = 2$	11,134,267	5.14x
$i = 5$	9,827,498	5.83x
$i = 10$	8,675,833	6.60x
$i = 20$	8,295,587	6.90x

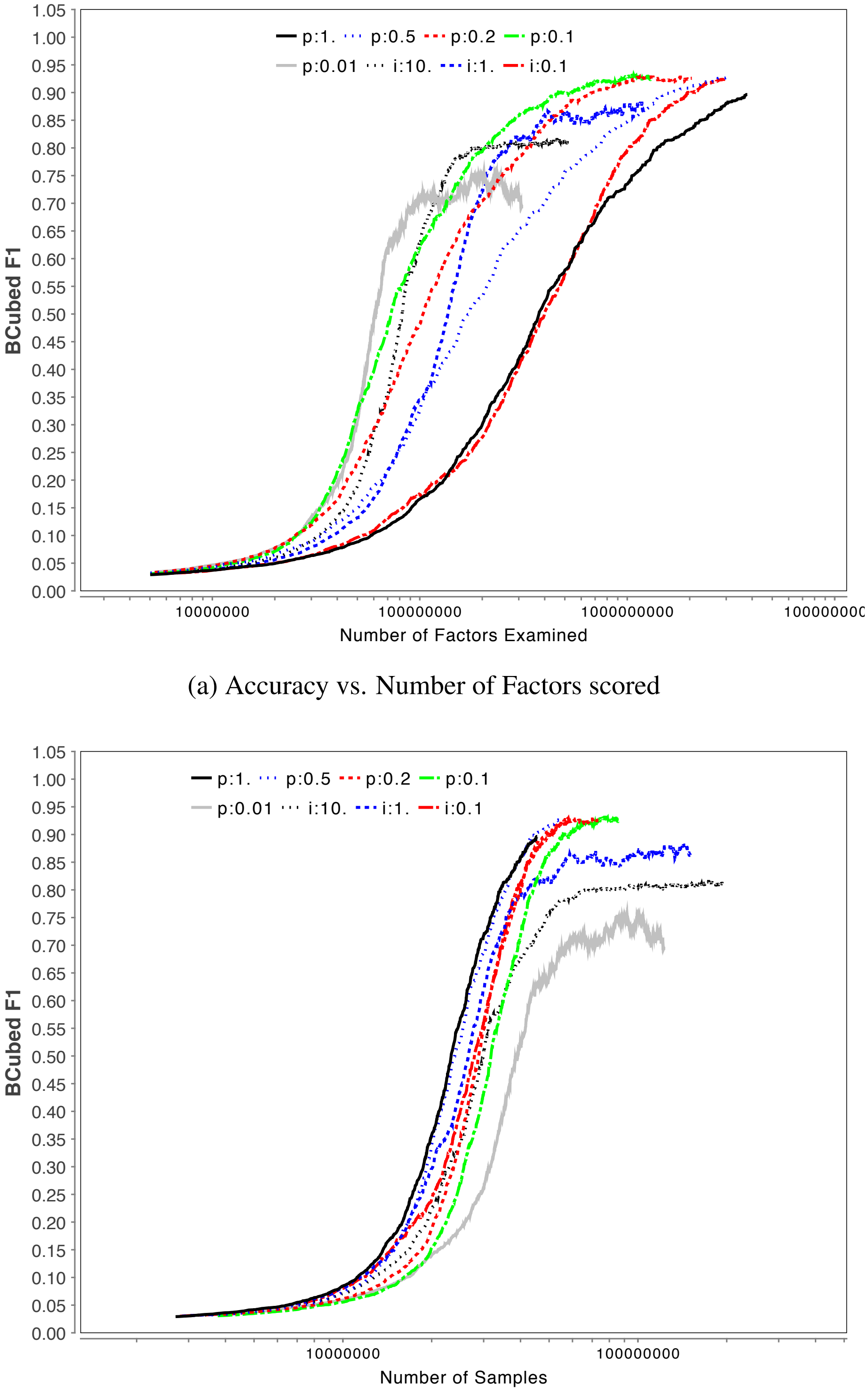
Table 5.1: **MCMCMC Speedups for Citation Resolution:** Speedups to obtain 90% B^3 F1

the size of the clusters. For author coreference, the number of author mentions and the number of references to an author entity can often be in the millions, making the evaluation of the MCMC proposals computationally expensive.

We use the publicly available DBLP dataset[2] of BibTex entries as our unlabeled set of mentions, which contains nearly 5 million authors. For evaluation of accuracy, we also include author mentions from the Rexa corpus[3] that contains $2,833$ mentions labeled for coreference. We use the same Metropolis-Hastings scheme that we employ in the problem of citation matching, however it uses manually-specified *blocking* to encourage fruitful proposals. As before, we initialize to the singleton configuration and run the experiments for a fixed number of samples, plotting accuracy versus the number of factors evaluated (Fig. 5.4a) as well as accuracy versus the number of samples generated (Fig. 5.4b). We also tabulate the relative speedups to obtain the desired accuracy level

[2] http://www.informatik.uni-trier.de/~ley/db/

[3] http://www2.selu.edu/Academics/Faculty/aculotta/data/rexa.html



(a) Accuracy vs. Number of Factors scored

(b) Accuracy versus Number of Samples

Figure 5.4: **MCMCMC for Large-Scale Author Resolution:** Performance of Different Sampling Strategies and Parameters for coreference over 5 million mentions. Plot with p refer to uniform sampling with proportion p of factors picked, while plots with i sample till confidence intervals are narrower than i.

Method	Factors Examined	Speedup
Baseline	1,395,330,603	1x
Uniform		
$p = 0.5$	689,254,134	2.02x
$p = 0.2$	327,616,794	4.26x
$p = 0.1$	206,157,705	6.77x
$p = 0.05$	152,069,987	9.17x
$p = 0.02$	142,689,770	9.78x
Variance		
$i = 0.00001$	1,442,091,344	0.96x
$i = 0.0001$	1,419,110,724	0.98x
$i = 0.001$	1,374,667,077	1.01x
$i = 0.1$	1,012,321,830	1.38x
$i = 1$	265,327,983	5.26x
$i = 10$	179,701,896	7.76x
$i = 100$	106,850,725	13.16x

Table 5.2: **MCMCMC Speedups for Author Resolution:** Speedups to reach 80% B^3 F1

in Table 5.2. Our proposed method achieves substantial savings on this task: speedups of 13.16 using the variance sampler and speedups of 9.78 using the uniform sampler. As expected, when we compare the performance using the number of generated samples, the approximate MCMC chains appear to converge more slowly; however, the overall convergence for our approach is substantially faster because evaluation of each sample is significantly cheaper. We also present results on using extreme approximations (for example, $p = 0.01$), resulting in convergence to a low accuracy.

5.1.4 Discussion

This section explores a stochastic MCMC algorithm that performs stochastic evaluation of the proposals during MCMC. Although the empirical results show impressive speedups to obtain accurate marginals and MAP configurations, we also show that the detailed balance does not hold for these chains. Thus, when we increase stochasticity of the approach, the chains often do not converge, for example $p = 0.01$ and $i = 10$ in Fig. 5.4 and $i = 0.1, 0.05$ for Fig. 5.1.

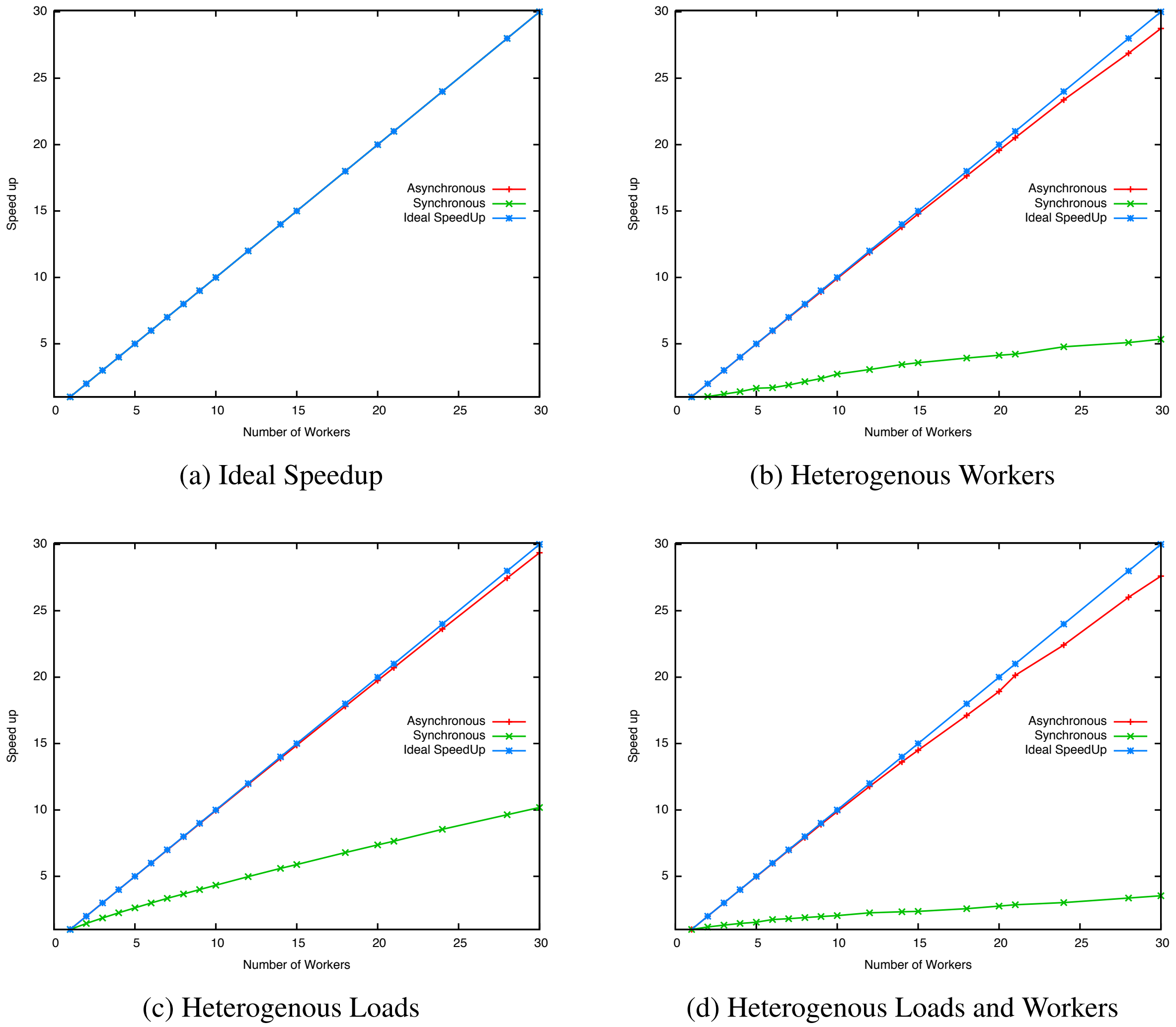
These results suggests sampling with unrestricted proposals in the synchronous distribution will add undesirable biases to our chains, however may be useful in initial stages of inference before switching to synchronous distribution with restrictions imposed on the proposal functions. In the rest of the chapter, we explore frameworks that are directly asynchronous, and address both the restrictions imposed on the proposal function, and the synchrony bottleneck inherent in the iterated Map-Reduce framework.

5.2 Synchronous Bottleneck, an illustration

A significant motivation for the rest of the chapter is the bottleneck created in distributed systems due to synchronous communication. This is a commonly studied problem in the popular Map-Reduce framework, in which it is often known as the *curse of the last reducer* [Suri and Vassilvitskii, 2011][4]. The problem can arise due to data skew, i.e. the variance in worker loads leads to some of the workers taking a longer time than others. Load-balancing, thus, has been a well-studied field in order to address this challenge. However, even if the loads are homogenized, it is not guaranteed that all the workers will perform equally well. Heterogeneity can arise from different architectures, memory access patterns, load from external processes, network load, and so on. It is not possible to compensate for all of these issues without constructing customized hardware dedicated to the task at hand; an expensive and increasingly impractical solution in today's world of elastic and cluster computing.

Data skew is also a concern for synchronously distributed MCMC. First, since we are partitioning a graph, it is not always possible to preserve locality and partition in a balanced manner, and we have to compromise one in order to maintain the other. Consider, for example, nodes of the graph that have an incredibly high degree; assigning all the neighbors of the node to the same worker will maintain locality, however will lead to unbalanced load. Graphs with power law degree distribution are common in large datasets and models, for example we describe a power law distribution of entity sizes in Singh et al. [2012a], and Gonzalez et al. [2012] describe techniques to handle such graphs in a distributed manner. A more-challenging concern arises from the fact our models may

[4] This is a version of Amdahl's Law [Amdahl, 1967], which is used to compute the maximum theoretical speedup in performance if only a part of the system is improved.



(a) Ideal Speedup

(b) Heterogenous Workers

(c) Heterogenous Loads

(d) Heterogenous Loads and Workers

Figure 5.5: **An Illustration of the Synchronous Bootleneck:** Comparison of an asynchronous architecture with a synchronous architecture on simulated loads and workers. The speedups from asynchronous distribution stay near linear, while the synchronous framework is not scalable for large number of heterogeneous workers or with heterogeneous loads.

be dynamic, and in general, the load on a worker can actually vary *during inference*. For example, in the entity resolution models described in the last chapter, the overall load is quadratic in the size of the entities, and thus if the sizes of the entities change during inference, the load on the worker can vary dramatically. This can be partially addressed by sampling less and communicating often, at an additional cost that substantially affect speedup.

We present an illustration of the adverse effects of the synchronous bottleneck on scalability. We create a number of synthetic loads and simulate their running time on synchronous and asynchronous framework, where the asynchronous approach is implemented as a queue of jobs that each worker queries once it has processed its previous load. We create ~ 2500 simulated loads and vary the number of workers from 1 to 30. Homogeneous loads (number of operations) and worker speeds (time per operation) are set to 1, while for heterogeneous loads and speeds we sample from the gamma distribution with $k = 2, \theta = 2$. We average the speedups over 20 runs, and compare ideal speedup (linear) with the speedups from synchronous and asynchronous distribution in Fig. 5.5. It is evident that heterogeneity in either the loads or worker speeds has a significant impact on the speedups of the synchronous framework, while the asynchronous framework still provides near linear speedups. Further when both loads and workers are heterogeneous, as shown in Fig. 5.5d, synchronous framework only provides a speedup of ~ 3 for 30 machines, while asynchronous is still scalable (~ 27 speedup for 30 machines).

Note that this comparison does not take into account the cost of communication, and it is possible to design synchronous frameworks that have cheaper communication than asynchronous. Further we also ignore the time for dequeuing a job for the asynchronous framework; conflicts can arise when a worker has to wait while the queue is serving a job to another worker. Nonetheless, we feel this simulation provides a valuable insight into the impact that the synchronous bottleneck can have on scalability.

5.3 Conflict-free Asynchronously Distributed MCMC

In Section 5.1 we propose stochastic evaluation of proposals to investigate the effect of dropping the restrictions placed by synchronous distribution. Although we show dropping the restrictions can benefit sampling, the synchronous bottleneck in the architecture is still the major concern, as we show on comparison with asynchronous distribution of the same load in Section 5.2. Here we focus on addressing the synchronous bottleneck, but maintain the restrictions imposed by exact model score computation.

In this section we propose an architecture for asynchronous distributed inference that maintains locks on variables for conflict-free inference. A central control on the locking/partitioning allows each worker to independently query for the set of variables to infer over. The locking ensures there is no overlap amongst the variables assigned to the workers, and hence the ratio of the model scores can be computed exactly for the restricted proposals. We show that the chains with such a distribution converge to the correct stationary distribution.

5.3.1 Architecture

The synchronously distributed MCMC technique proposed in the previous chapter provides efficient sampling, however faces restricted scalability due to the synchronous bottleneck. We want to design an architecture that maintains the benefits of distributed inference but communication amongst the workers is asynchronous.

Our proposed architecture is shown in Fig. 5.6. A distributed persistence layer maintains all the *current* values of the variables. A separate variable-locking mechanism coordinates the partitioning of the variables, and locking the variables that are assigned to a worker, and freeing the lock when the worker has finished inference. The overall algorithm works as follows:

1. The inference worker requests a partition to perform inference over.

2. The variable-locking mechanism selects a partition over the unlocked variables, hands them off to the inference worker, and marks the variables in the partition as locked.

3. The inference worker reads the current values of the variables from the distributed DB.

4. The worker performs inference over the partition assuming (correctly) that no other worker is sampling the same variables. Note that the same restrictions are imposed on sampling as in the previous chapter, i.e. only the proposals that examine the local factors for scoring can be proposed.

5. On completion (sampling for a fixed number of samples), the inference worker writes the new values of the variables to the distributed DB. There will be no conflicts since the variables cannot be assigned to any other worker.

6. The inference worker notifies the variable-locking mechanism the set of variables it sampled. The variable-locking mechanism marks the variables as unlocked. Repeat step 1.

For the above architecture to work for multiple inference workers, we only allow a single-threaded access to the variable-locking mechanism, i.e. it is only handling the requests of a single inference worker at a time. Although this adds a bottleneck, the task of variable-locking and assigning a partition is much faster than performing inference and communication, and hence we expect the impact this bottleneck to be negligible. This architecture also supports the various partitioning strategies from the previous chapter since the variable-locking mechanism is free to construct arbitrary partitions over the free variables while inference workers are performing inference. This is also a significant improvement over the synchronous distribution since inference workers in latter cannot perform inference before partitioning is complete, and vice versa. For the rest of the section, we assume that only a small fraction of variables are locked at any time, i.e. we focus on sampling for very large models while the load on each inference worker is relatively small.

5.3.2 Analysis

Our assumption that only a small fraction of the model is assigned to the workers is important for the analysis. In particular, if the model is exactly partitioned over the workers and all the variables are locked, then the variable locking mechanism would have no choice to assign the very same variables to the inference worker as the ones the worker performed inference on, since the rest

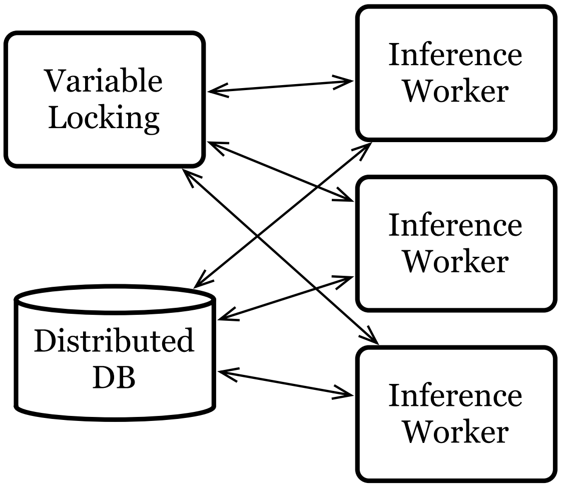


Figure 5.6: **Conflict-free Asynchronous Distributed Inference:** Variable locking is a lightweight index that maintains the locking status of each variable. Each inference worker requests locks and reads/writes to the DB completely asynchronously.

of the variables are locked. This would prevent the chain from being irreducible, hence resulting in non-stationarity. We further assume that workers perform inference over the partition assigned to them in finite time, i.e. each worker eventually unlocks the variables that were assigned to it. If this assumption is broken, i.e. if a worker fails, the variables will remain locked for the rest of sampling, resulting in reducible chains.

The rest of the analysis is similar to Section 4.3.2, and makes similar assumptions. To show correctness, we need to show exact model score ratio computation, irreducibility and aperiodicity of the proposals, and exact computation of the forward-backward ratio.

- **Model Score:** Since the proposals are restricted to examine the factors local to the workers, the model score is computed exactly.

- **Proposal Function:** Since we assume only a fraction of the variables are ever locked, it is easy to see that we will explore all the various partitions if any of the random partitioning schemes is used. Thus if the proposal function used in each worker is irreducible and aperiodic, the overall propsal function maintains irreducibility and aperiodicity.

- **F/B Ratio:** To show that the forward-backward ratio remains unchanged, we have to show that the probability of a variable being selected for sampling is independent of its value. Since the variable locking mechanism does not take the values of the variables into account, and since each variable is unlocked in finite time, the probability of selecting a variable is independent of its value. Note that this probability does not depend on the speed of the various inference workers since every variable has the same probability of being assigned to every inference worker.

Given these conditions, we can see that the chain will converge to the desired distribution. We can also use the latent variable partitioning scheme, however we cannot guarantee correctness since the forward-backward ratio cannot be computed exactly. However latent variable can still be used for MAP inference.

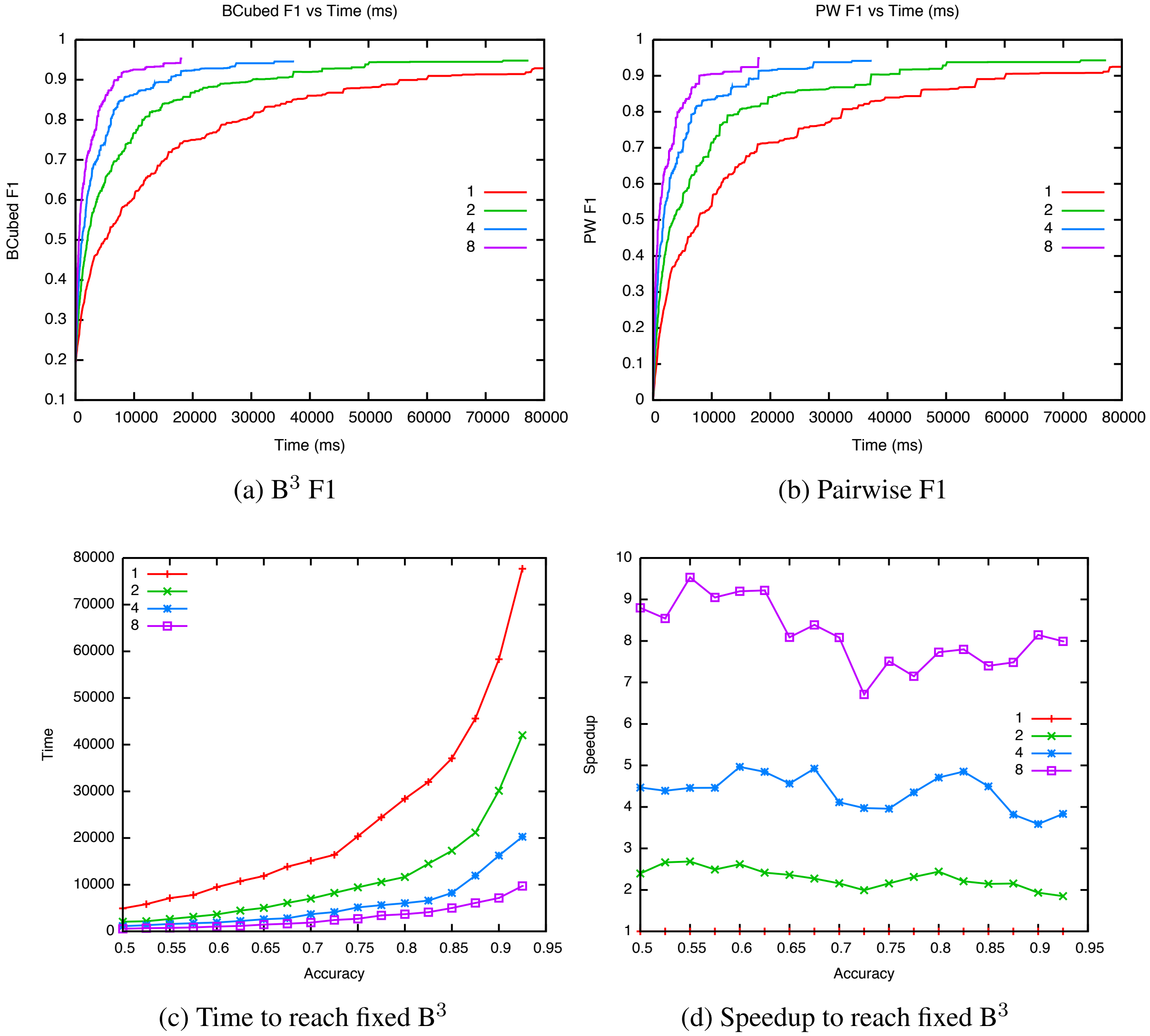
5.3.3 Experiments

To assess the utility of asynchronous distribution, we evaluate MCMC inference on two entity resolution tasks: citation deduplication and author disambiguation.

5.3.3.1 Citation Deduplication

In this section, we revisit the citation resolution task. In Section 4.4.2, we demonstrate that synchronous distribution for this task is able to achieve significant speedups for up to 4 inference workers, however is unable to provide additional gains for more workers. We also use the same dataset in Section 5.1.3.2 to show that using stochastic MCMC can used to speed-up single machine inference, the chains may converge to the desired distribution. Here, we will return to the distributed MCMC architecture, and investigate whether asynchronous distribution with locking is able to address the drawbacks of the synchronous case.

We present the performance of asynchronously distributed MCMC in Fig. 5.7, plotting the B^3 and pairwise F1 against the wall-clock running time, and presenting the time taken/speedup to obtain a fixed level of accuracy. As evident from the figures, increasing the number of workers



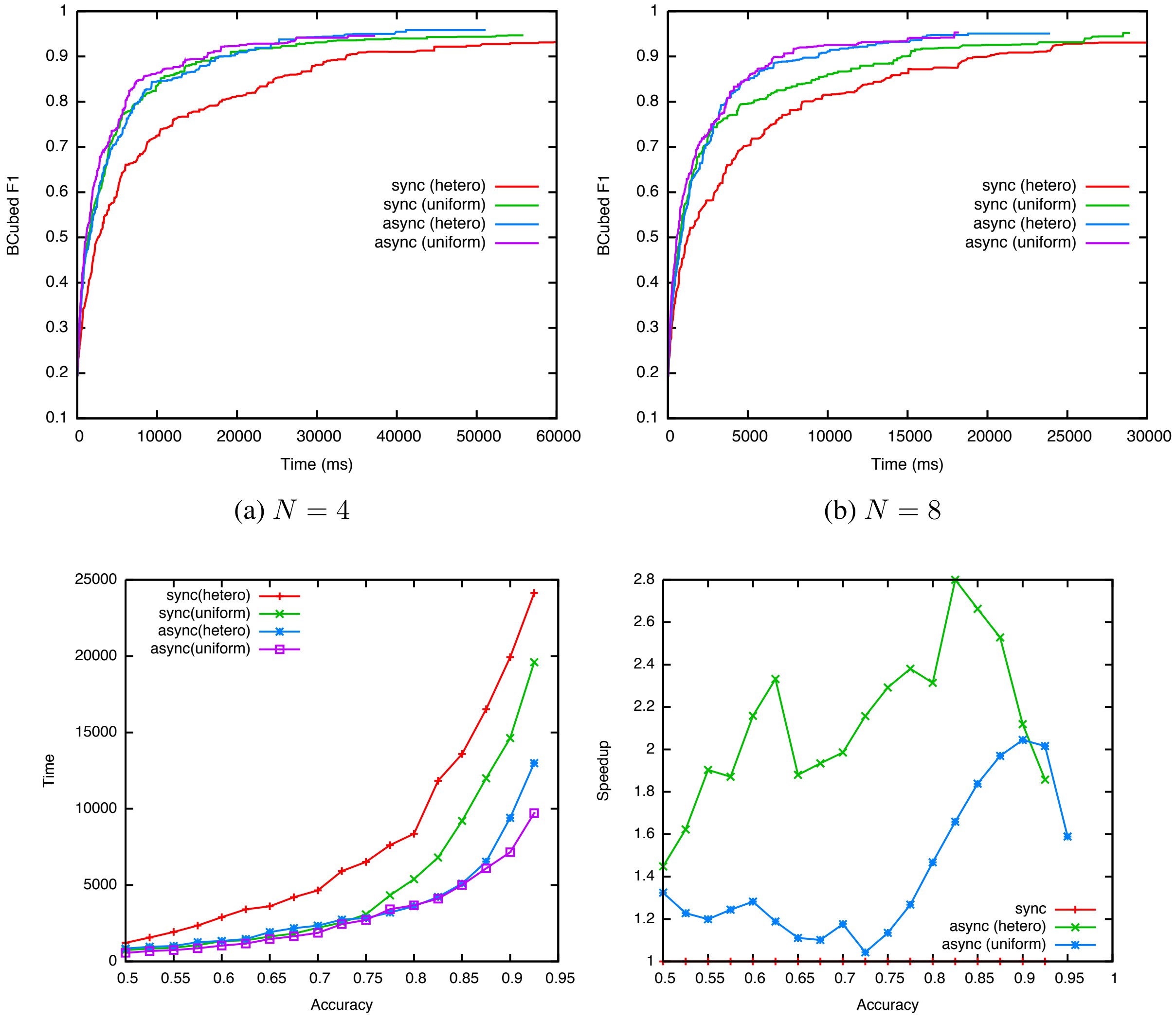
(a) B^3 F1

(b) Pairwise F1

(c) Time to reach fixed B^3

(d) Speedup to reach fixed B^3

Figure 5.7: **Speedups for Citation Resolution with Asynchronous MCMC:** Evaluation of scalability of Asynchronous MCMC by varying the number of workers for citation resolution.



(a) $N = 4$

(b) $N = 8$

(c) Time to obtain fixed B^3 ($N = 8$)

(d) Speedup to obtain fixed B^3 ($N = 8$)

Figure 5.8: **Comparing Synchronous versus Asynchronous for Citations:** Evaluation of scalability of Asynchronous MCMC compared to Synchronous MCMC by varying the number of workers for citation resolution on the Cora dataset.

