Efficient Inference Algorithms for Some Probabilistic Graphical Models

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Qiang Fu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. Arindam Banerjee

February, 2014

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school.

I would like to express my sincere gratitude to my advisor Prof. Arindam Banerjee for his invaluable support and guidance throughout my graduate study. I would also like to thank him for introducing me to the area of machine learning and data mining. His enthusiasm in research and his serious work attitude had a great influence on me. I feel lucky to have him as my advisor and I really enjoyed working with him.

I would like to thank Prof. Vipin Kumar, Zhi-li Zhang and Hui Zou for agreeing to be on my thesis committee.

I also want to express my sincere gratitude to my research collaborators and lab mates.

Dedication

To my parents.

Abstract

The probabilistic graphical model framework provides an essential tool to reason coherently from limited and noisy observations. The framework has been used in an enormous range of application domains, which include: natural language processing, computer vision, bioinformatic, robot navigation and many more. We propose several inference algorithms for some probabilistic graphical models. For Bayesian network graphical models, we focus on the problem of overlapping clustering, where a data point is allowed to belong to multiple clusters. We present an overlapping clustering algorithm based on multiplicative mixture models. We analyze a general setting where each component of the multiplicative mixture is from an exponential family, and present an efficient alternating maximization algorithm to learn the model and infer overlapping clusters. We also propose a Bayesian Overlapping Subspace Clustering (BOSC) model which is a hierarchical generative model for matrices with potentially overlapping uniform sub-block structures. The BOSC model can also handle matrices with missing entries. We propose an EM-style algorithm based on approximate inference using Gibbs sampling and parameter estimation using coordinate descent for the BOSC model. We propose an EM-style algorithm based on approximate inference using Gibbs sampling and parameter estimation using coordinate descent for the BOSC model.

We also consider Markov random field graphical models and address the problem of maximum a posteriori (MAP) inference. We first show that the drought detection problem from the climate science domain can be formulated as a MAP inference problem and propose an automatic drought detection problem. We then present a parallel MAP inference algorithm called Bethe-ADMM based on two ideas: tree-decomposition of the graph and the alternating direction method of multipliers (ADMM). However, unlike the standard ADMM, we use an inexact ADMM augmented with a Bethe-divergence based proximal function, which makes each subproblem in ADMM easy to solve in parallel using the sum-product algorithm. We rigorously prove global convergence of Bethe-ADMM. The proposed algorithm is extensively evaluated on both synthetic and real datasets to illustrate its effectiveness. Further, the parallel Bethe-ADMM is shown to scale almost linearly with increasing number of cores.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Probabilistic graphical models [103, 60] use a graph-based representation as the basis for compactly encoding a complex distribution over a high-dimensional space. The framework is quite general in that many of the commonly used statistical models (hidden Markov models, Ising models, Karman filters) can be treated as graphical models. These models have been widely applied across diverse fields such as statistical machine learning, computational biology, climate science, statistical physics, communication theory, and information retrieval.

Graphical models bring together graph theory and probability theory in a powerful formalism for multivariate statistical modeling. In a probabilistic graphical model, each node represents a random variable and the edge express probabilistic relationships between the random variables. The graph captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of variables. The two most common classes of graphical models are *Bayesian networks* and *Markov random fields*.

1.1 Examples of Graphical Models

We first give a brief review of Bayesian networks and Markov random fields respectively.

1.1.1 Bayesian Networks

The underlying semantics of Bayesian networks [82] are based on directed graphs and hence they are also called directed graphical models. A directed graphical model represents a factorization of the joint probability distribution over all random variables, where the nodes are the random variables, and edges correspond, intuitively, to direct influence of one node on another. One way to view the graph is as a data structure that provides the skeleton for representing the joint distribution compactly in a factorized way.

To be more specific, let G be a Bayesian network graph over the variables $X = \{X_1, \ldots, X_n\}$. Each random variable X_i in the network has an associated conditional probability distribution (CPD). The CPD for X_i, given its parents in the graph, which we denote as $Pa(X_i)$, is $P(X_i|Pa(X_i))$. It captures the conditional probability of the random variable, given its parents in the graph. CPDs can be described in a variety of ways. For discrete valued X_i, a simple and common representation for a CPD is a table which contains a row for each possible set of values for the parents of the node describing the probability of different values for X_i. The joint distribution specified by the graphical model can be expressed as a product over the CPDs:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \ .$$

A Bayesian network example [60] is shown in Fig 1.1. In this graphical model, we focus on two diseases – flu and hay fever: these are not mutually exclusive, as a patient can have either, both, or none. Thus, we might have two binary-valued random variables, Flu and Hayfever. We also have a 4-valued random variable Season, which is correlated both with Flu and Hayfever. We may also have two symptoms, Congestion and Muscle Pain, each of which is also binary-valued. Given the CPDs associated with each random variable, the joint distribution is factorized as follows:

$$P(S, F, H, C, M) = P(S)P(F|S)P(H|S)P(C|F, H)P(M|F) \ .$$

1.1.2 Markov Random Fields

Markov random fields [103] are based on undirected graphical models and hence they are also called undirected graphical models. These models are useful in modeling a variety

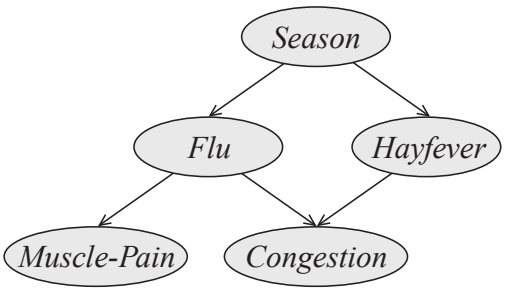


Figure 1.1: A simple Bayesian network with 5 random variables.

of phenomena where one cannot naturally ascribe a directionality to the interaction between variables. As in a Bayesian network, the nodes in the graph of a Markov random field graph represent the random variables, and the edges correspond to some notion of direct probabilistic interaction between the neighboring variables. In this thesis, we focus on the case where each node represents a discrete random variable, i.e., each node can take value from some discrete space $\mathcal{X} = \{1, \ldots, k\}$.

To parameterize a Markov random field, we define *potentials* over cliques of the undirected graph $G = \{V, E\}$, where V is the node set and E is the edge set. For example, in a pairwise Markov random filed, we define potentials over the node and edges. To be more specific, for each node $i \in V$, we have nodewise potential $h_i(X_i), \forall x_i \in \mathcal{X}$ and each edge (i, j), we have edgewise potential $h_{ij}(X_i, X_j), \forall X_i, X_j \in \mathcal{X}$. A pairwise Markov random field then defines a distribution over the random variables that factors according to the potentials:

$$P(X) = \frac{1}{z(X)} \left\{ \prod_{i \in V} h_i(X_i) \prod_{(i,j) \in E} h_{ij}(X_i, X_j) \right\} ,$$

where $z(x)$ is the partition function to make sure $P(X)$ is a valid distribution.

1.2 Inference and Learning in Graphical Models

Both directed and undirected graphical models represent a full joint probability distribution over the random variables and the graph structure often allows the joint distribution to be used effectively for *inference*, i.e., answering queries using the distribution. In particular, we want to compute the posterior probability of some random variables given evidence on others. For example, for the directed graphical model presented in Section 1.1.1, the inference problem can be to compute the probability of having flu if the season is winter and the patient has the congestion symptom:

$$P(Flu = true | Season = winter, Congestion = true, MusclePain = false) \ .$$

Another important inference task is the *maximum a posteriori* (MAP) problem, which it to find the most likely assignment to the random variables. For example, the MAP inference problem in the graphical model presented in Section 1.1.2 is to, given the node-wise and edgewise potentials $\mathbf{H}$, compute the assignment over all the random variables with the largest probability, i.e., $\max_X P(X|\mathbf{H})$.

In principle, inference problems can be solved by exhaustively summing out the joint distribution (for the posterior query) or directly search for the most likely assignment (for the MAP query). This approach is not very satisfactory though, as it results in the exponential blowup of the computation. Fortunately, many types of exact inference can be carried out for graphical models with bounded tree width and the idea is to use dynamic programming to avoid repeated computations [63]. However, for a large number of graphical models, exact inference is intractable and people resort to approximations. Broadly speaking, there are two major frameworks for approximate inference: variational approaches and sampling-based approaches. In variational algorithms, we define a class of tractable distribution Q and search for a particular instance of Q that best approximates the original joint distribution defined by the graphical model [103]. Constructing the class of tractable distribution varies by the inference problems and identifying the best approximation is usually formulated as an optimization problem. In sampling-based algorithms, we construct a Markov chain whose stationary distribution is the true joint distribution and approximate the distribution by obtaining samples from the Markov chain. Commonly used sampling techniques include Markov Chain Monte Carlo (MCMC) methods [6] and Gibbs sampling method [40].

Besides the inference tasks, there are also two *learning* tasks for probabilistic graphical models: parameter estimation and structure learning. In the parameter learning task, we assume the dependency structure is known and the learning task is to estimate CPDs in Bayesian networks, and the potential functions in Markov random fields. In the structure learning task, the goal is to extract the Bayesian network and Markov random field structure. We refer the readers to [60] for the learning task in graphical models.

1.3 Summary of Contributions

This thesis contributes several inference algorithms for probabilistic graphical models. For inference on Bayesian networks, our research work includes:

- We propose a probabilistic model for overlapping clustering, where each data point can potentially belong to multiple clusters. Our overlapping clustering model is based on Bayesian networks and we propose an efficient inference algorithm as the overlapping clustering algorithm.
- We propose another Bayesian network graphical model to identifying overlapping sub-blocks in given data matrices and propose an efficient sampling-based inference algorithm.
- We apply the above overlapping clustering algorithms to solve real-world application problems, such as microarray gene expression analysis and movie recommendation systems. The empirical results show that our algorithms beat the state of the art algorithms.

For inference on Markov random fields, we make the following contributions:

- We focus on an important climate science problem: drought detection from precipitation datasets. We formulate the detection problem as an MAP inference problem on an MRF. After the MAP estimation is computed, we can post-process the solution to identify major droughts. The experimental results show that our algorithm detects all the major droughts of the last century.

- We design a parallel MAP inference algorithm, which is well suited for large scale MRFs. The basic idea behind the parallel algorithm is that we divide the original problem into smaller sub-problems based on graph decomposition and each sub-problem is independent and can be solved in parallel. One novelty of our algorithm is that each sub-problem has closed-form solution, which makes our algorithm very efficient.

- We implement the parallel MAP inference algorithm using Message Passing Interface (MPI). We carefully design the message passing scheme to fully utilize the computing power provided by modern super computers and the empirical results show that we obtain almost linear speedup in the number of MPI processes.

1.4 Organization of the Thesis

The technical part of the thesis is divided into two parts, which we describe below:

In Part I, we address the overlapping clustering problem. We propose overlapping clustering models and efficient inference algorithms. This part of the thesis consists of the following chapters:

- In Chapter 2, we show the importance of efficient overlapping clustering algorithms and discuss the relevant literature. Since we formulate the overlapping clustering problem as an inference problem on a Bayesian network, we also give some preliminary background on constructing Bayesian networks.

- In Chapter 3, we introduce our overlapping clustering model, Multiplicative Mixture Model (MMM), where each mixture component is an exponential family distribution and derive efficient overlapping clustering algorithm. We also propose a kernelized overlapping clustering algorithm based on Gaussian MMMs. We show the efficacy of our algorithms on both simulated and gene-expression datasets. This chapter is primarily based on [33].

- In Chapter 4, we extend the MMM model and propose the Bayesian Overlapping Subspace Clustering (BOSC) model, which can find potentially overlapping sub-blocks in data matrices, automatically detect background noise and naturally

handle matrices with missing values. We also show the experimental results on both simulated and real datasets. This chapter is primarily based on [34, 108].

We discuss our research on the MAP inference problem in Part II and it is organized as follows:

- In Chapter 5, we formally define Markov random fields (MRFs) and the MAP inference problem and review the existing MAP inference algorithms in the literature.

- We start discussing MAP inference from Chapter 6. We use the drought detection problem as a motivating example. We formulate the problem as a MAP inference problem on a three-dimensional grid MRF, where each node represents a temporal/spatial location and can be in two states: dry or normal. We apply our drought detection algorithm on a high resolution precipitation dataset of the last century and our algorithm successfully detects all the major droughts. This chapter is primarily based on [35].

- In Chapter 7, we propose our parallel MAP inference algorithm Bethe-ADMM. In Bethe-ADMM, the MAP inference problem is decomposed into several subproblems on trees and each subproblem is independent and can be solved in parallel. Instead of using a quadratic penalty as in the conventional ADMM, Bethe-ADMM adopts a penalty term based on Bethe entropy on trees, which facilities efficient update on each subproblem. We rigorously prove the global convergence of the ADMM algorithm. We show that our Bethe-ADMM algorithm has similar or better performance than the existing algorithms in the MAP inference literature. This chapter is primarily based on [37, 38].

- In Chapter 8, we discuss efficient MPI implementation of the Bethe-ADMM algorithm and show empirically that our implementation scales almost linearly up to one thousand MPI processes. This chapter is primarily based on [36].

Finally, in Chapter 9, we review the main contributions of this thesis, and discuss future work.

Part I

Bayesian Overlapping Clustering

Chapter 2

Motivation and Related Work

In this Chapter, we show the importance of efficient overlapping clustering algorithms in Section 2.1 and review the relevant literature in Section 2.2. Since we formulate the overlapping clustering algorithm as an inference in a Bayesian network, we give the background knowledge on Bayesian networks in Section 2.3.

2.1 Motivation

One of the most important goals of unsupervised learning is to discover meaningful clusters in datasets. Clustering algorithms strive to discover groups or clusters of data points which are often represented as feature vectors so that data points in the same cluster are similar in some way. For example, if given the task of clustering movies, we might group them by genre (*Action*, *Animation*, *Comedy*, *Horror*, ...) or alternatively by color (*black-and-white*, *color*). Generally speaking, any particular dataset does not have a uniquely correct clustering and the desired clustering depends on the particular application. For instance, in gene-expression data analysis, one might want to cluster the genes according to the biological process they participate in. In social network analysis, one might want to cluster the actors by the communities they belong to.