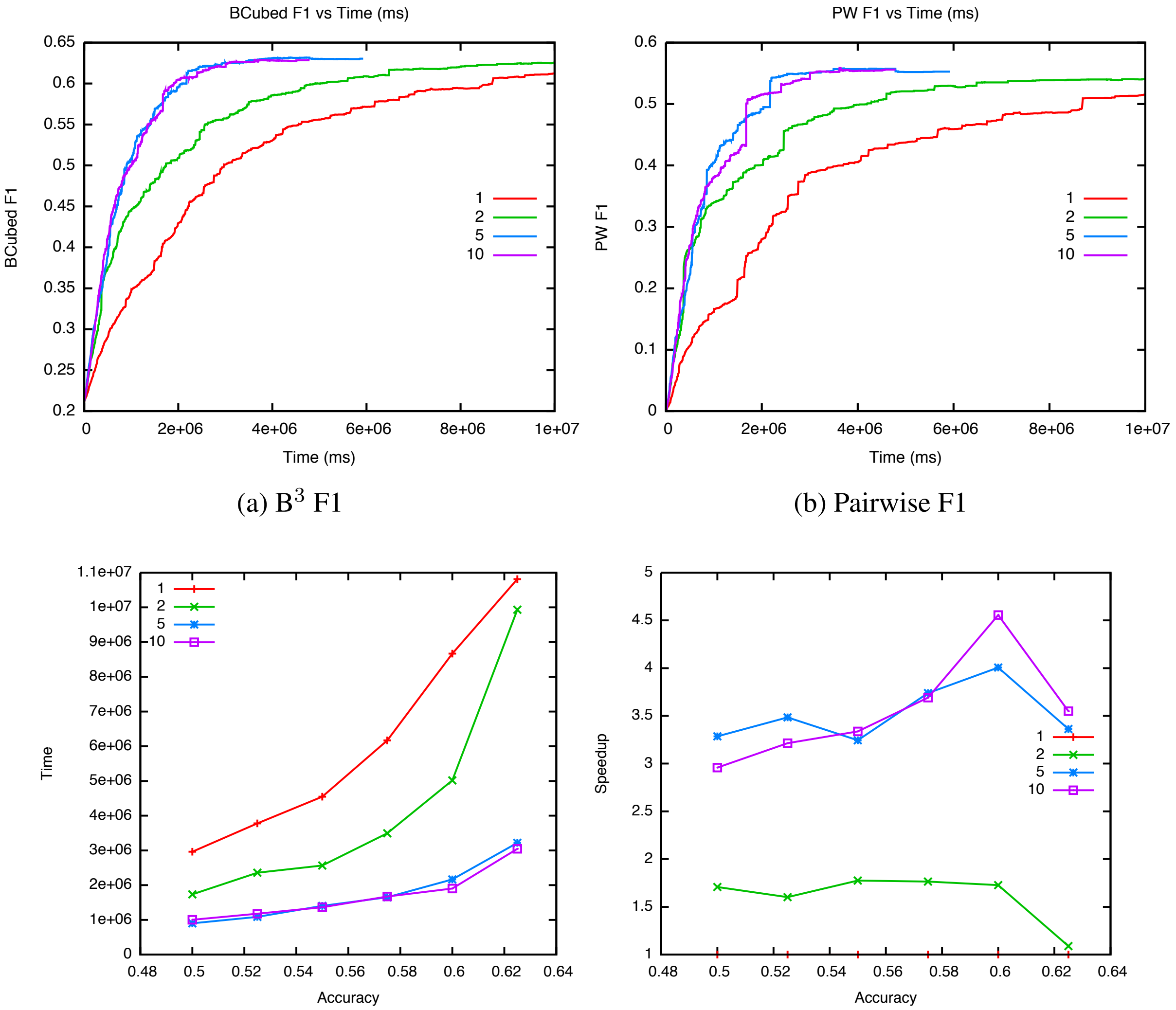
consistently improves the rate of convergence for both the metrics, with $N = 8$ providing a nearly linear speedup over $N = 1$. We describe the possible explanation for super-linear speedups for entity resolution in Section 4.4.2.

To compare against the synchronous distribution, we plot the accuracy of our previous experiment in Fig. 5.8 with *uniform* and *heterogeneous* workers. For $N = 4$, as shown in Fig. 5.8a, asynchronous and synchronous with uniform workers perform nearly identical. For $N = 8$, although the accuracy is identical for the initial part of the curve (where the entities are small and hence the load is uniform), we see from Figs. 5.7c and 5.7d that asynchronous (uniform) outperforms synchronous (uniform) substantially for higher accuracies (where entities are large and of different sizes), obtaining upto $1.5 - 2\times$ speedup over the synchronous architecture.

The processors are nearly identical in our experiment setup, the worker speeds are the same as each other, and hence the heterogeneity arises from the variance in the loads. To investigate the effect of heterogeneous workers, we artificially inject delays in some of the machines, such that the slowest machine is half as fast as the fastest (linearly interpolating the delays for the remaining workers). Synchronous distribution is substantially affected by this imbalance, performing significantly worse for both $N = 4$ and $N = 8$ in Figs. 5.8a and 5.8b, respectively. Asynchronous distribution, on the other hand, is robust to heterogeneous loads and workers, getting affected only in the slightest for both $N = 4$ and $N = 8$. In particular, we find asynchronous MCMC on heterogeneous workers obtains almost $3\times$ speedup over synchronous distribution on $N = 8$, as shown in Fig. 5.8d.

5.3.3.2 Large-Scale Author Disambiguation

We also evaluate our conflict-free asynchronous inference approach to large-scale author disambiguation, used earlier in Section 5.1.3.3. In particular, we use the same datasets and the model, but instead subsample a tenth of the dataset to obtain nearly $500,000$ mentions. Each worker requests 25000 mentions at a time, and performs 1 million sampling steps before releasing the mentions. A significant difference between the earlier results and experiments presented here is



(a) B^3 F1

(b) Pairwise F1

(c) Time to reach fixed B^3

(d) Speedup to reach fixed B^3

Figure 5.9: **Speedups for Author Disambiguation with Asynchronous MCMC:** Evaluation of scalability of Asynchronous MCMC by varying the number of workers.

that we do not use any manual blocking information, and hence we expect a slower convergence and lower accuracy. We perform 2500 rounds of sampling, and plot the accuracy of the labeled mentions over time in Fig. 5.9. The results indicate that even on this large dataset, the asynchronous distributed inference is able to scale with more cores. Specifically, $N = 5$ provides a $4\times$ speedup over $N = 1$ and $2.3\times$ speedup over $N = 2$, while $N = 2$ provides a $1.8\times$ speedup over $N = 1$, to achieve a B^3 F1 of 60%. However, interestingly, $N = 10$ does not provide any significant gains over $N = 5$. We suspect this is due to our implementation choice of merging variable locking and the value storage into a single thread that causes a bottleneck for large number of workers.

5.3.4 Discussion

In this section we introduce an approach to address the synchronous bottleneck prevalent in iterated Map-Reduce distribution frameworks. Using a central control on the variable locking and partitioning, we allow each inference worker to independently query for partitions and perform inference. By using a separate distributed DB to store the values of the variables, the central locking is relatively free of bottlenecks, further improving scalability. We show that the chains will converge to the correct distribution, and show results on entity resolution demonstrating the utility of asynchronous communication over synchronous distribution.

5.4 Lock-free Distributed Inference

Earlier in the chapter, in Section 5.1, we demonstrate that using subsets of factors to evaluate proposals can lead to impressive speedups, however the chains are not guaranteed to converge to the stationary distribution. On the other hand, we maintain the restrictions on the proposal function in Section 5.3, but introduce an asynchronous distributed inference architecture that addresses the synchrony bottleneck. Here we combine the two approaches of asynchronous communication and unrestricted proposals.

In this section, we propose a distributed inference algorithm similar to the one proposed in Section 5.3, but without any locks on the variables. The inference workers may perform sampling on overlapping sets of variables, and ignore cross-worker factors when evaluating proposals on each machine. When the inference worker is writing the values of the values to the persistence layer, it is possible a different value has been written, leading to a conflict. We propose a number of conflict-resolution techniques that vary in the computation cost and the quality of the resulting sample. Since we are interested in evaluating the correctness of the resulting chains, we evaluate the approach on a synthetic marginal inference task. We show that additional machines result in faster convergence. Further, amongst the proposed conflict resolution approaches, we show that the simple strategy of overwriting the conflict (fast but noisy) with the new value performs as well as resampling the variables (slow but accurate).

5.4.1 Architecture

Our proposed architecture for asynchronous MCMC is shown in Fig. 5.10, which is identical to the architecture in Section 5.3.1 but without the variable locking mechanism. The variables and their current state is maintained by the data repository, which may be a distributed file system or database. Inference is performed by independent workers that:

1. Request a set of variables and their current values, and the current value of the neighboring variables, from the repository.

2. Perform independent inference by sampling the local variables and unrestricted proposal function.

3. After sampling for a fixed number of samples, use the accepted proposals to write a set of values back to the repository, and repeat (1).

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

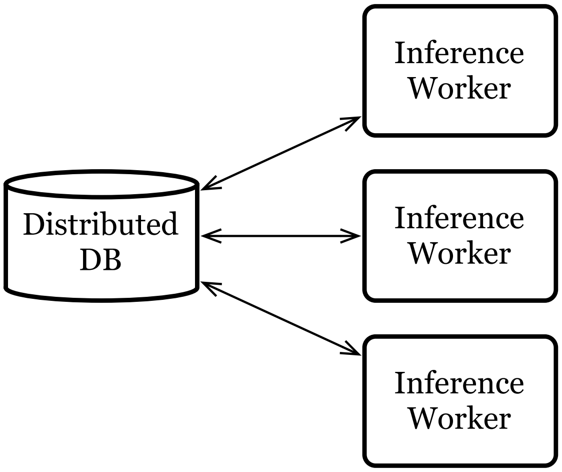


Figure 5.10: **Lock-free Asynchronous Distributed Inference:** Each worker independently reads the values of the variables when performing inference. Since there are no-locks, it is possible that the value of the variable has been changed during inference. Hence, a conflict-resolution strategy needs to be adopted before the worker can submit the variable values to the DB.

made by other workers that are using the obsolete value. Due to the restriction in communication, the workers can only detect new values when writing back to the repository after inference. Thus the inference workers will need to resolve the conflict created between the value they used for inference with the new value in the repository.

5.4.2 Conflict Resolution Strategies

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 in order to generate a large number of inaccurate samples, or to generate fewer, higher quality samples. In particular, given the set of accepted proposals for a inference worker as $\mathbf{y}^1, \ldots, \mathbf{y}^m$, and $\mathbf{y}_0$ is the value of the variables when inference was initiated, we study different strategies for resolving the value for variable Y_k when the new value y_k is the repository differs from y_k^0. Although we describe the strategies for a single changed variable below, they can be extended to multiple conflicts.

- **Overwrite:** The simplest and the quickest strategy is to simply ignore the changed variables with the assumption that the accepted proposals are still as informative in the new context. This corresponds to $Y_k = y_k^m$. This technique ignores the model score of the conflict, and in general, will often write values to the repository that have a low score according to the model.

- **Combined Proposal:** Since directly over-writing the value can lead to low-scoring values in the repository, we need to take the model score into account. A direct approach of incorporating model score 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. Formally, we create a joint proposal $(\mathbf{y}^0|Y_k = y_k) \rightarrow \mathbf{y}^m$ and compute the acceptance probability using only the model score ratio.

- **Separate Evaluations:** The previous approach evaluates all the accepted proposals as a single proposal. However, it is possible that a large number of accepted proposals are still high-scoring, however the the aggregated proposal is low-scoring, and hence, may be rejected. The inference, thus, loses out on the high quality samples. Instead, we propose a finer-grained search for good proposals, by re-evaluating the proposals. This method will reset the values of the variables to $(\mathbf{y}^0|Y_k = y_k)$, and iterate through all the samples $\mathbf{y}^1, \ldots, \mathbf{y}^m$ to evaluate them separately. Although this method is likely to retain the high-scoring proposals, it will be substantially slower than the previous approaches.

- **Restricted to Proposals Neighboring the Conflict:** Out of all the proposals that were accepted, some of the proposals lie in the Markov blanket of the changed variable and are directly affect by the change, while others are not. To improve efficiency of the previous approach, we iterate through all proposals, but only evaluate ones that are directly affected by a changed variable. The set of changed variables grows every time the worker accepts a proposal. In particular, we initialize the set of changed variables as $C = \{Y_k\}$ and the current sample to $(\mathbf{y}^0|Y_k = y_k)$. For each proposal $\mathbf{y}^i \rightarrow \mathbf{y}^{i+1}$, we only evaluate it if any variable in C touches the factors used to evaluate the proposal. If the proposal is accepted, we append C with the changed variables. This strategy will reduce the number of proposal evaluations massively since variables in C will not be in the neighborhood of most proposals (depending on the model).

- **Resampling the Variables:** Many of the above approaches take a subset of the accepted proposals, re-evaluate them, and potentially reject some of them. This is two significant drawbacks for correctness. First, since these proposals are being evaluated twice, the transition probability for such proposals is not correct (double scores the transition). Second, it is not possible to compute the forward-backward ratio of such proposals, since it relies on the proposal being accepted in the first place, and not just on the probability of a conflict (which is easier to compute). Instead, we propose to reject the old proposals, and resample all the changed variables given the new context. This way the accepted proposals are evaluated only once, and hence their transition probability is well-defined. Further, since the probability of a conflict of the variable does not depend on the value but only on the properties of the factor graph (we assume a static graph), the forward-backward ratio is the same as that of the proposal function.

5.4.3 Analysis

Our analysis of correctness of distributed MCMC relies on discovering the sequential ordering of the parallel computations, and analyzing the properties of the chain resulting from the single-machine ordering in terms of whether the model and forward-backward ratios are computed correctly. For our results on synchronous and conflict-free distribution, we can sequentially order the proposals since we make a strong restriction that proposals that examine factors across workers, i.e. the ones that will prevent a sequential ordering of the proposals, cannot be proposed. For synchronous distribution, any ordering of proposals within each iteration is a valid one, while for asynchronous workers, the locking imposes partial ordering constraints that can be used to define a total ordering over the proposals. We then analyze these chains (see Section 4.3 and Section 5.3.2).

With our lock-free architecture, since the writes are atomic, we can use the writes to create an ordering amongst the proposal, i.e. the proposals are ordered according the order in which the workers write to the repository. In particular, for a worker i that started sampling when the repository is at $\mathbf{y}^0$ and ended when the repository is at $\mathbf{y}^t$, the proposal evaluations of worker i need to be a valid chain as if it was executed *after* $\mathbf{y}^t$, independent of $\mathbf{y}^0$. For the case of no conflict (i.e. no writes between $\mathbf{y}^0$ and $\mathbf{y}^t$ to variables assigned to i), the analysis is same as the conflict-free case and demonstrating valid chain after $\mathbf{y}^0$ is adequate. However, in case of a conflict, the proposal evaluations within the worker are invalid since they are conditioned on $\mathbf{y}^0$ and not $\mathbf{y}^t$, and thus the worker has to resample in order to generate a set of proposals that can be ordered sequentially after $\mathbf{y}^t$. Amongst our proposed conflict resolution strategies, resampling is the only one that meets this criterion; the other strategies are much faster however they break the Markov chain assumptions. For our analysis we make all the same assumptions as Section 5.3.2, including the assumption that only a small fraction of variables are being sampled at any time, which provides a non-zero probability that there is no conflict when a worker is writing values to the repository.

We show that the chain is a valid MCMC chain by induction. For the first worker that writes values to the repository, there are no conflicts, and all of its proposals have been evaluated correctly (as shown in Section 5.3.2 for conflict-free distribution). For the purpose of induction, assume that all proposals up to the i^{th} write have been evaluated correctly, and thus the assignment of the variables $\mathbf{y}^i$ is a valid state of the chain. When the $(i+1)^{\text{th}}$ write is taking place, there can be two cases:

1. If no other writes have been performed on the overlapping variables, then the proposals have been evaluated exactly as well (as shown in Section 5.3.2), and $\mathbf{y}^{i+1}$ is a valid state, or

2. In case of a conflict, the resampling strategy drops all the proposals that cannot be ordered after $\mathbf{y}^i$, initializes a new chain at $\mathbf{y}^i$, and generates new proposals, effectively restarting the sampling that will end at $\mathbf{y}^{i+1+j}$ for some $j \geq 0$. Since there is a non zero-probability that

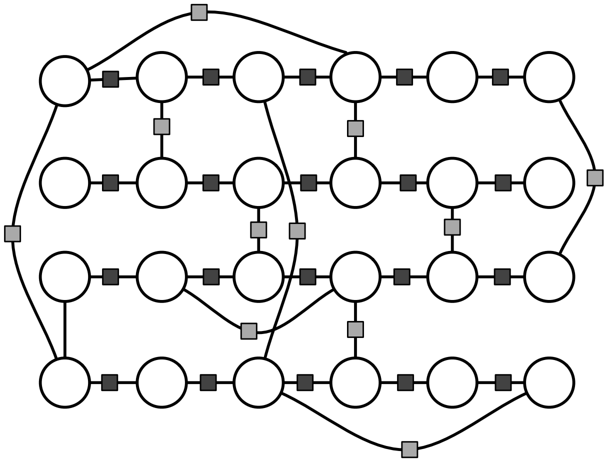


Figure 5.11: **Skip-Chain model:** The model consists of chains of factors, with *skip* factors between variables in separate chains, creating a dense, loopy structure.

there will be no conflicts when sampling is finished, eventually the chain will reach case (1), resulting in exact samples.

The probability that a variable has a conflict depends only on the graph structure, and not on its value. Hence the forward-backward ratio remains the same as the conflict-free architecture.

5.4.4 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 [Sutton and McCallum, 2007a]. 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. 5.11), creating a loopy dense graph for which inference is incredibly expensive. 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 $(-5, 5)$.

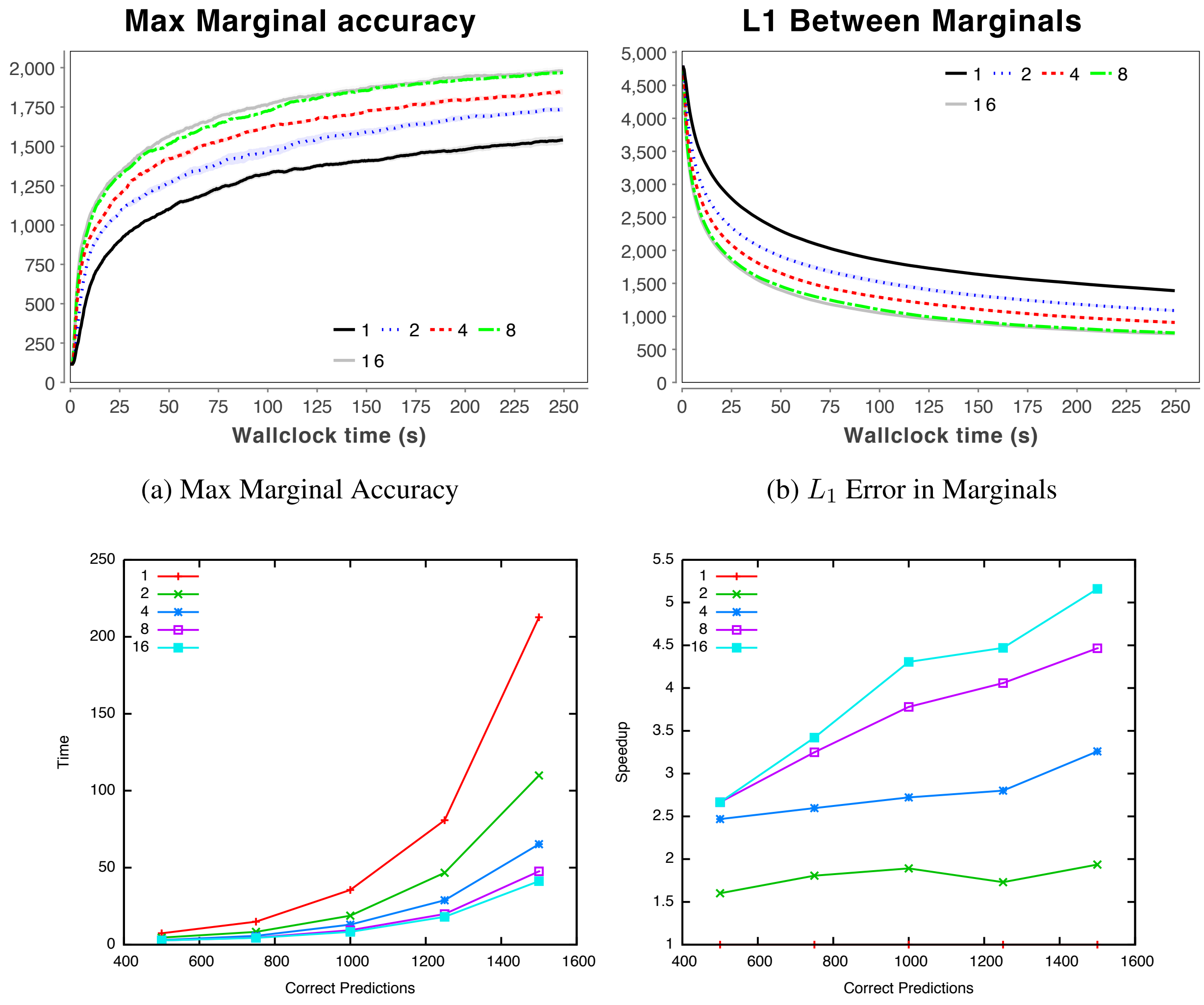
For evaluation, we run three single worker Gibbs chains independently till they have converged (by comparing within-chain variance to cross-chain). We treat the empirical distribution of the samples as the "true" marginals to compare convergence against. We run three random initialization chains for 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.

Figure 5.12 shows the quality of the marginals versus running time for different numbers of workers using our exact "resampling" strategy (Gibbs). The plot clearly shows that a larger number of workers result in faster convergence to better marginals. As we can see in Figs. 5.12c and 5.12d, this speedup is not linear, however even 16 workers provide small but noticeable advantage over 8. To compare the various ways to handle conflicts, we examine their convergence for a fixed number of workers in Fig. 5.13. Combined and Separate perform substantially worse than others on both accuracy and L1, but most surprisingly the simple strategy of Overwrite works as well the exact Gibbs resampling.

5.5 Related Work

Our contributions in this chapter focus on asynchronous updates for MCMC.

A number of existing approaches in statistics are related to our Monte Carlo MCMC algorithm. Leskovec and Faloutsos [2006] propose techniques to sample a graph to compute certain graph statistics with associated confidence. Christen and Fox [2005] also propose an approach to efficiently evaluate a proposal, however, once accepted, they score all the factors. Murray and Ghahramani [2004] propose an approximate MCMC technique for Bayesian models that estimates the partition function instead of computing it exactly. Related work has also applied such ideas for robust learning, for example Kok and Domingos [2005], based on earlier work by Hulten and Domingos [2002], uniformly sample the groundings of an MLN to estimate the likelihood.



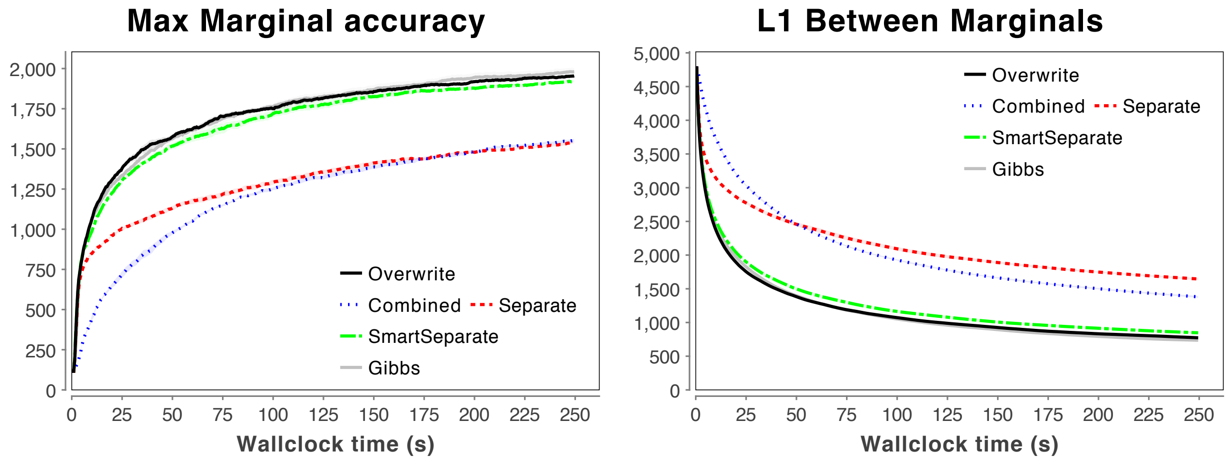
(a) Max Marginal Accuracy

(b) L_1 Error in Marginals

(c) Time to reach fixed accuracy

(d) Speedup to reach fixed accuracy

Figure 5.12: **Lock-free Marginal Inference:** Accuracy of the marginals as the number of cores is varied (using the Gibbs conflict resolution)



(a) Max Marginal Accuracy

(b) L_1 Error in Marginals

Figure 5.13: **Conflict Resolution for Lock-free Inference:** Accuracy of marginals by the various conflict resolution strategies (for 16 cores)

Gonzalez et al. [2011] most similar to our conflict-free distributed inference approach since it performs partially synchronous Gibbs sampling, providing linear speedups on a multi-core setup. However, they use analysis of global properties of the graph and assume shared memory architecture, both assumptions that are not possible in really large, dynamic graphs.

Since our work has been published, a number of related approaches have addressed similar problems. Recent work on stochastic MCMC [Korattikara et al., 2014, Bardenet et al., 2014] extends our Monte Carlo MCMC approach, and provides convergence guarantees for the resulting Markov chains. Similar to our approach on asynchronous lock-free MCMC, work by Ahmed et al. [2012] uses a strategy similar to "overwrite" for coordination-free updates for LDA-like models, and are not able to guarantee convergence either. Pan et al. [2013] use coordination-free distributed MCMC for performing scalable inference based on optimal concurrency control from the database community [Kung and Robinson, 1981].

5.6 Conclusion and Future Work

We identify a number of important drawbacks in the synchronous distributed MCMC framework. First, the partitioning of the model imposes strict restrictions on the proposals that can be explored on each worker. Second, the Map-Reduce framework contains an inherent synchrony bottleneck that is exacerbated with multiple iterations and dynamic loads. In this chapter, we propose a number of alternatives that collectively explore asynchronous updates for MCMC, and address these concerns. In Section 5.1, we propose a stochastic evaluation of the proposal function, demonstrating impressive empirical speedup, but negative theoretical properties, when factors are ignored during inference. We describe the synchrony bottleneck in Section 5.2, and propose an asynchronous version of the distributed MCMC that is conflict-free, however imposes restrictions on the proposals to guarantee convergence in Section 5.3. In Section 5.4, we explore an extension that drops the locking requirement to allow potentially conflicting updates, and propose a number of conflict-resolution strategies ranging from exact but slow, to inaccurate but fast. Experiment results for all the proposed techniques show asynchronous updates can be useful in practice for large, dense graphical models.

A number of possible future studies are apparent. We would like to provide an analysis of a broader range of algorithms, such as showing convergence for multiple conflict resolution strategies. The issue of combining the approaches proposed in this chapter with adaptive MCMC is intriguing and worth exploring. A large-scale deployment on a cluster with many compute nodes, and on a large graphical model, would be useful in validating the scalability of the proposed approaches.

CHAPTER 6

BELIEF PROPAGATION OVERVIEW

> *Knowledge is like money: to be of value it must circulate, and in circulating it can increase in quantity and, hopefully, in value.*
>
> Louis L'Amour

In this chapter we provide an overview of belief propagation, a widely studied variational inference technique for marginal inference. Since the model distribution p is often intractable to perform inference over, variational approaches usually introduce an approximating distribution q that lies in a tractable family, and perform optimization to discover the q that is *closest* to the desired distribution p. A popular instantiation of variational inference is belief propagation, which optimizes the variational objective by employing message passing over variables and the factors. The algorithm is deterministic, exact for tree-shaped models, and often produces accurate marginals in practice. Belief Propagation is primarily used for computing marginals, however a simple modification results in the Max-Product algorithm that can be used to compute the MAP configuration, and is often more efficient than alternative formulations [Yanover et al., 2006]. We contrast belief propagation with sampling in Section 6.1, followed by a detailed introduction to the BP algorithm and its variants in Section 6.2, focusing particularly on generalized BP in Section 6.3. Finally, in Section 6.4, we describe the advantages and disadvantages of BP for large, dense graphical models.

6.1 Sampling versus Belief Propagation

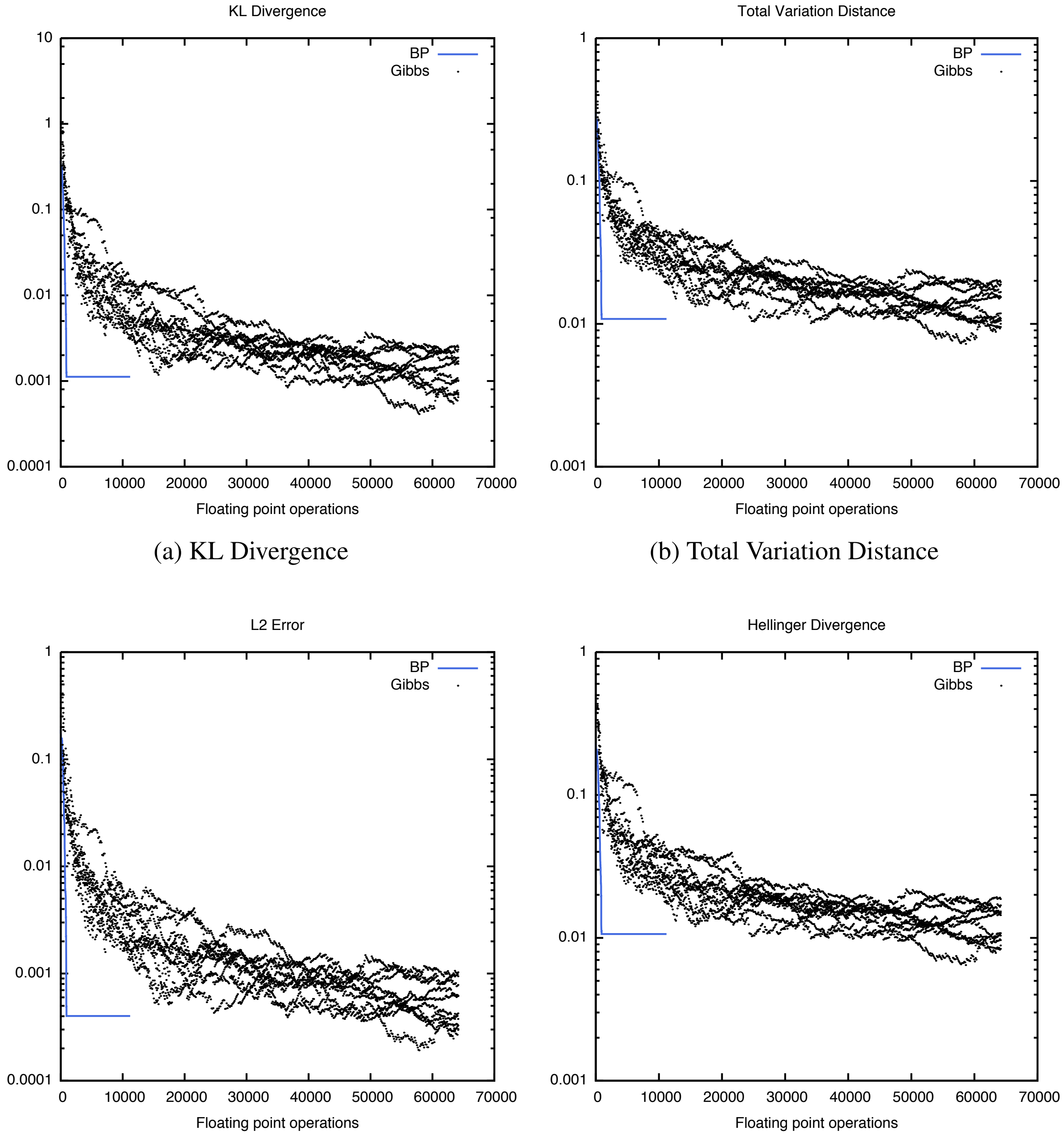
In the chapters so far, we describe approaches to scale sampling-based inference, and demonstrate its utility on large graphical models. However, sampling is not appropriate for all situations, and can sometimes take a long time to converge. First, if the model contains strong dependencies (large potentials) then MCMC has trouble getting out of local maximas. Second, sampling can take a long time before it escapes the dependence on the initial configuration and starts drawing samples from the model distribution (burn-in period). Furthermore, it is non-trivial to detect when the chains have converged (mixing time). Third, the utility of MCMC for computing marginals over the variables is limited. It usually involves running a number of parallel chains, and using the number of samples to compute marginals over, specifying both of which is more of an art than a science. Last, the MCMC process cannot easily utilize tractable sub-structures in the model, i.e. if the model mostly consists of independent chains, the inference can be performed exactly, however naive samplers are unable to do so. Depending on the number of machines available, and the complexity and scale of the model, even the distributed inference approaches introduced in the first part of the thesis may not be practical.

Belief propagation addresses a number of these concerns. Deterministic nature of the algorithm is often desirable in practice. By maintaining the distribution over all the values of the variables (that are directly multiplied with the potential), belief propagation can often escape local modes that accompany strong dependencies. Although initialization of messages can have an effect on convergence, in practice the maximum entropic messages (uniform over the domain) often work well. Since the messages can be computed independently, parallelizing belief propagation is often trivial (we elaborate in Section 6.4). Finally, the schedule of message computation can be varied to efficiently utilize tractable sub-structures such that trees and chains.

The choice of inference algorithm eventually depends on the model and particular needs of the application, since sampling does provide a number of benefits over belief propagation. First, the variational approximation may not be tight enough for certain models, resulting in poor anytime properties (quickly obtains marginals that may be inaccurate but further inference does not improve them). MCMC, on the other hand, can obtain accurate marginals for complex marginals in the asymptote. Second, the variational objective may not be easy to optimize, and often non-convex approximations are introduced for efficiency, resulting in no guarantees of convergence, or convergence to the global optima if the algorithms do converge. Although some of the convergence considerations are well-understood (we provide a brief survey in Section 6.2.3), for example cycles of strong dependencies prove challenging, it is often not clear in practice how the algorithm will behave for a given model. Third, since BP represents the full marginals for each variable and factor, it scales poorly with large domains and high-order factors (see Section 6.4 for an overview). Except in restricted cases [Minka and Winn, 2008], BP is not able to perform inference for general dynamic factor graphs (the set of instantiated factors depend on the value of the variables) without unrolling all the factors. Finally, depending on the structure of the factor potentials and the variational approximation desired, it is not trivial to implement the inference algorithm, and we often need to derive the message update rules specific to the factors. Given the various advantages and disadvantages for sampling and belief propagation, a number of approaches known collectively as Rao-Blackwellization [Doucet et al., 2000] have been proposed to combine the two, including application to parallel inference for large graphs [Gonzalez et al., 2011].

To illustrate the behavior of sampling as compared to belief propagation, we generate synthetic model of 4×4 grid containing binary variables, with local log factor scores sampled from shifted gaussians, $\psi_i = \mathcal{N}(0,1) \pm 1$ and pairwise factors $\psi_{ij} = 1$ if $y_i = y_j$, -1 otherwise. We show the accuracy of marginals obtained by BP and 10 chains of Gibbs sampling in Fig. 6.1. For all the of marginal error metrics, we notice that BP converges very quickly to low-error marginals while Gibbs sampling takes a substantially longer time to reach the same level of accuracy. On the other hand, if Gibbs is run for a longer time, it achieves marginals with a lower error. Note that although the marginals for BP are fairly accurate for this model, the variational objective for a different model may result in marginals with considerably higher error.

Belief Propagation is especially appropriate for joint inference (Section 2.4.3). Although the composition of multiple tasks results in a dense, intractable model, the individual tasks themselves are often small and tractable. Given these subs-structures, belief propagation can utilize their tractability to converge faster for the global, joint model. Second, the additional factors that represent dependencies between the tasks are often very strong. Sampling is likely to use a large number of samples to escape these dependencies, and requires “joint” proposals that are difficult to design. On the other hand, belief propagation directly incorporates factors with strong potentials. Last, the



(a) KL Divergence

(b) Total Variation Distance

(c) L_2 Error

(d) Hellinger Distance

Figure 6.1: Comparison of Sampling with Belief Propagation

number of maximas in the joint configuration space is much larger than the number of maximas in the separate spaces of individual tasks. Sampling will require a larger number of samples to escape these maximas, however belief propagation can keep the spaces of individual tasks separate.

6.2 Belief Propagation

We now cover background material on belief propagation for marginal inference. We introduce the algorithm and the message computations of belief propagation in Section 6.2.1. To provide understand of the BP fixed points, Section 6.2.2 describes the variational objective and the Bethe approximation, and present the convergence properties in Section 6.2.3.

6.2.1 Algorithm

Belief propagation is a form of marginal inference based on message-passing over the edges of the factor graph. Since each factor can be seen as inducing a distribution over its neighborhood using ψ_f, the belief propagation can be interpreted as the factors *negotiating* with each other on the marginals of the variables. The algorithm maintains a set of directed *messages* for all edges of the graph that are a real-valued functions over the domain of the variable they touch, i.e. $\forall f \in \mathcal{F}, Y_i \in \mathbf{Y}, \boldsymbol{m}_{f\to i}$ and $\boldsymbol{m}_{i\to f}$ represent a distribution over $y_i \in \mathcal{Y}_i$. The algorithm consists of iterating through the factors in the model, and updating the messages in the neighborhood of the factor to be consistent with each other and with the factor potential (as they all represent distributions over the same variable). Iteratively, a message is selected and updated using the neighboring messages.

We present the outline of belief propagation in Algorithm 6.1. A number of variations of belief propagation arise from various ways the message queue is prioritized, ranging from simply iterating through the factors or picking randomly, to tree-based schedules, and to dynamic schedules based on the state of inference. Initially belief propagation was defined as a synchronous algorithm, however the asynchronous version described here has been shown to converge faster [Elidan et al., 2006]. Approaches also vary in the message update rules. Next we describe the standard loopy

Algorithm 6.1 Belief Propagation: Queue-based framework for Belief Propagation

```
procedure BELIEFPROPAGATION($\mathcal{G}$)
    $\forall f \in \mathcal{F}$; $Q_m \overset{+}{\leftarrow} f$;          ▷ Add all factors to the message queue
    while converged do
        $f \leftarrow Q_m.pop$                  ▷ If prioritizing, factor with maximum priority
        MESSAGEPASSING($f$, $\mathcal{G}$)
        for $Y \leftarrow \mathcal{N}(f); f' \leftarrow \mathcal{N}(Y)$ do
            $Q_m \overset{+}{\leftarrow} f'$                ▷ Update message priorities if prioritizing
        end for
    end while                                   ▷ Converged on whole graph
end procedure
procedure MESSAGEPASSING($f$, $\mathcal{G}$)
    $\forall Y_i \leftarrow \mathcal{N}(f)$, Compute $\boldsymbol{m}_{i\to f}$      ▷ Compute all incoming messages, Eq. (6.1)
    $\forall Y_i \leftarrow \mathcal{N}(f)$, Compute $\boldsymbol{m}_{f\to i}$      ▷ Compute all outgoing messages, Eq. (6.2)
end procedure
```

belief propagation algorithm that we will use in the rest of the paper, and mention a few of the variations.

6.2.1.1 Message Computations

In loopy belief propagation [Pearl, 1988], the message from a variable i to a factor f is computed as follows ($\boldsymbol{m}_{i\to f} : \mathcal{Y}_i \to \mathcal{R}$):

$$\boldsymbol{m}_{i\to f}(y \in \mathcal{Y}_i) = \prod_{f' \in \mathcal{N}(Y_i), f' \neq f} \boldsymbol{m}_{f'\to i}(y) \tag{6.1}$$

While the message from a factor f to a variable i is computed as follows ($\boldsymbol{m}_{f\to i} : \mathcal{Y}_i \to \mathcal{R}$):

$$\boldsymbol{m}_{f\to i}(y \in \mathcal{Y}_i) = \sum_{\mathbf{y}'_f : y'_i = y} \exp\{\psi_f(\mathbf{y}'_f)\} \prod_{Y'_i \in \mathbf{Y}_f, i' \neq i} \boldsymbol{m}_{i'\to f}(y'_{i'}) \tag{6.2}$$

The belief propagation iteratively updates the messages until convergence. Note that all messages $\boldsymbol{m}_{i\to f}$ are initialized to be uniform, and messages $\boldsymbol{m}_{f\to i}$ may be initialized to the distribution induced by ψ_f. The messages are often represented in the $\log$ space for numerical and efficiency benefits. The final marginals can be computed for each variable i from its incoming messages:

$$p_i(y \in \mathcal{Y}_i) = \frac{\prod_{f:Y_i \in \mathbf{Y}_f} \boldsymbol{m}_{f \to i}(y)}{\sum_{y' \in \mathcal{Y}_i} \prod_{f:Y_i \in \mathbf{Y}_f} \boldsymbol{m}_{f \to i}(y')} \tag{6.3}$$

A variation of the belief propagation can also be used to perform MAP inference. In particular, if we replace the summation in Eq. (6.2) with a maximization, the resulting algorithm is known as Max-Product and can be used for MAP inference. Similar to the choice of schedules for regular belief propagation, the schedule that propagates messages from the leaves to root (and back) computes the exact MAP configuration for tree-shaped models.

6.2.1.2 Dynamic Message Scheduling

The *schedule* of which messages to propagate has a significant impact on the quality of the resulting marginals, and and may be controlled by specifying the priority in the message queue. In fact, for tree-shaped models (where the factors form a tree), a fixed schedule that goes from leaves to a root (and back) results in exact marginals on the variables. For such models, we include the direction of messages in the queue when adding a factor, and only include the factors in the reverse direction when the leaves (as per the BFS ordering) of the model are reached. For non-tree shaped, loopy models there are no such guarantees, however tree-based schedules have also been employed successfully for loopy models [Wainwright et al., 2003].

More recently, a number of dynamic scheduling algorithms have been proposed for loopy models. By using the *residual* of the messages, i.e. the change in the message in the last update, belief propagation inference can target message passing to the areas of the graph that would benefit the most from an update [Elidan et al., 2006]. Extensions to this approach have proposed more efficient alternatives [Sutton and McCallum, 2007b], alternatives to message residuals [Ihler et al., 2005b], and use in parallel inference Gonzalez et al. [2009a].

6.2.2 Optimization perspective of BP

To understand the runtime-properties of belief propagation, and to analyze its extensions, it is important to characterize the objective that the message updates are trying to optimize. With this aim, we describe the variational inference framework for marginal inference in Section 6.2.2.1, followed by the description of the Bethe approximation for loopy belief propagation in Section 6.2.2.2.

6.2.2.1 Variational Inference

When performing approximate inference, we represent the approximate marginals $\boldsymbol{\mu} \equiv (\boldsymbol{\mu}_{\mathcal{X}}, \boldsymbol{\mu}_{\mathcal{F}})$ that contain elements for every assignment to the variables $\boldsymbol{\mu}_{\mathcal{X}} \equiv \mu_i(y_i), \forall Y_i, y_i \in \mathcal{Y}$ and factors $\boldsymbol{\mu}_{\mathcal{F}} \equiv \mu_f(\mathbf{y}_f), \forall f, \mathbf{y}_f \in \boldsymbol{\mathcal{Y}}_f$. The variational marginals $\boldsymbol{\mu}$ may be used to define a distribution $q_{\boldsymbol{\mu}}(\mathbf{y})$ over our configuration space. Correct marginals corresponds to $\boldsymbol{\mu}_{\mathcal{F}} = \{\ldots, p(\mathbf{y}_f), \ldots\}$ and $\boldsymbol{\mu}_{\mathcal{X}} = \{\ldots, p(y_i), \ldots\}$ as defined in Section 2.2.2. Minimizing the KL divergence between the desired marginals defined by p and approximate marginals results in the following optimization:

$$\min_{\boldsymbol{\mu}\in\mathcal{M}} KL(q_{\boldsymbol{\mu}}||p) = \min_{\boldsymbol{\mu}\in\mathcal{M}} \sum_{\mathbf{y}} q_{\boldsymbol{\mu}}(\mathbf{y}) \log q_{\boldsymbol{\mu}}(\mathbf{y}) - q_{\boldsymbol{\mu}}(\mathbf{y}) \log p(\mathbf{y}) \tag{6.4}$$

$$\Rightarrow \min_{\boldsymbol{\mu}\in\mathcal{M}} -\sum_{\mathbf{y}} q_{\boldsymbol{\mu}}(\mathbf{y})\pi(\mathbf{y}) + \log Z \sum_{\mathbf{y}} q_{\boldsymbol{\mu}}(\mathbf{y}) + \sum_{\mathbf{y}} q_{\boldsymbol{\mu}}(\mathbf{y}) \log q_{\boldsymbol{\mu}}(\mathbf{y}) \tag{6.5}$$

$$\Rightarrow \min_{\boldsymbol{\mu}\in\mathcal{M}} -\sum_{f\in\mathcal{F}} \sum_{\mathbf{y}_f} \boldsymbol{\mu}_f(\mathbf{y}_f)\psi_f(\mathbf{y}_f) + \log Z - H(\boldsymbol{\mu}) \tag{6.6}$$

$$\Rightarrow \max_{\boldsymbol{\mu}\in\mathcal{M}} \sum_{f\in\mathcal{F}} \sum_{\mathbf{y}_f} \mu_f(\mathbf{y}_f)\psi_f(\mathbf{y}_f) + H(\boldsymbol{\mu}) \tag{6.7}$$

where $\mathcal{M}$ is the set of *realizable* mean vectors $\boldsymbol{\mu}$ that constrains $\boldsymbol{\mu}$ to be valid distributions, and $H(\boldsymbol{\mu})$ is the entropy of the distribution that yields $\boldsymbol{\mu}$. The maximizer $\boldsymbol{\mu}^* \in \mathcal{M}$ is the mean vector that corresponds to the result of marginal inference for Equation (6.7). The corresponding minimization (negative of the variational objective) is often known as the Gibbs free energy.

6.2.2.2 Bethe approximation

Both the polytope $\mathcal{M}$ and the entropy H need to be approximated in order to efficiently solve the maximization. Belief propagation approximates $\mathcal{M}$ using the *local polytope*:

$$\mathcal{L} \triangleq \Bigg\{ \boldsymbol{\mu} \geq 0,\ \ \forall f \in \mathcal{F} : \sum_{\mathbf{y}_f} \mu_f\left(\mathbf{y}_f\right) = 1, \\ \forall f, i \in \mathcal{N}(f), y_i \in \mathcal{Y}_i : \sum_{\mathbf{y}'_f, y'_i = y_i} \mu_f\left(\mathbf{y}'_f\right) = \mu_i\left(y'_i\right) \Bigg\}, \tag{6.8}$$

that defines consistency and normalization constraints locally for each factor, ignoring global consistency. BP also approximates the entropy using Bethe approximation:

$$H_B\left(\boldsymbol{\mu}\right) \triangleq \sum_f H\left(\boldsymbol{\mu}_f\right) - \sum_i \left(d_i - 1\right) H\left(\boldsymbol{\mu}_i\right), \tag{6.9}$$

The combination of using the local marginal polytope and the Bethe approximation to entropy leads to the the Bethe optimization (or Bethe free energy for the minimization):

$$\max_{\boldsymbol{\mu} \in \mathcal{L}} \sum_{f \in \mathcal{F}} \sum_{\mathbf{y}_f} \mu_f(\mathbf{y}_f) \psi_f(\mathbf{y}_f) + H_B\left(\boldsymbol{\mu}\right) \tag{6.10}$$

Note that although we are relaxing the marginal polytope constraint, Bethe entropy in fact is not an upper bound on the complete entropy, hence the above approximation is not an upper bound on the variational objective.

To show that belief propagation is related to the Bethe optimization, we write the Lagrangian relaxation of this optimization as:

$$L_{\text{BP}}\left(\boldsymbol{\mu}, \boldsymbol{\lambda}\right) \triangleq \sum_{f \in \mathcal{F}} \sum_{\mathbf{y}_f} \mu_f(\mathbf{y}_f) \psi_f(\mathbf{y}_f) + H_B(\boldsymbol{\mu}) + \sum_f \lambda_f C_f(\boldsymbol{\mu}) + \sum_f \sum_{i \in \mathcal{N}(f)} \sum_{y_i} \lambda_{fi}^{y_i} C_{f,i,y_i}(\boldsymbol{\mu}) \tag{6.11}$$

where

$$C_{f,i,y_i} = \mu_i(y_i) - \sum_{\mathbf{y}'_f : y'_i = y_i} \mu_f(\mathbf{y}_f) \tag{6.12}$$

$$C_f = 1 - \sum_{\mathbf{y}_f} \mu_f(\mathbf{y}_f) \tag{6.13}$$

are the constraints that correspond to the local polytope $\mathcal{L}$. BP messages correspond to the dual variables, i.e. $\boldsymbol{m}_{f\to i}(y_i) \propto \exp \lambda_{fi}^{y_i}$. If the messages converge, Heskes [2002] show that the marginals correspond to a $\boldsymbol{\mu}^*$ and $\boldsymbol{\lambda}^*$ at a saddle point of L_{BP}, i.e. $\nabla_{\boldsymbol{\mu}} L_{\text{BP}}\left(\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*\right) = 0$ and $\nabla_{\boldsymbol{\lambda}} L_{\text{BP}}\left(\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*\right) = 0$. In other words: at convergence BP marginals are locally consistent and locally optimal.

6.2.3 Convergence and Convergent Variants

Although in practice belief propagation often converges and computes accurate marginals [Murphy et al., 1999], it is not guaranteed to do so. A number of approaches have characterized the convergence properties of BP to better understand the fixed points [Tatikonda and Jordan, 2002, Ihler et al., 2005b, Mooij and Kappen, 2012], and use damping of messages to ensure convergence [Heskes, 2004]. Apart from non-convergence, it's also possible that loopy belief propagation does converge, but to a poor local minima.

Extensions to loopy belief propagation have been proposed to that instead optimize convex objectives [Globerson and Jaakkola, 2007a]. Tree-reweighted belief propagation [Wainwright et al., 2003] uses a distribution over spanning trees of the model to define a convex objective for message passing. Although it provides strong convergence guarantees [Roosta et al., 2008], the distribution over spanning trees is difficult to specify, and loopy belief propagation often produces better marginals. Hazan and Shashua [2008] generalize TRBP to propose a family of convergent objectives using counting numbers, which they use in conjunction with the model structure to identify convex objectives *closest* to the Bethe objective. By realizing that these convex approximations still fare poorly as compared to Bethe approximation, Meshi et al. [2009] propose computing the counting numbers to directly find a convex approximation of Bethe, using the model structure and the factor potentials to do so.

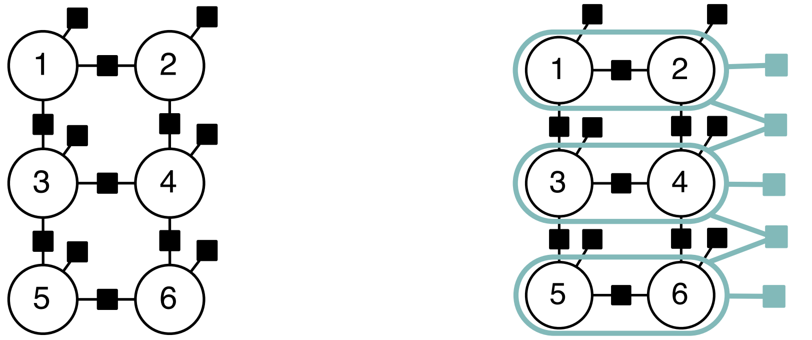
In the next section, we will describe a class of algorithms that specify messages on *clusters* of variables. By operating on junction graphs (or region graphs), these approaches optimize a tighter approximation than Bethe, and are exact on a larger class of models (all low-tree width models). However, the computational complexity of such algorithms is exponential in the size of the clusters, and these also face the non-convergence and convergence to a local minima issues of belief propagation.

6.3 Generalized Belief Propagation

As we show in Eq. (6.7), the true marginals maximize the variational objective. Belief propagation, however, optimizes an approximation known as the Bethe free energy, as defined in Eq. (6.10). Kikuchi [1951] generalizes the Bethe free energy and propose a family of approximations that are tighter than Bethe energy, known collectively as Kikuchi free energies. Yedidia et al. [2000] introduce generalized belief propagation (GBP), a family of of message passing algorithms that send messages between *clusters* of variables and factors (of which loopy belief propagation is a special case), and show that GBP optimizes the Kikuchi free energy. In this section we provide background on cluster graphs and the GBP algorithm, and then define the Kikuchi approximation that GBP optimizes.

Give our graphical model $\mathcal{G}$ containing variables $\mathbf{Y}$ and factors $\mathcal{F}$, we define a cluster graphical model $\mathcal{C}_\mathcal{G}$ containing cluster variables $\mathbf{C}$ and cluster factors $\mathbf{\Phi}$. Each variable Y belong to one and only cluster in the cluster graph, i.e. $\forall C \in \mathcal{C}, C \subseteq \mathbf{Y}$, $\bigcup_{C \in \mathcal{C}} C = \mathbf{Y}$ and $\forall C_i, C_j \in \mathcal{C}, C_i \cap \mathcal{C}_j = \emptyset$[1]. Each cluster factor $\Psi \in \mathbf{\Psi}$ contains all the factors ψ_f that neighbor variables only in the cluster variables of Ψ, i.e. $\forall f \in \Psi, \forall C \in \mathcal{N}(\Psi), \exists Y$ s.t. $Y \in C \wedge Y \in \mathcal{N}(f)$. Hence given

[1] Note that although generalized BP supports overlapping clusters/regions, we assume cluster variables as used in Pearl's clustering method [Pearl, 1988]. Yedidia et al. [2000] show GBP is a generalization of the cluster graph approach, however one can also include overlapping clusters in the above approach by creating auxiliary copy of the variables and additional deterministic equality factors.



(a) Factor Graph $\mathcal{G}$

(b) Cluster Graph for $\mathcal{G}$, $\mathcal{C}_\mathcal{G}$

Figure 6.2: Example of a Graphical Model and a Cluster Graph

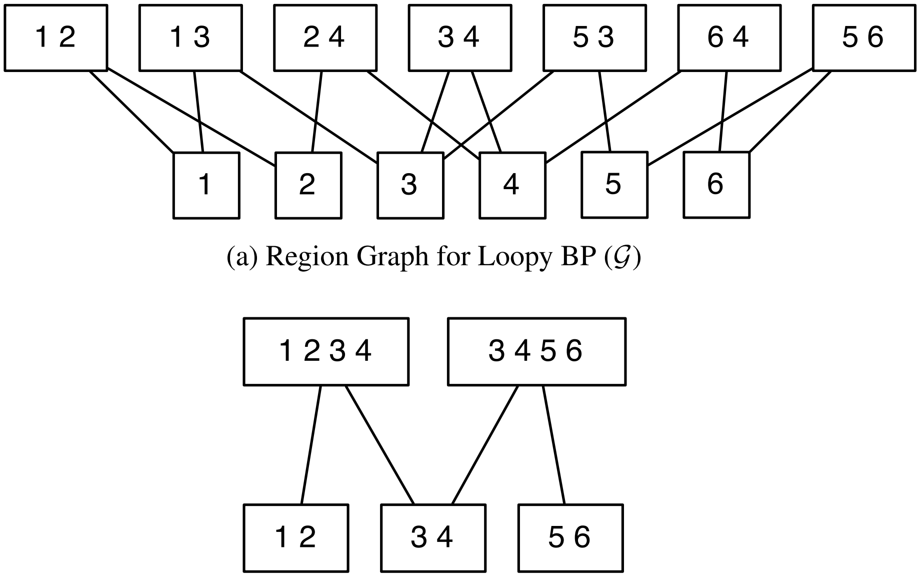
the graphical model $\mathcal{G}$ and a partitioning of its variables $\mathbf{C}$, the cluster graphical model $\mathcal{C}_\mathcal{G}$ is directly determined based on the variable-factor neighborhoods. It is straightforward to see that the probability distribution defined by $\mathcal{C}_\mathcal{G}$ is same as that by $\mathcal{G}$, i.e. the following holds, where $\Psi(\mathbf{y}_\Psi) = \sum_{f\in\Psi}\psi_f(\mathbf{y}_f)$:

$$p_\mathcal{G}(\mathbf{Y}) = \frac{\exp\sum_{f\in\mathcal{F}}\psi_f(\mathbf{y}_f)}{\sum_{\mathbf{y}\in\mathcal{Y}}\exp\sum_{f\in\mathcal{F}}\psi_f(\mathbf{y}_f)} \tag{6.14}$$

$$= \frac{\exp\sum_{\Psi\in\boldsymbol{\Psi}}\Psi(\mathbf{y}_\Psi)}{\sum_{\mathbf{y}\in\mathcal{Y}}\exp\sum_{\Psi\in\boldsymbol{\Psi}}\Psi(\mathbf{y}_\Psi)} = p_{\mathcal{C}_\mathcal{G}}(\mathbf{y}) \tag{6.15}$$

Figure 6.2a shows a simple 6 variable factor graph, and a possible cluster graph with 3 cluster variables and 5 cluster factors is shown in Fig. 6.2b.

Inference for cluster graphs can be performed using generalized belief propagation. Generalized BP operates on a *region graph* that contains subsets of variables as nodes/regions, with a hierarchy amongst the regions as defined by subset containment. The edges in the region graph exist between a region and its *direct* sub-regions, and message passing operates along these edges. The region graphs with the original set of variables and factors consist of a separate region for each factor (assuming each variable includes a local factor), resulting in a 2-layer region graph (see Fig. 6.3a for an example). The cluster graph formulation described above also results in a



(a) Region Graph for Loopy BP ($\mathcal{G}$)

(b) Region Graph for Cluster BP ($\mathcal{C}_\mathcal{G}$)

Figure 6.3: Region Graphs for Generalized Belief Propagation for models from Fig. 6.2

similar region graph, defined over the cluster factors (again, assuming a local cluster factor accompanies each cluster variable), as shown in Fig. 6.3b.

For a given region graph, it can be shown that the fixed point of generalized BP corresponds to the Kikuchi approximation, which generalizes the Bethe approximation to region graphs. In particular,

$$\max \sum_{r \in \mathcal{R}} c_r \left(\sum_{\mathbf{y}_r} \mu_r(\mathbf{y}_r) \Psi_r(\mathbf{y}_r) + H_r(\mu_r) \right) \tag{6.16}$$

where $\Psi_r(\mathbf{y}_r) = \sum_{f_r} \psi_{f_r}(\mathbf{y}_{f_r})$, $H_r(\mu_r) = -\sum_{\mathbf{y}_r} \mu_r(\mathbf{y}_r) \log \mu_r(\mathbf{y}_r)$, and $c_r = 1 - \sum_{s \in \text{super}(r)} c_s$. This approximation is formulated for a general region graph, and using the region graphs as presented in Fig. 6.3 we can define the optimization for loopy belief propagation and cluster belief propagation (the former, not surprisingly, results in the Bethe objective from Eq. (6.10)).

Kikuchi approximation to the original variational objective can be much tighter than the Bethe approximation. If the cluster graph forms a tree, generalized BP is exact even if the original model contains loops (for example the model in Fig. 6.2b). Further, large clusters lead to tighter approximations even if the resulting cluster graph contains cycles, and generalized BP takes fewer iterations of message passing to achieve the fixed point. Unfortunately, the computational cost increases exponentially with the size of the clusters, and hence generalized belief propagation is often not practical for model where the variables have a large domain, or the factors are incredibly dense such that the Kikuchi approximation is not much tighter.

6.4 Belief Propagation for Large, Dense Models

Since the message passing operations are local, and asynchronous updates to messages have been shown to converge well, we expect belief propagation to scale to large models by parallelizing the message computations. Unfortunately, there are a number of drawbacks for using BP on large, dense models that we describe next.