In machine learning and data mining literature, there has been a huge amount of previous work on different methods for clustering data, including mixture modeling [73], k-means clustering [48] and spectral clustering [95, 81]. While these methods have been widely used in practice, many of them make a strict assumption that each data

point belongs to one and only one cluster; that is, there are k exhaustive and mutually exclusive clusters explaining the data. However, in many real-life applications, some data points may actually belong to multiple clusters. For example, in the context of microarray data analysis, since a particular gene may participate in several biological processes, ideally it may belong to serval clusters each of which represents a set of genes that participate in a common biological process. In social network analysis, since a actor can belong to multiple communities, e.g., he/she is a student at the *University of Minnesota*, works for *Target* and lives in *Loring Park* neighborhood, he/she should be placed in multiple community clusters. Similarly, when we cluster movies by genre, obviously some movies should be in several clusters, e.g, the movie *Transformers* belongs to the *science-fiction* cluster, the *action* cluster and the *thriller* cluster. A related desideratum is to allow some data point not to belong to any cluster at all, possibly because the data point is an outlier or because of the reasons based on domain semantics, e.g., a loner in a social network. More generally, such desiderata attempts to relax the inherent assumption that the entire set of objects available for cluster analysis is actually a part of the cluster structure and the dataset does not contain any outliers.

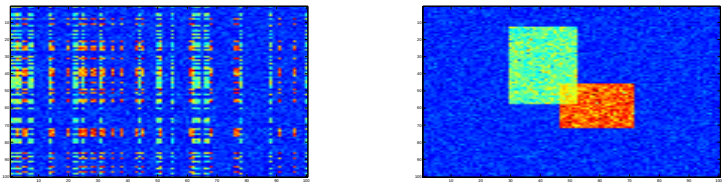
Since the problem of automatically finding overlapping clusters, where an object (data point) can potentially belong to one or more clusters, has been gaining importance in a wide variety of application domains, there is a pressing need for research on overlapping clustering algorithms. Although there are several overlapping clustering algorithms in the machine learning literature, they are either very slow for large datasets [50] or works only with Gaussian data [10]. In this thesis, we present *Multiplicative Mixture Models* (MMMs) as an appropriate framework for overlapping clustering. They are designed to generate overlapping from a Bayesian perspective. MMMs can work with a variety of data types, including real data and categorical data. They can also handle outliers in the dataset. We also propose a fast and scalable overlapping clustering algorithm based on MMMs.

Although MMMs are useful for overlapping clustering, it can only cluster objects with respect to all the features. However, sometimes the cluster structure of objects is often hidden in a subset of features and a meaningful cluster contains objects with only a subset of features. In this thesis, we focus on the matrix setting, where each row represents an object and each column represents a feature. In some applications, we

may want to find dense/uniform sub-blocks in a given data matrix and a dense/uniform sub-block consists of a subset of objects that have similar feature values for a subset of features. For example, in micro-array gene expression data analysis, one would like to find a set of genes which co-express under a set of experimental conditions. In a recommendation system, a uniform sub-block indicates a group of users who have similar ratings for a group of movies.

While progress has been made in the development of subspace [78] clustering and co-clustering algorithms [94], the existing formulations often lack the flexibility needed to solve the problem of finding uniform sub-blocks. The lack of flexibility is often inherited from properties of typical clustering formulations, such as kmeans or graph cuts. In the current context, the desiderata can be captured by the following three requirements. First, *the sub-blocks may overlap*, so that some entries may belong to more than one sub-block. For example, in gene expression analysis, a gene can have multiple functions and hence co-express with different groups under different experimental conditions. Most clustering/co-clustering formulations are not designed to discover overlapping clusters. Second, *not all rows and columns may be a part of a sub-block*, and the formulation has to be flexible enough to allow that. Most existing clustering/co-clustering formulations assume that all points belong to some cluster/co-cluster, and the corresponding algorithms have no capacity to identify background noise automatically. Finally, *the matrix may have missing entries*. For example, in a movie recommendation system, the rating matrix has many missing entries, because users cannot rate all the movies. In practice, one often imputes the missing values with row/column statistics or has a heuristic work around. Ideally, we want the model formulation to be able to work with sparse matrices and in fact use the sparsity to a computational advantage. Figure 2.1 shows a simple example of raw matrix data and a suitable row-column permutation to reveal the overlapping dense block structure.

In this thesis, we also extend the multiplicative mixture model and present a *Bayesian Overlapping Subspace Clustering* model which can find potentially overlapping sub-blocks, automatically detect the background noise, and naturally handle matrices with missing entries.



(a) Raw Data (b) Ideal Output

Figure 2.1: An example problem: (a) Raw data with latent overlapping co-clustering structure, (b) Ideal output from an algorithm, where rows and columns have been permuted to reveal the structure discovered.

2.2 Related Work

2.2.1 Overlapping Clustering Algorithms

Our MMMs are closely related to the Product of Experts (PoE) model proposed by Hinton [51]. The PoE model with k experts has $p(\boldsymbol{x}|\Theta) = \frac{1}{c}\prod_{j=1}^{k} p_j(\boldsymbol{x}|\theta_j)$. The MMM is different from PoE in that it uses a binary vector $\boldsymbol{z}$ to select experts. If $\boldsymbol{z}$ is the all 1 vector in MMMs, then we exactly obtain the PoE model. For general $\boldsymbol{z}$, $p(\boldsymbol{x}|\boldsymbol{z},\Theta)$ is a PoE model over a subset of experts, chosen according to $\boldsymbol{z}$. Training a PoE model by maximizing the log-likelihood can be difficult because it requires the evaluation of an intractable average over $p(\boldsymbol{x}|\Theta)$. The average often requires approximate methods like MCMC sampling to approximate it. But MCMC sampling is computational expensive and results in high-variance estimates of the required averages. Hinton [51] advocates the use of contrastive divergence. It is $KL(P_{\text{data}}||P_{\text{model}}) - KL(P_{\text{data}}||P_k)$, where P_{data} is the empirical data distribution, P_{model} is the current estimate of the model distribution and P_k is the distribution based on k steps of sampling.

A non-parametric Bayesian model for overlapping clustering, due to [50], is also closely related to the proposed MMMs. Their treatment focuses on the use of non-parametric priors based on the Indian Buffet Process [44], and uses MCMC to sample the model parameters Θ, which can be very expensive for large datasets. The analysis for the case when $\boldsymbol{z}$ is all zero was not explicitly handled. In comparison, our proposed MMMs assume knowledge of k, the number of components, although an extension to the non-parametric setting appears straightforward, following ideas in [15]. Further, we

analyze the all zero $\boldsymbol{z}$ case to detect outlying or noisy points, and propose an efficient alternating minimization algorithm for model learning based on optimization.

Another class of overlapping clustering models combines the expectation parameters of component distributions, rather than the natural parameters as in MMMs. For example, Battle et al. [10] use such an idea to discover overlapping processes from gene expression data. Their algorithm works with the observed real gene expression profiles X (genes $\times$ experiments), a hidden binary membership matrix Z (genes $\times$ processes) containing the membership of each gene in each process, and a hidden real activity matrix A (processes $\times$ experiments) containing the activity of each process for each experimental condition. The assumption of their model is $E[\boldsymbol{x}_i] = A\boldsymbol{z}_i$, i.e., each $\boldsymbol{x}_i$ is generated from a Gaussian distribution with mean $A\boldsymbol{z}_i$, which is sum of the activity levels of the processes that contribute to the generation of $\boldsymbol{x}_i$. Banerjee et al. [8] generates the model from Gaussians to other exponential family distributions. However, the additive combination rule $E[\boldsymbol{x}_i] = A\boldsymbol{z}_i$ does not capture the intuition of overlapping clusters, but rather of multiple processes that add together. Several related models with a same generative structure have also appeared in the literature in the form of factorial or multiple cause models [41, 52].

2.2.2 Co-clustering Algorithms

Our work on BOSC model is closely related to research in co-clustering algorithms which simultaneously cluster rows and columns of a matrix. Co-clustering is originally introduced by Hartigan [47] and the algorithm uses a local greedy splitting procedure to identify hierarchical row and column clusters. It adopts SSQ (sum of squares) as the cost function. Suppose the matrix is A and the non-overlapping partition is $B_1, \ldots, B_p$, SSQ is defined as $\sum_p \sum_{i,j \in B_p} (A_{ij} - b_p)^2$, where b_p is the average value of A_{ij} in the co-cluster B_p. The splitting terminates when the sum of squares SSQ is small. Many other co-clustering algorithms have been proposed to address the above partition problem based on various cost functions. Since they are all partitional algorithms, they does not allow overlapping structure. Dhillon et al. [28] propose an information-theoretic co-clustering algorithm that views a non-negative matrix as an empirical joint distribution of two discrete random variables and co-clustering is defined as a pair of maps from rows to row-clusters and from columns to column-clusters. Clearly, these maps induce

clustered random variables. The algorithm tries to find the optimal co-clustering that minimizes the difference in mutual information between the original random variables and the mutual information between the clustered random variables. The algorithm is guaranteed to improve the quality of co-clustering gradually but it works only with non-negative matrices. Cho et al. [24] develops two k-means like minimum sum squared co-clustering algorithms : one with its cost function based on [47] and the other one based on mean square residue formulated by Cheng and Church [23]. Bregman co-clustering [7] is a very efficient, generalized co-clustering framework that works with any distance measure known as Bregman divergence. It includes several previously proposed co-clustering algorithm like [28] and [24] as special cases. Deodhar et al. [27] extends the Bregman co-clustering framework so that only a predetermined number of rows and columns are assigned to the co-clusters. Thus their algorithm can automatically detect and prune away outlier data points which do not show interesting patterns. Recently, Shan and Banerjee propose a Bayesian co-clustering algorithm [94]. It views co-clustering as a generative mixture modeling problem. Each row and column have a mixed membership respectively, from which row and column clusters are generated. Each entry is then generated given the corresponding row and column cluster. The algorithm works with any exponential family distribution and can handle matrices with missing values. The algorithm is a soft co-clustering algorithm, but still does not allow overlapping. Shafie and Milios [93] propose an overlapping co-clustering algorithm by extending the one-way overlapping clustering algorithm in [8]. Their algorithm uses a heuristic simultaneously on rows and columns and thus does not have a theoretical guarantee in terms of accuracy.

Recent advances in co-clustering (bi-clustering) have often centered around problems in bioinformatics, and gene-expression analysis in particular where the goal is to identify subgroups of genes which have similar expression patterns for a subset of experiments. A comprehensive survey of co-clustering methods for gene-expression analysis can be found in [68]. Cheng and Church [23] were among the first to apply co-clustering to gene expression data. They propose a greedy search heuristic to generate arbitrarily positioned, overlapping co-clusters that satisfy a certain homogeneity constraint, called mean square residue. The algorithm is a sequential algorithm, in that it produces a co-cluster one at a time. It cannot effectively handle missing entries. Lazzeroni and

Owen [65] propose the Plaid model, which assumes the expression value in a co-cluster is the sum of main effect, gene effect, condition effect and a noise term which is sampled from a normal distribution with zero mean. Plaid directly models overlapping co-clusters by assuming an additive model where the expression value is the sum of the effect from all the co-clusters a gene belongs to. The algorithm is still a sequential algorithm and does not have a nice way of handling missing values. Tang et al. [100] introduce the Interrelated Two-way Clustering (ITWC) that combines the results of one-way clustering on both dimensions of the data matrix in order to produce biclusters. After normalizing the rows of the data matrix, they compute the vector-angle cosine value between each row and a predefined pattern to test whether the row values vary much among the columns and remove ones with little variation. After that they use a correlation coefficient as similarity measure to measure the strength of the linear relationship between two rows or two column and perform two-way clustering. Yang et al. [109] present an algorithm called Flexible Overlapped biClustering (FLOC) that simultaneously produces k co-clusters whose mean residues are less than a predefined threshold. FLOC incrementally moves a row or column out of into a co-cluster depending on whether the row or column included in that co-cluster or not. The algorithm can produce arbitrarily positioned, overlapping co-clusters. Kluger et al. [59] apply a spectral co-clustering algorithm on gene expression data. The largest several left and right singular vectors of the normalized gene expression are computed and then a final clustering step using k-means and normalized cuts is applied to the data projected to the topmost singular vectors. The algorithm is a partitional algorithm and it models the gene expression matrix as a bipartite graph with non-negative edge weights, thus they are restricted to non-negative matrices.

It is worthwhile to note that some co-clustering algorithms used to analysis gene expression datasets have different definitions of co-cluster compared to ours. The order preserving sub-matrix algorithm [12] tries to identify the co-clusters where the values of rows induce a linear order across the columns. Their work focuses on the relative order of the columns in the co-cluster rather than on the uniformity of the actual values in the data matrix as our model does. Furthermore, they define a complete model as the pair (J, π) where J is a set of s columns and $\pi = (j_1, \ldots, j_s)$ is a linear ordering of the column J. They say that a row supports (J, π) if the s corresponding values, ordered according

to the permutation π are monotonically increasing. Ben-Dor et al. [12] aim at finding a complete model with highest statistically significant support. The algorithm identifies one co-cluster at a time. Tanay et al. [99] propose the Statistical-Algorithmic Methods for Bicluster Analysis (SAMBA) algorithm. It views the elements of the matrix as symbolic values and try to discover subsets of rows and columns with coherent behaviors regardless of the exact numeric values in the data matrix. Tanay et al. define a co-cluster as a subset of genes that jointly respond across a subset of conditions. A gene is considered to respond a certain condition if its expression level changes significantly at that condition with respect to its normal level. The matrix is modeled as a bipartite graph whose two parts corresponds conditions and genes respectively, with one edge for each significant expression change. SAMBA's goal is to discover sub-graphs with an overall coherent evolution. In order to do that it is assumed that all the genes that the genes in a given co-cluster are up-regulated in the subset of conditions that form the co-cluster and the goal is then to find the largest co-cluster with this co-evolution property.

2.2.3 Subspace and Projected Clustering Algorithms

There are two closely related classes of problems, viz subspace and projected clustering, in the data mining literature. Many of them rely heavily on heuristic and tuning parameters. Subspace clustering algorithms essentially search for cluster structure in all possible subspaces of the feature space according to a suitable definition of a cluster. CLIQUE [5] is one of the first algorithms proposed to find clusters within subspaces. It uses static grid to divide each dimension into bins and selects the bins with densities above a threshold. Once the dense subspace are found, they are sorted by coverage, which is defined as the fraction of the dataset covered by the dense units in the subspace. The subspace with the largest coverage are kept. The algorithm then finds adjacent dense grid units in each of the selected subspaces using a depth first search. clusters are formed by combing these units using a greedy growth scheme. Since CLIQUE uses a bottom-up search method, it produces overlapping clusters. ENCLUS [22] is another subspace clustering algorithm based on CLIQUE. The difference is that ENCLUS measures entropy instead of density. The intuition is that a subspace with clusters has lower entropy than a subspace without clusters. MAFIA [43] is another extension of CLIQUE

that uses an adaptive grid based on the distribution of data to improve efficiency and cluster quality. SUBCLU [56] is a grid-free approach that can detect subspace clusters with more general orientation and shape than grid-based approaches, but it still uses a global density threshold. Recently, Moise and Sander [78] propose STATPC algorithm. Since the statistically significant regions are typically redundant, they formulate the problem as extracting a reduced, non redundant set of statistically significant regions from the data axis-parallel regions. The STATPC algorithm is actually an approximation algorithm to find a minimal set of these regions.