6.4.1 Expense of each message computation

The graphical models of our interest often contain variables with large domains, and factors with a large number of neighbors (high-order factors). This is especially common in joint inference, in which combining multiple tasks naturally increases the neighborhood of the factors that capture the dependencies across the tasks. Even though the probability distribution is extremely complex, such variables and factors do not pose a computational challenge for sampling since it operates on a per value basis. Belief propagation, on the other hand, maintains a full distribution on the domain of the variables, and hence is directly affected by large domains. More importantly, the message computations for factors iterates through all the value assignments to the neighboring variables, requiring $O(|\mathcal{Y}|^{|\mathcal{N}(f)|})$ operations. In Chapter 7 we focus on techniques to reduce the number of operations for marginalizing such factors.

6.4.2 Too many messages or iterations to converge

As we show in Section 6.2.2, loopy belief propagation optimizes the Bethe free energy that is exact for tree-shaped model, but is a substantially poorer approximation for models with large treewidths. Apart from the fact that this objective may result in poor marginals, a poor approximation also indicates belief propagation will require a larger number of iterations of message passing to converge. Although this is a concern even on a single machine, more iterations of message passing on a distributed system implies more communication between nodes, which is a costly operation. In Chapter 8 we focus primarily on designing approximations that reduce the number of iterations (across machine communication) at the expense of additional *local* computations.

6.4.3 Related Work

Due to the importance of belief propagation marginal inference, a number of techniques have scaled belief propagation to large models. As we mention above, belief propagation can be parallelized by computing the messages in parallel for general graphs. Kozlov and Singh [1994, 1996] improve upon this by additionally parallelizing the computations within each message. [Gonzalez et al., 2009a] adaptively identify subsets of variables to perform inference over for each core, essentially combining synchronous parallelism and message scheduling in the shared memory setting. However, inference on massive data sets requires *distributed* inference frameworks deployed over large clusters where the shared-memory assumption of the multi-core setting does not hold. Work on distributed belief propagation [Gonzalez et al., 2009b] performs independent inference on the partitions of the model, asynchronously communicating the messages; the model is partitioned such that within-partition variables are more likely to communicate. A similar approach by Schwing et al. [2011] consists of independent inference interleaved with synchronous communication phases (similar to running multiple Map-Reduce iterations). Ihler et al. [2005a] address the communication cost of distributed inference messages by sampling. There has been work on distributing message passing for specific models [Paskin et al., 2005, Funiak et al., 2006, Smola and Narayanamurthy, 2010, Strandmark and Kahl, 2010] however it is difficult to use the resulting algorithms for other tasks. The concerns with scaling belief propagation that we raise earlier are only partially addressed by these existing approaches.

Due to the widespread applicability of belief propagation as the choice of inference, studying approximations to belief propagation is an active field of research in machine learning. A number of approaches have studied pruning of variables values for more efficient inference. Early work by Coughlan and Ferreira [2002] prunes values when their current belief falls under a threshold, followed by a dynamic extension by Coughlan and Shen [2007]. Also similar, Komodakis and Tziritas [2007] combine label pruning with entropy-based message scheduling. Instead of approximating the variable domains, efficient approximations to messages have also been proposed. Song et al. [2011] introduce a kernel-based representation of the messages to provide efficient update computations independent of the domain size. Ihler et al. [2004, 2005b] provide a theoretical framework to analyze the bounds on errors for BP with message approximations. Work by Sudderth et al. [2003] and the following generalization by Ihler and McAllester [2009] maintains samples on messages. Noorshams and Wainwright [2011] directly address the complexity of factor marginalization by proposing stochastic updates on the message computations, thus reducing the polynomial dependence on the domain size.

Motivated by real-world applications of joint inference, our work presented in the next two chapters is focused on combining the insights from parallel and distributed computing with approximations to messages. Both provide advantages that are mutually beneficial, can can lead to approaches that are able to obtain accurate marginals for complex models with high-order factors and large domains, with minimal communication and synchronization cost.

CHAPTER 7

BELIEF PROPAGATION WITH SPARSE DOMAINS

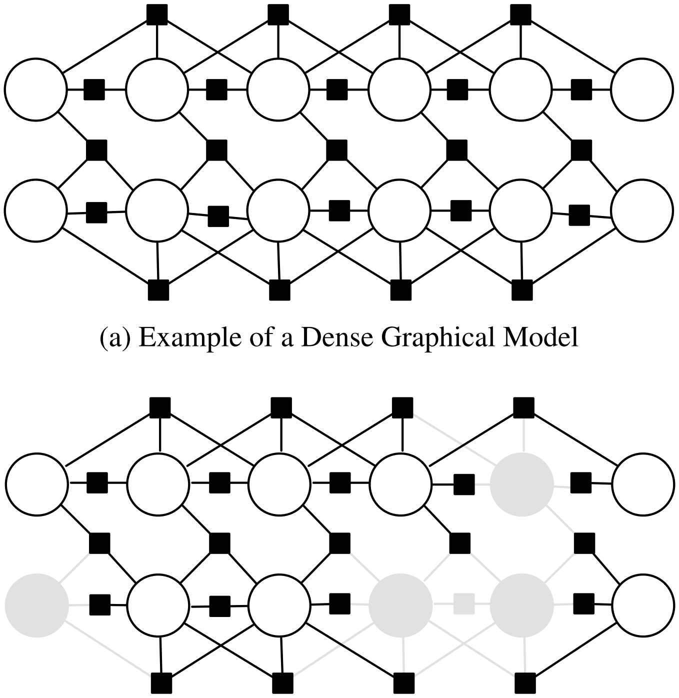
Knowledge is power only if man knows what facts not to bother with.
Robert Staughton Lynd

For marginal inference on graphical models, belief propagation (BP) has been the algorithm of choice due to impressive empirical results on many models. These models often contain many variables and factors, however the domain of each variable (the set of values that the variable can take) and the neighborhood of the factors is usually small. In this chapter, we focus on inference for models that contain variables with large domains and higher-order factors, for which BP is often intractable (Section 6.4). Large domain problems, unfortunately, are becoming increasingly important due to growing interest in joint inference in applications such as computer vision [Yao et al., 2012], natural language processing [Finkel and Manning, 2009], and bioinformatics [Wei and Pan, 2012]. These models represent the joint distribution over multiple tasks, resulting in variables with large domains and factors that neighbor a number of such variables. There is a need for efficient marginal inference that represents the full distribution across these multiple tasks, and current BP methods are impractically slow.

The primary reason BP is unsuitable for large domains is the cost of message computations and representation, which is on the order of the cross-product of the neighbors' domains. In this chapter we propose techniques that prevent the computation of beliefs on *all* the values, instead approximate the belief by defining it only on a subset of the values, and assume a zero probability on the rest of the values (*Sparsification*). This provides a number of benefits to belief propagation: (1) Computation of messages for each variable is efficient as it is proportional to the size of the sparse domains, (2) Messages for high-order factors can also be computed quickly for the same reason, (3) Reduction of the sparse domain to a single value allows *conditioning* of the model on the variable value, simplifying and reducing the structural complexity of the graph (speeding up inference), and (4) conditioning the model on single-valued variables can facilitate independence, resulting in higher speedups from parallelism.

First, in Section 7.1, we describe our belief propagation algorithm for single-value sparsity. Our approach is motivated by the observation that the marginal probabilities of random variables are often extremely peaked, where a single value has a substantially higher marginal probability compared to the rest of the values. We propose a simple technique to identify such high-probability values during the early stages of inference, to allow efficient inference for the remaining iterations. Further, we continually monitor the marginals and revoke sparsity if the beliefs imply so, thus dynamically revisiting and correcting our sparsity decisions. To overcome the restrictions of a single-value sparsity and a fixed level of approximation, we propose a novel *anytime* belief propagation algorithm with sparse domains in Section 7.2. The algorithm makes strong sparsity assumption at the early stages of inference, but dynamically grows the domains of variables as inference progresses. We use message-scheduling based belief propagation to ensure the consistent marginals throughout the run, and improve the quality of the marginals to eventually obtain the BP marginals.

We evaluate the two contributions on a number of models that have variables with a large domain. On linear chains for which belief propagation cannot be parallelized over more than 2 cores, we demonstrate single-value sparsity based inference that obtains high speedups with as many as 8 processors, while retaining much of the accuracy of exact inference. On synthetic grids with large domains, we use anytime BP to obtain up to 12 times speedup over regular belief propagation. For a real-world information extraction task, we introduce a model for joint inference of within-document entity types, relations, and coreference. Single-value sparsity allows joint inference to obtain higher accuracy on all three tasks over their independent, non-joint models, achieving an error reduction of up to 12.4% for entity types. For fine-grained entity type and relation prediction, we obtain more than 5 times speedup over the baseline by using Anytime BP.



(a) Example of a Dense Graphical Model

(b) Graph after inducing single-value sparsity (grey)

Figure 7.1: **Inducing Single-Value Sparsity:** Illustration of the benefit of inducing single-value sparsity that results in *deterministic* nodes. Along with decreasing the overall complexity of the model (tree width) and reducing factor neighborhood, it also results in independent components.

Work presented in this chapter has appeared as Singh et al. [2011a], Singh et al. [2013a], and Singh et al. [2013b], which can be referred to for more details.

7.1 Single-Value Sparsity

Variables with large domains are quite common in natural language processing, for example for fine-grained entity type prediction or relation extraction with a large set of possible relations. Some approaches to joint inference over multiple tasks combine variables of multiple tasks into a single one, with a domain that is the cross product of the domains of the individual tasks. Computing messages for factors that neighbor such variables is extremely expensive as the complexity is polynomial in size of the domain. This cost can be prohibitive even for exact inference in chains and trees, making BP difficult for use when models are large and/or inference needs to be fast. There is a need for efficient approximate inference that is able to handle variables with large domains, and provide speedups over exact inference in chain- and tree-shaped models.

Since belief propagation is not directly applicable, we propose to extend the algorithm to induce single-value sparsity. Our main extension stems from the observation that variables often have probability mass concentrated on a single value, and these variable marginals often peak during the initial stages of inference, without changing substantially during the rest of the course of inference. We detect such low-entropy marginals in earlier phases and *fix* the variables to their single high-probability values. The proposed algorithm continuously reassesses the sparsity decisions through the course of inference. Sparsifying the domain to a single-value provides a number of benefits to belief propagation. First, since the domain now contains only a single value, the factors that neighbor the variable can marginalize much more efficiently. Second, these fixed variables result in fewer cycles in the model and allow decomposition of the model into independent inference problems by partitioning at these fixed variables. Lastly, factors that only neighbor fixed variables can be effectively removed during inference, reducing the amount of messages that are passed. These benefits are illustrated in Fig. 7.1.

We also explore the utility of the above approach for fast, parallel inference for chains and trees. The problem of inference for tree-shaped models (such as chains) also plays an important role as a large number of real-world problems are best modeled as linear-chains or trees [Tang et al., 2006, Peng and McCallum, 2006, Settles, 2005, Liu et al., 2006, Torralba et al., 2010]. A fast approximate inference approach can allow inference over very large models (such as those that model documents instead of sentences) for which exact inference is slow. Second, even for smaller (sentence level) models, the computational limits for certain tasks may be too severe to allow exact inference, such as for query analysis for real-time search [Li et al., 2009]. Fixing the domain to a single value makes the variable deterministic, blocking the information from propagating across it. We utilize this property to effectively split the chain at the deterministic variables, facilitating parallelism.

7.1.1 Inducing Single Value Sparsity

In this work, we introduce an approach to utilize the inherent sparsity in variable marginals for faster inference. As described above, inducing *value sparsity* provides a number of benefits to belief propagation. For early detection of peaked marginals during inference, we examine the marginals of all the variables that have changed after every message propagation. When the marginal probability of a value for a variable according to the current set of beliefs goes above a predetermined probability threshold ζ, we approximate the marginals by replacing the distribution with a *deterministic* distribution that has probability of 1 for the mode of the distribution (and 0 for the others). By setting the variable to its maximum probability value, inference treats it as a fixed variable for the rest of inference.

The parameter ζ decides when to induce the sparsity on a variable, i.e. ζ defines the assumption that if the mode of the marginal has probability $\geq \zeta$, the marginal is not likely to be affected significantly by its neighboring variables. The parameter ζ directly controls the computational efficiency and accuracy trade-off, which we study in our evaluation in Section 7.1.5.1.

7.1.2 Revisiting Sparsity

Marginals of a variable may vary considerably over the course of inference, and committing to a deterministic value at an early stage of inference can lead to substantial errors. We allow revisiting of our sparsity decisions by storing the actual marginals as a separate value $\boldsymbol{m}'_i$. After every message propagation, we update $\boldsymbol{m}'_i$ and compare the resulting distribution probabilities with ζ. If the probability of all the values is $< \zeta$, we *unsparsify* the variable by setting $\boldsymbol{m}_i$ to $\boldsymbol{m}'_i$. We also allow modification of the deterministic value if the mode of $\boldsymbol{m}'_i$ is different from $\boldsymbol{m}_i$ and the probability of the mode is $\geq \zeta$.

7.1.3 Algorithm

We outline our approach in Algorithm 7.1 that consists of minor variations on Algorithm 6.1. The underlying message-passing uses a queue of factors to operate upon. Each set of message computation consumes a factor, updates the outgoing messages and marginals of the neighboring

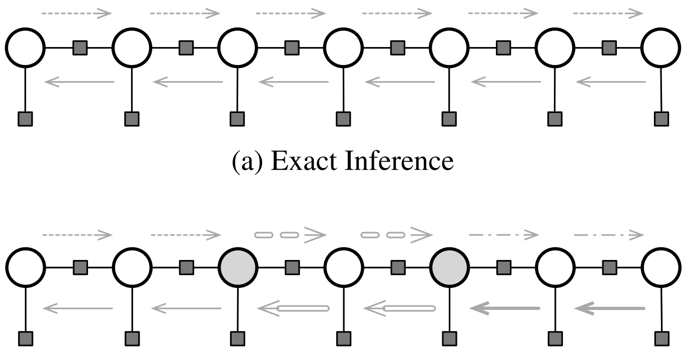
Algorithm 7.1 Single-Value Sparsity: Belief Propagation with Single-Value Sparsity

```
procedure SINGLEVALUESPARSITY($\mathcal{G}, \zeta$)
    $Q_m \overset{+}{\leftarrow} f; \forall f \in \mathcal{F}$
    while converged do                          ▷ If prioritizing, $\max(Q_m) < \epsilon$
        $f \leftarrow Q_m.pop$                  ▷ If prioritizing, factor with maximum priority
        MESSAGEPASSING($f$, $\mathcal{G}$)      ▷ Compute messages and update neighbor marginals
        for $y \leftarrow \mathcal{N}_f$ do
            SPARSIFY($y$, $\zeta$)              ▷ If $\max_y p_m(Y) \geq \zeta$, sparsify outgoing messages
            for $f' \leftarrow \mathcal{N}(y)$ do     ▷ Unless marginal of $y$ is unchanged
                $Q_m \overset{+}{\leftarrow} f'$      ▷ Update priorities if prioritizing
            end for
        end for
    end while
end procedure
```

variables, and appends the neighboring factors to the queue (with de-duplication). This framework directly supports message scheduling (Section 6.2.2.2); we use a heap-based priority queue that updates the priorities of factors as they are added. The framework can be deployed on multiple cores directly; the message updates are independent of each other and can be parallelized, and thus the speedup is restricted by the size of the queue.

We include a few additional operations in the above message passing framework to support single-value sparsity. First, if we encounter a factor for which all neighboring variables are fixed, we remove the factor from the queue and do not compute its messages. Once the messages of a factor have been computed, we examine the marginals of the neighboring variables, and sparsify or unsparsify their domains as appropriate. If the marginal of the variable is unchanged, for example the variable marginal is sparse with the same value, we do not add its neighboring factors to the queue. The combination of these modifications implements our single-value sparsity approach.

For single-value sparsity on tree-shaped models, when the marginal of a variable becomes peaked, we add factors in both directions to the queue (instead of only in one direction in the non-sparse case), since the variable has effectively partitioned the chain/tree into independent components.



(a) Exact Inference

(b) Inference with Sparse Values

Figure 7.2: **Parallel Inference in Chains:** Messages with the same arrow style are required to be assigned to the same core. By inducing sparsity on the shaded variables (b), upto 6 cores can be used, as opposed to only 2 in exact inference (a).

7.1.4 Parallelism and Sparsity for Chains

As we mention above, the number of threads that can compute messages simultaneously is restricted by the queue size. For exact inference, the queue is initialized with only 2 computations, and each step consumes a single message and adds a single message, thereby limiting the size of the queue to ≤ 2. This results in a speedup of 2 when using 2 cores, and increasing the number of cores do not provide any benefit. With our approximation, however, computations that results in a deterministic variable consume a single factor, but potentially add two more computations to the queue (in both directions). The queue can therefore grow to sizes > 2, thus allowing multiple cores can be utilized to provide speedups. Alternatively, one can view the deterministic variables as “islands of certainty” that each split the chain into two. This results in multiple forward and backward passes that are independent of each other. See Fig. 7.2 for an example.

Although we describe our approach on linear-chain models, the algorithm generalizes to the case of the tree. Exact inference in the tree is defined by selecting a root, and performing message passing in *upstream* and *downstream* passes. However, the queue size is not limited to 2; particularly the queue at the beginning of *upstream* pass and the end of *downstream* pass contains as many factors as leaves of the tree. Thus we can achieve speedups > 2 even for exact inference (see Xia and Prasanna [2008]). When including our approximation, we expect further speedups. In exact inference, as the message propagate up the tree, the size of the queue shrinks. In our approach, we are splitting the tree into multiple smaller trees, thus providing potential for utilizing multiple cores at every stage of inference. However, we do not expect the speedups to be as significant as in chains.

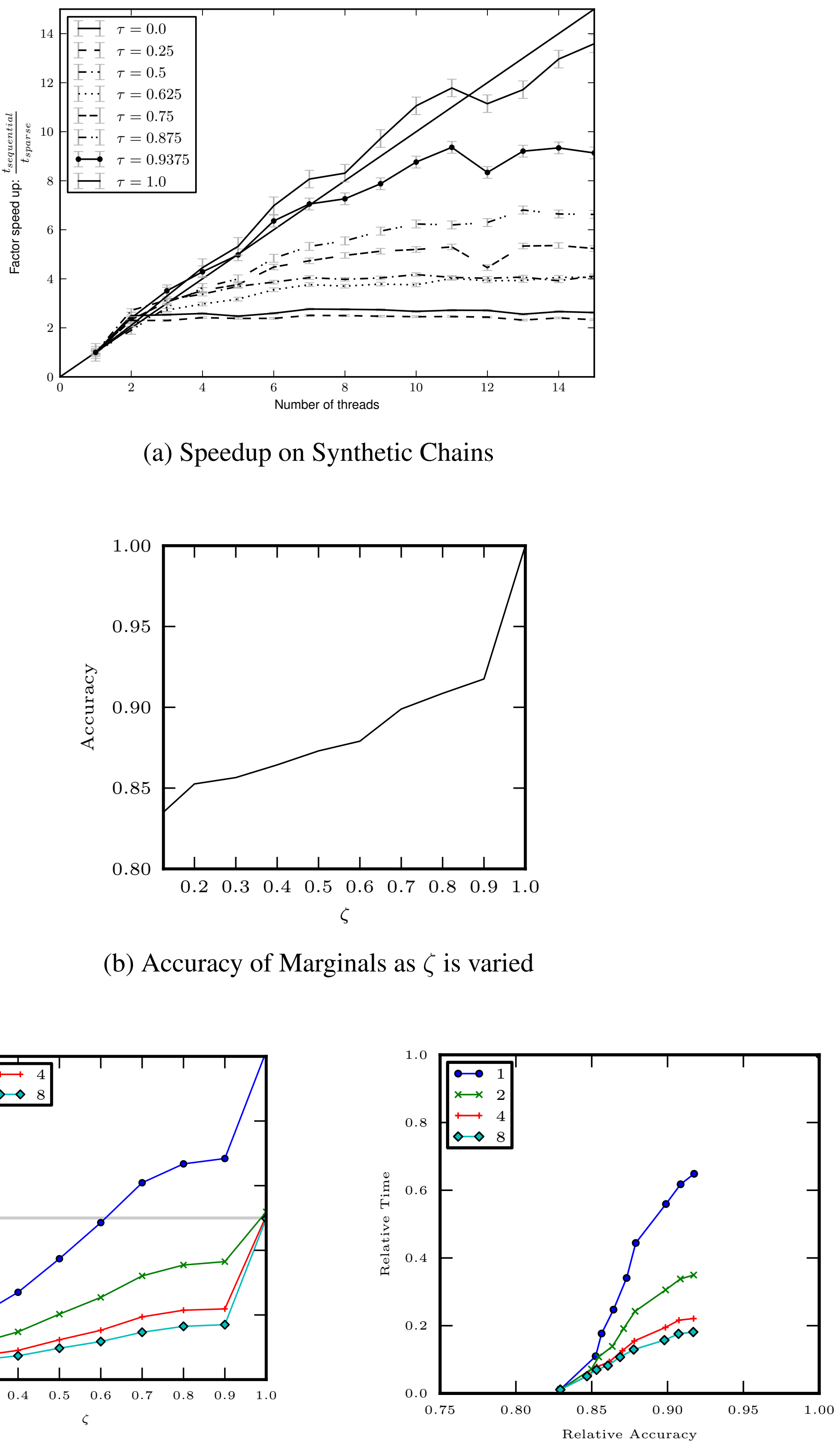
7.1.5 Experiments

In this section, we compare the accuracy and the speedups of our single-value sparsity approach to exact inference on linear chains by evaluating on both synthetic and real-world models. Further, our single-value sparsity enables joint inference amongst three within-document tasks; we define a novel joint model and demonstrate improved accuracy for all the three tasks, made feasible by our approach.

7.1.5.1 Parallel Inference for Chains

As described above, the speedup provided by multiple cores depends on the size of the queue of computations, i.e. linear speedups can be obtained if the queue size is always greater than the number of cores. However, for linear chains the queue size doesn't depend on ζ as much as it depends on *which* variables turn deterministic during inference. To study the behavior of our system as we vary the position of deterministic variables, we create synthetic chains with random potentials of length 128 with 15 deterministic variables each. The position of the deterministic variables within the chain is controlled by τ, where $\tau = 1$ denotes ideal placement (uniformly across the chain) and $\tau = 0$ denotes the worst placement (all deterministic placed together). Figure 7.3a shows the speedups obtained when using multiple cores (averaged over 100 iterations) and varying τ. We observe near-linear speedups when the placement is close to ideal, and never worse than the speedup obtained using parallel exact inference.

To observe the performance of our approach on real-world dataset, we run inference on a model of part of speech tagging on the CONLL 2003 dataset [Tjong Kim Sang and De Meulder, 2003]. A linear chain model with features on the observed word (capitalization, lowercase, etc.), chunk,



(a) Speedup on Synthetic Chains

(b) Accuracy of Marginals as ζ is varied

(c) Relative Running Time

(d) Tradeoff between Accuracy and Time

Figure 7.3: Accuracy and Speedups for Chains with Single-Value Sparsity

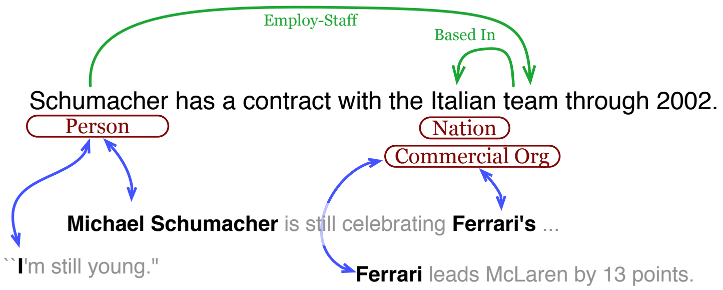
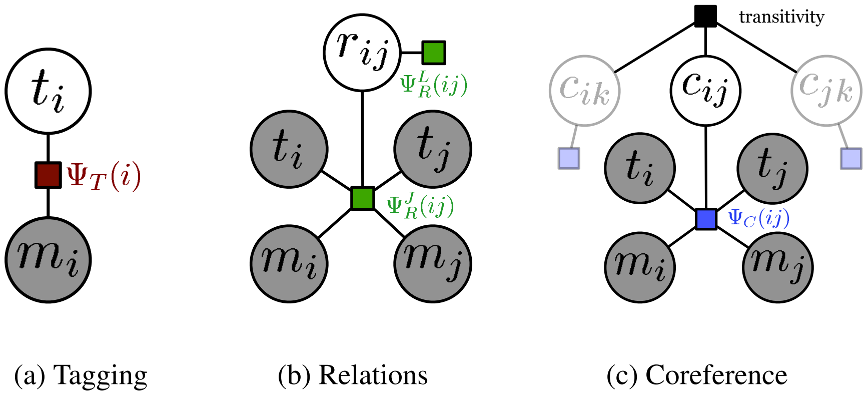

Employ-Staff
Based In
Schumacher has a contract with the Italian team through 2002.
Person
Nation
Commercial Org
Michael Schumacher is still celebrating Ferrari's ...
``I'm still young."
Ferrari leads McLaren by 13 points.

Figure 7.4: **Information Extraction:** 3 mentions labeled with entity types (red), relations (green), and coreference (blue links).

and context features is created and trained using max-likelihood with exact inference. We plot the performance of our system on 1000 sentences from the test set (and filter by size < 10) in Fig. 7.3 as we vary ζ. The drop-off in accuracy (relative to the exact marginals) as we adjust our approximation factor ζ is quite small, as shown in Fig. 7.3b. On the other hand, Fig. 7.3c shows impressive gains in running time over the sequential and the parallel exact inference (solid gray line). Figure 7.3d summarizes the accuracy versus time trade-off as the number of cores and the approximation factor is varied, demonstrating marginals can be obtained much faster for a fixed level of accuracy as the number of cores is increased.

7.1.5.2 Joint Inference of Types, Relations and Coreference

In this section, we use sparse belief propagation on a single, joint probabilistic graphical model for classification of entity mentions (*entity tagging*), clustering of mentions that refer to the same entity (*coreference resolution*), and identification of the relations between these entities (*relation extraction*). Figure 7.4 shows an annotated example sentence. The input for this task is the set mention boundaries ($\mathbf{m}$) and the sentences of a document. For each mention m_i, the output of entity tagging is a label t_i from a predefined set of labels $\mathcal{T}$ (for example PERSON, ORGANIZATION, LOCATION, etc.). Relation extraction labels each mention pair in the same sentence with its relation as expressed in that sentence, or NONE if no relation is expressed. We represent this task as variables r_{ij} that represent the type of the relation where m_i is the first argument, m_j the second



(a) Tagging (b) Relations (c) Coreference

Figure 7.5: **Individual Classification Models:** where the observed/fixed variables are shaded in grey. For brevity, we have only included (i, j) in the factor labels, and have omitted t_i, t_j, m_i and m_j.

argument, and the type comes from a predefined set of labels $\mathcal{R}$.[1] For coreference, we represent the task as classification of pairs of mentions as coreferent or not, i.e. for pairs of mentions m_i and m_j that appear in the same document, there is a variable $c_{ij} \in \{0, 1\}$. These decisions are symmetric ($c_{ij} \equiv c_{ji}$), and we only include one of these variables in our models. Coreference also requires the link decisions to be transitive.

The models used in practice model each task individually, for example the tagging model only relies of the features of the observed mention to predict the entity type [Bender et al., 2003]. The relation extraction and coreference models both condition on the *fixed* entity tags to predict the type of the relation or coreference [Zhou et al., 2005, Soon et al., 2001, Bengston and Roth, 2008]. Transitivity is captured using $O(n^3)$ deterministic factors, but this is often intractable. Instead, transitive closures of the coreferent pairs is computed as a post processing step. These models are shown in Fig. 7.5. Unfortunately, it is not possible with such models for entity tagging to benefit from the coreference or the relation extraction decisions, resulting in a *uni-directional* flow of information out of the entity tagging model. Further, this can cause a cascade of error; incorrect entity tags result in adverse effect on relations and coreference. However, evidence at the relation

[1] Note that r_{ij} and r_{ji} represent different relations.

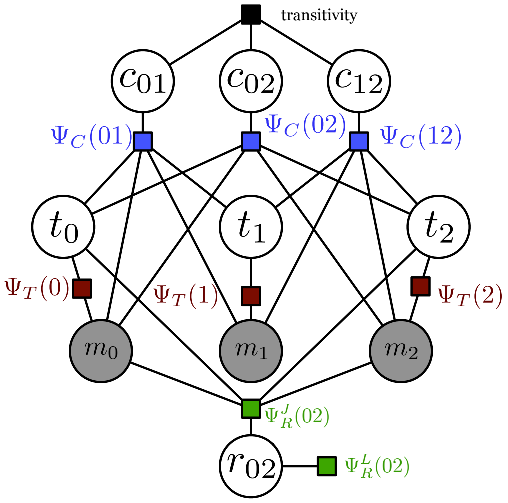


Figure 7.6: **Joint Model of Tagging, Relations, and Coreference:** The joint model for a document with 3 observed mentions (for illustration), two of which belong to the same sentence (m_0 and m_2).

or coreference levels should be used to improve the entity tagging task. Certain relations, for example, only appear between entities of a specific type. Similarly, for two mentions that are coreferent, by definition their entity types have to be the same. Isolated models do not have any direct mechanism to facilitate this *bi-directional* flow of information.

Joint Model: We need to define a model that directly represents the dependencies between the three tasks by modeling the joint distribution over them. Since the individual models defined in the previous section represent the individual tasks, we construct a joint model by combining all the variables and factors into a single graphical model, but do not *fix* the entity tags to be observed. See Fig. 7.4 for an illustration of the joint model as defined over 3 mentions. Note that even with such a small set of mentions, the underlying joint model is quite complex and dense. Formally, the probability for a setting to all the document variables is:

$$p(\mathbf{t},\mathbf{r},\mathbf{c}|\mathbf{m}) \propto \prod_{t_i\in\mathbf{t}} \psi_T(m_i,t_i) \prod_{c_{ij}\in\mathbf{c}} \psi_C(c_{ij},m_i,m_j,t_i,t_j) \prod_{r_{ij}\in\mathbf{r}} \psi_R^L(m_i,m_j,r_{ij})\psi_R^J(r_{ij},m_i,m_j,t_i,t_j)$$

The factors here denote different distributions than for the individual classification models. Instead of representing a distribution over the labels of a single task *conditioned on* the predictions from

Data	#Mentions	#Coreference	#Relation
Train	15,640	637,160	82,479
Dev	5,545	244,461	34,057
Test	6,598	342,942	38,270

Table 7.1: **ACE Dataset:** Number of variables in the various folds.

another task, these factors now directly represent the *joint* (unnormalized) distribution over the tasks that they are defined over. The coreference resolution and relation extraction are not directly connected in the model, as the dependency between these two tasks is much weaker in practice. Nonetheless they are not independent. As part of the same graphical model, information can flow between the two via entity tags, resulting in indirect improvements to relation extraction when coreference improves, and vice versa.