Projected clustering algorithms define a cluster as a subset of data points (rows) and features (columns) where the data points are similar when projected on this subset of features and dissimilar when projected on the other features. PROCLUS [3] is the first projected clustering algorithm which selects a set of medoids and iteratively refines the clustering. Initially some data points are chosen as the medoids. But before assigning every data point to the nearest medoids, each medoid is first assigned a set of neighboring objects that are close to it in the input space to form a tentative cluster. For each tentative cluster, all dimension are sorted according to the average distance between the projections of the mediod and the neighbor objects. l dimensions with the smallest average distance are selected as the relevant dimensions for each cluster, where l is the user parameter. Normal object assignment then resumes, but the distance between an object and a medoid is computed using only the selected dimensions. Medoids with few assigned objects are replaced by some other objects to start a new iteration. ORCLUS [4] is proposed to improve PROCLUS. DOC [83] is a hypercube approach projected clustering algorithm, where each cluster is defined as a hypercube with width 2ω, where ω is a user parameter. To find a cluster, a pivot point is randomly chosen as the cluster center and a small set of data points is randomly sampled to form a tentative cluster around the pivot point. A dimension is selected if and only if the distance between the projected values of every sample and the pivot point on the dimension is no more than ω. The tentative cluster is thus bounded by a hypercube with width 2ω. All data points falling into the hypercube are grouped to form a candidate cluster and the candidate cluster with the best evaluation score is accepted. HARP [113] is a hierarchical projected clustering algorithm. In HARP, at the beginning, each data point is treated as a cluster and subsequently merged to form larger clusters.

2.3 Preliminaries

2.3.1 Exponential Family Distributions

An exponential family distribution can be written in the form:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left\{\langle\phi(\boldsymbol{x}), \boldsymbol{\theta}\rangle - Z(\boldsymbol{\theta})\right\} \ , \tag{2.1}$$

where $\boldsymbol{\theta}$ is the natural parameter, $\phi(\boldsymbol{x})$ is the sufficient statistics and $Z(\boldsymbol{\theta})$ is the cumulant function or log partition function. $Z(\boldsymbol{\theta})$ is defined by the following integral to make sure $p(\boldsymbol{x}|\boldsymbol{\theta})$ is a valid probability distribution:

$$Z(\boldsymbol{\theta}) = \log \int \exp\langle\phi(\boldsymbol{x}), \boldsymbol{\theta}\rangle d\boldsymbol{x} \ . \tag{2.2}$$

Many commonly used probability density distributions, e.g., Bernoulli, Dirichlet, Gaussian, can be written as exponential family distributions. Here, we use Bernoulli distribution as an example and derive the exponential family distribution in the form (2.1)

$$\begin{aligned} p(x|\alpha) &= \alpha^x(1-\alpha)^{1-x} && (2.3)\\ &= \exp\left(\log\left(\alpha^x(1-\alpha)^{1-x}\right)\right) && (2.4)\\ &= \exp\left(x\log\alpha + (1-x)\log(1-\alpha)\right) && (2.5)\\ &= \exp\left(x\log\frac{\alpha}{1-\alpha} + \log(1-\alpha)\right) && (2.6)\\ &= \exp\left(x\theta - \log(1+e^{\theta})\right) \ , && (2.7) \end{aligned}$$

where $\phi(x) = x, \theta = \log\frac{\alpha}{1-\alpha}$ and $Z(\theta) = \log(1+e^{\theta})$.

A probability distribution $p(\boldsymbol{\theta})$ is called a conjugate prior to the exponential family distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ if the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{x})$ has the same form as $p(\boldsymbol{\theta})$. For example, the Gaussian distribution is conjugate to itself: if the likelihood function $p(\boldsymbol{x}|\boldsymbol{\theta})$ is Gaussian, choosing a Guassian prior over the mean will ensure that the posterior distribution is also a Gaussian. Conjugate priors are extremely useful tools in Bayesian statistics, since they make things a lot more analytically tractable. In the remainder of this section, we show that Beta distribution is the conjugate prior to

Bernoulli distribution. The Beta-Bernoulli conjugate prior will be used in Chapter 3. Recall that the Beta probability density function is as follows:

$$p(\theta|\alpha,\beta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \ , \tag{2.8}$$

where B is the beta function to ensure that the total probability integrates to 1. Assume the Bernoulli distribution takes the form:

$$p(x|\theta) = \theta^{x}(1-\theta)^{1-x} \ , \tag{2.9}$$

Then the posterior distribution can be written as:

$$p(\theta|x,\alpha,\beta) = \frac{p(\theta|\alpha,\beta)p(x|\theta)}{\int p(\theta|\alpha,\beta)p(x|\theta)d\theta} \tag{2.10}$$

$$= \frac{\theta^{x}(1-\theta)^{1-x}\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} \frac{B(\alpha,\beta)}{\int \theta^{x}(1-\theta)^{1-x}\theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta} \tag{2.11}$$

$$= \frac{\theta^{\alpha+x-1}(1-\theta)^{\beta-x}}{B(\alpha+x,\beta-x+1)} \ , \tag{2.12}$$

which is exactly a Beta distribution parameterized by $\alpha + x$ and $\beta - x + 1$.

2.3.2 Finite Mixture Models

Assume we have a dataset $\mathbf{X}$ of N data points $x_1, \ldots, x_N$. In a finite mixture model, the dataset is modeled by a mixture of k probability distributions:

$$p(\mathbf{X}|\mathbf{\Theta},\mathbf{\Pi}) = \prod_{i=1}^{N}\sum_{j=1}^{k}\pi_j p(\boldsymbol{x}_i|\boldsymbol{\theta}_j) \ , \tag{2.13}$$

where π_j is the weight on the jth component ($\sum_j \pi_j = 1$) and $p(\boldsymbol{x}|\boldsymbol{\theta}_j)$ is the probability density function for the jth component parameterized by $\boldsymbol{\theta}_j$. To generate a data point using the finite mixture model, we first sample a component from the discrete distribution $\mathbf{\Pi}$ and then generate the data point using the corresponding probability density distribution.

Most of the existing literature has focussed on the case where the component distributions are exponential family distributions, and the most widely used mixture models are Gaussian mixture models, where each $p(\boldsymbol{x}|\boldsymbol{\theta})$ is a Gaussian distribution. Maximum

likelihood learning for finite mixture models can be done by applying the Expectation-Maximization (EM) algorithm [26]. The EM algorithm iterates between two steps: (i) For each data point $\boldsymbol{x}_i$, compute the posterior distribution $p(c_i = j|\boldsymbol{x}_i)$ conditioned on the current parameter $\boldsymbol{\Theta}$ estimation. The posterior distribution can be computed as:

$$p(c_i = j|\boldsymbol{x}_i) = \frac{\pi_j p(\boldsymbol{x}_i|\boldsymbol{\theta}_j)}{\sum_{j'} \pi_{j'} p(\boldsymbol{x}_i|\boldsymbol{\theta}_{j'})} \ , \tag{2.14}$$

where c_i is the latent variable for $\boldsymbol{x}_i$ and $c_i = j$ indicates that the data point is generated by the jth component. (ii)Optimize the model parameter $\boldsymbol{\Theta}$ given the posterior distributions computed in the first step. These two steps are iterated until convergence.

Chapter 3

Multiplicative Mixture Model for Overlapping Clustering

In this Chapter, we present Multiplicative Mixture Models as an appropriate framework for overlapping clustering. In Section 3.1, we present MMMs using exponential family distributions as mixture components. Section 3.2 presents an efficient overlapping clustering algorithm that alternates between inference and parameter estimation. In section 3.3, we propose a kernelized overlapping clustering algorithm based on Gaussian MMMs. The kernelized algorithm can allow non-linear separators for clusters. In Section 3.4, we present experimental results on both simulated and real datasets to demonstrate the efficacy of MMMs for overlapping clustering.

3.1 Multiplicative Mixture Models

Consider the traditional additive mixture model with k components whose density function is given by:

$$p(\boldsymbol{x}|\Theta) = \sum_{j=1}^{k} \pi_j p_j(\boldsymbol{x}|\theta_j) \ , \tag{3.1}$$

where π_j is the mixing weight for component j, $p_j(\boldsymbol{x}|\theta_j)$ is the probability density function for component j parameterized by θ_j and $\boldsymbol{x}$ is the data point under consideration. From a generative model perspective, one first samples a component j with probability π_j, and then sample $\boldsymbol{x} \sim p_j(.|\theta_j)$. Clearly, the model assumes each $\boldsymbol{x}$ to have been

generated from one component, making the model unsuitable for overlapping clustering.

A few alternative approaches to mixture modeling based overlapping clustering have been proposed in recent years [10, 8, 88]. In this paper, we consider a multiplicative mixture model motivated by the product-of-experts model of [51], more recently presented in the context of mixture modeling by [50]. For k mixture components, we assume a (latent) binary vector $\boldsymbol{z} = [z_1, \ldots, z_k]$ such that the conditional probability

$$p(\boldsymbol{x}|\boldsymbol{z}, \Theta) = \frac{1}{c(\boldsymbol{z})} \prod_{j=1}^{k} p_j(\boldsymbol{x}|\theta_j)^{z_j} \ , \tag{3.2}$$

where $c(\boldsymbol{z})$ is a normalization constant. The latent boolean vector $\boldsymbol{z}$ indicates which components participated in generating $\boldsymbol{x}$, and $z_j \in \{0, 1\}$ without any restrictions. If $\pi(\boldsymbol{z})$ defines an appropriate prior over $\boldsymbol{z}$, then we have

$$p(\boldsymbol{x}|\Theta) = \sum_{\boldsymbol{z}} \frac{\pi(\boldsymbol{z})}{c(\boldsymbol{z})} \prod_{j=1}^{k} p_j(\boldsymbol{x}|\theta_j)^{z_j} \ . \tag{3.3}$$

From a generative model perspective, one samples $\boldsymbol{z}$ with probability $\pi(\boldsymbol{z})$, and then samples $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}, \Theta)$. Since $\boldsymbol{z}$ can have multiple components as 1, the model is clearly well suited for overlapping clustering.

There are, however, two issues with the above multiplicative model. First, the model may not be well defined when $\boldsymbol{z} = \mathbf{0}$, the all zeros vector. We emphasize that this case should not be ignored by setting $\pi(\mathbf{0}) = 0$, or something equivalent. In several real life datasets, there are points which do not naturally belong to any cluster. Rather than forcing them into an existing cluster, it may be more meaningful to have a model which can potentially leave a few points un-clustered, depending on the structure of the data. Secondly, since $\boldsymbol{z}$ is a boolean vector of size k, inference methods may need to go over all 2^k possible states for each data point. Even with $k = 20$, it amounts to considering a million states for each point in each iteration. In practice, we want inference algorithms that are a few orders of magnitude faster, while maintaining reasonable accuracy. We focus on the modeling issues in the rest of this section, and develop efficient algorithms in Section 3.2.

3.1.1 Exponential Family Mixtures

To make the discussion concrete, we focus on MMMs where the components are exponential family distributions. Recall that a distribution is in the exponential family if the density function with respect to a base measure can be written in the form:

$$p(\boldsymbol{x}|\theta) = \frac{dP(\boldsymbol{x}|\theta)}{dP_0(\boldsymbol{x})} = \exp\{s(\boldsymbol{x})^T\theta - \psi(\theta)\} \ , \tag{3.4}$$

where θ is the natural parameter, $s(\boldsymbol{x})$ is the sufficient statistic, and $\psi(\theta)$ is the cumulant function, which is a convex function of Legendre type [87]. Without loss of generality, we assume $\psi(0) = 0$. With the component distributions being from the same exponential family, the conditional probability in (3.2) becomes

$$p(\boldsymbol{x}|\Theta, \boldsymbol{z}) = \frac{1}{c(\boldsymbol{z})} \exp\left\{ \sum_{j=1}^{k} z_j s(\boldsymbol{x})^T\theta_j - z_j\psi(\theta_j) \right\} \ . \tag{3.5}$$

The normalization $c(\boldsymbol{z})$ must be such that

$$\begin{aligned} 1 &= \frac{1}{c(\boldsymbol{z})} \int \exp\left\{ s(\boldsymbol{x})^T \sum_{j=1}^{k} z_j\theta_j - \sum_{j=1}^{k} z_j\psi(\theta_j) \right\} dP_0(\boldsymbol{x}) \\ &= \frac{\exp\{-\sum_{j=1}^{k} z_j\psi(\theta_j)\}}{c(\boldsymbol{z})} \int \exp\left\{ s(\boldsymbol{x})^T \sum_{j=1}^{k} z_j\theta_j \right\} dP_0(\boldsymbol{x}) \ . \end{aligned}$$

By definition of the cumulant function, we have $\int \exp\{s(\boldsymbol{x})^T\theta\} dP_0(x) = \exp\{\psi(\theta)\}$. Now if the natural parameter is $\sum_{j=1}^{k} z_j\theta_j$,

$$\int \exp\left\{ s(\boldsymbol{x})^T \sum_{j=1}^{k} z_j\theta_j \right\} dP_0(\boldsymbol{x}) = \exp\left\{ \psi\left(\sum_{j=1}^{k} z_j\theta_j \right) \right\} \ .$$

It follows that

$$c(\boldsymbol{z}) = \exp\left\{ \psi\left(\sum_{j=1}^{k} z_j\theta_j \right) - \sum_{j=1}^{k} z_j\psi(\theta_j) \right\} \ .$$

Making use of the closed form for $c(\boldsymbol{z})$, we have

$$p(\boldsymbol{x}|\Theta, \boldsymbol{z}) = \exp\left\{ s(\boldsymbol{x})^T \sum_{j} z_j\theta_j - \psi\left(\sum_{j=1}^{k} z_j\theta_j \right) \right\} \ . \tag{3.6}$$

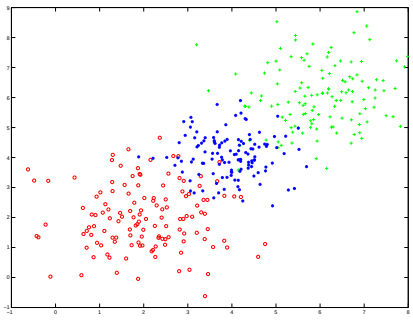


Figure 3.1: Multiplicative Mixture with 2 components. Blue points correspond to $\boldsymbol{z} = [1\ 1]$.

So $p(\boldsymbol{x}|\Theta, \boldsymbol{z})$ is a distribution in the same exponential family as the component distributions, with natural parameter $\sum_j z_j \theta_j$.

Consider the component distributions to be multi-variate Gaussians. For any given $\boldsymbol{z}$, a direct calculation shows that $p(\boldsymbol{x}|\boldsymbol{z}, \Theta)$ is a Gaussian distribution with mean $\bar{\mu}$ and covariance $\bar{\Sigma}$ where

$$\bar{\boldsymbol{\Sigma}}^{-1} = \sum_{j=1}^{k+1} z_j \boldsymbol{\Sigma}_j^{-1} \quad \text{and} \quad \bar{\boldsymbol{\mu}} = \bar{\boldsymbol{\Sigma}} \left(\sum_{j=1}^{k+1} z_j \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\mu}_j \right) .$$

Figure 3.1 shows samples drawn from a multiplicative mixture of two 2-dimensional Gaussian distributions with different choices of $\boldsymbol{z}$. The red and green points respectively correspond to $\boldsymbol{z} = [1\ \ 0]$ and $\boldsymbol{z} = [0\ \ 1]$, i.e., drawn from the component Gaussians; the blue points correspond to $\boldsymbol{z} = [1\ \ 1]$, i.e., the overlapping points, and hence come from another Gaussian distribution as discussed above.

3.1.2 The Noise Component

When $\boldsymbol{z} = \boldsymbol{0}$, from (3.5) it follows that $p(\boldsymbol{x}|\boldsymbol{z}, \Theta) = 1$, which may not be a well defined density function depending on the domain of $\boldsymbol{x}$, e.g., when $\boldsymbol{x} \in \mathbb{R}^d$, $\int_{\mathbb{R}^d} p(\boldsymbol{x}|\boldsymbol{z}, \Theta) = \infty$, as well as the choice of the base measure $P_0(\boldsymbol{x})$. For example, the product model is not meaningful even for 1-dimensional Gaussians with $\boldsymbol{z} = \boldsymbol{0}$. Intuitively, the points corresponding to $\boldsymbol{z} = \boldsymbol{0}$ may be considered as "noise" in that they do not follow the cluster structure implied by the multiplicative mixture model. To incorporate this into

the generative model, we introduce another parametric exponential family as the noise component. The noise component does not necessarily come from the same exponential family as other base components.

Introducing another (latent) boolean variable z_{k+1} for the noise component, which is 1 only when $\boldsymbol{z} = \mathbf{0}$, and 0 otherwise, the conditional probability for the new model is given by

$$p(\boldsymbol{x}|\boldsymbol{z}, z_{k+1}, \Theta) = \frac{1}{c(\boldsymbol{z})} \prod_{j=1}^{k+1} p_j(\boldsymbol{x}|\theta_j)^{z_j} \ . \tag{3.7}$$

3.1.3 Generative Model

A complete specification of the model requires an appropriate prior $\pi(\boldsymbol{z})$ over $\boldsymbol{z}$. Since $\boldsymbol{z}$ is a boolean vector, we assume each component z_j to be sampled from a Bernoulli distribution ϕ_j, which itself has been drawn from a Beta distribution $\text{Beta}(\alpha_j, \beta_j)$. The generative model (Figure 3.2) for a sample $\boldsymbol{x}$ can be described as follows:

1. Draw $\phi_j|\{\alpha_j, \beta_j\} \sim \text{Beta}(\alpha_j, \beta_j)$, for $j = 1, \ldots, k$.
2. Draw $z_j|\phi_j \sim \text{Bernoulli}(\phi_j)$, for $j = 1, \ldots, k$.
3. If $\boldsymbol{z} = \mathbf{0}, z_{k+1} = 1$, else $z_{k+1} = 0$.
4. Draw $\boldsymbol{x}|\{\boldsymbol{z}, z_{k+1}, \Theta\} \sim \frac{1}{c(\boldsymbol{z})} \prod_{j=1}^{k+1} p_j(\boldsymbol{x}|\theta_j)^{z_j}$.

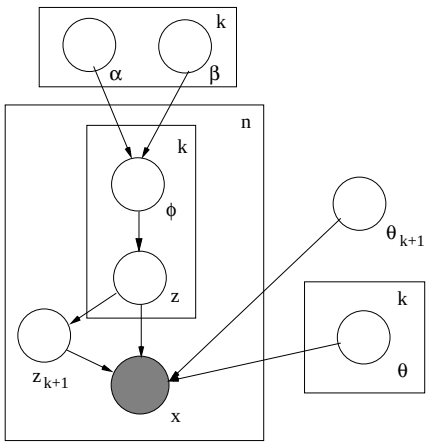


Figure 3.2: Bayesian overlapping clustering model.

Based on the above model, the joint distribution

$$p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\phi}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta) = \frac{1}{c(\boldsymbol{z})} \left(\prod_{j=1}^{k} p(\phi_j|\alpha_j, \beta_j) p(z_j|\phi_j) \right) \left(\prod_{j=1}^{k+1} p(\boldsymbol{x}|\theta_j)^{z_j} \right).$$

The marginal distribution $p(\boldsymbol{x}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta)$ can be obtained by integrating out the latent variables $(\phi_j, z_j), j = 1, \ldots, k$.

3.2 Overlapping Clustering Algorithm

Given a set of data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the task in overlapping clustering based on MMMs is to simultaneously estimate the set of parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta)$ in the model, as well as infer the latent cluster assignment vector $\boldsymbol{z}$ for each data point $\boldsymbol{x}$. In this paper, we formulate the problem as one of finding the mode of the joint distribution of the observable and the corresponding latent cluster assignment $p(\boldsymbol{x}, \boldsymbol{z}|\alpha, \beta, \Theta)$. Noting that $(\boldsymbol{x}, \boldsymbol{z})$ for different data points are conditionally independent, the problem can be posed as maximizing the following objective function:

$$\begin{aligned} L(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta) &= \sum_{i=1}^{n} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta) \\ &= \sum_{i=1}^{n} \log p(\boldsymbol{z}_i|\boldsymbol{\alpha}, \boldsymbol{\beta}) + \sum_{i=1}^{n} \log p(\boldsymbol{x}_i|\boldsymbol{z}_i, \Theta) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} \log \left(\int_{\phi_{i,j}} p(z_{i,j}|\phi_{i,j}) p(\phi_{i,j}|\alpha_j, \beta_j) \; d\phi_{i,j} \right) + \sum_{i=1}^{n} \left(\sum_{j=1}^{k+1} z_{i,j} \log p(\boldsymbol{x}_i|\theta_j) - \log c(\boldsymbol{z}_i) \right). \end{aligned} \tag{3.8}$$

Based on the above objective function, we propose an EM-style alternating maximization algorithm to do inference and estimation. In the E- or inference step, given a set of parameter values $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta)$, we optimize L with respect to $\boldsymbol{z}_i, i = 1, \ldots, n$. In the M- or estimation step, for a given set of overlapping clusterings $\boldsymbol{z}$, we optimize L over the parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta)$. The alternating iterations are assumed to have converged when either no $\boldsymbol{z}_i$ changes in the inference step, or when the maximum absolute change over all parameters in the estimation step is below a threshold.

3.2.1 Inference

First, we focus on the inference step, which maximizes L over $\boldsymbol{z}$ given the parameters. A naive approach to optimizing over $\boldsymbol{z}$ is to try every possible value of $\boldsymbol{z}$, and choose the one which gives the highest log-likelihood. Such an approach has to go over 2^k possibilities for each $\boldsymbol{x}$ in each iteration. As a result, such an approach will be computationally inefficient and impractical even for moderate k. An alternative approach is to use a fast heuristic which ensures that the log-likelihood is non-decreasing. We follow this strategy by adopting an idea from the literature [8].

For any $\boldsymbol{x}$, let $\boldsymbol{z}_0$ be the assignment vector from the previous inference step, let $\mathbf{e}_j, j = 1, \ldots, k$, be the boolean vector with the j^{th} component being 1, and all else zero, and E be the set of all such vectors. The heuristic tries k threads $t_j, j = 1, \ldots, k$, each starting with $\boldsymbol{z}_{1j} = \boldsymbol{z}_0 + \mathbf{e}_j, j = 1, \ldots, k$. In any thread, the algorithm first computes the log-likelihood for $\boldsymbol{z}_{1j}$; then, the algorithm finds the best assignment among $\boldsymbol{z}_{2jj'} = \boldsymbol{z}_{1j} + \mathbf{e}_{j'}$, where $\mathbf{e}_{j'} \in E \setminus \{\mathbf{e}_j\}$; in the next step, the best assignment among $\boldsymbol{z}_{3jj'j''} = \boldsymbol{z}_{2jj'} + \mathbf{e}_{j''}$, where $\mathbf{e}_{j''} \in E \setminus \{\mathbf{e}_j, \mathbf{e}_{j'}\}$; and so on. If the best $\boldsymbol{z}$ at any step is better than the best at the next step, the thread terminates setting $\boldsymbol{z}_{j*} = \boldsymbol{z}$. Finally, the algorithm picks the best $\boldsymbol{z}_{j*}$ among $j = 1, \ldots, k$. Since there are k threads, each thread has at most k steps, and each step has at most k evaluations of the log-likelihood, the complexity of the heuristic is $O(k^3)$ (the number of evaluations is at most $k\binom{k}{2}$). Furthermore, it is guaranteed to give an assignment $\boldsymbol{z}$ that is at least as good as the old assignment, so that the log-likelihood is non-decreasing over iterations. In practice, the heuristic took much less than worst case $k\binom{k}{2}$ and was very fast, making it appropriate for large datasets with moderate to large k.

3.2.2 Estimation

In the estimation step, for a given set of overlapping cluster assignments, we optimize L over the parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \Theta)$. The optimization can be broken into independent two parts—over the parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the Beta distributions, and over the natural parameters Θ of the component exponential family distributions. For a given set of $\boldsymbol{z}$, let m_j be the total number of $z_{i,j}$ that are 1, so that $(n - m_j)$ is the total number of $z_{i,j}$ that are 0. A direct calculation based on taking derivatives w.r.t. (α_j, β_j) and setting

it to 0 shows that the optimal parameters satisfy the following equation:

$$\frac{\alpha_j}{\beta_j} = \frac{m_j}{n - m_j} \ .$$

Setting $\beta_j = 1$, we only update $\alpha_j = m_j/(n - m_j), j = 1, \ldots, k$ in each iteration.

The dependency on the component model parameters is captured by the second term in (4.11). We show that for any exponential family distribution, the objective function L is concave in each θ_j given all other parameters are held constant. Using (3.6) in (4.11), the objective function can be written as a function of Θ given by

$$\begin{aligned} f(\Theta) &= \sum_{i=1}^{n} \left(\sum_{j=1}^{k+1} z_{i,j} \log p(\boldsymbol{x}_i|\theta_j) - \log c(\boldsymbol{z}_i) \right) \\ &= \sum_{i=1}^{n} \left[s(\boldsymbol{x}_i)^T \sum_{j=1}^{k+1} z_{i,j}\theta_j - \psi \left(\sum_{j=1}^{k+1} z_{i,j}\theta_j \right) \right] \ . \end{aligned}$$

Since ψ is the cumulant of an exponential family, it is a convex function of Legendre type, implying that it is in C^∞. Computing the second derivative of $f(\Theta)$ with respect to θ_j, we have

$$\nabla^2_{\theta_j} f(\Theta) = -\sum_{i=1}^{n} z_{i,j} \nabla^2_{\theta_j} \psi \left(\sum_{h=1}^{k+1} z_{i,h}\theta_h \right) \ ,$$

which is negative, since ψ is a convex function implying $\nabla^2\psi$ is positive. Hence, $f(\Theta)$ is a concave function of θ_j. In order to find the maximizer θ_j^* given all the other parameters $\theta_h, h \neq j$, taking gradient and setting it to 0, we obtain

$$\theta_j^* = \sum_{i:z_{i,j}=1} \sum_{\substack{h=1 \\ h\neq j}}^{k+1} z_{i,h}\theta_h + (\nabla\psi)^{-1} \left(\sum_{i:z_{i,j}=1} s(\boldsymbol{x}_i) \right) \ .$$

Since ψ is a Legendre function, the function $(\nabla\psi)^{-1}$ will be well defined and equal to $\nabla\phi$, where $\phi = \psi^*$, the conjugate of the cumulant function ψ. The actual update equation for any exponential family can be derived by plugging in the specific cumulant function ψ and sufficient statistics $s(\boldsymbol{x})$.

3.3 Kernelized Overlapping Clustering

In this section, we show that the proposed multiplicative model can be kernelized, and the overlapping clustering algorithm can be extended to the general case. There are two

key advantages to the kernelized extension: (i) Individual base clusters can be separated by non-linear boundaries, making the approach applicable to more complex data, and (ii) Overlapping clustering can be applied to structured data, such as strings, trees, graphs, etc., for which a meaningful kernel can be defined [91]. To make the kernelized extension, we implicitly map the data points to a high dimensional space and assume that in the high dimensional space there are k spherical Gaussian clusters. If $\phi(.)$ is the mapping function, a Gaussian in the high dimensional space can be represented as :

$$p(\phi(\boldsymbol{x})|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp\left(-\frac{1}{2}(\phi(\boldsymbol{x}) - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\phi(\boldsymbol{x}) - \boldsymbol{\mu})\right)}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}}$$
$$= \frac{\exp\left(-\frac{a}{2}\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x})\rangle + a\langle\phi(\boldsymbol{x}), \boldsymbol{\mu}\rangle - \frac{a}{2}\langle\boldsymbol{\mu}, \boldsymbol{\mu}\rangle\right)}{(2\pi)^{D/2}a^{-D/2}} ,$$

where $a = \frac{1}{\sigma^2}$ is the inverse of the Gaussian variance so that $\boldsymbol{\Sigma}^{-1} = a\mathbb{I}$, $\boldsymbol{\mu}$ is the mean of the Gaussian and D is the high dimension. Plugging the above expression into (3.7), the log-likelihood of MMM with respect to a single data point $\boldsymbol{x}$ becomes :

$$\log p(\phi(\boldsymbol{x})|\boldsymbol{z}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}) = -\frac{D}{2}\log(2\pi) + \frac{D}{2}\log(\bar{a}) - \frac{\bar{a}}{2}\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x})\rangle + \bar{a}\langle\phi(\boldsymbol{x}), \bar{\boldsymbol{\mu}}\rangle - \frac{\bar{a}}{2}\langle\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\mu}}\rangle ,$$

(3.9)

where $\bar{a} = \sum_{j=1}^{k+1} z_j a_j$ and $\bar{\boldsymbol{\mu}} = \sum_{j=1}^{k+1} z_j a_j \boldsymbol{\mu}_j / \bar{a}$.