Inference: Due to the number of variables, non-trivial domain sizes, strong dependencies, and a loopy structure, belief propagation cannot be applied directly. However, joint inference in this model an ideal use case for single-value belief propagation. A large number of factors exist amongst binary coreference variables, suggesting that sparsifying the binary variables can provide substantial benefits in speed. Further, the factors have strong dependencies across tasks that naturally sparsify, for example given a single choice for the relation, the entity type of the argument may only be able to take a single value. We perform 4 rounds of iteration with a relative high $\zeta = 0.8$. Since transitivity is not a *learned* potential, but instead deterministically propagates decisions, we incorporate transitivity into the inference technique by directly propagating the positive coreference decisions over their transitive closure after every iteration of message passing.

Results: We use the Automatic Content Extraction (ACE) 2004 English dataset for the experiments, a standard labeled corpus for the three tasks [Doddington et al., 2004]. ACE consists of 443 documents from 4 distinct news domains, with $7,790$ sentences and $172,506$ tokens. Counts of each type of variable are shown in Table 7.1. For these experiments, we use gold mention boundaries, and the coarse-grained labels for tagging and relations (7 and 8 respectively).

Model	Tagging	Relations			Coreference	
	Accuracy	**Prec**	**Recall**	**F1**	**PW F1**	**B^3 F1**
Tagging → Coreference, Relations	80.23	53.22	54.92	54.05	53.94	76.34
Tagging + Coreference	81.24	-	-	-	57.59	78.06
Tagging + Relations	81.77	54.93	54.02	54.47	-	-
Tagging + Relations + Coreference	**82.69**	56.06	54.74	**55.39**	**58.39**	**78.50**

Table 7.2: **Joint Inference with Single-Value Sparsity:** Evaluation on Tagging using per mention accuracy, Relations using pairwise precision/recall, and Coreference using the pairwise decision and B^3 metrics. The baselines models consist of independent models for tagging, and models for relations and coreference condition on the output of tagging (denoted using $\rightarrow$). Our contribution consists of three joint models, denoted by $+$.

We train the isolated models using standard features, described in detail in Singh et al. [2013a]. Our model for entity tagging achieves an accuracy of 80.2%, which is impressive considering many of the mentions are pronouns and pronominals with little evidence in the context. Our relation extraction model achieves an F1 score of 54.05% which is comparable to existing research that uses only predicted entity tags (we obtain 61% F1 with gold entity tags). The coreference model achieves a macro B^3 F1 score of 76.34%, which is competitive with related approaches.

We first present joint inference between pairs of tasks. In particular, we separately evaluate the result of joint inference between entity tagging and each of the other two tasks. The results, when compared to the isolated models, are shown in Table 7.2. Allowing uncertainty in entity tags improves the accuracies of both the tasks, demonstrating the importance of propagating uncertainty along the pipeline. Further, there are significant error reductions for entity tagging, corroborating the need for flow of information from relations and coreference to the tagging model. When performing inference together on all the three tasks, we achieved further improvements for all of them, most significantly an error reduction of 12.4% for the entity tagging task.

7.2 Anytime Belief Propagation

The primary reason BP is unsuitable for large domains and higher-order factors is the cost of message computations and representation, which is on the order of the cross-product of the neighbors' domains. Existing extensions to BP that address this concern, including single-value sparsity (Section 7.1), use determinism [Shen et al., 2007], state-space pruning [Coughlan and Ferreira, 2002], sampling [Ihler and McAllester, 2009], smoothing [Sudderth et al., 2003], stochastic updates [Noorshams and Wainwright, 2011] and dynamic domains [Coughlan and Shen, 2007]. These techniques use parameters that define the desired level of approximation, and return the approximate marginals at convergence. This results in poor *anytime* properties such as: 1) providing accurate and consistent marginals when stopped early, 2) improving the approximation when run longer, and 3) converging to the fixed point of BP. Since these algorithms try to directly achieve the desired approximation, the marginals *during* inference cannot be characterized, and are often inconsistent with each other. Further, the relationship of the parameter that controls the approximation to the quality of the intermediate marginals is often unclear. As a result, these approaches are not suitable for applications that require consistent, anytime marginals but are willing to trade-off error for speed, for example applications in vision and natural language processing that involve real-time tracking or user interactions. There is a need for an anytime algorithm that can be interrupted to obtain consistent marginals corresponding to fixed points of a well-defined objective, and can improve the quality of the marginals over the execution period, eventually obtaining the BP marginals.

To this end, we propose a class of message passing algorithms that works on sparse (partially instantiated) domains, and compute accurate, anytime-consistent marginals. Initialized with a sparse domain for each variable, the approach alternates between two phases: (1) augmenting values to sparse variable domains, and (2) converging to a fixed point of the approximate marginal inference objective as defined by these sparse domains. We tighten our approximate marginal inference objective by selecting the value to add to the sparse domains by estimating the impact of adding the value to the variational objective; this is an accurate prioritization scheme that depends on the instantiated domains and requires runtime computation. We also provide an alternate prioritization scheme based on the gradient of the primal objective that can be computed a priori, and provides constant time selection of the value to add. To converge to a fixed point of the approximate marginal objective, we perform message passing on the sparse domains. Since naive schedules that update messages in a round robin or random fashion are wasteful, we use residual-based dynamic prioritization [Elidan et al., 2006]. Inference can be interrupted to obtain consistent marginals at a fixed point defined over the instantiated domains, and longer execution results in more accurate marginals, eventually optimizing the BP objective.

7.2.1 Overview of the Approach

In practice, graphical models are often defined over variables with large domains and factors that neighbor many variables. Message passing based algorithms perform poorly for such models since the complexity of message computation for a factor is $\mathrm{O}\left(|\mathcal{Y}|^{|\mathcal{N}_f|}\right)$, where $\mathcal{Y}$ is the domain of each variable. Further, if inference is interrupted, the resulting marginals are not locally consistent, nor do they correspond to a fixed point of a well-defined objective. In this section, we describe an algorithm that meets the following desiderata: (1) anytime property that results in consistent marginals, (2) more iterations improve the accuracy of marginals, and (3) convergence to BP marginals (as obtained at a fixed point of BP). To obtain accurate marginals quickly, we utilize the current state of inference to instantiate high-likelihood values before picking low probability ones.

Instead of directly performing inference on the complete model, our approach maintains *partial* domains for each variable. Message passing on these sparse domains converges to a fixed point of a well-defined objective (Section 7.2.2). This is followed by incrementally *growing* the domains (Section 7.2.3), and resuming message passing on the new set of domains till convergence. At any point, the marginals are close to a fixed point of the sparse BP objective, and we tighten this objective over time by growing the domains. If the algorithm is not interrupted, entire domains are instantiated, and the marginals converge to a fixed point of the complete BP objective.

7.2.2 Belief Propagation with Sparse Domains

First we study the propagation of messages when the domains of each variables have been partially instantiated (and are assumed to be fixed here). Reconsider the BP dual objective for complete domains, defined earlier in Eq. (6.11) on Page 106:

$$L_{\text{BP}}\left(\boldsymbol{\mu},\boldsymbol{\lambda}\right)\triangleq\sum_{f\in\mathcal{F}}\sum_{\mathbf{y}_f}\mu_f(\mathbf{y}_f)\psi_f(\mathbf{y}_f)+H_B\left(\boldsymbol{\mu}\right)$$
$$+\sum_{f}\lambda_f C_f\left(\boldsymbol{\mu}\right)+\sum_{f}\sum_{i\in N(f)}\sum_{y_i}\lambda_{fi}^{y_i}C_{f,i,y_i}\left(\boldsymbol{\mu}\right)$$

where recall that $\boldsymbol{\mu}$ refers to the variable and factor marginals, $\boldsymbol{\lambda}$ is (log) proportional to the messages, H_B is the Bethe Entropy, and C_f and C_{f,i,y_i} refer to the local consistency constraints on the marginals.

Let $\mathcal{S}_i\subseteq\mathcal{Y}, |\mathcal{S}_i|\geq 1$ be the set of values associated with the instantiated domain for variable Y_i. During message passing, we *fix* the marginals corresponding to the non-instantiated domain to be zero, i.e. $\forall y_i\in\mathcal{Y}-\mathcal{S}_i, \mu_i(y_i)=0$. By setting these values in the BP dual objective, we obtain the optimization defined only over the sparse domains:

$$L_{\text{SBP}}\left(\boldsymbol{\mu},\boldsymbol{\lambda},\mathcal{S}\right)\triangleq\sum_{f}\sum_{\mathbf{y}_f\in\mathcal{S}_f}\mu_f(\mathbf{y}_f)\psi_f(\mathbf{y}_f)+H_B\left(\boldsymbol{\mu}\right)$$
$$+\sum_{f}\lambda_f C_f\left(\boldsymbol{\mu}\right)+\sum_{f}\sum_{i\in N(f)}\sum_{y_i\in\mathcal{S}_i}\lambda_{fi}^{y_i}C_{f,i,y_i}\left(\boldsymbol{\mu}\right)\tag{7.1}$$

Note that $L_{\text{SBP}}(\boldsymbol{\mu},\boldsymbol{\lambda},\mathcal{Y}^n)=L_{\text{BP}}(\boldsymbol{\mu},\boldsymbol{\lambda})$. Message computations for this approximate objective, including the summations in the updates, are defined sparsely over the instantiated domains. In general, for a factor f, the computation of its outgoing messages requires $\mathrm{O}\left(\prod_{Y_i\in\mathcal{N}_f}|\mathcal{S}_i|\right)$ operations, as opposed to $\mathrm{O}\left(|D|^{|\mathcal{N}_f|}\right)$ for whole domains. Variables for which $|\mathcal{S}_i|=1$ are treated as *observed*, similar to Section 7.1.

7.2.3 Growing the Domains

As expected, BP on sparse domains is much faster than on whole domains, however it is optimizing a different, approximate objective (7.1). The approximation can be tightened by growing the instantiated domains, that is, as the sparsity constraints of $\mu_i(y_i)=0$ are removed, we obtain more accurate marginals when message passing for newly instantiated domain converges. However, identifying *which* values to add is crucial for good anytime performance, and we propose two approaches here based on the gradient of the variational and the primal objectives.

7.2.3.1 Dynamic Prioritization of Values

When inference with partial domains converges, we obtain marginals that are locally consistent, and define a saddle point of the approximation, i.e. Eq. (7.1). We would like to add the value y_i to $\mathcal{S}_i$ for which removing the constraint $\mu_i(v_i) = 0$ will have the most impact on the approximate objective L_{SBP}. In other words, we select y_i for which the gradient $\left.\frac{\partial L_{\text{SBP}}}{\partial \mu_i(y_i)}\right|_{\mu_i(y_i)=0}$ is largest. From Eq. (7.1) we derive

$$\frac{\partial L_{SBP}}{\partial \mu_i(y_i)} = (d_i - 1)(1 + \log \mu_i(y_i)) + \sum_{f \in \mathcal{N}(Y_i)} \lambda_{fi}^{y_i}. \tag{7.2}$$

Although $\log \mu_i(y_i) \to -\infty$ when $\mu_i(y_i) \to 0$, we ignore the term as it appears for all i and y_i[2]. The rest of the gradient is the priority:

$$\pi_i(y_i) = d_i + \sum_{f \in \mathcal{N}(Y_i)} \lambda_{fi}^{y_i} \tag{7.3}$$

Since $\lambda_{fi}^{y_i}$ is undefined for $y_i \notin \mathcal{S}_i$, we estimate it by performing a single round of message update over the sparse domains:

$$\lambda_{fi}^{y_i} = \log \sum_{\mathbf{y}_f \in \mathcal{S}_f / Y_i} \exp \left\{ \psi_f(\mathbf{y}_f) + \sum_{Y_j \in \mathcal{N}_f / Y_i} \lambda_{jf}^{\mathbf{y}_f(j)} \right\} \tag{7.4}$$

To compute priority of all values for a variable Y_i, this computation requires $\mathrm{O}\Big(\sum_f |\mathcal{Y} - \mathcal{S}_i| \prod_{\mathcal{N}_f / x_i} |\mathcal{S}_i|\Big)$ operations, i.e. an efficient $\mathrm{O}\Big(|\mathcal{Y}||\mathcal{S}|^{\mathcal{N}_f - 1}\Big)$. Since we only need to identify the value with the high-

[2] For an alternative justification, consider the approximation to L_{SBP} that replaces the variable entropy $-\sum_p p \log p$ with its second order Taylor approximation $\sum_p p(1-p)$. The gradient at $\mu_i(y_i) = 0$ of the approximation is $d_i + \sum_{f \in \mathcal{N}(Y_i)} \lambda_{fi}^{y_i}$.

est priority, we can improve this search by sorting factor scores ψ, and further, we only update the priorities for the variables that participated in message passing.

7.2.3.2 Precomputed Priorities of Values

Although the dynamic strategy selects the value that improves the approximation the most, it also spends time on computations that may not result in a corresponding benefit. As an alternative, we propose a prioritization that precomputes the order of the values to add; even though this does not take the current beliefs into account, the resulting savings in speed may compensate. Intuitively, we want to add values to the domain that have the highest marginals in the final solution. Although the final marginals cannot be computed directly, we estimate them by enforcing a single constraint $\mu_i(y_i) = \sum_{\mathbf{y}_f/Y_i} \mu_f(\mathbf{y}_f)$ and performing greedy coordinate ascent for each f on the primal objective in (6.10). We set the gradient w.r.t. $\mu_f(\mathbf{y}_f)$ to zero to obtain:

$$\pi_i(y_i) = \hat{\mu}_i(y_i) = \sum_{\mathbf{y}'_f, y'_i = y_s} \hat{\mu}_f\left(\mathbf{y}'_f\right) = \sum_{f \in \mathcal{N}(Y_i)} \log \sum_{\mathbf{y}_f \in \mathcal{Y}_f/Y_i} \exp \psi_f(\mathbf{y}_f) \tag{7.5}$$

Since π_i is independent of the current messages or marginals, this priority can be precomputed and identifies the next value to add in constant time. Nonetheless, this is an approximation, and its relative benefit over dynamic prioritization will be empirically evaluated.

7.2.4 Dynamic Message Scheduling

After the selected value has been added to its respective domain, we perform message passing as described in Section 7.2.2 to converge to a fixed point of the new objective. A naive scheduling of messages during this phase (random or round-robin) results in wasteful computations since the marginals are already consistent and favorable in most areas of the graph. Identifying the regions of the graph that are inconsistent and avoiding wasteful updates is crucial for fast convergence of message passing. Fortunately there has been existing work to identify schedules for messages that speed up BP convergence [Elidan et al., 2006, Sutton and McCallum, 2007b] by dynamically prioritizing message updates in areas with least consistency. We use these dynamic schedules to identify messages to propagate once the domains have grown. As the choice of the message norm, we use the dynamic range of the change in the messages as the *residual*, as explored by Ihler et al. [2005b] and Sutton and McCallum [2007b]. Instead of computing individual message priority, we aggregate over the messages of a factor; this significantly reduces the number of entries to prioritize over. Formally, a factor priority is:

$$\pi(f) = \max_{Y_i \in \mathcal{N}_f} \max_{y_i, y_j \in \mathcal{S}_i} \log \frac{e(y_i)}{e(y_j)}, \quad e(y_i) = \frac{\boldsymbol{m}_{fi}(y_i)}{\boldsymbol{m}'_{fi}(y_i)} \tag{7.6}$$

As shown by Elidan et al. [2006], residuals of this form bound the reduction in distance between the factor's messages and its fixed point. This allows us to use the priorities in two ways: first, we pick the message with the highest priority, since it indicates the part of the graph that is least locally consistent. Second, the maximum priority over all the factors is an indication of convergence and consistency; a low max residual implies a low bound on the distance to convergence.

7.2.5 Algorithm

The proposed approach is outlined in Algorithm 7.2. We initialize the sparse domains using a single value for each variable with the highest priority. The domain priority queue contains the priorities for the rest of the values of the variables, which remain fixed or are updated depending on the prioritization scheme of choice (Section 7.2.3). Once message passing has converged to obtain locally-consistent marginals (according to some small ϵ), we select another value to add to the domains using one of the value priority schemes, and continue till all the domains are fully-instantiated. If the algorithm is interrupted at any point, we return either the current marginals, or the last converged, locally-consistent marginals. We use a heap-based priority queue for both messages and domain values, in which update and deletion take $O(\log n)$, where n is often smaller than the number of factors and total possible values.

Algorithm 7.2 Anytime Belief Propagation: Q_d represents the domain priority queue and Q_m the message priority queue. The outer loop in here grows domains by selecting the value from the top of the queue Q_d in Algorithm 7.3. With the updated domains, we perform priority-based BP, propagating local changes to Q_m and Q_d in Algorithm 7.4.

```
$\forall x_i, \mathcal{S}_i \leftarrow \{v_i\}$                          ▷ where $v_i = \arg\max_{v_s} \pi_i(v_s)$
$Q_d \overset{+}{\leftarrow} \langle (i, v_i), \pi_i(v_i) \rangle$        ▷ $\forall x_i, v_i \in \mathcal{Y}_i - \mathcal{S}_i$
$Q_m = \{\}$
while $|Q_d| > 0$ do
    GrowDomain($\mathcal{S}$, $Q_d$)                  ▷ Add a value to a domain, Algorithm 7.3
    ConvergeUsingBP($\mathcal{S}$, $Q_m$)             ▷ Converge to a fixed point, Algorithm 7.4
end while                                             ▷ Converged on full domains
```

Algorithm 7.3 Grow Domain for Anytime BP: Growing Domains by a single value (Section 7.2.3)

```
procedure GrowDomain($\mathcal{S}, Q_d$)
    $(i, v_p) \leftarrow Q_d.pop$                          ▷ Select value to add
    $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{v_p\}$     ▷ Add value to domain
    for $f \leftarrow \mathcal{N}(x_i)$ do
        $Q_m \overset{+}{\leftarrow} \langle f, \pi(f) \rangle$    ▷ Update message priority
    end for
end procedure
```

Algorithm 7.4 Sparse Message Passing: BP on Sparse Domains (Sections 7.2.2 and 7.2.4)

```
procedure ConvergeUsingBP($\mathcal{S}, Q_m$)
    while $max(Q_m) > \epsilon$ do
        $f \leftarrow Q_m.pop$                          ▷ Factor with max priority
        MessagePassing($f$, $\mathcal{G}$)             ▷ Compute messages and update neighbor marginals
        for $x_j \leftarrow \mathcal{N}_f; f' \leftarrow \mathcal{N}(x_j)$ do
            $Q_m \overset{+}{\leftarrow} \langle f', \pi(f') \rangle$                ▷ Update message priorities
            $Q_d \overset{+}{\leftarrow} \langle (k, v_q), \pi_k(v_k) \rangle$      ▷ $\forall x_k \in \mathcal{N}_{f'}, \forall v_k$
        end for
    end while                                          ▷ Converged on sparse domains
end procedure
```

7.2.6 Anytime Convergent Belief Propagation

To address the convergence issues of belief propagation, there has been work on convergent message passing algorithms using *damping* and convex approximations [Yuille, 2002, Wainwright et al., 2003, Hazan and Shashua, 2008], however these approaches often come with a cost to speed and/or accuracy, and introduce several extra parameters that need tuning. Although our work focuses on faster convergence to a BP fixed point due to its importance in practice, we can incorporate these convergent versions into our framework. Message passing on partial domains (Section 7.2.2), for example, can optimize the convex approximations of BP instead; this corresponds to using an algorithm such as tree re-weighted BP in the inner loop of of our approach (Algorithm 7.4). Additionally, we can also incorporate other heuristics used in practice to compensate for non-convergence, such as damping or running for some maximum number of iterations, and then grow the domains regardless; although we may lose low consistency marginals, we retain the anytime property.

7.2.7 Experiments

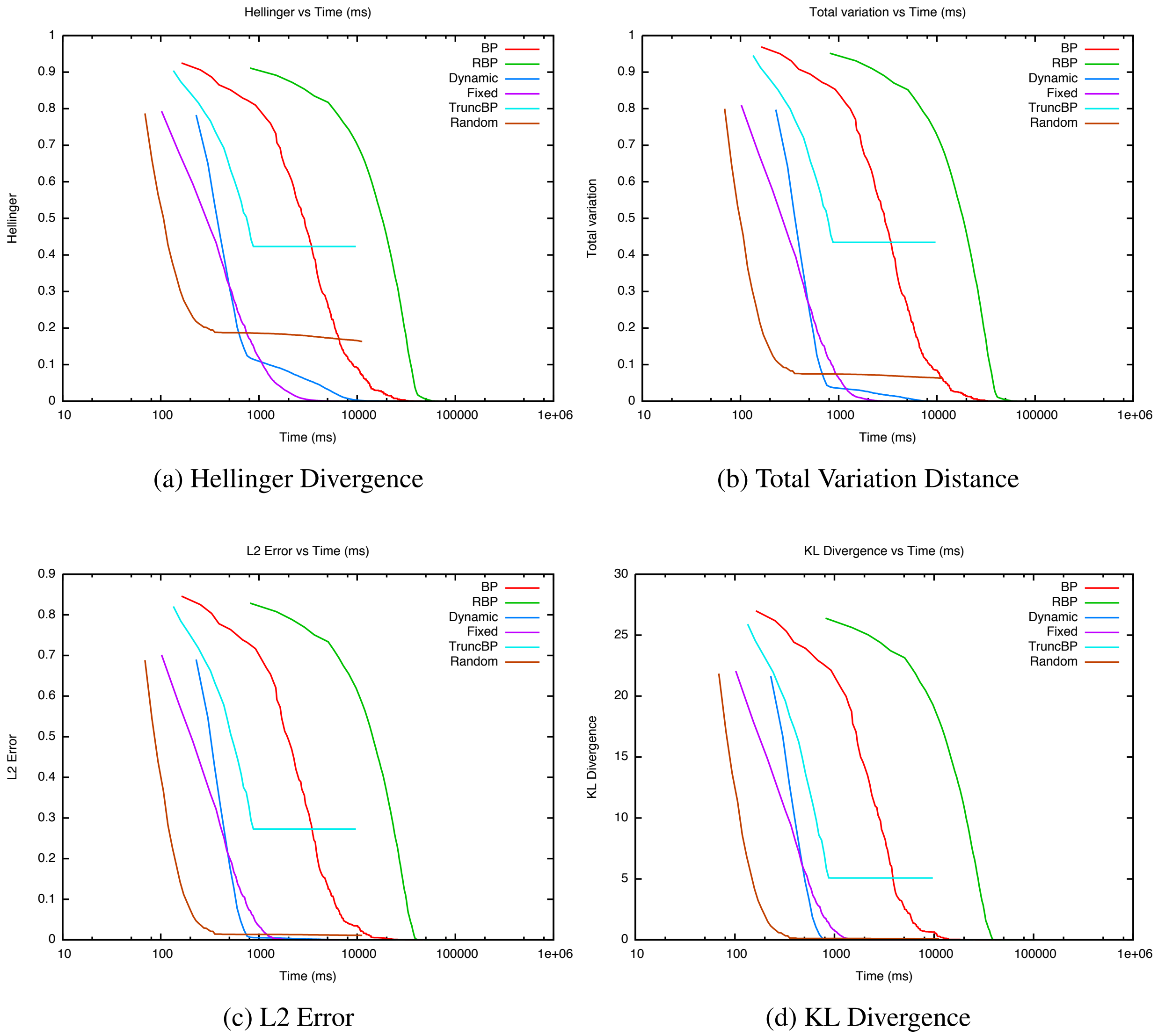
We evaluate our approaches by examining the convergence rate and accuracy of the marginals for the following methods. Our primary baseline is *Belief Propagation* (**BP**) using random, asynchronous scheduling. We also evaluate *Residual BP* (**RBP**) that uses the dynamic range for message scheduling, as described in Section 7.2.4. Our first baseline that uses sparsity, *Truncated Domain* (**TruncBP**), is initialized with domains that contain a fixed fraction of values (0.25), sorted according to precomputed priorities (Section 7.2.3). We do not modify the domains after initialization, and perform message passing as in Section 7.2.2. We evaluate on three variations of our framework. *Random Instantiation* (**Random**) is the simplest version in which the value to be added to the domains is selected at random, followed by priority based message passing. Our approach to prioritization that uses the incoming messages to estimate the gradient of the dual objective is called **Dynamic**, while the approach that uses precomputed priorities is named **Fixed**. We set ϵ, the maximum residual for convergence, to 10^{-10}.

7.2.7.1 Grids

Our first testbed for evaluation consists of synthetically generated grid models that are often used as inference benchmarks. We generate models for an $n \times n$ grid of variables (each with a domain size of L), consisting of unary and pairwise factors. We design the factors to induce a consistent, decaying distribution on the neighbor domains by using geometric distributions with random Gaussian noise, i.e. $\gamma(k) = k \log(1-p) + \log(p) + \mathcal{N}(0, 0.1)$, where $p \in [0.0, 1.0]$ is uniformly sampled for each factor. In particular, for unary factors $\psi_i(v_i) = \gamma(v_i)$ while for pairwise factors $\psi_{ij}(v_i, v_j) = \gamma(v_i + v_j L)$. Experiments are performed on $n = 5, 10$, and averaged over 10 runs, with the per-variable domain size $L = 10, 20, 50, 100, 250$.

Convergence Rate: To examine the runtime error, we compare against the marginals obtained by BP at convergence in Fig. 7.7. It is possible that these marginals are not the *true* marginals since BP is an approximate algorithm. However our objective in this work is to reach a BP fixed point faster, thus comparing the runtime marginals to ones obtained at convergence is valid. We compare a number of error metrics in Fig. 7.7. The runtime performance for our approaches is significantly better than BP; up to $\mathbf{12}$ times faster to obtain L_2 error of 10^{-7}. TruncBP is efficient, however converges to an inaccurate solution, suggesting that prefixed sparsity in domains is not desirable. Similarly, Random is initially fast, since adding *any* value has a significant impact, however as the selections become crucial, the rate of convergence slows down considerably. Although both Fixed and Dynamic approach provide desirable trajectories, Fixed is much faster initially due to constant time growth of domains. However as messages and marginals become accurate, the dynamic prioritization that utilizes them becomes faster, and eventually overtakes the Fixed approach. These experiments also demonstrate the inefficiency of RBP for models with large domains.

Runtime Behavior: To examine the anytime local consistency, we examine the average residuals on the current messages (see Fig. 7.8a). As described earlier, the *max*-residuals are an upper-bound on the reduction in distance to the locally-consistent BP fixed point, and low residuals imply a consistent set of marginals for the objective defined over the instantiated domain. Since we converge to a consistent solution on partially instantiated domains before growing domains,



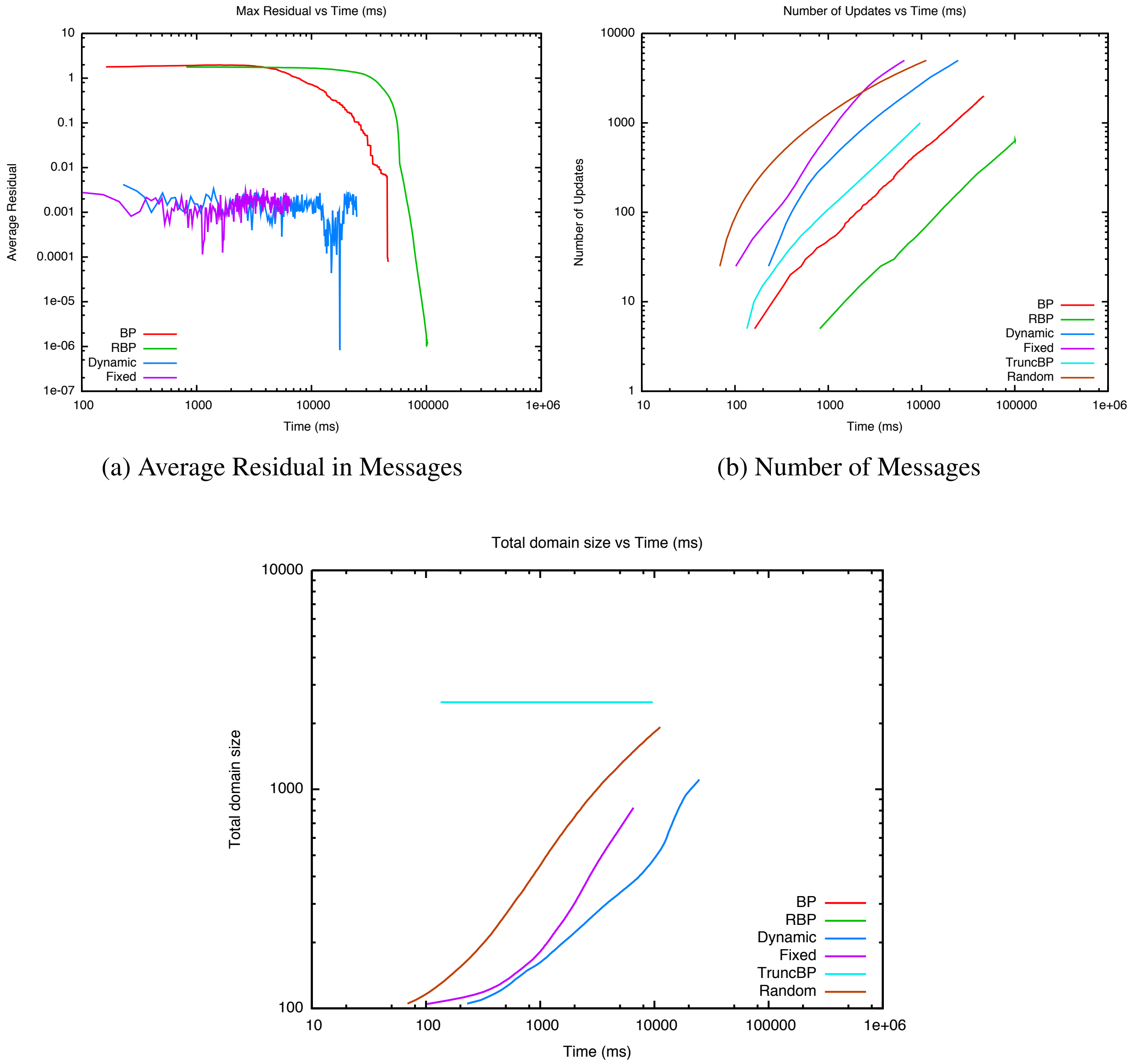
(a) Hellinger Divergence

(b) Total Variation Distance

(c) L2 Error

(d) KL Divergence

Figure 7.7: **Error on Marginals:** for 10×10 grid with domain size of 100, averaged over 10 runs.