A direct calculation for the estimation step shows when each component in MMM is a Gaussian, the mean of each Gaussian $\boldsymbol{\mu}_j, j = 1, \ldots, k$ can be estimated using an appropriate linear combination of all the data points $\phi(\boldsymbol{x}_i), i = 1, \ldots, n$. Let $\boldsymbol{\mu}_j = \sum_{i=1}^{n} c_{i,j}\phi(\boldsymbol{x}_i)$ and $c_{i,j} \in \mathbb{R}$, we have:

$$\langle\phi(\boldsymbol{x}), \bar{\boldsymbol{\mu}}\rangle = \frac{1}{\bar{a}}\sum_{j=1}^{k+1}\langle\phi(\boldsymbol{x}), z_j a_j \boldsymbol{\mu}_j\rangle = \frac{1}{\bar{a}}\sum_{j=1}^{k+1}\sum_{i=1}^{n} z_j a_j c_{i,j}\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i)\rangle ,$$
$$\langle\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\mu}}\rangle = \frac{1}{\bar{a}^2}\langle\sum_{j=1}^{k+1} z_j a_j \boldsymbol{\mu}_j, \sum_{j'=1}^{k+1} z_{j'} a_{j'} \boldsymbol{\mu}_{j'}\rangle$$
$$= \frac{1}{\bar{a}^2}\sum_{j=1}^{k+1}\sum_{j'=1}^{k+1} z_j z_{j'} a_j a_{j'} \sum_{i=1}^{n}\sum_{i'=1}^{n} c_{i,j} c_{i',j'}\langle\phi(x_i), \phi(x_{i'})\rangle .$$

Suppose the kernel similarity matrix is K, replacing the inner product $\langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_{i'})\rangle$ with $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$, and plugging in the kernelized terms back in (3.9), we obtain the

Variance	Ratio 1	# Noisy Points	Ratio 2
0.01	0.9572 ± 0.0048	9	0.8889 ± 0.0000
0.05	0.9320 ± 0.0037	7	0.5429 ± 0.1195
0.10	0.9240 ± 0.0081	4	0.4500 ± 0.1118
0.20	0.8824 ± 0.0193	5	0.4400 ± 0.0894

Table 3.1: Clustering accuracy on 4 simulated datasets.

objective function for kernelized overlapping clustering algorithm. The inference and estimation step remains the same, except that we need to estimate $c_{i,j}, j = 1, \ldots, k+1$, instead of $\boldsymbol{\mu}_j, j = 1, \ldots, k+1$.

3.4 Experimental Results

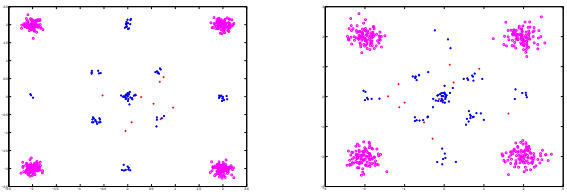
In this section, we present experimental results on simulated datasets, UCI benchmark datasets and a microarray gene expression dataset.

3.4.1 Simulated Datasets

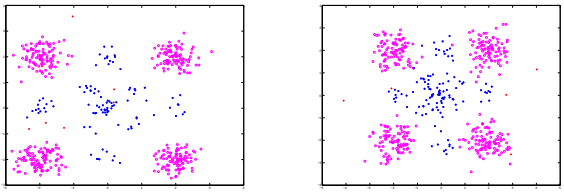
We start with a simple setting. Consider four 2-dimensional spherical Gaussian clusters which centers at (2,2), (-2,2), (2,-2) and (-2,-2) and the noisy cluster centers at (0,0). We create 4 datasets and each dataset has 400 data points. The variances of the datasets are 0.01, 0.05, 0.10 and 0.20, one for each. The data distribution plots for two datasets are in Figure 2. The magenta points belong to one cluster, the blue points are overlapping, and the red points are noise. Note that as the variance increases, the cluster structure is less clear.

We run the overlapping clustering algorithm on all 4 data sets and compute two ratios: Ratio 1 is the fraction of data points that get the correct cluster assignment, and Ratio 2 is the fraction of the noisy data points which are correctly detected by the algorithm (Table 3.1). When the variance is small, the algorithm works very well in getting the correct overlapping cluster assignments as well as detecting outliers. As expected, the performance degrades with increasing variance.

We also test our algorithm on a larger dataset, which has 2000 points from ten 15-dimensional Gaussian clusters (Table 3.2). The parameters in the generative model



(a) variance = 0.01 (b) variance = 0.05



(c) variance = 0.10 (d) variance = 0.20

Figure 3.3: Data distributions with different variances.

are learned by running the overlapping clustering algorithm on the Pendigits dataset in UCI Machine Learning Repository.

# Errors	0	1	2	3	4	5
Mean	0.8629	0.0949	0.0264	0.0115	0.0039	0.0004
Std	0.0313	0.0079	0.0119	0.0149	0.0062	0.0009

Table 3.2: Clustering accuracy on simulated data: Row 1 measures the number of cluster assignment errors made, and Row 2 and 3 are the mean and standard deviation of the corresponding fraction of the dataset.

3.4.2 UCI Datasets Based on Overlapping Clustering Algorithm without Kernels

We run the overlapping clustering algorithm on 8 UCI datasets (Table 3.3). For all experiments reported, we set k to be the true number of classes, and use multivariate Gaussian with diagonal covariance matrix to model each cluster. Since the proposed algorithm is based on alternate maximization, careful initialization is necessary. We

	Iris	Ionosphere	Vowel	Wdbc	Pima	Segment	Landsat	Pendigits
k	3	2	11	2	2	10	6	10
d	4	32	10	30	8	16	36	15
n	150	351	528	569	768	2310	6435	10922

Table 3.3: Data Sets.

adopt the semi-supervised seeding approach [2]: we randomly select 10% of the data points from each class and each base cluster is initialized using the means and variances of the selected data points from the class. We run the algorithm on each dataset 5 times with different initialization and report the result based on the one which has the highest log-likelihood.

To make comparisons, we use 2 baseline algorithms. The first one is the overlapping clustering algorithm described in [9], which we refer to as BSK algorithm. The second one is the EM algorithm based on Gaussian mixture models. To get overlapping clustering from EM, we threshold the posterior probability: for a given threshold t, if for any cluster j the posterior probability $p(j|x) \geq t$, we consider $\boldsymbol{x}$ belongs to cluster j. The initialization and convergence criterion are the same for all the algorithms.

Since the UCI datasets do not have overlapping labels, we evaluate the algorithms using predictions based on the overlapping clustering, e.g., points with multiple cluster assignments are possibly close to the boundary of classes, and are good candidates for support vectors in a Support Vector Machine (SVM) classifier. Before getting to the actual experiments, we note that a dataset can be split into three subsets based on the cluster assignments: (i) pure data points, which belong to only one cluster, (ii) overlapping data points, which belong to more than one cluster, and (iii) noisy data points, which do not belong to any cluster.

Experiment 1: We study the overlapping data points with the following hypothesis—overlapping points lie close to the boundary of classes, and have higher chance of becoming support vectors in a SVM classifier. To test the hypothesis, we train a SVM classifier, based on LIBSVM [20], using linear kernel and default parameter settings on each dataset, and obtain the support vectors. Then, for each dataset, we compute the following three ratios: Ratio 1 is the fraction of support vectors in the data set, i.e., $\frac{|\text{Support Vectors}|}{n}$. Ratio 2 (Precision) is the fraction of overlapping points that are support vectors, i.e., $\frac{|\text{Overlapping} \cap \text{Support Vectors}|}{|\text{Overlapping}|}$ and Ratio 3 (Recall) is the fraction of

support vectors that are overlapping points, i.e., $\frac{|\text{Overlapping} \cap \text{Support Vectors}|}{|\text{Support Vectors}|}$.

Based on our hypothesis, we expect Ratio 1 $<$ Ratio 2. The result is listed in Table 3.4. For the overlapping clustering algorithm, the inequality is true on 7 datasets. We also find that the set of overlapping data points detected by our algorithm has a reasonable intersection with that of support vectors. However, BSK algorithm either fails to find any overlapping points on 6 datasets (Ratio 2 is N/A) or finds only few overlapping data points (9 for Ionosphere and 6 for Segment). For EM algorithm, Ratio 2 is larger than Ratio 1 in most cases, but Ratio 3 is usually very small, which indicates that EM algorithm tends to give few overlapping points. This observation can be explained as follows : if one of the posterior probabilities $p(j|\boldsymbol{x})$ is large, all the other posterior probabilities will become relatively smaller since $\sum_{j=1}^{k} p(j|\boldsymbol{x}) = 1$. So when we threshold on the posterior probability, we get very few overlapping data points, making EM unsuitable for overlapping clustering. The phenomenon is obvious for datasets with larger values of k, such as Landsat, Vowel, and Segment.

Experiment 2: The second experiment involves the union of overlapping and noisy data points. Since the overlapping points belong to multiple clusters and noisy data points do not belong to any cluster, the hypothesis is that they should have higher classification error rate. Assuming there are m overlapping and noisy points, the second experiment proceeds as follows: We train a SVM using the entire dataset, test it on the overlapping and noisy data points to get Error Rate 1. Then we test the SVM on m random training samples (repeated 10 times), and get Error Rate 2. Now, our hypothesis implies Error Rate 1 $>$ Error Rate 2. We also calculate the pair-wise p-value. If our hypothesis is valid, we should expect a low p-value.

The experiment results are listed in Table 5. For overlapping clustering algorithm, the inequality holds on most of the datasets. As for BSK, only 3 datasets support the hypothesis. The result based on EM is listed in Table 6. Error Rate 1 is larger than Error Rate 2 in most cases and the p-value is low especially for small threshold.

Experiment 3: The third experiment involves pure data points. The experiment proceeds as follows: The SVM is trained using a 10-fold cross validation on the original data set. In each fold, for the test points which got the correct label, we compute the fraction of pure data points (Ratio 1). For the test points which got the wrong label, we compute the fraction of pure data points (Ratio 2). The hypothesis is that since

pure points are not close to class boundaries, the correctly predicted points will mostly be pure, i.e., Ratio 1 > Ratio 2.

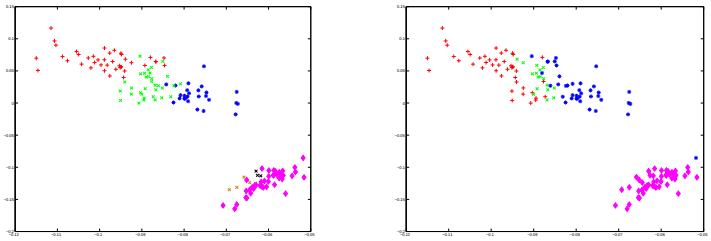
The results are listed in Table 3.7 along with the 10-fold cross validation classification error rate and the p-values for a paired t-test. For overlapping clustering algorithm, the inequality holds for most of the datasets. However, for BSK, only 2 datasets support the hypothesis. The results based on EM is listed in Table 3.8. As we can see, the p-value is relatively larger in quite a few cases.

3.4.3 UCI Datasets Based on Kernelized Overlapping Clustering Algorithm

We also test the kernelized overlapping clustering algorithm on the 8 UCI datasets. We use RBF kernel $\exp(-\frac{\|\boldsymbol{x}-\mathbf{y}\|^2}{d})$ in the algorithm, where d is the dimension of the original dataset. To match the assumption of using spherical Gaussians, we first z-score the datasets. The baseline algorithm is the overlapping clustering algorithm without kernels. After running both algorithms using five different initializations, we report the results based on the initialization which leads to the highest log-likelihood for the overlapping clustering algorithm without kernels. The SVM is trained using RBF kernel with default settings. We report the results of kernelized overlapping clustering algorithm on different dimensions D[1] . We still compared both algorithms using the 3 experiments. Table 3.9 represents the results of experiment 1. It shows that the kernelized overlapping clustering algorithm can have higher precision (Ratio 2) than that of baseline algorithm on most of the datasets, while maintaining higher or similar recall (Ratio 3). For Experiment 2, the hypothesis Error Rate 1 > Error Rate 2 is supported on all the datasets by both two algorithms. For Experiment 3, the hypothesis Ratio 1 > Ratio 2 holds on most of the datasets for both algorithms. We do not show the detailed results due to lack of space.

We use the results on z-scored Iris as an example to show that kernelization helps to obtain better clustering quality. Figure 3.4 shows the clustering effect of both algorithms on a two-dimensional space. The red, blue and magenta points represent pure points and others being overlapping points. The cluster represented by the magenta points is far away from the other two clusters and ideally all the points at the right bottom corner

[1] As it is proved in [91], the largest possible D is the size of the dataset n.



(a) Without kernel (b) With kernel

Figure 3.4: The clustering effect of two algorithms on z-scored Iris (best seen in color).

should be labeled as pure points. However, the overlapping algorithm without kernels finds 7 points (brown and black) as overlapping points. The kernelized algorithm only assigns one point (blue) to the blue cluster.

3.4.4 Microarray Gene Expresssion Dataset

To find an application for the overlapping clustering algorithm, we make use of a microarray gene expression dataset. The dataset is in the form of a matrix, where each row represents a gene and each column represents an experimental condition. The entry of the matrix is a measurement of the activity of a gene under a certain experimental condition. Our goal is to cluster the genes into multiple biological processes based on the expression profiles. Since many genes are known to be multi-functional, we would expect that some genes participate in more than one biological processes, thus overlapping clustering is a natural approach for the problem.

We evaluated our algorithm on a yeast microarray gene expression dataset. The dataset consists of 4062 genes and 215 experimental conditions. We report results on 1354 genes that have significant changes in the gene expression, i.e., 1/3 of the genes that have the highest variances of gene expression over the 215 experimental conditions. The number of clusters k is fixed to be 30. As before, we compare our overlapping clustering algorithm with BSK algorithm [10] and EM. We initialize all the algorithms based on the preliminary clustering result given by kmeans.

Overall, our overlapping clustering algorithm predicts that 556 genes participate in only one process, 552 in two, 219 in three and 27 in four or more, while BSK algorithm discovers that 95 genes do not belong to any process and 397 participate in only one

process, 383 in two, 255 in three and 224 in four or more. For EM algorithm, we set the posterior probability threshold to be 0.01. Based on the analysis in the previous section, it is not surprising to see that EM gives very few overlapping genes even under this low threshold: it predicts 1324 genes participate in only one process, 27 in two and 3 in three or more. This result further illustrates that EM should not be considered for overlapping clustering on complex datasets.