(a) Average Residual in Messages

(b) Number of Messages

(c) Instantiated Domain Size

Figure 7.8: **Runtime Analysis:** for Fig. 7.7: (a) average message residual (signifying local consistency), (b) number of messages, and (c) total size of the instantiated domains (maximum 10^4), all axes are in log-scale.

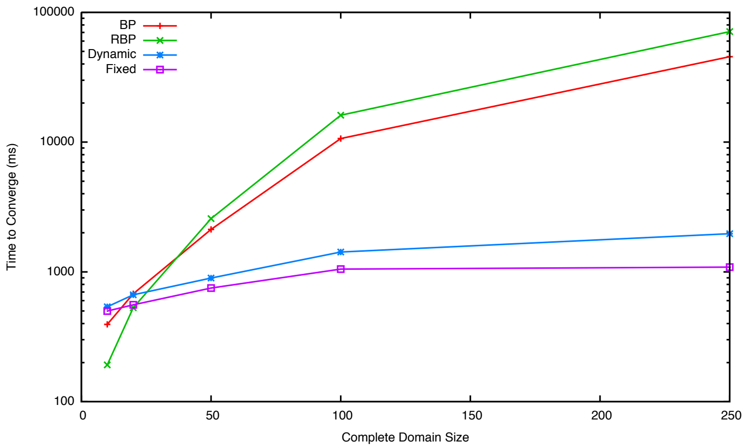


Figure 7.9: **Convergence for different domain sizes:** Time to achieve a low L_2 error (10^{-4}) of the marginals computed over 10 runs of 5×5 grids.

our approaches demonstrate low residuals throughout. In comparison, the residuals for existing techniques remain significantly higher for most of the runtime (note the log-scale for Y-axis), lowering only near convergence. We also present the number of messages computed as a function of time in Fig. 7.8b; even though our approaches send many more messages, they are faster due to the sparse domains (Fig. 7.8c).

Varying the Domain Size: We also examine the performance as the total domain size is varied. We run 10 runs of 5×5 synthetic grids and observe the time to converge for different domain sizes to a low L_2 error in Fig. 7.9. Although our proposed approaches are slower on problems with small domains, they obtain significantly higher speedups on larger domains ($25-40$ times on 250 labels). Fixed fares somewhat better than the dynamic approach, as the domain size has an adverse impact on the priority computations of the latter. Note that although per iteration complexity for BP and RBP is quadratic in the domain size, the time to converge depends on the model potentials. We also demonstrate that although message scheduling is helpful when $L = 10$, BP outperforms RBP for larger domain sizes since the cost of computing message priorities supersedes their benefit.

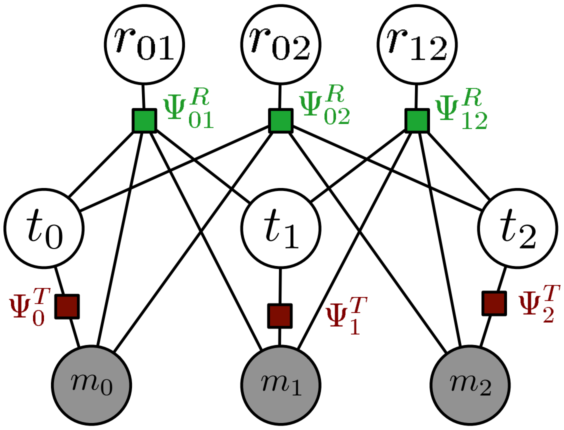


Figure 7.10: **Joint Information Extraction:** (a) Sentence consisting of 3 entities, with the annotated *types* and *relations*. (b) Instantiated model with observed entities (m_i) and their type (t_i) and relation (r_{ij}) variables.

# Entities	4	6	8
# Vars	16	36	64
# Factors	28	66	120
BP	41,193	91,396	198,374
RBP	54,577	117,850	241,870
Fixed	24,099	26,981	49,227
Dynamic	24,931	36,432	41,371

Table 7.3: **Speed of Anytime Joint Inference** : Avg time taken (in milliseconds) to obtain a low error in marginals ($L_2 < 0.001$)

7.2.7.2 Joint Inference for Types and Relations

We also evaluate on the real-world task of joint information extraction. Given a sentence and its entities, a crucial step for information extraction is to determine the entity types, and to identify the relations between the entities (see Fig. 7.4 for an example, we do not model coreference here). The domain for entity types consists of fine-grained labels such as PERSON, COUNTY, WATERBODY, PROJECTILEWEAPON, etc., while the relations are labeled with BASEDIN, STAFFEMPLOYEE, SPOUSEOF, and so on. We represent the distribution over variables of these dependent tasks jointly as a single model (shown in Fig. 7.10), containing joint factors Ψ^R_{ij} that neighbor variables of both the tasks. The domain for entity types contains 42 labels and the domain for relations contains 24 labels, resulting in $42,336$ neighbor assignments for joint factors Ψ^R_{ij}. We run inference on

ACE 2004, and average over 5 runs. Table 7.3 shows the time taken to obtain a low error in marginals, as compared to the marginals at convergence. For smaller sentences, sparsity does not help much since BP converges in a few iterations. However, for longer sentences containing many more entities, we observe a significant speedup (up to **6** times). Further, although **Fixed** is sufficient for smaller models, **Dynamic** is faster for the large models; taking messages into account is crucial as the structure becomes complex. These improvements are less dramatic than on grids, we suspect this is due to the medium sized domains, and the extremely dense structure of the model (each entity variable is connected to n relation variables). Such structures violate the locality assumptions made in our prioritization; on the other hand, the regular structure prevalent in common graphical models is conducive to our approach.

7.3 Related Work

Considerable effort has been invested in designing message passing algorithms that converge quickly for models with large domain variables. Our proposed approaches are extensions to, and combination of, a number of related techniques.

Approximate Inference for Chains: Some existing work addresses approximate inference in chains. Tsuruoka and Tsujii [2005] present a heuristic for performing inference in "easiest-first" order, avoiding full bidirectional inference. Shen et al. [2007] have proposed learning the order of inference instead of employing runtime heuristics. These differ from our proposed approach as they do not revisit earlier decisions, address parallelism or computational resources.

Sparse Message Passing: The bottleneck introduced by large domains has motivated several approaches that use *sparse* message approximations, some of which we outline here. The single-value sparsity is partly inspired by cutset conditioning introduced in [Pearl, 1988] which identifies variables in a model that, if fixed to a single value, results in tree-shaped model. Although we focus on approximate inference, work on cutset conditioning can be combined with our approach to prioritize the variables to be fixed. Song et al. [2011] propose a kernel-based representation of the messages to provide efficient update computations independent of the domain size. Early work by Coughlan and Ferreira [2002] prunes values when their current belief falls under a threshold. Ihler et al. [2004, 2005b] provide a theoretical framework to analyze the bounds on errors for BP with message approximations. Also relevant to our approach is work by Sudderth et al. [2003] and the following generalization by Ihler and McAllester [2009] that maintains samples on messages, factors and variables, iteratively improving their approximation to minimize the error. Coughlan and Shen [2007] use threshold based pruning of the label spaces to perform efficient inference. Also similar, Komodakis and Tziritas [2007] combine label pruning with entropy-based message scheduling. All these approaches operate on a fixed level of approximation; in contrast our work *grows* domain when needed, is guaranteed to consider all values, and eventually give full BP marginals. As an exception to this, Noorshams and Wainwright [2011] propose stochastic updates on the message computations, thus reducing the polynomial dependence on the domain size, leading to faster convergence. In contrast to their approach, we constantly work in a near-primal-feasible region (with respect to local consistency), providing the anytime flavor. We also differ by directly using the gradient of the primal or the dual objective to select high likelihood values to add to the sparse domains.

Similar to our approach, a number of researchers propose modifications to BP that perform inference without visiting all the factors. Recent work introduces dynamic schedules to prioritize amongst the factors [Coughlan and Shen, 2007, Sutton and McCallum, 2007b] that has been used to only visit a small fraction of the factors [Riedel and Smith, 2010]. Gonzalez et al. [2009a] utilize these schedules to facilitate parallelization. Structured prediction cascades [Weiss and Taskar, 2010] also dynamically pick the set of values to use during inference, however differ from our approach since the utility of the approach relies on learning parameters for this purpose, while our proposed approach does not make such assumptions about the parameters of the model.

Message Passing Schedules: Scheduling between messages was introduced by Elidan et al. [2006], followed by an efficient approximation [Sutton and McCallum, 2007b]. These schedulers operate on full domains, and for problems with large domains, can often be slower than BP, as we show in our experiments (Fig. 7.9). Instead of including all the factors in the model, Riedel et al.

[2010a] explicitly sparsify the model by selecting only the factors with a high priority. Similarly, Choi and Darwiche [2009] perform inference with similar approximations, then improve the solution by compensating for them at a later stage. Our approach is one of the first to investigate the interactions between residual-based message scheduling and value prioritization, and make it practical on models with large domains.

MAP Inference: Although MAP inference is different, many of the ideas that we use are shared by the MAP inference literature. A number of approaches relax the problem, and iteratively tighten it by reintroducing constraints till the optimum is reached. These differ from each other based on how the problem is relaxed, and how the constraints are discovered, for example, using equality constraints [Choi and Darwiche, 2009, Sontag et al., 2011], integrality constraints [Globerson and Jaakkola, 2007b] and cutting planes [Sontag and Jaakkola, 2008]. Our approach is similar, but introduces sparsity constraints for efficiency, and removes them iteratively to reduce the approximation. Amongst the work in value sparsity, Sontag et al. [2009] introduce higher-order constraints that represent the marginal polytope, and partition cluster domains for efficiency. Although our work also selects subsets of the values to represent, and may be applied directly to a cluster graph, the objective is to improve the anytime properties of convergence to a local polytope fixed point. Recent work by Belanger et al. [2012] and Wang and Koller [2013] uses value sparsity for efficient MAP inference, in contrast, our approach targets marginal inference. Our work differs from MAP in another crucial way; obtaining a global optimum for MAP is possible without exploring all values, while sparsifying the domains in marginal inference necessarily results in an unavoidable approximation error.

7.4 Conclusion and Future Work

In this work we are primarily motivated with problems that have high-order factors and large domain variables, but may not have many variables or factors. We introduced two set of algorithms have explore the use of sparsity in variable domains for more efficient inference. In the first part of this chapter, we introduce single-value sparsity, a straightforward extension to belief propagation that encourages variables to have deterministic marginals. The approach allows approximate inference to parallelize for chains, and facilitates tractable inference on a joint within-document information extraction task. In the second part of this chapter, we describe a novel family of *anytime* message passing algorithms designed for marginal inference on problems with large domains. The approaches maintain sparse domains, and efficiently compute updates that quickly reach the fixed point of an approximate objective by using dynamic message scheduling. Further, by growing domains based on the gradient of the objective, we improve the approximation iteratively, eventually obtaining the BP marginals.

In future work, we will use tools to analyze the effects of approximate messages [Ihler et al., 2005b] for a better theoretical understanding of our approaches. We also want to study the effect of sparsity on global properties, for example, does the sparsity induce an exploitable structure such as low tree-width? There are a number of further alternatives that we would like to explore, such as growing domains by more than a single value at a time, inducing sparsity at the level of factor assignments, and so on. Research is also needed to combine our approach with convergent marginal inference algorithms.

CHAPTER 8

BELIEF PROPAGATION WITH CLUSTER MESSAGE COMPRESSION

> *With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*
>
> John von Neumann

For scaling belief propagation to large, dense models frequent in joint inference, we may need to employ distributed processing, and utilize structure-specific inference that can provide efficient inference for sub-structures of the model. In the previous chapter we focus mainly on variables with large domains, and introduce approaches that utilize the low-entropy marginals to induce sparsity in the earlier stages of inference. Although our approaches provide efficient inference for a number of tasks, the complexity in the model arises from the domain size and not size of the model, and it is not clear how the approaches would generalize to larger models, or to the distributed setting. Further it is difficult to combine the sparsity based algorithms with customized algorithms that may be more efficient for specific parts of the model.

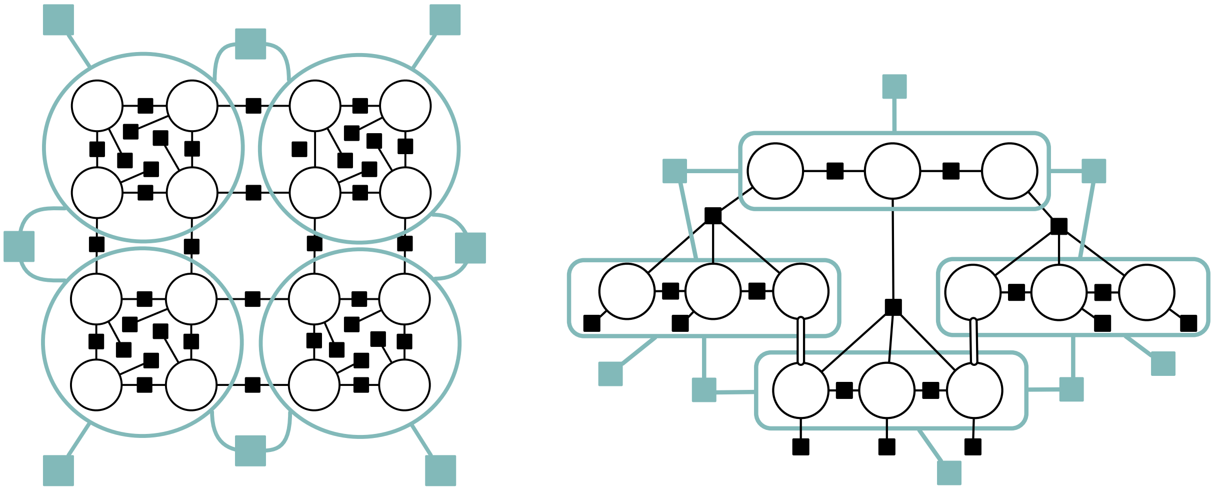
Let us consider a straightforward solution to distributing belief propagation. Since message passing is naively distributable, we can partition the model into components (each variable is assigned a single component), and compute messages within each component in parallel. Occasionally, variables lying on the boundary of the components will have to communicate their messages to their neighbors that lie on a separate node; we can use asynchronous, node-to-node communication to achieve this. Unfortunately, a number of weaknesses are inherent in this framework: (1) Since asynchronous, node-to-node communication is often slow in most distributed processing architectures, the framework should communicate minimally, and each message should be informative. The above framework does not differentiate between within-node and across-node messages, making it difficult to take the cost of communication into account. (2) Although loopy belief propagation works well on small models, it may be a poor choice for large graphs due to non-convergence or convergence to a poor solution (Section 6.2.2.2), requiring a large number of iterations before obtaining reasonable marginals. (3) If there are existing inference algorithms for sub-structures, for example sampling for large-domain variables or efficient MAP algorithms, this framework for distribution cannot utilize them. It is clear that this proposed framework for belief propagation lacks a number of desired properties.

In this chapter we introduce a framework for efficient inference that builds upon generalized belief propagation. The clusters of variables for generalized BP are defined to reflect the exploitable sub-structures of the model and the communication boundaries across the nodes (Section 8.1). Performing message passing with complete cluster messages would result in convergence to accurate marginals in with very few messages (iterations), however complete messages are not practical since both the computation and the size of each message is prohibitively expensive (exponential in the number of variables in the cluster). Instead in Section 8.2 we propose a number of approximations of cluster messages that trade-off size (communication cost) and information content (number of iterations), ranging from the special case of loopy belief propagation as small but information poor messages, to full generalized BP messages that are informative but impractical to compute/communicate. Since our approximations are based on sampling and search techniques, we can exploit sub-structures for which such algorithms are available.

We provide convergence analysis using expectation propagation in Section 8.3. Empirical evidence, presented in Section 8.4, demonstrates convergence to more accurate marginals compared to loopy belief propagation on a single machine, and with less communication for the distributed setting, on a number of synthetic and real-world joint inference tasks.

8.1 Cluster Graph for Modular Belief Propagation

In this section we describe our formulation of *modular* belief propagation as as instance of cluster graph based generalized belief propagation (we refer the reader to Section 6.3 for a review of cluster graphs and generalized BP). We then represent both distributed BP and joint inference



(a) Distributed BP

(b) Joint Inference

Figure 8.1: **Examples of Cluster Graphs:** (a) shows an example of a 4×4 grid distributed over 4 machines, where the factors between cluster variables represent inter-machine communications, and (b) shows a joint inference task between two chains. The top task contains 3 variables, while the bottom 7, and the model is similar to factorial CRFs with a window of 3. Variables shared between clusters are repeated with an equality factor (double-stroke) between them. Note that inference for each *local* factor is efficient (they are chains), however inference for factors across tasks is not.

between multiple tasks as instances of this framework. Let $\mathcal{G}$ be the given graphical model, consisting of variables $\mathbf{Y}$ and factors $\mathcal{F}$. In the cluster graph representation, we further assume $\mathcal{C}$ is a set of clusters where each $C \in \mathcal{C}$ consists of a subset of variables $C \subseteq \mathbf{Y}$. For distributed inference, we treat all the variables assigned to each node as part of a single cluster, as illustrated in Fig. 8.1a. Each local cluster factor represents the part of the model that is local to the computation node, while cluster factors across the cluster variables represents the dependencies across the machines. For joint inference, we can treat groups of variables for each task in a single cluster, as shown in Fig. 8.1b. The local cluster factors in this represent factors that define the distribution for each task, while the cluster factors across the clusters represent dependencies across the tasks. When performing generalized BP on the cluster graph, the cluster messages (messages to/from cluster variables and cluster factors) represent the joint distribution over the variables that are part of the cluster.

Message passing on the cluster graph using generalized BP converges to the Kikuchi approximation to the variational objective, a tighter approximation than loopy belief propagation (i.e. Bethe approximation), and in much fewer iterations of message passing; both desired properties for distributed BP and joint inference. Unfortunately the size fully-specified cluster messages is $|\mathcal{Y}|^{|C|}$. Such messages are prohibitively expensive both to compute and communicate, and we do not expect to use generalized BP for inference. On the other hand, it is possible to consider fully-factorized messages as approximations to the fully-specified message, i.e. $\boldsymbol{m}_{C\rightarrow\Psi} = \prod_{Y_i \in C} \boldsymbol{m}_{i\rightarrow\Psi}$. This is equivalent to performing loopy belief propagation on the underlying model; the messages are small in size ($|\mathcal{Y}| \times |C|$) and easy to compute and communicate. However, since these message fail to capture the dependencies accurately, they result in convergence to the Bethe fixed point (a poor objective), and take a large number of iterations to get there.

8.2 Approximations of Cluster Messages

The previous section suggests an inherent trade-off in cluster belief propagation. If the cluster messages are fully-specified and exact, they are informative and capture the dependencies accurately, resulting in accurate marginals and fewer iterations of message passing; however the messages are exponential in size. On the other hand, factorized cluster messages are small and efficient to compute and communicate, however lack the information necessary to converge fast to a good solution. In this section we define a number of representations of cluster messages $\boldsymbol{m}_{C\rightarrow\Psi}$ that together span a spectrum of representation/message size trade-offs from factorized to fully-specified.

8.2.1 Factorization over Individual Variables (Loopy BP)

Analogous to performing regular belief propagation, we can represent the joint distribution over all the variables in the cluster variable as a product of *local* distributions on each variable. As we mention above, this corresponds to $\boldsymbol{m}_{C\rightarrow\Psi} = \prod_{Y_i \in C} \boldsymbol{m}_{i\rightarrow\Psi}$. Although this is reasonably small in size ($|\mathcal{Y}| \times |C|$), it does not capture any global correlations present amongst the variables. This is very restrictive in the loopy models that we are interested in, and can cause belief propagation to take many iterations to converge. Furthermore, depending on the structure of the *consuming* factor f, this may not lead to efficient marginalization, i.e. the marginalization may iterate over the complete cross-product of the domains.

8.2.2 Sampling

To represent the global correlations that are ignored by the fully-factorized messages, the joint distribution can be approximated using a set of samples $\mathcal{S}$, where each sample $\mathbf{y}_s \in \mathcal{S}$ is a complete configuration of the inner variables $Y \in C$. The probability of each configuration can be set proportional to the counts, i.e. $\boldsymbol{m}_{C\rightarrow\Psi}(\mathbf{y}) \propto \sum_{s\in\mathcal{S}} \mathbb{I}_{(\mathbf{y}==\mathbf{y}_s)}$. However, since the log model score π is often easy to compute, we can also set $\boldsymbol{m}_{C\rightarrow\Psi}(\mathbf{y}) \propto \exp\pi(\mathbf{y})$; this way we are using sampling to identify a representative set of configurations, but not relying on sampling to be exact to compute the probabilities. Probability of zero is assigned to the configuration that do not appear in the set of samples. This facilitates efficient marginalization since marginalization only needs to iterate over the set of samples (instead of all possible configurations, as in the previous approximation). By using methods such as MCMC, the sampling task itself can be performed tractably (the size of the message is $\propto \mathcal{S}$, and the computation time is approximately linear in the number of samples). This approach also provides control over the degree of approximation and performance; a larger set of samples can represent the joint distribution better, however computation and marginalization is slower.

8.2.3 k-best Configurations

Samples can often be a very inefficient way to represent the joint distribution. If the the distribution contains multiple peaked modes, a large number of samples may be required to obtain samples from each of the nodes. On the other hand, if the distribution is relatively flat, a large number of samples may be required to discover the modes. Instead we would like methods that directly represent the modes of the distribution. To achieve this, top k complete assignments to the variables $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ can be selected (ranked according to the model probability), resulting in the k-best approximation. The probability of the selected configurations may be set uniformly,

$\boldsymbol{m}_{C\rightarrow\Psi}(\mathbf{y}) = 1/k$, however since we can compute the model score, we set $\boldsymbol{m}_{C\rightarrow\Psi}(\mathbf{y}) \propto \exp\pi(\mathbf{y})$. This approximation also assumes 0 probability mass outside the k-best configurations, and the size of the messages is $\propto k$. Similar to sampling, k allows a direct trade-off between approximation and performance, however by selecting the *top* configuration, it can adequately capture the modes of the distribution. Further, we can utilize existing literature on k-Best algorithms (including beam-search for approximate k-Best) to compute the messages efficiently.

8.2.4 Merged and FRAK Messages

The sparse messages described by k-best configurations (Section 8.2.3) and set of samples (Section 8.2.2) are efficient to perform marginalization over, capture global properties of the joint distribution, can utilize existing efficient algorithms to compute samples/modes, and allow a smooth trade-off between message size and representation capability. However, they assume that the probability outside the sparse set is 0. This restricts the consuming factor to represent the distribution only on a limited set of configurations, and thus it cannot induce the distribution to prefer configurations outside the sparse set of samples/configurations.

Instead, along with the set of samples/configurations, we introduce a "default" probability that is assumed to be spread uniformly over the rest of the space (*merged* set of configurations). To compute the probability mass assigned to the *merged* space, we compute $\boldsymbol{m}_{C\rightarrow\Psi}(\mathbf{y}) \propto \frac{1}{|C|-k}\left(Z - \sum_{\mathbf{y}_k}\exp\pi(\mathbf{y}_k)\right)$, which can be computed exactly or approximately depending on the structure of the local factors of C (for example, loopy BP is exact for chains and trees). If the total size of the domain is known, as is usually the case ($|\mathcal{Y}|^{|C|}$), the marginalization of these messages by the receiving factor f is still computationally tractable. Further since it assigns a probability mass to the *rest* of the domain, the downstream messages can express preference for rejecting all of the selected samples/configurations.

For computing the sparse set of configurations, we would like to utilize MAP inference instead of sampling or k-Best since MAP algorithms are often much faster and often exact. Since MAP provides the 1-best configuration, we maintain a list of previously selected configurations, and augment the list with the 1-best if it is not already in the list. This allows the system to adaptively select states during inference, growing the list to more configurations if the list of selected configurations is not a good representation of the message. We call this approximation the factorized, relaxed and adaptive k-best (FRAK) lists, retaining all the benefits of the sparse representation (small size, representation of global properties, fast marginalization) with ability to adapt in size during inference.

8.2.5 Other Approximations

Although we restrict our experiments to the approximations listed above, the framework allows alternative forms of approximation that may be more appropriate for certain classes of problems. For example, it is possible to project the joint distribution over the variables in a cluster to a simpler family of models (known as the approximating family), such as a chain or a tree [Welling et al., 2005]. Full factorized messages described above in Section 8.2.1 is a special case where the approximating family consists of completely independent factors. Although working with these messages is intractable in general (as mentioned above), depending on the structure within the consuming potential, certain approximations can still be tractable. For example, if the consuming cluster factor itself is a product of independent factors, then the fully-factorized approximation messages offer an appropriate approximation.

Since different variables have different forms of dependencies, it may be possible to combine the approximations above, for example use a fully-factorized distribution for a subset of the variables of C, while using a fully-specified or FRAK based distribution for the remaining variables. It is also possible to utilize context specific independence by using algebraic decision diagrams as the choice of representation [Gogate and Domingos, 2013].

8.3 Expectation Propagation

Expectation Propagation (EP; Minka [2001]) provides a theoretical framework for us to study the approximations. EP generalizes the belief propagation algorithm to allow each message to have a tractable form that approximates the original distribution. In particular, it requires the approximating family to be in the exponential family. The EP algorithm consists of the following three steps to compute a message: 1) multiplication of incoming messages with the factor score, 2) minimizing the KL divergence between the distribution computed in previous step and the approximation family, and 3) computing the outgoing message by dividing the incoming message. If the three operations can be computed efficiently and exactly for a given choice of approximation, EP defines the fixed point of the resulting algorithm. Unsurprisingly, choice of fully factorized marginals reduces the EP objective to the Bethe free energy. Welling et al. [2005] extend EP to generalized belief propagation.

Let us describe EP message computations as it applies to cluster BP with approximate messages. Consider a cluster variable C consisting of variables $\mathbf{Y}_C$, and a potential over it, Ψ_C. We want to approximate the message $\boldsymbol{m}_{C\to a}$ from the cluster variable C to a cluster factor Ψ_a, where the message from Ψ_a to C is $\boldsymbol{m}_{a\to C}$. Given a sufficient statistics function $\eta : \boldsymbol{\mathcal{Y}}_C \to \mathcal{R}^d$ that defines the approximating family, Expectation Propagation first computes the moments μ_η of η over the distribution $\boldsymbol{m}_{a\to C} \cdot \Psi_a$:

$$\mu_\eta \leftarrow \mathbb{E}_{m_{a\to C}\cdot\Psi_a}\left[\eta\right] \tag{8.1}$$

Then a *matching* distribution is computed, $\texttt{match}(\mu_\eta)$, that matches the moments μ_η of the full message. The projection step is followed by dividing out the incoming message $\boldsymbol{m}_{a\to C}$ to compute the desired message $\boldsymbol{m}_{C\to a}$.

$$\boldsymbol{m}_{C\to a} \leftarrow \frac{\texttt{match}(\mu_\eta)}{\boldsymbol{m}_{a\to C}} \tag{8.2}$$

Different choices of the sufficient statistics represent different approximations.

- **Exact messages:** Performing generalized belief propagation (i.e. with full messages) will correspond to η of size $k = |\boldsymbol{\mathcal{Y}}_C|$ such that $\eta(\mathbf{y}_C)$ is 0 everywhere except for 1 at the index of the value $\mathbf{y}_C$. This is equivalent to running full generalized belief propagation.

- **Factorized:** For the fully-factorized case (Section 8.2.1), which is equivalent to performing loopy belief propagation, η contains a single bit for each individual variable and value combination. This is much smaller than the full message; $|\mathcal{Y}| \times |C|$ in this case versus $|\mathcal{Y}|^{|C|}$ in the full message case, where $|C|$ is the number of variables in $\mathbf{Y}_C$ and $|\mathcal{Y}|$ is the size of the domain of individual variables in $\mathbf{Y}_C$.

- **Samples/k-Best:** Similarly, for the k-best or sampling configurations (Sections 8.2.2 and 8.2.3), the sufficient statistics $\eta = \{\eta_s\}_{s\in\mathcal{S}}$, where $\mathcal{S}$ is the set of k-best configurations ($|\mathcal{S}| = K$) or samples, and $\eta_s(\mathbf{y}_C) = \delta_{\mathbf{y}_s,\mathbf{y}_C}$. In other words, $\eta(\mathbf{y}_i)$ is a vector that is 0 everywhere except at the index corresponding to a configuration $\mathbf{y}_i = \mathbf{y}_s$ that is part of the k-best list or samples[1]. The sparse matching distribution does not match these expectations exactly, instead they assign zero probability outside of $\mathcal{S}$, and renormalize probabilities within. Note that the size of η is $|\mathcal{S}|$ here, which is much smaller than $|\mathcal{Y}|^{|C|}$.

- **Merged and FRAK Messages:** For the *merged* messages, the statistics are same as for k-Best, with an additional bit η_m that is 1 if $\mathbf{y}_C \notin \mathcal{S}$. The matching distribution implicitly divides the probability mass on η_m equally amongst the merged configurations. FRAK may incrementally extend the list $\mathcal{S}$ by adding the MAP state if absent, making it difficult to directly apply EP (EP requires a fixed approximation family). However, FRAK will either fully expand to all states (hopefully not), or arrive to a point where the following iterations do not add any more items to the lists. In either case the remaining iterations will perform conventional EP with fixed $\mathcal{S}$.