To evaluate whether the cluster assignments for the genes are reasonable from a biological perspective, we check if the genes in each learned biological process show any enrichment for known annotations. We make use of Gene Ontology Term Finder[2] online tool, which searches for shared annotations given a set of genes and computes an associated p-value. The p-value measures the probability of observing a group of genes to be annotated with a certain annotation purely by chance. If a cluster of genes indeed correspond to known biological processes, we would expect a low p-value. We consider an annotation to be significant if the p-value associated with it is less than 10^{-4}. Both the overlapping clustering algorithm and BSK algorithm discover 94 different significant annotations. Among the 62 common significant annotations, the overlapping clustering algorithm performs better in 37 (60%) of them with lower p-values. In case a significant annotation presents in more than one learned processes in any algorithm, we pick the one with the lowest p-value.

We also test the kernelized overlapping clustering algorithm on the z-scored dataset. We try three different RBF kernels, $\exp(-\frac{\|\boldsymbol{x}-\mathbf{y}\|^2}{\sigma^2}), \sigma^2 = 250, 500, 750$, and specify the high dimension D to be 1354, the number of genes we use in the experiment. The detailed result is listed in Table 3.10. Figure 3.5 shows the scatter plot of the negative log p-value of the common significant annotations discovered by kernelized overlapping clustering algorithm, overlapping clustering algorithm without kernels, BSK algorithm and EM algorithm. As the results show, the proposed overlapping algorithm, especially the kernelized version, performs favorably compared to the baseline algorithms.

[2] http://db.yeastgenome.org/cgi-bin/GO/goTermFinder.pl

	Ratio 1	Ratio 2 (Precision)					Ratio 3 (Recall)				
		MMM	BSK	Thresholded EM			MMM	BSK	Thresholded EM		
				0.01	0.1	0.2			0.01	0.1	0.2
Iris	0.1800	0.6250	N/A	0.5172	0.6250	**0.6364**	**0.5556**	0	**0.5556**	0.3704	0.2592
Ionosphere	0.7493	**0.9223**	0.7778	0.8462	0.7143	0.6667	**0.3612**	0.0266	0.0418	0.0190	0.0076
Vowel	0.6913	**0.8537**	N/A	0.6892	0.7609	0.7778	0.1918	0	**0.2795**	0.0959	0.0575
Wdbc	0.1002	0.2857	N/A	0.6000	0.7000	**0.7778**	**0.6667**	0	0.1579	0.1228	0.1228
Pima	0.5208	0.6626	N/A	0.6049	**0.7226**	0.6629	0.2700	0	**0.4975**	0.2475	0.1475
Segment	0.1242	0.1338	**0.6667**	0.2289	0.2157	0.1818	**0.5958**	0.0139	0.0662	0.0383	0.0209
Landsat	0.2810	0.3872	N/A	0.5582	**0.5882**	0.5645	**0.7129**	0	0.0769	0.0387	0.0194
Pendigits	**0.0890**	0.0658	N/A	0.0687	0.0622	0.0388	**0.1779**	0	0.0327	0.0133	0.0051

Table 3.4: Experiment 1: Overlapping points have larger fraction of support vectors, i.e, Ratio 2 > Ratio 1. MMM performs substantially better than BSK. Thresholded EM can have reasonable precision for some (high) thresholds, but gives poor recall on many datasets.

	m	Error Rate 1	Error Rate 2	p-value
Iris	55	**0.0182**	0.0091 ± 0.0096	0.0150
Ionosphere	103	**0.4272**	0.3524 ± 0.0465	0.0007
Vowel	89	**0.1461**	0.1213 ± 0.0197	0.0032
Wdbc	182	**0.0934**	0.0357 ± 0.0074	0.0000
Pima	209	**0.2727**	0.2263 ± 0.0293	0.0007
Segment	1336	**0.0314**	0.0283 ± 0.0035	0.0200
Landsat	3580	**0.1422**	0.1100 ± 0.0040	0.0000
Pendigits	2875	0.0132	**0.018** ± 0.0021	0.0000

(a) Overlapping clustering algorithm

	m	Error Rate 1	Error Rate 2	p-value
Iris	12	0.0000	0.0000	N/A
Ionosphere	163	0.2638	**0.3429** ± 0.0357	0.0001
Vowel	77	0.0130	**0.1247** ± 0.0282	0.0000
Wdbc	6	**0.5000**	0.0833 ± 0.1179	0.0000
Pima	54	0.0370	**0.2593** ± 0.0611	0.0000
Segment	489	**0.0470**	0.0327 ± 0.0079	0.0003
Landsat	1018	0.1071	0.1050 ± 0.0043	0.1662
Pendigits	1459	**0.0295**	0.0192 ± 0.0026	0.0000

(b) BSK algorithm

Table 3.5: Experiment 2: Overlapping and noisy points have higher error rates in prediction, i.e., Error Rate 1 > Error Rate 2. MMM performs better than BSK.

	Thresholded EM											
	0.01				0.1				0.2			
	m	Error Rate 1	Error rate 2	p-value	m	Error Rate 1	Error rate 2	p-value	m	Error Rate 1	Error rate 2	p-value
Iris	29	**0.0345**	0.0069 ± 0.0145	0.0002	16	0.0000	**0.0063** ± 0.0198	0.3434	11	0.0000	**0.0364** ± 0.0469	0.0368
Ionosphere	13	**0.5383**	0.3692 ± 0.1076	0.0008	7	**0.5714**	0.3286 ± 0.0964	0.0000	3	0.3333	**0.5333** ± 0.2811	0.0510
Vowel	148	**0.1486**	0.1311 ± 0.0169	0.0095	46	0.1304	0.1261 ± 0.0521	0.7976	27	0.1111	0.1407 ± 0.0737	0.2353
Wdbc	15	**0.3333**	0.0267 ± 0.0344	0.0000	10	**0.5000**	0.0100 ± 0.0316	0.0000	9	**0.5556**	0.0222 ± 0.0468	0.0000
Pima	329	**0.2614**	0.2240 ± 0.0162	0.0000	137	**0.3504**	0.2204 ± 0.0228	0.0000	89	**0.3258**	0.2101 ± 0.0305	0.0000
Segment	83	**0.0482**	0.0253 ± 0.0144	0.0007	51	**0.0784**	0.0373 ± 0.0313	0.0024	33	**0.0909**	0.0091 ± 0.0287	0.0000
Landsat	249	**0.1767**	0.1048 ± 0.0132	0.0000	119	**0.1849**	0.1143 ± 0.0292	0.0000	62	**0.1452**	0.0887 ± 0.0382	0.0012
Pendigits	466	0.0129	**0.0170** ± 0.0031	0.0025	209	0.0048	**0.0234** ± 0.0080	0.0000	129	0.0078	**0.0171** ± 0.0095	0.0130

Table 3.6: Experiment 2 using Thresholded EM. Thresholded EM has similar performance with MMM when the threshold is small.

	Error Rate	Ratio 1	Ratio 2	p-value
Iris	0.0133 ± 0.0422	0.6405 ± 0.1092	N/A	N/A
Ionosphere	0.3914 ± 0.0632	**0.7384** ± 0.0755	0.6378 ± 0.1079	0.0590
Vowel	0.2077 ± 0.0451	**0.8510** ± 0.0764	0.7585 ± 0.1285	0.0683
Wdbc	0.0446 ± 0.0256	**0.6963** ± 0.0721	0.1783 ± 0.2097	0.0001
Pima	0.2355 ± 0.0374	**0.7482** ± 0.0725	0.6770 ± 0.0670	0.0172
Segment	0.0381 ± 0.0100	0.4250 ± 0.0385	0.3468 ± 0.1358	0.1526
Landsat	0.1320 ± 0.0169	**0.4711** ± 0.0158	0.2594 ± 0.0473	0.0000
Pendigits	0.0285 ± 0.0061	0.7372 ± 0.0184	**0.7907** ± 0.0709	0.0390

(a) Overlapping clustering algorithm

	Ratio 1	Ratio 2	p-value
Iris	0.9190 ± 0.0756	N/A	N/A
Ionosphere	0.4633 ± 0.0681	**0.6509** ± 0.1312	0.0024
Vowel	0.8254 ± 0.0279	**0.9806** ± 0.0415	0.0000
Wdbc	0.9945 ± 0.0089	0.8917 ± 0.1845	0.1182
Pima	0.9136 ± 0.0341	**0.9905** ± 0.0202	0.0001
Segment	**0.7925** ± 0.0143	0.6810 ± 0.1540	0.0484
Landsat	0.8444 ± 0.0214	0.8251 ± 0.0330	0.2004
Pendigits	**0.8703** ± 0.0126	0.7794 ± 0.0603	0.0006

(b) BSK algorithm

Table 3.7: Experiment 3: Pure points have higher predictive accuracy, i.e., Ratio 1 > Ratio 2. MMM performs better than BSK.

	Thresholded EM								
	0.01			0.1			0.2		
	Ratio 1	Ratio 2	p-value	Ratio 1	Ratio 2	p-value	Ratio 1	Ratio 2	p-value
Iris	0.8179 ± 0.0988	N/A	N/A	0.8990 ± 0.0899	N/A	N/A	0.9333 ± 0.0629	N/A	N/A
Ionosphere	0.9715 ± 0.0391	0.9495 ± 0.0700	0.3168	0.9854 ± 0.0237	0.9707 ± 0.0505	0.4450	0.9899 ± 0.0214	0.9929 ± 0.0226	0.7871
Vowel	0.7380 ± 0.0738	0.6469 ± 0.1961	0.1657	0.9233 ± 0.0306	0.8709 ± 0.1313	0.2187	0.9561 ± 0.0289	0.9225 ± 0.0645	0.1327
Wdbc	**0.9814** ± 0.0152	0.8217 ± 0.2097	0.0437	**0.9907** ± 0.0098	0.8217 ± 0.2097	0.0354	**0.9925** ± 0.0097	0.8217 ±0.2097	0.0328
Pima	**0.5927** ± 0.0623	0.5081 ± 0.0978	0.0756	**0.8542** ± 0.0685	0.7355 ± 0.1196	0.0301	0.8974 ± 0.0539	0.8437 ±0.0833	0.1294
Segment	0.9654 ± 0.0069	0.9394 ± 0.0834	0.3565	0.9793 ± 0.0060	0.9465 ± 0.0709	0.1641	0.9865 ± 0.0047	0.9662 ±0.0591	0.2867
Landsat	**0.9661** ± 0.0056	0.9302 ± 0.0298	0.0029	**0.9844** ± 0.0055	0.9632 ± 0.0245	0.0249	0.9912 ± 0.0042	0.9857 ±0.0136	0.2169
Pendigits	0.9570 ± 0.0052	**0.9794** ± 0.0199	0.0100	0.9805 ± 0.0028	**0.9971** ± 0.0090	0.0005	0.9880 ± 0.0027	**0.9971** ±0.0090	0.0202

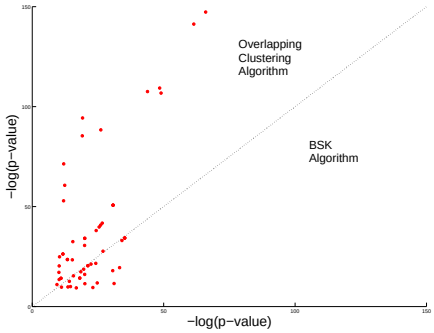
Table 3.8: Experiment 3 using Thresholded EM. The p-value is relatively large for some datasets.

	Ratio 1	Ratio 2 (Precision)					Ratio 3 (Recall)				
		MMM	Kernelized MMM				MMM	Kernelized MMM			
			$0.1n$	$0.2n$	$0.3n$	$0.4n$		$0.1n$	$0.2n$	$0.3n$	$0.4n$
Iris	0.34	**0.7917**	0.6538	0.6071	0.5862	0.5862	**0.3725**	0.3333	0.3333	0.3333	0.3333
Ionosphere	0.8803	0.9266	**0.9621**	0.9535	0.9520	0.9520	0.3269	**0.4110**	0.3981	0.3851	0.3851
Vowel	0.8864	**0.9881**	0.9134	0.9128	0.9171	0.9202	0.1774	0.2479	0.3803	0.4252	**0.4679**
Wdbc	0.2091	0.4198	**0.7500**	0.6250	0.7000	0.7000	**0.4622**	0.0756	0.0840	0.1176	0.1176
Pima	0.5664	0.7477	**1.0000**	**1.0000**	**1.0000**	**1.000**	**0.3678**	0.0138	0.0161	0.0184	0.0184
Segment	0.2801	0.3336	**0.3755**	0.3723	0.3716	0.3713	0.6569	0.7063	0.7141	**0.7156**	**0.7156**
Landsat	0.3111	0.3709	**0.4537**	0.4495	0.4481	0.4478	**0.6903**	0.6269	0.6354	0.6359	0.6359
Pendigits	0.1433	0.1548	0.1696	0.1697	**0.1700**	**0.1700**	0.2616	0.4025	0.4032	**0.4044**	**0.4044**

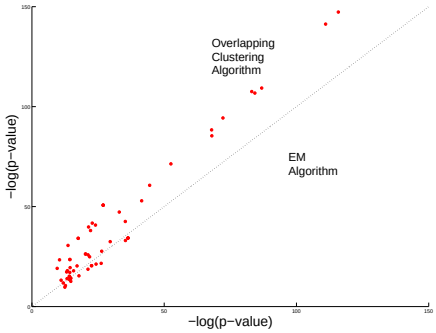
Table 3.9: Experiment 1 using both kernelized and unkernelized MMM on z-scored datasets. Kernelized MMM is run with several choices of D as fraction of the dataset size n. Kernelized MMM often has higher precision and maintains higher or similar recall on most of the datasets.

Kernels	# Significant Anno.	BSK	Unkernelized Algo.
$\exp(-\frac{\|\boldsymbol{x}-\mathbf{y}\|^2}{250})$	107	67% (43/64)	67% (48/72)
$\exp(-\frac{\|\boldsymbol{x}-\mathbf{y}\|^2}{500})$	109	68% (43/63)	63% (54/86)
$\exp(-\frac{\|\boldsymbol{x}-\mathbf{y}\|^2}{750})$	101	75% (42/56)	60% (49/82)

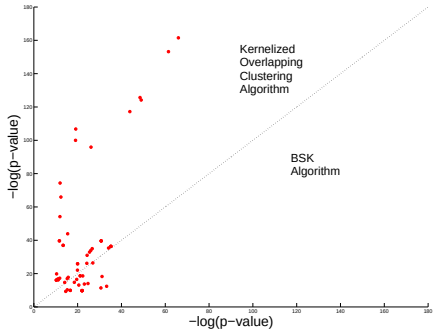
Table 3.10: Kernelized overlapping algorithm consistently performs better than the two baselines in terms of enrichment. For the fractions (a/b), b is the number of common significant annotations, and a is the number of times the kernelized algorithm performs better.



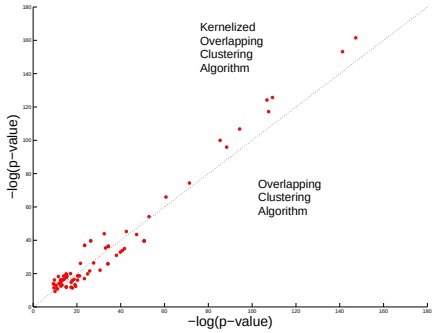
(a) Overlap vs BSK



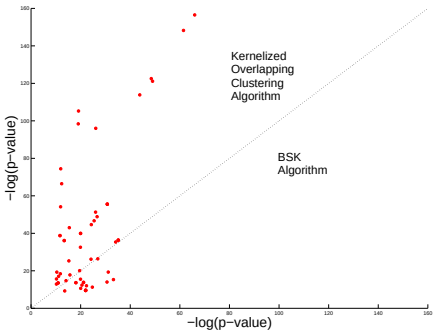
(b) Overlap vs EM



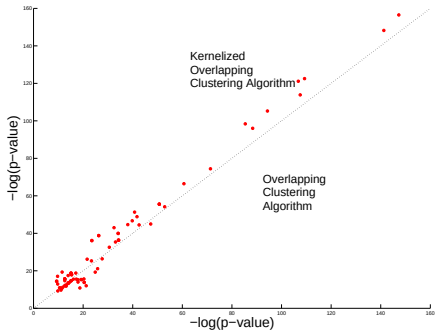
(c) Kernel ($\sigma^2 = 250$) vs BSK



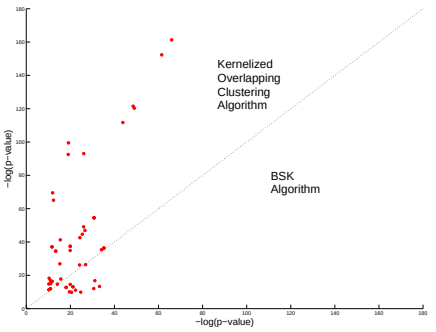
(d) Kernel ($\sigma^2 = 250$) vs Overlap



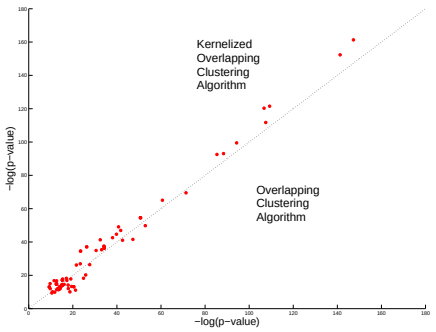
(e) Kernel ($\sigma^2 = 500$) vs BSK



(f) Kernel ($\sigma^2 = 500$) vs Overlap



(g) Kernel ($\sigma^2 = 750$) vs BSK



(h) Kernel ($\sigma^2 = 750$) vs Overlap

Figure 3.5: Scatter plot of the negative log p-value of the common significant annotations discovered by all the algorithms. Both the kernelized and unkernelized overlapping clustering algorithms perform consistently better than the baseline algorithms.

Chapter 4

Bayesian Overlapping Subspace Clustering

In this Chapter, we present a Bayesian Overlapping Subspace Clustering algorithm which can find potentially overlapping sub-blocks, automatically detect the background noise and naturally handle matrices with missing values. We propose the Bayesian Overlapping Subspace Clustering model in Section 4.1 and present an EM-style algorithm to learn the sub-block assignments in Section 4.2. The experimental results on both simulated and real datasets are presented in Section 4.3.

4.1 Bayesian Overlapping Subspace Clustering Model

The proposed Bayesian Overlapping Subspace Clustering model assumes that the number of sub-blocks k is given as an input. Each sub-block is modeled using a parametric distribution $p(\cdot|\boldsymbol{\theta}_j), [j]_1^k$ $([j]_1^k \equiv j = 1, \ldots, k)$ from any suitable exponential family. The noise entries are modeled using another distribution $p(\cdot|\boldsymbol{\theta}_{k+1})$ from the same family. However, the generative model for the observed data matrix is rather different from traditional mixture models [73] as well as the more recent mixed membership models such as LDA [16, 45].

Suppose the data matrix X has m rows and n columns, possibly with several missing entries. The main idea behind the proposed model is as follows: Each row u and each column v respectively have k-dimensional latent bit vectors $\boldsymbol{z}_r^u$ and $\boldsymbol{z}_c^v$ which indicate

their sub-block memberships. The sub-block membership for any entry x_{uv} in the matrix is obtained by an element-wise (Hadamard) product of the corresponding row and column bit vectors, i.e., $\boldsymbol{z} = \boldsymbol{z}_r^u \odot \boldsymbol{z}_c^v$. Given the sub-block membership $\boldsymbol{z}$ and the block distributions, the actual observation x_{uv} is assumed to be generated by a multiplicative mixture model so that

$$p(x_{uv}|\boldsymbol{z}_r^u, \boldsymbol{z}_c^v, \Theta) = \begin{cases} \frac{1}{c(\boldsymbol{z})} \prod_{j=1}^{k} p_j(x_{uv}|\boldsymbol{\theta}_j)^{z_j} & \text{if } \boldsymbol{z} \neq \mathbf{0} \ , \\ p(x_{uv}|\boldsymbol{\theta}_{k+1}) & \text{otherwise} \ , \end{cases} \tag{4.1}$$

where $c(\boldsymbol{z})$ is a normalization factor to guarantee that $p(\cdot|\boldsymbol{z}_r^u, \boldsymbol{z}_c^v, \Theta)$ is a valid distribution. If $\boldsymbol{z} = \boldsymbol{z}_r^u \odot \boldsymbol{z}_c^v = \mathbf{0}$, the all zeros vector, then x_{uv} is assumed to be generated from the noise component $p(\cdot|\boldsymbol{\theta}_{k+1})$. In the sequel, we will use $[\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^v = \mathbf{0}]$ to denote the indicator of this event. Figure 2.1 shows an example of a matrix generated from such a model with two dense blocks. The Hadamard product based generation ensures that the matrix has uniform/dense sub-blocks with possible overlaps while treating certain rows/columns as noise.

Since it can be tricky to work directly with latent bit vectors, we introduce suitable Bayesian priors on the sub-block memberships. In particular, the proposed model assumes that there are k Beta distributions $\text{Beta}(\alpha_r^j, \beta_r^j), [j]_1^k$ corresponding to the rows and k Beta distributions $\text{Beta}(\alpha_c^j, \beta_c^j), [j]_1^k$ corresponding to the columns. Let $\pi_r^{u,j}$ denote the Bernoulli parameter sampled from $\text{Beta}(\alpha_r^j, \beta_r^j)$ for row u and sub-block j where $[u]_1^m$ and $[j]_1^k$. Similarly, let $\pi_c^{v,j}$ denote the Bernoulli parameter sampled from $\text{Beta}(\alpha_c^j, \beta_c^j)$ for column v and sub-block j, where $[v]_1^n$ and $[j]_1^k$. The Beta-Bernoulli distributions are assumed to be the priors for the latent row and column membership vectors $\boldsymbol{z}_r^u$ and $\boldsymbol{z}_c^v$.

The proposed model is shown as a plate diagram in Figure 4.1. In particular, the generative process is as follows:

1. For each co-cluster $[j]_1^k$:
   (a) For each row $u, [u]_1^m$:
      (i) sample $\pi_r^{u,j} \sim \text{Beta}(\alpha_r^j, \beta_r^j)$,
      (ii) sample $z_r^{u,j} \sim \text{Bernoulli}(\pi_r^{u,j})$.
   (b) For each column $v, [v]_1^n$:

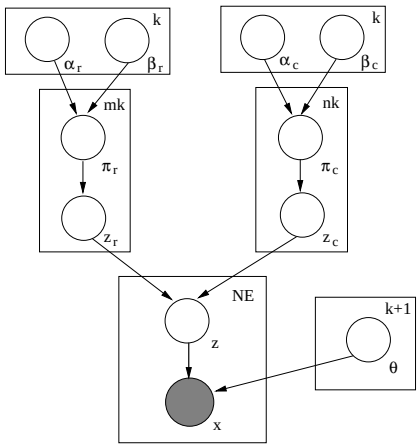


Figure 4.1: Bayesian Overlapping Subspace Clustering model. $\boldsymbol{z} = \boldsymbol{z}_r \odot \boldsymbol{z}_c$. NE is the number of non-missing entries in the matrix.

(i) sample $\pi_c^{v,j} \sim \text{Beta}(\alpha_c^j, \beta_c^j)$,

(ii) sample $z_c^{v,j} \sim \text{Bernoulli}(\pi_c^{v,j})$.

2. For each (non-missing) matrix entry $x_{uv}, [u]_1^m [v]_1^n$, sample

$$x_{uv} \sim \begin{cases} \frac{1}{c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^v)} \prod_{j=1}^k p_(x_{u,v}|\theta_j, \boldsymbol{z}_r^{u,j}, \boldsymbol{z}_c^{v,j}) & \text{if } z_r^u \odot \boldsymbol{z}_c^v \neq \mathbf{0} \ , \\ p(x_{u,v}|\theta_{k+1}) & \text{otherwise} \ . \end{cases}$$

Since only the observed entries in the matrix are assumed to be generated by the above process, the model naturally handles matrices with missing values.

Note that our Bayesian Overlapping Subspace Clustering model is different from the co-clustering framework [47, 94], which defines a co-cluster as the entries belonging to both a row and column cluster. Thus the rows of a co-cluster are restricted to those of the corresponding row cluster and the columns of a co-cluster are restricted to those of the corresponding column cluster. However, our model gives the algorithm much more flexibility to assign rows and columns to dense sub-blocks by modeling the blocks directly.

4.2 Analysis and Algorithm

Let Π_r and Π_c be $m \times k$ and $n \times k$ latent matrices which have the Bernoulli parameters for each row and column, Z_r and Z_c be $m \times k$ and $n \times k$ matrices that have the latent row and column sub-block assignments for each row and column. For convenience of notation, let ς_{uv} be a indicator variable for observed entries in the matrix, i.e., $\varsigma_{uv} = 1$ if entry x_{uv} is not missing, and 0 otherwise. Then the joint distribution over all observed and latent variables is given by

$$p(X, Z_r, Z_c, \Pi_r, \Pi_c | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta) = p(\Pi_r | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) p(\Pi_c | \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c) p(Z_r | \Pi_r) p(Z_c | \Pi_c) p(X | \Theta, Z_r, Z_c) \ .$$

Since the observations are statistically independent given Z_r, Z_c, we have

$$p(X | \Theta, Z_r, Z_c) = \prod_{u=1}^{m} \prod_{v=1}^{n} p(x_{uv} | \Theta, \boldsymbol{z}_r^u, \boldsymbol{z}_c^v)^{\varsigma_{uv}} \ . \tag{4.2}$$

Marginalizing over all latent variables, the conditional probability of generating the matrix X given the parameters $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$ is given by

$$\begin{aligned} & p(X | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta) \\ & = \int_{\Pi_r, \Pi_c} \sum_{Z_r, Z_c} p(X, Z_r, Z_c, \Pi_r, \Pi_c | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta) d\Pi_r d\Pi_c \ . \end{aligned} \tag{4.3}$$

Π_r and Π_c are integrated out in (4.3) because of conjugacy : they are generated by Beta distributions which are conjugate priors to Bernoulli distributions which generate Z_r and Z_c. Thus (4.3) does not depend on Π_r and Π_c. It is also important to note the conditional probability of observing X as in (4.3) is not the product of the conditional probability of observing each entry, i.e.,

$$p(X | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta) \neq \prod_{u=1}^{m} \prod_{v=1}^{n} p(x_{u,v} | \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)^{\varsigma_{uv}} \ .$$

The equality does not hold because the entries in the matrix are not conditionally independent given the parameters $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$. According to the generative process, $\boldsymbol{z}_r^u$ and $\boldsymbol{z}_c^v$ are sampled only once for each row and each column, so that the observations in the same row/column get coupled. Note that this is a crucial departure from several related mixture models which assume the joint probability of all observations to be simply a product of the marginal probabilities.

Given the entire matrix X, the learning task is to compute the model parameters $(\boldsymbol{\alpha}_r^\star, \boldsymbol{\beta}_r^\star, \boldsymbol{\alpha}_c^\star, \boldsymbol{\beta}_c^\star, \Theta^\star)$ which maximize $\log p(X|\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$. A general approach is to use expectation maximization (EM) algorithm [74]. However, direct calculation of $\log p(X|\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$ is intractable, indicating that a direct application of EM is not possible. In this section, we propose an EM-like algorithm alternating between approximate inference and optimal parameter estimation to tackle the learning task. In the E-step, since (4.3) does not depend on Π_r and Π_c, we approximate the expectation $E[\log p(X, Z_r, Z_c|\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)]$ with respect to the posterior probability of (Z_r, Z_c). In the M step, we estimate parameters $(\boldsymbol{\alpha}_r^\star, \boldsymbol{\beta}_r^\star, \boldsymbol{\alpha}_c^\star, \boldsymbol{\beta}_c^\star, \Theta^\star)$ which maximize the expectation. We iterate between the E- and M-steps until the algorithm converges.

4.2.1 Inference

In the E-step, given the model parameters $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$, the goal is to estimate the expectation of the log-likelihood $E[\log p(X, Z_r, Z_c|\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)]$ where the expectation is with respect to the posterior probability $p(Z_r, Z_c|X, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$. We use Gibbs sampling to approximate the expectation [45, 40]. In particular, we compute the conditional probabilities of each row variable $z_r^{u,j}$ and column variable $z_c^{v,j}$ and construct a Markov chain based on the conditional probabilities. On convergence, the chain will draw samples from the posterior joint distribution of (Z_r, Z_c), which in turn can be used to get an approximate estimate of the expected log-likelihood.