In this section we have presented our various approximations in context of the approximating families of expectation propagation, and described how the matching distributions are computed for each. This characterizes the fixed point of cluster belief propagation with compressed messages.

[1] If $\mathbf{y}_i$ is not part of the n-best list, then $\eta(\mathbf{y}_i)$ is 0 everywhere.

8.4 Experiments

We explore the utility of our approach on a number of different evaluations. First, we compare the representation capability and the storage size of our approximations for single cluster messages, outside the context of inference. We then present performance of our approximations on synthetic factorial chains with large domains and a wide window, for which running belief propagation is inefficient. To evaluate the utility of our approach for compressing messages for distributed BP, we simulate distributed processing on large grids, and compare accuracy of the marginals to the amount of bandwidth used. Finally, we present results on real-world joint inference task of citation segmentation and resolution.

8.4.1 Single Message

In order to gain a better understanding of the various approximations afforded by our proposed techniques, we compare the proposed approximate distributions to the true distribution. We generate a random 3×3 synthetic grid, with each variable domain of size 3 and local and pairwise factors generated randomly from the normal distribution. Specifically, the local factors $\psi_i(y_i)$ are sampled independently from $\mathcal{N}(0,1)$, while the pairwise factors $\psi_{ij}(y_i, y_j) = \begin{cases} \theta_{ij}, \text{if } y_i = y_j \\ -\theta_{ij}, \text{o.w.} \end{cases}$, where $\theta_{ij} \sim \mathcal{N}(0,1)$. The total number of values over which the message is defined is $3^9 = 19,683$. The model is shown in Fig. 8.2.

We compute the approximate cluster messages for the above factor using Factorized, Sampling, k-Best, and the FRAK methods, varying the different parameters such as the k and number of samples. We also compute the fully-specified message for this model that defines the true distribution, and compare the accuracy and the storage used by the approximate distributions to the fully-specified message. We present the results in Fig. 8.3. Although the Factorized message is small in size, it achieves a relatively high error, suggesting it does not represent the distribution well. Although our proposed approximations are bigger in size, they achieve much lower

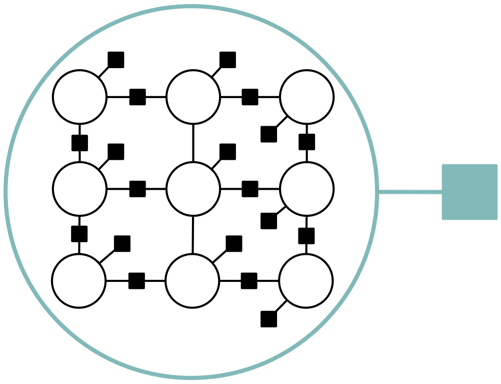


Figure 8.2: **Grid Model a Cluster Variable and Cluster Potential:** Each variable has a domain size of 3, resulting in $19,683$ values in the completely specified cluster message.

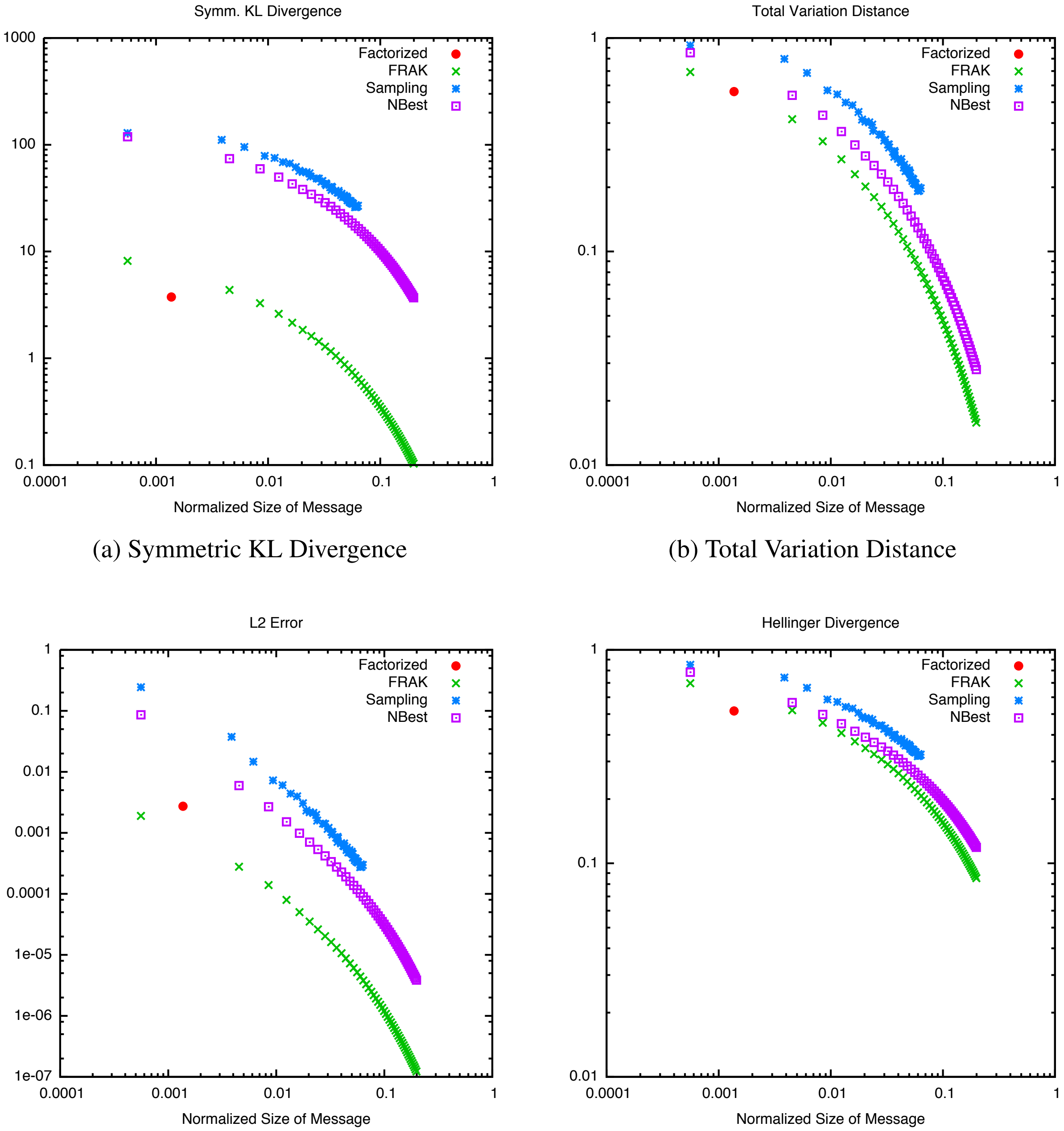
error than factorized messages[2], providing a smooth tradeoff between the message size and the approximation. Further, FRAK outperforms all of the remaining approximations, indicating that representation of the merged probability is crucial for representing such distributions.

8.4.2 Factorial Chains

To closely represent a real-world joint inference task, we generate a large number of factorial CRFs (introduced by Sutton et al. [2007]). These models are commonly used to represent two tagging tasks over the same sequence, where the tasks are usually divided into *top* and *bottom* tasks. Each variable in the top task may be connected to a number of bottom variables in the context, with the clique size defined by the context window w. We generate random potentials, and compare: 1) Regular BP (fully-factorized messages), 2) k-Best (incrementally adds the next best configuration), and 3) FRAK, our combination of sparse and merged messages. With large window and domain sizes, computing even the regular BP messages is expensive, and we expect the sparse representations of messages can outperform loopy BP in speed.

Figure 8.4 summarizes the performance on these chains. For small clique size ($w = 3$) we compare the performance of the three approaches using error when compared to exact marginals.

[2] With the exception of KL divergence. Since Sampling and k-Best messages contain 0 probabilities, the KL is not defined. We use a small probability (10^{-10}) for the purpose of evaluation



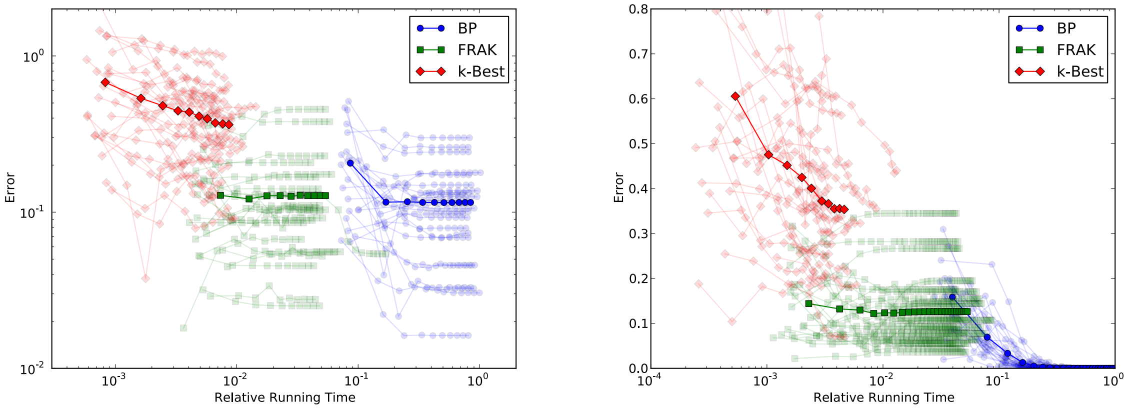
(a) Symmetric KL Divergence

(b) Total Variation Distance

(c) L2 Error

(d) Hellinger

Figure 8.3: **Accuracy of Cluster Message Approximations**: compared on the memory used to represent the messages (normalized by the size of the fully specified message). Error is computed relative to the distribution of the fully-specified message.



(a) Small Cliques (Error to Exact Marginals)

(b) Large Cliques (Error to BP Marginals)

Figure 8.4: Convergence of Approximate Cluster Messages on Synthetic Factorial Chains

As shown in Fig. 8.4a, the FRAK approach converges to the true solution much faster than regular BP. Although k-best is fast, the approximation it is making prevents it from reaching close to the exact marginals. We also ran experiments on a larger clique size of $w = 9$ in Fig. 8.4b. Since we cannot compute exact marginals in this case, we compare against the BP marginals (at convergence). FRAK approach is still converging faster than BP (it is not clear whether to better marginals or not) while still computing more accurate marginals than k-best.

8.4.3 Grids

So far, we have evaluated the FRAK sparse-merged messages, demonstrating the trade-offs for single cluster messages in Section 8.4.1 and the accuracy on dense, high-dimensional factorial chains common in joint inference in Section 8.4.2. In this section we now turn to the utility of such messages for distributed inference, in particular, we show that approximate cluster messages, due to smaller storage than full messages of GBP but more representation power than factorized messages of Loopy BP, provide convergence to lower accuracy marginals with less computation/communication.

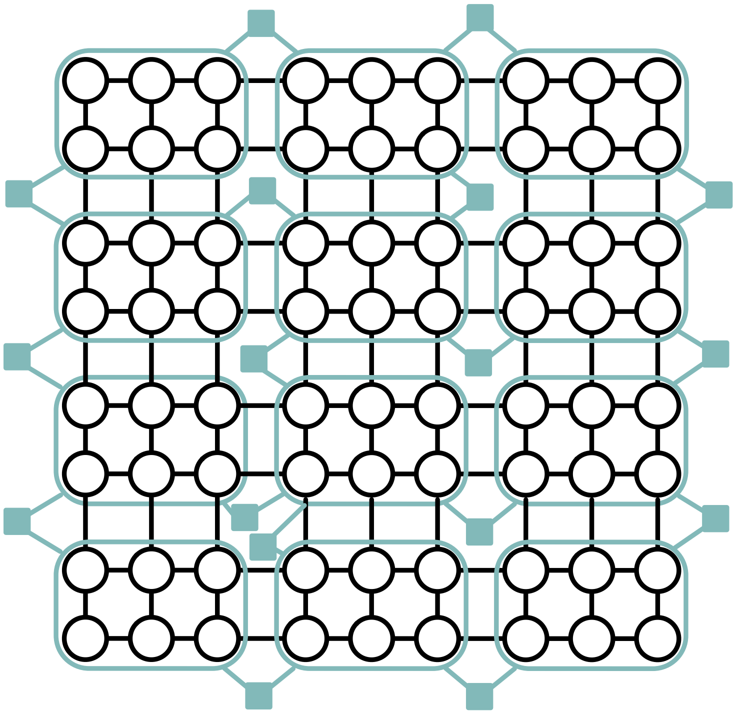
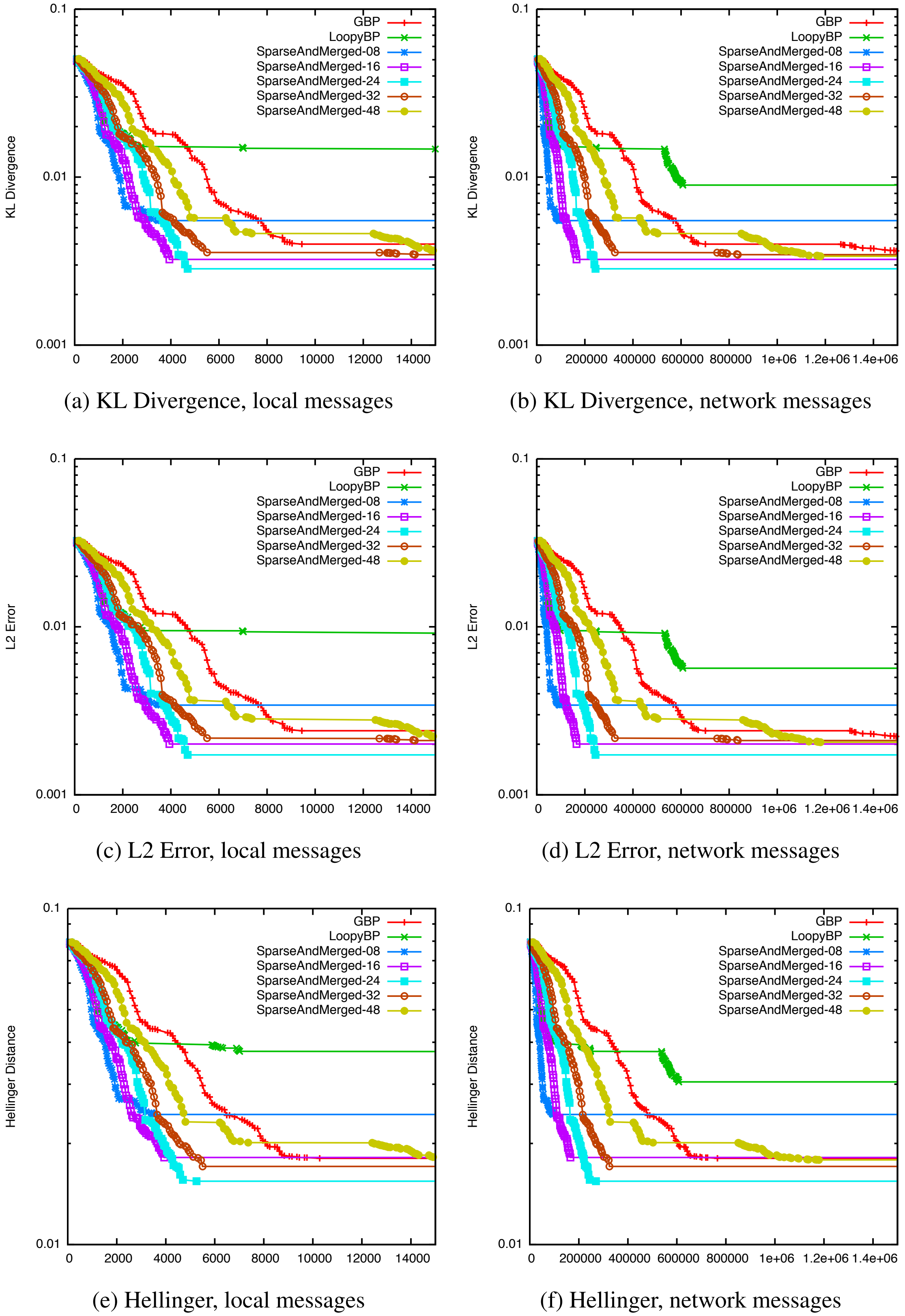


Figure 8.5: **Cluster Model over Synthetic 9×8 Grids:** Consisting of 3×2 sized clusters. Each cluster variable is assigned to a single node/processor of computing its messages, i.e. we simulate 12 processors for the above model. Local factors have been omitted for both the regular and the cluster graph for clarity, along with the black squares that represent regular pairwise factors.

We create synthetic grids with 9×8 binary variables. The underlying model consists of local factors (randomly generated from $\mathcal{N}(0, 0.25)$ and pairwise factors (random generated $e_{ij} \sim \mathcal{N}(0, 1)$, with $\psi_{ij}(y_i, y_j) = (-1)^{\mathbb{I}_{y_i \neq y_j}} e_{ij}$). We create cluster variables consisting of 3×2 grids, and generate the corresponding cluster potentials. The model and cluster graph is shown in Fig. 8.5. We perform inference in the cluster graph using Loopy BP, GBP, and the sparse-merged messages (k-Best and sampling do not perform well in this task). We compare the accuracy of the marginals to true marginals (computed using a separate cluster chain) and measure message sizes for each update. We repeat our experiments 20 times, and average the results.

The results of the experiment are presented in Fig. 8.6. In the left column (Figs. 8.6a, 8.6c and 8.6e), we plot the accuracy of the marginals with the total size of transmitted messages. A number of aspects are evident from these figures. First, we notice that Loopy BP stays at a high-



(a) KL Divergence, local messages

(b) KL Divergence, network messages

(c) L2 Error, local messages

(d) L2 Error, network messages

(e) Hellinger, local messages

(f) Hellinger, network messages

Figure 8.6: **Cluster Message Compressions on Synthetic Grids:** showing that the sparse and merged approximation converges to an accurate solution and is substantially faster.

error for the course of inference. Second, GBP converges to a much lower error than Loopy BP, demonstrating that the cluster Kikuchi approximation leads to more accurate marginals. Third, the sparse messages offer much lower error with smaller message sizes, and further, provide a trade-off with $k = 8$ converging to a slightly higher error quickly. This plots, however, do not differentiate between *local* (no approximation) and *remote* (with approximations) messages, and thus are unfair to Loopy BP and other approximations that are small in size. In the right column (Figs. 8.6b, 8.6d and 8.6f) we present the same results with messages across the (simulated) network associated with a much higher cost ($100\times$ that of the local messages). Although these plots capture a much large proportion of Loopy BP (compare the final error to the left plots), we see that Loopy BP still converges to a much higher error. Further, our sparse-merged approximations, since they are much smaller in size than full messages, converge to a low error substantially faster (fewer bits communicated) than GBP. From these observations, it is apparent that the sparse-merged cluster messages provide much more accurate inference techniques as compared to Loopy BP, but are much cheaper to compute and communicate than the GBP messages.

8.4.4 Joint Segmentation and Entity Resolution

To evaluate our approach on a real-world joint inference problem, we apply it to the task of citation segmentation and resolution. We study the problem that, given a number of citation strings of scientific publications, identifies the author, title and venue fields within each citation, and resolves citations that are duplicates of each other. This task is usually solved as a pipeline with the segmentation task forming the first component that consumes raw citation text, and identifies the *author, title* and *venue* fields. The entity resolution (or *coreference*) component takes a pair of citations as an input and predicts whether the citations refer to the same underlying paper. We use the Cora dataset [McCallum et al., 1999] used in evaluation of MCMC-based joint inference approaches [Poon and Domingos, 2007, Singh et al., 2009] and contains 1295 labeled citations. This is the same dataset we use earlier for evaluating MCMC on citation disambiguation, see Sections 4.4.2, 5.1.3.2 and 5.3.3.1, although using a substantially different model.

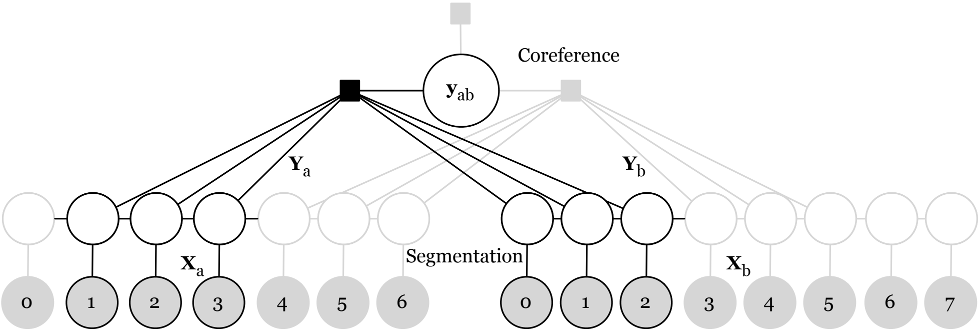


Figure 8.7: **Joint model of Segmentation and Coreference:** Citations $\mathbf{X}_a$ and $\mathbf{X}_b$ have an identical substring $\mathbf{X}_a(1, 2, 3)$=$\mathbf{X}_b(0, 1, 2)$

Our model for segmentation consists of a *linear-chain* CRF, consisting of *local* factors between each token and its label, and *transition* factors between neighboring label variables. Independent of other components, exact inference can be performed efficiently for this model using the forward-backward algorithm. The model of coreference, being the downstream component, consumes the output *labels* of the citation pair (a and b) it is defined over. We would like to the model to capture the intuition that citations that refer to the same paper should have the same tags on the identical tokens, i.e. if the same sequence appears in both citations a and b is labeled as the citations' titles, then this provides evidence for the citations being coreferent. For every subsequence of tokens of fixed length (4 in our experiments) between pairs that are string-identical, the model contains a factor consisting of the common subsequence labels, and the coreference variable (see Fig. 8.7). Note that conditioned on a fixed assignment to the label variables of the segmentation layer, exact inference for coreference is very efficient (in fact, the problem reduces to classification). However, if there is uncertainty in the citation segmentation, coreference has to incorporate the incoming message on the labeling of the substrings. Since this space is very large, messages that contain the full distribution are intractable.

We compare sampling and k-Best approaches with our proposed FRAK approach on the resulting accuracy of coreference, and present the experiment results in Table 8.1. Note that the $k = 1$ baseline corresponds to the pipeline system of conditioning the downstream task on the best

Method		Prec	Rec	**F1**	**Time**
					(hours)
k-Best	$n = 1$	98.8	42.3	59.3	0.08
	$k = 2$	98.9	40.6	57.6	0.16
	$k = 5$	99.1	46.5	63.3	0.4
	$k = 10$	98.8	48.1	64.8	0.8
	$k = 50$	98.2	53.6	69.3	4.0
	$k = 100$	98.2	54.3	69.9	8.0
	$k = 150$	98.2	54.8	70.3	12.0
Sampling	$s = 25$	96.3	67.6	79.4	3.1
	$s = 100$	96.9	67.3	79.5	10.0
FRAK		95.3	69.9	**80.7**	0.16

Table 8.1: **Approximate Cluster Messages for Citation Resolution:** Pairwise evaluation of citation resolution when performed jointly with citation segmentation.

prediction of the upstream task. As we can see from the table, Sampling outperforms k-best on this task, but does not match our performance. FRAK messages, by assigning non-zero probability to the "merged" states, allow the coreference layer to recover from the mistakes made in the segmentation layer. FRAK shows substantial accuracy gains; an error reduction of 52.5% is achieved over the pipeline 1-Best approach. Further, our system takes approximately 10 minutes to run, which is much faster than sampling ($s = 100$ takes ~ 10 hours) and k-Best ($k = 150$ takes ~ 12 hours).

8.5 Related Work

A number of existing approaches are closely related to the contributions in this chapter.

Distributed BP: Various approaches have been proposed to mitigate the costly communication of messages for distributed belief propagation. Gonzalez et al. [2009b] partition the model such that inter-node messages are less relevant for accurate marginals than the intra-node messages. It is also possible to use bulk synchronous model for distributed processing over graphs [Malewicz et al., 2010] to reduce the communication overhead due to asynchronous processing. The idea of using samples as an approximation to messages for distributed belief propagation was also proposed by Ihler et al. [2005a], however for acyclic, non-cluster graphs.

GBP and Cluster Graphs: Generalized belief propagation was introduced by Yedidia et al. [2000], describing the algorithm for a fixed region graph and fully-specified messages, and its relation to Kikuchi approximations. Welling et al. [2005] analyze the approximations of GBP, and describe connections to expectation propagation Minka [2001]. More recently, independent invented from out approach, Gogate and Domingos [2013] explore message passing with structured representation of factor potentials such as sparse hash-tables and algebraic decision diagrams, and describe convergence using expectation propagation. They mention that non-parametric BP [Sudderth et al., 2003] and particle BP Ihler and McAllester [2009] that represent each message as a set of samples (similar to Section 8.2.2) can be analyzed using EP. Heskes and Zoeter [2002] is similar to our work in that it frames approximate inference for DBNs as expectation propagation. Further, the convergent algorithm they propose which is orthogonal to our contribution; there is potential to combine the approaches. Although we do not describe how the clusters are identified in this chapter, approaches have been proposed to describe fruitful regions [Welling, 2004], however we imagine we can utilize splash propagation to partition the models [Gonzalez et al., 2009a].

Approximate Joint Inference: A number of approaches have proposed various approximations to allow efficient but approximate joint inference. Finkel et al. [2005] uses sampling as an approach to approximate the messages between joint inference tasks [Finkel and Manning, 2009], while Wellner et al. [2004] uses k-best list of configurations. Both of these assume 0 probability outside the sparse set of configurations. Even though Gibbs sampling [Finkel et al., 2005] performs only slightly worse for joint inference, our approach provides significant advantages. First, sampling is comparatively very slow. Second, there is inherent variance in the results, as the set of samples may not be representative (due to local minima), and multiple chains may be required. Further, there are a number of hyper-parameters that need to be tuned, such as number of samples, burn-in period, iterations between samples, etc. that can severely impact the results. Factorized frontier algorithm [Murphy and Weiss, 2001] and its variants aim to reduce the complexity of message passing between components from $|\mathcal{Y}|^{|C|}$ to $|\mathcal{Y}|^{|\mathcal{N}(f)|}$, where $\mathcal{N}(f)$ the max fan-in of variables. In most joint-inference problems of our focus, the fan-in is large enough that $|\mathcal{Y}|^{|\mathcal{N}(f)|}$ is still very high, making these approaches intractable. Further, our cluster variables are designed to encourage $|C| = |\mathcal{N}(f)|$ for joint inference.

8.6 Conclusion and Future Work

For distributed inference over large models that span across machines, or for joint inference over multiple tasks with large domains, loopy belief propagation is not practical due to slow marginalization, large number of iterations required to converge, and low accuracy of the resulting marginals. Generalized belief propagation addresses some of these concerns by obtaining the fixed point of a considerably tighter approximation in much fewer iterations, however marginalization is still slow and the messages are exponentially larger. There is a need for approximate inference that can overcome these problems, and perhaps combine the advantages of these two approaches.

In this chapter we introduced a cluster graph based formulation of distributed BP and joint inference that allowed us to present a number of approximations of cluster messages. These approximations span the spectrum of message sizes and approximation factor, providing an easy way to trade-off accuracy with storage/computation time. We also proposed a novel approximation scheme called FRAK that combines the low-storage and fast computation of sparse approximations (such as sampling and k-Best) with a more accurate and adaptive representation of the true message. Results on synthetic models demonstrate FRAK outperforms BP, Sampling and k-Best based messages. Further, on a real-world joint task of citation segmentation and resolution, FRAK is able to obtain an error reduction of 52.5% over the pipeline approach, while being much faster that Sampling or k-Best.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

> *We can only see a short distance ahead, but we can see plenty there that needs to be done.*
>
> Alan Turing

This thesis was motivated by the need to represent complex distributions over the ever-growing datasets available to machine learning, and to allow probabilistic queries efficiently and in a scalable manner on such models. Factor graphs are commonly used to compactly represent the complex dependencies in such distributions, however the resulting models contain massive number of random variables with large domains, along with a dense connectivity of high-order factors. Exact inference for such models is intractable, and even approximate inference schemes such as Markov chain Monte Carlo (MCMC) and belief propagation (BP) do not scale adequately. Instead of making approximations in the complexity of the model (as is common in existing literature), this thesis set out to investigate the utility of scalable approximate inference techniques that make dynamic sparsity approximations and distribute the computations to facilitate efficiency, allowing the models to retain complexity and expressiveness.

9.1 Summary of Contributions

Our first set of contributions focused on extensions to MCMC techniques (Chapter 3 provides an overview). Even though MCMC is an inherently sequential algorithm, in Chapter 4 we found that a Map-reduce based distributed MCMC algorithm that uses a combination of restricted proposals and informed partitioning of the variables can scale to large, dense models. We further discovered that a collection of asynchronous alternatives in Chapter 5 that use stochastic evaluation of the proposals and perform asynchronous communication for distributed processing are able to address some of the undesirable characteristics arising from the synchronous distribution.

The second part of the thesis focused on the belief propagation methods, introduced and contrasted with MCMC in Chapter 6. We found that our proposed techniques in Chapter 7 that induce sparsity on the variable domains in a dynamic and adaptive manner are able to address the large domains and high-order factors, along with providing anytime characteristics and facilitating efficiency and parallelism. Framing joint inference and distributed message passing as generalized BP on a cluster graph in Chapter 8 results in a family of algorithms that we showed can trade-off computation/communication time with accuracy of inference.

9.2 Potential Impact and Lessons Learned

This thesis contributes to existing knowledge in a number of ways. Since the proposed inference algorithms are general-purpose, their application is not limited to the empirical evaluation presented in this thesis. The proposed algorithms can be applied to complex models over large datasets from any application domain, and further, enables future studies on novel probabilistic models that are currently considered intractable. This thesis has also demonstrated the utility of approximate inference with complex models over exact/near-exact inference on simpler models, indicating, for example, the utility of joint modeling and inference of multiple tasks for accurate decision-making with uncertainty. This work also adds to a growing body of literature on understanding the interplay between systems and machine learning, in particular relating theoretical approximations in inference to efficiency in computation and communication.

9.3 Limitations and Future Work

A number of important limitations of the work need to be considered, many of which bring up interesting questions that require further investigation.

9.3.1 Theoretical Analysis

Although we present analysis for a number of our approaches, the generalisability of our results is restricted due to lack of the following theoretical insights.

- We show the validity of synchronously distributed MCMC chains only for random partitioning (Section 4.3), even though the informed partitioning performs much better in practice.

  Future Work: Along with showing validity of these chains, further work needs to be done to identify the properties of the informed partitioning that provide this advantage, and to characterize the speedups.

- Stochastic evaluation of proposals with a fixed level of approximation is limited by lack of detailed balance (Section 5.1.2).

  Future Work: Further investigation is needed to show an adaptive version with vanishing approximation results in valid chains, akin to Andrieu and Thoms [2008] and Atchadé et al. [2011].

- Asynchronously distributed MCMC is shown to maintain valid MCMC chains in a restrictive setting (Sections 5.3.2 and 5.4.3), however fails to demonstrate correctness in the general case of MCMC with overlapping variables.

  Future Work: Possible future studies might extend related work in parallel tempering [Earl and Deem, 2005], distributed optimization [Pan et al., 2013], and asynchronous topic models [Asuncion et al., 2009] to our setting.

- Our current investigation into value sparsity for belief propagation (Chapter 7) was unable to provide convergence analysis.

  Future Work: It would be interesting to see if future work is able to characterize the fixed point of single value sparsity or provide *regret* on the sparse values that are chosen for anytime belief propagation.

- **Future Work:** More research is needed to better understand the convergence with approximate cluster messages for generalized belief propagation (Chapter 8), in particular the theoretical framework introduced by Ihler et al. [2004] may be suitable for such analysis.

9.3.2 Systems Considerations

While this thesis primarily deals with machine learning, we mention a number of systems aspects such as synchronous and asynchronous communication in Section 5.2 and bandwidth conservation by compressing probability distribution in Chapter 8. However, this study fails to consider the following:

- This study restricts itself to fairly simple communication patterns, i.e. we either use synchronous communication (Section 4.1), via a central repository (*star*, in Sections 5.3 and 5.4), or network customized to the model (cluster graph in Section 8.1).

  Future Work: Further work should explore direct, point-to-point communication, or more sophisticated network topologies such as rings and trees.

- The proposed techniques are also limited by the ways memory access is specified; in particular we assume either data is in the processor memory, or needs to be communicated via the network.

  Future Work: What is needed is a study of different data access patterns, such as from RAM, solid-state drives, disk drives, and network file systems, and their effect on the choice of inference.

- The algorithms presented in this work have been designed, for the most part, to be architecture agnostic. This can, however, sometimes be a disadvantage.

  Future Work: In future, we recommend investigation of approaches that are *parameterized* by system considerations, such as memory limit, memory access speed, communication speed etc. so as to adapt to different existing (and future) architectures.

- **Future Work:** Another interaction of machine learning and systems can take in form of an application of scalable inference to probabilistic databases. We describe this in the next section.

9.3.3 Potential Applications

In this work we present empirical evaluation on a number of different applications, including cross-document coreference resolution, author disambiguation, citation segmentation and deduplication, named entity tagging, relation extraction, within-document coreference, and part of speech tagging. Our work, however, need not be limited to these applications:

- One of the primary motivation for this work has been to perform joint inference. Although we have proposed a number of joint models of information extraction that did not exist in the literature, the improved results suggest this paradigm can be fruitful.

  Future Work: Further studies should explore joint inference for tasks and domains that are not modeled yet, perhaps using the algorithms presented in this thesis to scale.

- We perform corpus-level inference on cross-document coreference, and perform joint inference on complex models of within-document coreference.

  Future Work: It is clear that a combination of these, i.e. joint inference of cross-document information extraction, should be an interesting potential application of this approach.

- Apart from synthetic experiments, the empirical evaluation in this work has limited itself to information extraction and natural language tasks.

  Future Work: These are not the only applications of graphical models, and in fact, graphical models have been used in many different domains such as computer vision, bio-informatics, speech recognition, medical applications, and robotics. Many of these applications can benefit from scalable inference alternatives.

- In recent years, a number of proposed approaches for representing uncertainty in databases in a scalable manner have been introduced as *probabilistic databases* (Section 2.4.4). The underlying uncertainty is sometimes represented by graphical models, with MCMC [Wick et al., 2010] or message passing [Singh and Graepel, 2013] as the choice of inference.

Future Work: This suggests further work should, therefore, use the presented approaches for scalable probabilistic databases.

BIBLIOGRAPHY

A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola. Scalable inference in latent variable models. In *International conference on Web search and data mining (WSDM)*, 2012. 3.4.3, 4, 4.5, 5.5

Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Spring Joint Computer Conference of the American Federation of Information Processing Societies (AFIPS)*, pages 483–485, New York, NY, USA, 1967. ACM. doi: 10.1145/1465482.1465560. URL `http://doi.acm.org/10.1145/1465482.1465560`. 4

Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, 2008. 5.1.3, 9.3.1

Apache Storm. The Storm Project. URL `http://storm.incubator.apache.org/`. 2.3.5

Arthur Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Neural Information Processing Systems (NIPS)*, pages 81–88, 2009. 3.4.3, 4.5, 9.3.1

Yves Atchadé, Gersende Fort, Eric Moulines, and Pierre Priouret. *Bayesian Time Series Models*, chapter Adaptive Markov chain Monte Carlo: Theory and Methods. Cambridge University Press, 2011. 5.1.3, 9.3.1

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *International Conference on Language Resources and Evaluation (LREC) Workshop on Linguistics Coreference*, pages 563–566, 1998. 2

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up markov chain monte carlo: an adaptive subsampling approach. In *International Conference on Machine Learning (ICML)*, 2014. 5.5

David Belanger, Alexandre Passos, Sebastian Riedel, and Andrew McCallum. MAP inference in chains using column generation. In *Neural Information Processing Systems (NIPS)*, 2012. 7.3

O. Bender, F.J. Och, and H. Ney. Maximum entropy models for named entity recognition. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 148–151. Association for Computational Linguistics, 2003. 7.1.5.2

Eric Bengston and Dan Roth. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2008. 7.1.5.2

AE Brockwell. Parallel markov chain monte carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, 2006. 3.4.3

Nader H. Bshouty and Philip M. Long. Finding planted partitions in nearly linear time using arrested spectral clustering. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning (ICML)*, pages 135–142, Haifa, Israel, June 2010. Omnipress. 4.4.3

Razvan C. Bunescu and Raymond J. Mooney. Learning to Extract Relations from the Web using Minimal Supervision. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, June 2007. 1.1

Arthur Choi and Adnan Darwiche. Relax then compensate: On max-product belief propagation and more. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 2009. 7.3

J. Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical Statistics*, 14(4):pp. 795–810, 2005. 5.5

Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, Yuanyuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. In B Schölkopf, J Platt, and T Hoffman, editors, *Neural Information Processing Systems (NIPS)*, pages 281–288. MIT Press, 2007. 2.3.5

Common Crawl Foundation. Common Crawl Dataset, November 2011. URL `http://www.commoncrawl.org/`. 1.1

G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990. 2.2.3

James Coughlan and Huiying Shen. Dynamic quantization for belief propagation in sparse spaces. *Computer Vision and Image Understanding*, 106:47–58, April 2007. ISSN 1077-3142. 6.4.3, 7.2, 7.3

James M. Coughlan and Sabino J. Ferreira. Finding deformable shapes using loopy belief propagation. In *European Conference on Computer Vision (ECCV)*, pages 453–468, 2002. 6.4.3, 7.2, 7.3

Aron Culotta. *Learning and inference in weighted logic with application to natural language processing*. PhD thesis, University of Massachusetts, 2008. 3

Aron Culotta, Michael Wick, and Andrew McCallum. First-order probabilistic models for coreference resolution. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2007. 2.4.1, 4.4.2

Nilesh N. Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Comm. of the ACM*, 52(7):86–94, 2009. 2.4.4

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Symposium on Operating Systems Design & Implementation (OSDI)*, 2004. 2.3.3, 2.3.5, 4, 4.1

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The Automatic Content Extraction (ACE) program–tasks, data, and evaluation. In *Proceedings of LREC*, volume 4, pages 837–840. Citeseer, 2004. 7.1.5.2

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. 3.4.3

Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000. 6.1

David J. Earl and Michael W. Deem. Parallel Tempering: Theory, Applications, and New Perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005. 3.4.3, 9.3.1

David Ediger, Karl Jiang, E. Jason Riedy, and David A. Bader. Graphct: Multithreaded algorithms for massive graph analysis. In *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2012. 2.3.5

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Uncertainty in Artificial Intelligence (UAI)*, 2006. 6.2.1, 6.2.1.2, 7.2, 7.2.4, 7.2.4, 7.3

Jenny R. Finkel and Christopher D. Manning. Joint parsing and named entity recognition. In *North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, 2009. 1.1, 2.4.3, 7, 8.5

Jenny R. Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, 2005. 1.1, 8.5

Jenny R. Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2006. 2.4.3

Stanislav Funiak, Carlos Guestrin, Mark Paskin, and Rahul Sukthankar. Distributed inference in dynamical systems. In *Neural Information Processing Systems (NIPS)*, 2006. 6.4.3

Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990. 3.2.3

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6): 721–741, 1984. 3.2.3

Amir Globerson and Tommi Jaakkola. Convergent propagation algorithms via oriented trees. In *Uncertainty in Artificial Intelligence (UAI)*, 2007a. 6.2.3

Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Neural Information Processing Systems (NIPS)*, 2007b. 7.3

Vibhav Gogate and Pedro Domingos. Structured message passing. In *Uncertainty in Artificial Intelligence (UAI)*, 2013. 8.2.5, 8.5

Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*, 2009a. 6.2.1.2, 6.4.3, 7.3, 8.5

Joseph Gonzalez, Yucheng Low, Carlos Guestrin, and David O'Hallaron. Distributed parallel inference on large factor graphs. In *Uncertainty in Artificial Intelligence (UAI)*, 2009b. 2.3.2, 2.3.5, 4.5, 6.4.3, 8.5

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. 2.3.5, 3.4.3, 4, 4.2.3, 5.5, 6.1

Joseph Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012. 2.3.5, 5.2

Chung Heong Gooi and James Allan. Cross-document coreference on a large scale corpus. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 9–16, 2004. 4.4.3

V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):652–656, Jun 1994. ISSN 0162-8828. doi: 10.1109/34.295910. 3.3

W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996. 2.3.5

Aria Haghighi and Dan Klein. Simple coreference resolution with rich syntactic and semantic features. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1152–1161, 2009. 1.1, 2.4.1

Aria Haghighi and Dan Klein. An entity-level approach to information extraction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 291–295, Uppsala, Sweden, July 2010. Association for Computational Linguistics. 1.1, 2.4.1

W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. 2.2.3.1, 3.2.2, 1

Tamir Hazan and Amnon Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *Uncertainty in Artificial Intelligence (UAI)*, 2008. 2.2.3.2, 6.2.3, 7.2.6

Tom Heskes. Stable fixed points of loopy belief propagation are local minima of the bethe free energy. In S. Thrun and K. Obermayer, editors, *Neural Information Processing Systems (NIPS)*, 2002. URL http://books.nips.cc/papers/files/nips15/LT04.pdf. 6.2.2.2

Tom Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16 (11):2379–2413, November 2004. ISSN 0899-7667. doi: 10.1162/0899766041941943. URL http://dx.doi.org/10.1162/0899766041941943. 6.2.3

Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 216–223, 2002. 8.5

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 541–550, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. 1.1, 2.4.2, 5.1.3.1

Geoff Hulten and Pedro Domingos. Mining complex models from arbitrarily large databases in constant time. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 525–531, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775124. 5.5

Alexander Ihler and David McAllester. Particle belief propagation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 256–263, 2009. 6.4.3, 7.2, 7.3, 8.5

Alexander Ihler, John W. Fisher III, and Alan S. Willsky. Using sample-based representations under communications constraints. Technical Report TR2601, Laboratory for Information and Decision Systems, 2005a. 6.4.3, 8.5

Alexander Ihler, John W. Fisher III, Alan S. Willsky, and Maxwell Chickering. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6: 905–936, 2005b. 6.2.1.2, 6.2.3, 6.4.3, 7.2.4, 7.3, 7.4

Alexander T. Ihler, John W. Fisher III, and Alan S. Willsky. Message errors in belief propagation. In *Neural Information Processing Systems (NIPS)*, 2004. 6.4.3, 7.3, 9.3.1

Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *European Conference on Computer Systems (EuroSys)*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-636-3. doi: http://doi.acm.org/10.1145/1272996.1273005. 2.3.5

E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925. 2.1.2

P. Jacob, C.P. Robert, and M.H. Smith. Using parallel computation to improve independent metropolis–hastings based estimation. *Journal of Computational and Graphical Statistics*, 20 (3):616–635, 2011. 3.4.3

Claus S Jensen, Uffe Kjærulff, and Augustine Kong. Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42(6):647–666, 1995. 3.2.3

Rohit J. Kate and Raymond J. Mooney. Joint entity and relation extraction using card-pyramid parsing. In *Conference on Computational Natural Language Learning (CoNLL)*, 2010. 1.1

Ryoichi Kikuchi. A theory of cooperative phenomena. *Phys. Rev.*, 81:988–1003, Mar 1951. doi: 10.1103/PhysRev.81.988. URL `http://link.aps.org/doi/10.1103/PhysRev.81.988`. 6.3

R. Kindermann, J.L. Snell, and American Mathematical Society. *Markov random fields and their applications*. American Mathematical Society Providence, RI, 1980. 2.1.2

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220 (4598):671, 1983. 2.2.3.1, 3.3

Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *International Conference on Machine Learning (ICML)*, pages 441–448, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102407. 5.5

Nikos Komodakis and Georgios Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing*, 16(11):2649 –2661, 2007. 6.4.3, 7.3

A. Korattikara, Y. Chen, and M. Welling. Austerity in mcmc land: Cutting the metropolis-hastings budget. In *International Conference on Machine Learning (ICML)*, 2014. 5.5

A.V. Kozlov and J.P. Singh. A parallel lauritzen-spiegelhalter algorithm for probabilistic inference. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 320–329. ACM, 1994. 6.4.3

A.V. Kozlov and J.P. Singh. Parallel implementations of probabilistic inference. *Computer*, 29 (12):33–40, 1996. 6.4.3

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. 2.1.2

Hsiang-Tsung Kung and John T Robinson. On optimistic methods for concurrency control. *ACM Transactions on Database Systems (TODS)*, 6(2):213–226, 1981. 5.5

Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012. 2.3.5

Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 631–636, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150479. 5.5

Xiao Li, Ye-Yi Wang, and Alex Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *International ACM conference on research and development in information retrieval (SIGIR)*, pages 572–579. ACM, 2009. ISBN 978-1-60558-483-6. doi: http://doi.acm.org/10.1145/1571941.1572039. 7.1

Jun S Liu, Wing Hung Wong, and Augustine Kong. Covariance structure of the gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81 (1):27–40, 1994. 3.2.3

Yan Liu, Jaime G. Carbonell, Peter Weigele, and Vanathi Gopalakrishnan. Protein fold recognition using segmentation conditional random fields (scrfs). *Journal of Computational Biology*, 13(2): 394–406, 2006. 7.1

Dan Lovell, Ryan P. Adams, and Vikash K. Mansinghka. Parallel markov chain monte carlo for dirichlet process mixtures. In *Neural Information Processing Systems (NIPS), Big Learning Workshop on Algorithms, Systems, and Tools for Learning at Scale*, 2012. 3.4.3, 4, 4.5

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010. 2.3.5, 4.5

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. *PVLDB*, 2012. 2.3.5

Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *ACM symposium on Principles of Distributed Computing (PODC)*, pages 6–6, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-396-9. doi: http://doi.acm.org/10.1145/1582716.1582723. 2.3.5

Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *International Conference on Management of Data (SIGMOD)*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0032-2. doi: http://doi.acm.org/10.1145/1807167.1807184. 2.3.5, 8.5

J. Mayfield, D. Alexander, B. Dorr, J. Eisner, T. Elsayed, T. Finin, C. Fink, M. Freedman, N. Garera, P. McNamee, et al. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009. 2.4.1

David McAllester, Michael Collins, and Fernando Pereira. Case-factor diagrams for structured probabilistic modeling. In *AUAI*, pages 382–391, 2004. ISBN 0-9749039-0-6. 2.1.2, 4.1.1

Andrew McCallum and David Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003. 2.4.3

Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*, 2003. 2.4.1

Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Neural Information Processing Systems (NIPS)*, 2004. 4.4.2, 5.1.3.3

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A machine learning approach to building domain-specific search engines. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1999. 4.4.2, 8.4.4

Andrew McCallum, Kamal Nigam, and Lyle Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 169–178, 2000. 4.3.1

Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009. 2.1.2, 3.4.3, 4.1.1, 3

Ofer Meshi, Ariel Jaimovich, Amir Globerson, and Nir Friedman. Convexifying the bethe free energy. In *Uncertainty in Artificial Intelligence (UAI)*, pages 402–410, 2009. 6.2.3

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087, 1953. 2.2.3.1, 3.2.2, 1

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence (UAI)*, pages 362–369, 2001. 8.3, 8.5

Tom Minka and John M. Winn. Gates. In *Neural Information Processing Systems (NIPS)*, pages 1073–1080, 2008. 2.1.2, 4.1.1, 6.1

Joris Mooij and Hilbert Kappen. Sufficient conditions for convergence of loopy belief propagation. *arXiv preprint arXiv:1207.1405*, 2012. 6.2.3

Kevin Murphy and Yair Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Uncertainty in Artificial Intelligence (UAI)*, 2001. 8.5

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, pages 467–475, 1999. 6.2.3

Iain Murray and Zoubin Ghahramani. Bayesian learning in undirected graphical models: Approximate MCMC algorithms. In *Uncertainty in Artificial Intelligence (UAI)*, 2004. 5.5

Lawrence Murray. Distributed markov chain monte carlo. In *Neural Information Processing Systems (NIPS), Workshop on Learning on Cores, Clusters and Clouds (LCCC)*, 2010. 3.4.3

David Newman, Padhraic Smyth, and Mark Steyvers. Scalable parallel topic models. *Journal of Intelligence Community Research and Development*, 2006. 3.4.3

Robert Nishihara, Iain Murray, and Ryan Adams. Generalizing elliptical slice sampling for parallel mcmc. In *Neural Information Processing Systems (NIPS), Big Learning Workshop on Algorithms, Systems, and Tools for Learning at Scale*, 2012. 3.4.3, 4

Nima Noorshams and Martin J. Wainwright. Stochastic belief propagation: Low-complexity message-passing with guarantees. In *Communication, Control, and Computing (Allerton)*, 2011. 6.4.3, 7.2, 7.3

Xinghao Pan, Joseph E. Gonzalez, Stefanie Jegelka, Tamara Broderick, and Michael Jordan. Optimistic concurrency control for distributed unsupervised learning. In *Neural Information Processing Systems (NIPS)*, pages 1403–1411. 2013. 5.5, 9.3.1

Mark Paskin, Carlos Guestrin, and Jim McFadden. A robust architecture for distributed inference in sensor networks. In *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005. 6.4.3

H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Neural Information Processing Systems (NIPS)*, 2003. 4.4.2

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988. 2.2.3.2, 6.2.1.1, 1, 7.3

Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Inf. Process. Manage.*, 42(4):963–979, July 2006. 7.1

Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI Conference on Artificial Intelligence*, 2006. 3.4.3

Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *AAAI Conference on Artificial Intelligence*, pages 913–918, 2007. 1.1, 2.4.3, 4.4.2, 8.4.4

Hoifung Poon, Pedro Domingos, and Marc Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *AAAI Conference on Artificial Intelligence*, 2008. 3.4.3

Delip Rao, Paul McNamee, and Mark Dredze. Streaming cross document entity coreference resolution. In *International Conference on Computational Linguistics (COLING)*, pages 1050–1058, Beijing, China, August 2010. Coling 2010 Organizing Committee. 2.4.1

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2): 107–136, 2006. ISSN 0885-6125. 3.4.3

Sebastian Riedel and David A. Smith. Relaxed marginal inference and its application to dependency parsing. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 760–768, 2010. 7.3

Sebastian Riedel, David A. Smith, and Andrew McCallum. Inference by minimizing size, divergence, or their sum. In *Uncertainty in Artificial Intelligence (UAI)*, 2010a. 7.3

Sebastian Riedel, Limin Yao, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2010b. 2.4.2

Tanya Gazelle Roosta, Martin J Wainwright, and Shankar S Sastry. Convergence analysis of reweighted sum-product algorithms. *IEEE Transactions on Signal Processing*, 56(9):4293–4305, 2008. 6.2.3

J. Rosenthal. Parallel computing and monte carlo algorithms. *Far East Journal of Theoretical Statistics*, 4(2):207–236, 2000. 3.4.3

Evan Sandhaus. The New York Times annotated corpus. *Linguistic Data Consortium*, 2008. 1.1, 4.4.3

Alexander Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Distributed message passing for large scale graphical models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 6.4.3

Prithviraj Sen, Amol Deshpande, and Lise Getoor. Prdb: managing and exploiting rich correlations in probabilistic databases. *The VLDB Journal*, 18:1065–1090, October 2009. ISSN 1066-8888. 2.4.4

B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005. 7.1

Libin Shen, Giorgio Satta, and Aravind Joshi. Guided learning for bidirectional sequence classification. In *Association for Computational Linguistics (ACL)*, 2007. 7.2, 7.3

S.E. Shimony. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2):399–410, 1994. 2.2.3

S. Singh and T. Graepel. Automated probabilistic modeling for relational data. In *ACM Conference of Information and Knowledge Management (CIKM)*, 2013. 2.4.4, 9.3.3

Sameer Singh and Andrew McCallum. Towards asynchronous distributed MCMC inference for large graphical models. In *Neural Information Processing Systems (NIPS), Big Learning Workshop on Algorithms, Systems, and Tools for Learning at Scale*, 2011. 1.3, 5

Sameer Singh, Karl Schultz, and Andrew McCallum. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *Machine Learning and Knowledge Discovery in Databases (Lecture Notes in Computer Science) and European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 414–429, 2009. 1.1, 2.4.3, 3.4.3, 4.4.2, 8.4.4

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 (LCCC)*, 2010. 1.3, 4

Sameer Singh, Brian Martin, and Andrew McCallum. Inducing value sparsity for parallel inference in tree-shaped models. In *Neural Information Processing Systems (NIPS), Workshop on Computational Trade-offs in Statistical Learning (COST)*, 2011a. 1.3, 7

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)*, 2011b. 1.1, 1.3, 2.4.1, 4, 4.4.3

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. WikiLinks: Large-scale cross-document coreference corpus labeled via links to wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst, 2012a. 4.4.3, 5.2

Sameer Singh, Michael Wick, and Andrew McCallum. Monte carlo MCMC: Efficient inference by approximate sampling. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2012b. 1.3, 5

Sameer Singh, Michael Wick, and Andrew McCallum. Monte carlo MCMC: Efficient inference by sampling factors. In *NAACL/HLT Workshop on Knowledge Extraction (AKBC-WEKEX)*, 2012c. 1.3, 5

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint inference of entities, relations, and coreference. In *CIKM Workshop on Automated Knowledge Base Construction (AKBC)*, 2013a. 1.3, 7, 7.1.5.2

Sameer Singh, Sebastian Riedel, and Andrew McCallum. Anytime belief propagation using sparse domains. In *Neural Information Processing Systems (NIPS) Workshop on Resource Efficient Machine Learning (ResEff)*, 2013b. 1.3, 7

Parag Singla and Pedro Domingos. Entity resolution with Markov logic. In *International Conference on Data Mining (ICDM)*, pages 572–582, 2006. 2.4.1

A.F.M. Smith and G.O. Roberts. Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 3–23, 1993. 2.2.3.1, 3.2.1

David A. Smith and Jason Eisner. Dependency parsing by belief propagation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 145–156, Honolulu, October 2008. 4.1.1

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. 3.4.3, 4, 4.5, 6.4.3

Le Song, Arthur Gretton, Danny Bickson, Yucheng Low, and Carlos Guestrin. Kernel belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*, 2011. 6.4.3, 7.3

David Sontag and Tommi Jaakkola. New outer bounds on the marginal polytope. In *Neural Information Processing Systems (NIPS)*, pages 1393–1400, Cambridge, MA, 2008. 7.3

David Sontag, Amir Globerson, and Tommi Jaakkola. Clusters and coarse partitions in LP relaxations. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1537–1544. MIT Press, 2009. 7.3

David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011. 7.3

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, Dec 2001. 7.1.5.2

P. Strandmark and F. Kahl. Parallel and distributed graph cuts by dual decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2085–2092, 2010. 6.4.3

I. Strid. Efficient parallelisation of metropolis-hastings algorithms using a prefetching approach. *Computational Statistics & Data Analysis*, 54(11):2814–2835, 2010. 3.4.3

Amar Subramanya and Jeff A. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2009. 2.3.2

E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition (CVPR)*, 2003. 6.4.3, 7.2, 7.3, 8.5

Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 607–614, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963491. URL http://doi.acm.org/10.1145/1963405.1963491. 5, 5.2

Charles Sutton and Andrew McCallum. Joint parsing and semantic role labeling. In *Conference on Computational Natural Language Learning (CoNLL)*, 2005. 2.4.3

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. 2007a. 5.4.4

Charles Sutton and Andrew McCallum. Improved dynamic schedules for belief propagation. In *Uncertainty in Artificial Intelligence (UAI)*, 2007b. 6.2.1.2, 7.2.4, 7.3

Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 2011. To appear. 4.1.1

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research (JMLR)*, 8:693–723, 2007. 8.4.2

Jie Tang, MingCai Hong, Juan-Zi Li, and Bangyong Liang. Tree-structured conditional random fields for semantic annotation. In *International Semantic Web Conference*, pages 640–653, 2006. 7.1

Sekhar C Tatikonda and Michael I Jordan. Loopy belief propagation and gibbs measures. In *Uncertainty in Artificial Intelligence (UAI)*, pages 493–500. Morgan Kaufmann Publishers Inc., 2002. 6.2.3

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 142–147. Association for Computational Linguistics, 2003. 7.1.5.1

A. Torralba, K. P. Murphy, and W. T. Freeman. Using the forest to see the trees: exploiting context for visual object detection and localization. *Communications of the ACM*, 53(3):107–114, March 2010. 7.1

Yoshimasa Tsuruoka and Jun'ichi Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 467–474, 2005. 7.3

Deepak Venugopal and Vibhav Gogate. Dynamic blocking and collapsing for gibbs sampling. In *Uncertainty in Artificial Intelligence (UAI)*, 2013. 3.2.3, 4.5

Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *Artificial Intelligence and Statistics (AISTATS)*, 2003. 2.2.3.2, 6.2.1.2, 6.2.3, 7.2.6

Huayan Wang and Daphne Koller. A fast and exact energy minimization algorithm for cycle mrfs. In *International Conference on Machine Learning (ICML)*, 2013. 7.3

Peng Wei and Wei Pan. Bayesian joint modeling of multiple gene networks and diverse genomic data to identify target genes of a transcription factor. *Annals of Applied Stats*, 6(1):334–355, 2012. 7

David J. Weiss and Ben Taskar. Structured prediction cascades. In *Artificial Intelligence and Statistics (AISTATS)*, 2010. 7.3

Max Welling. On the choice of regions for generalized belief propagation. In *Uncertainty in Artificial Intelligence (UAI)*, pages 585–592, 2004. 8.5

Max Welling, Thomas Minka, and Yee Whye Teh. Structured Region Graphs: Morphing EP into GBP. In *Uncertainty in Artificial Intelligence (UAI)*, pages 609–616, 2005. 8.2.5, 8.3, 8.5

Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Uncertainty in Artificial Intelligence (UAI)*, pages 593–601, 2004. 2.4.3, 8.5

Tom White. *Hadoop: The Definitive Guide*. O'Reilly, 2009. 2.3.5

Michael Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An entity-based model for coreference resolution. In *SIAM International Conference on Data Mining (SDM)*, 2009. 3.4.3

Michael Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and mcmc. In *Conference on Very Large Data Bases (VLDB)*, 2010. 2.4.4, 9.3.3

Michael Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. Samplerank: Training factor graphs with atomic gradients. In *International Conference on Machine Learning (ICML)*, 2011. 4.4.2

Michael Wick, Sameer Singh, Harshal Pandya, and Andrew McCallum. A joint model for discovering and linking entities. In *CIKM Workshop on Automated Knowledge Base Construction (AKBC)*, 2013. 1.3, 5

D.J. Wilkinson. Parallel bayesian computation. In *Handbook of Parallel Computing and Statistics*, chapter 18. Dekker, 2004. 3.4.3

Yinglong Xia and V.K. Prasanna. Junction tree decomposition for parallel exact inference. In *Parallel and Distributed Processing Symposium*, pages 1 –12, 2008. 7.1.4

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. 3.4.3

Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research (JMLR)*, 7:1887–1907, December 2006. ISSN 1532-4435. 6

Jian Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 7

Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2010. 1.1, 5.1.3.1

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *Neural Information Processing Systems (NIPS)*, number 13, pages 689–695, December 2000. 6.3, 1, 8.5

Xiaofeng Yu and Wai Lam. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *International Conference on Computational Linguistics (COLING)*, pages 1399–1407, Beijing, China, August 2010. Coling 2010 Organizing Committee. 1.1

Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Úlfar Erlingsson, Pradeep Kumar Gunda, and Jon Currey. Dryadlinq: A system for general-purpose distributed data-parallel computing using a high-level language. In Richard Draves and Robbert van Renesse, editors, *Symposium on Operating Systems Design & Implementation (OSDI)*, pages 1–14. USENIX Association, 2008. 2.3.5

A. L. Yuille. Cccp algorithms to minimize the bethe and kikuchi free energies: convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, July 2002. ISSN 0899-7667. 2.2.3.2, 7.2.6

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *USENIX Conference on Hot topics in cloud computing (HotCloud)*, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. 2.3.5

Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 423–438. ACM, 2013. 2.3.5

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 427–434, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. 2.4.2, 7.1.5.2