If $Z_r^{-(u,j)}$ denotes the binary matrix Z_r excluding $z_r^{u,j}$, the conditional probability of $z_r^{u,j} = 1$ is given by:

$$p(z_r^{u,j} = 1|Z_r^{-(u,j)}, Z_c, X, \Theta) \propto p(X|Z_r, Z_c, \Theta)p(z_r^{u,j} = 1|Z_r^{-(u,j)}) \ ,$$

where $p(X|Z_r, Z_c, \Theta)$ is as in (4.2) and

$$p(z_r^{u,j} = 1|Z_r^{-(u,j)}) = \int_0^1 p(z_r^{u,j} = 1|\pi_r^{u,j})p(\pi_r^{u,j})d\pi_r^{u,j} = \frac{\alpha_r^j}{\alpha_r^j + \beta_r^j} \ . \tag{4.4}$$

Note that (4.4) does not include the count on how many rows are assigned to sub-block j and hence has a different form compared to that for LDA [45], because the Bernoulli

parameters Π_r and Π_c are sampled once for each row and column. Now,

$$
\begin{aligned}
& p(z_r^{u,j} = 1 | Z_r^{-(u,j)}, Z_c, X, \Theta) \\
& \propto \prod_{p=1}^{m} \prod_{q=1}^{n} p(x_{p,q} | \boldsymbol{z}_r^p, \boldsymbol{z}_c^q, \Theta)^{\varsigma_{pq}} . \frac{\alpha_r^j}{\alpha_r^j + \beta_r^j} \ , && (4.5) \\
& \propto \prod_{q=1}^{n} p(x_{u,q} | \boldsymbol{z}_r^u, \boldsymbol{z}_c^q, \Theta)^{\varsigma_{uq}} . \frac{\alpha_r^j}{\alpha_r^j + \beta_r^j} \ , && (4.6) \\
& \propto \prod_{q=1}^{n} \left(\frac{p(x_{u,q} | \boldsymbol{\theta}_j)^{z_c^{q,j}} . p(x_{u,q} | \boldsymbol{\theta}_{k+1})^{[\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q = \mathbf{0}]}}{c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q)} \right)^{\varsigma_{uq}} . \frac{\alpha_r^j}{\alpha_r^j + \beta_r^j} \ , && (4.7)
\end{aligned}
$$

where (4.6) follows since the probability of generating the entries in the rows except u does not depend on the value of $z_r^{u,j}$, and (4.7) follows since whether the entry $x_{u,q}$ belongs to sub-blocks other than j does not play a role in deciding the value of $z_r^{u,j}$ in the product term other than the overall normalization term $c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q)$.

The probability of $z_r^{u,j} = 0$ can be derived similarly as

$$
p(z_r^{u,j} = 0 | Z_r^{-(u,j)}, Z_c, X, \Theta) \propto \prod_{q=1}^{n} \left(\frac{p(x_{u,q} | \boldsymbol{\theta}_{k+1})^{[\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q = \mathbf{0}]}}{c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q)} \right)^{\varsigma_{uq}} . \frac{\beta_r^j}{\alpha_r^j + \beta_r^j} \ . \tag{4.8}
$$

Following the same argument, we derive the probability of $z_c^{v,j}$ as:

$$
p(z_c^{v,j} = 1 | Z_r, Z_c^{-(v,j)}, X, \Theta) \propto \prod_{p=1}^{m} \left(\frac{p(x_{p,v} | \boldsymbol{\theta}_j)^{z_r^{p,j}} . p(x_{p,v} | \boldsymbol{\theta}_{k+1})^{[\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q = \mathbf{0}]}}{c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q)} \right)^{\varsigma_{pv}} . \frac{\alpha_c^j}{\alpha_c^j + \beta_c^j} \ , \tag{4.9}
$$

and

$$
p(z_c^{v,j} = 0 | Z_r, Z_c^{-(v,j)}, X, \Theta) \propto \prod_{p=1}^{m} \left(\frac{p(x_{p,v} | \boldsymbol{\theta}_{k+1})^{[\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q = \mathbf{0}]}}{c(\boldsymbol{z}_r^u \odot \boldsymbol{z}_c^q)} \right)^{\varsigma_{pv}} . \frac{\beta_c^j}{\alpha_c^j + \beta_c^j} \ . \tag{4.10}
$$

The true underlying posterior distribution of (Z_r, Z_c) may have multiple modes. To prevent the sampling algorithm from getting stuck in local modes, we modify the Gibbs sampler using simulated annealing [58]. Given a temperature parameter T, the sampling

is done following

$$p^T(z_r^{u,j}=0|\cdots)=\frac{p(z_r^{u,j}=0|\cdots)^{\frac{1}{T}}}{p(z_r^{u,j}=0|\cdots)^{\frac{1}{T}}+p(z_r^{u,j}=1|\cdots)^{\frac{1}{T}}}\ ,$$
$$p^T(z_r^{u,j}=1|\cdots)=\frac{p(z_r^{u,j}=1|\cdots)^{\frac{1}{T}}}{p(z_r^{u,j}=0|\cdots)^{\frac{1}{T}}+p(z_r^{u,j}=1|\cdots)^{\frac{1}{T}}}\ .$$

When T is high, the probability distribution is almost uniform, and when T is low, more emphasis is given to high probability states. In practice, we start with a relatively high T and gradually decrease T to 1, when the sampling distribution is exactly the posterior distribution of Z_r and Z_c.

The sampling is run for enough iterations till it converges. Then we sample from the stationary distribution (with suitable gaps) to obtain N independent and identically distributed samples of (Z_r, Z_c), where N is a predefined large number. From the samples, the expectation of the log-likelihood can be empirically estimated as: $\frac{1}{N}\sum_{s=1}^{N}\log p(X, Z_{r,s}, Z_{c,s}|\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)$, where $Z_{r,s}$ and $Z_{c,s}$ correspond to the s^{th} samples.

4.2.2 Estimation

In M-step, we estimate $(\boldsymbol{\alpha}_r^\star, \boldsymbol{\beta}_r^\star, \boldsymbol{\alpha}_c^\star, \boldsymbol{\beta}_c^\star, \Theta^\star)$ which maximizes the expectation. Note that, given Z_r and Z_c, each entry in the matrix is statistically independent of each other. So the parameter estimation problem can be formulated as maximizing the following expected log-likelihood objective function:

$$\begin{aligned}
&L(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c, \Theta)\\
&=\sum_{s=1}^{N}\log p(X, Z_{r,s}, Z_{c,s}|\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_c, \Theta)\\
&=\sum_{s=1}^{N}\log p(Z_{r,s}|\boldsymbol{\alpha}_r)+\sum_{s=1}^{N}\log p(Z_{c,s}|\boldsymbol{\alpha}_c)+\sum_{s=1}^{N}\log p(X|Z_{r,s}, Z_{c,s}, \Theta)\\
&=\sum_{s=1}^{N}\sum_{u=1}^{m}\sum_{j=1}^{k}\log p(z_{r,s}^{u,j}|\alpha_r^j)+\sum_{s=1}^{N}\sum_{v=1}^{n}\sum_{j=1}^{k}\log p(z_{c,s}^{v,j}|\alpha_c^j)+\sum_{s=1}^{N}\sum_{u=1}^{m}\sum_{v=1}^{n}\varsigma_{uv}\log p(x_{u,v}|\boldsymbol{z}_{r,s}^u, \boldsymbol{z}_{c,s}^v, \Theta)\ .
\end{aligned}\tag{4.11}$$

The optimization can be broken into two independent parts—over the parameters $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r, \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c)$ of the Beta distributions, and over the natural parameters Θ of the exponential family distributions.

A direct calculation based on taking derivatives w.r.t. $(\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_c)$ and setting them to $\mathbf{0}$ shows that only the ratios $\frac{\boldsymbol{\alpha}_r}{\boldsymbol{\beta}_r}$ and $\frac{\boldsymbol{\alpha}_c}{\boldsymbol{\beta}_c}$ matter. So we set $\boldsymbol{\beta}_r = \boldsymbol{\beta}_c = \mathbb{1}$, the all ones vector, and only update $\boldsymbol{\alpha}_r$ and $\boldsymbol{\alpha}_c$ in each iteration. The optimal $(\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_c)$ parameters satisfy the following equation:

$$\alpha_r^{j\star} = \frac{\sum_{s=1}^{N}\sum_{u=1}^{m} z_{r,s}^{u,j}}{Nm - \sum_{s=1}^{N}\sum_{u=1}^{m} z_{r,s}^{u,j}} \ , \tag{4.12}$$

$$\alpha_c^{j\star} = \frac{\sum_{s=1}^{N}\sum_{v=1}^{n} z_{c,s}^{v,j}}{Nn - \sum_{s=1}^{N}\sum_{v=1}^{n} z_{c,s}^{v,j}} \ , \tag{4.13}$$

where $j = 1, \ldots, k$.

A distribution is in the exponential family if the density function with respect to a base measure can be written in the form:

$$p(x|\boldsymbol{\theta}) = \frac{dP(x|\boldsymbol{\theta})}{dP_0(x)} = \exp\{s(x)^T\boldsymbol{\theta} - \psi(\boldsymbol{\theta})\} \ , \tag{4.14}$$

where $\boldsymbol{\theta}$ is the natural parameter, $s(x)$ is the sufficient statistic, and $\psi(\boldsymbol{\theta})$ is the cumulant function, which is a convex function of Legendre type [87]. Using (4.14) and ignoring the noise component for simplicity, the last term of the objective function (4.11) can be rewritten in a function of Θ:

$$f(\Theta) = \sum_{s=1}^{N}\sum_{u=1}^{m}\sum_{v=1}^{n} \varsigma_{uv} s(x_{uv})^T \sum_{j=1}^{k} z_{r,s}^{u,j} z_{c,s}^{v,j} \boldsymbol{\theta}_j - \sum_{s=1}^{N}\sum_{u=1}^{m}\sum_{v=1}^{n} \varsigma_{uv} \psi\left(\sum_{j=1}^{k} z_{r,s}^{u,j} z_{c,s}^{v,j} \boldsymbol{\theta}_j\right) \ . \tag{4.15}$$

Computing the second derivative of $f(\Theta)$ with respect to $\boldsymbol{\theta}_j$, we have

$$\frac{\partial^2 f(\Theta)}{\partial \boldsymbol{\theta}_j^2} = -\sum_{s=1}^{N}\sum_{u=1}^{m}\sum_{v=1}^{n} \varsigma_{uv} z_{r,s}^{u,j} z_{c,s}^{v,j} \frac{\partial^2 \psi\left(\sum_{h=1}^{k} z_{r,s}^{u,h} z_{c,s}^{v,h} \boldsymbol{\theta}_h\right)}{\partial \boldsymbol{\theta}_j^2} \ ,$$

which is negative since ψ is a convex function so the second derivation of ψ is non-negative. Hence, $f(\Theta)$ is a concave function of each $\boldsymbol{\theta}_j$ (not necessarily jointly concave on Θ). We propose to use a co-ordinate descent algorithm for estimating Θ, where we optimize only one parameter while keeping the others fixed [102]. In order to find the

maximizer $\boldsymbol{\theta}_j^*$ given all the other parameters $\boldsymbol{\theta}_h, h \neq j$, taking gradient and setting it to 0, we obtain

$$\boldsymbol{\theta}_j^\star = \sum_{\varsigma_{uv} z_{r,s}^{u,j} z_{c,s}^{v,j}=1} \sum_{\substack{h=1 \\ h\neq j}}^{k} z_{r,s}^{u,h} z_{c,s}^{v,h} \boldsymbol{\theta}_h + (\nabla\psi)^{-1} \left(\sum_{\varsigma_{uv} z_{r,s}^{u,j} z_{c,s}^{v,j}=1} s(x_{uv}) \right) . \tag{4.16}$$

Since ψ is a Legendre function, the function $(\nabla\psi)^{-1}$ will be well defined and equal to $\nabla\phi$, where $\phi = \psi^*$, the conjugate of the cumulant function ψ. Hence, the above updates can be efficiently implemented.

4.3 Experimental Results

In this section, we present experimental results on simulated datasets, a microarray gene expression dataset and a movie recommendation dataset. First, we introduce some additional notation to be used in this section: T_{start} denotes the initial temperature parameter in simulated annealing, $f_T < 1$ denotes the multiplicative factor by which the temperature goes down every I_T iterations; and N is the number of samples drawn from the stationary distribution.

Since we obtain several samples from the Markov chain after it converges, the final row and column sub-block assignments are decided by the 'vote for majority' fashion : if a row/column belongs to a sub-block in more than half of the samples, we consider the row/column belongs to that corresponding sub-block.

The implementation of the baseline algorithms [78, 94, 33] are kindly provided by the corresponding authors. The implementation of [65] is downloaded from the author's webpage.

4.3.1 Simulated Datasets

We first do experiment on two simulated datasets which are easy to visualize. Both datasets are in the form of a 200×100 matrix, whose entries are initially sampled from a background noise distribution. For the first dataset D_1, we introduce 3 non-overlapping uniform blocks (normally distributed with different means) to replace certain sub-blocks in the matrix (Figure 4.2(a)). The actual dataset is obtained by randomly permuting

the rows and columns of the matrix, so that the block structure is not apparent (Figure 4.2(b)). For the second dataset D_2, we introduce 4 mildly overlapping dense blocks where the overlapping entries are generated from the multiplicative model in (4.1) (Figure 4.3(a)). As before, the actual dataset is obtained by a random row/column permutation (Figure 4.3(b)). We also do experiment on two other simulated datasets with larger number of sub-blocks, one with 10 blocks and the other with 15 blocks. And we do not provide label information to STATPC on these two datasets.

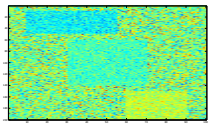
We compare the performance of the proposed algorithm to a state-of-the-art subspace clustering algorithm called STATPC [78] and an overlapping clustering algorithm [33], which we call MMM algorithm. STATPC finds non-redundant and statistically (overlapping) regions in high dimensional data. MMM finds overlapping clusters and automatically detects the noisy data points. To make the three algorithms comparable, MMM is used to cluster the entries in the matrix, instead of the rows. STATPC can make use of the row cluster labels if available—we give it substantial advantage by providing the true cluster labels for all the rows. For D_1, since there is no overlap, we provide the true cluster label. For D_2, we provide the true cluster labels for the non-overlapping rows, and one of the true cluster labels for the overlapping rows. On the contrary, the proposed algorithm does not use any form of supervision. In particular, we run kmeans on the matrix entries to get the initial estimate of the component parameter values for BOSC and MMM—in this case, means and standard deviations of each Gaussian component. However, kmeans does not capture the structure of the matrix, because it rarely assigns entries to the correct sub-blocks. The noise component is initialized with the mean and standard deviation across all entries in the matrix. We use $T_{start} = 10$, $f_T = 0.67$, $I_T = 50$ and $N = 50$. We do the experiment three times on each dataset. Since we get the same results, we only report one set of results.

We quantitatively measure the performance by calculating the clustering accuracy (Table 4.1). We consider an output sub-block corresponds to a ground truth sub-block if that ground truth sub-block is the majority in that output sub-block. The overall clustering accuracy is computed along all the output sub-blocks generated by each algorithm. In particular, suppose c_1 is the number of ground truth sub-blocks, c_2 is the number of output sub-blocks, and $a_{ij}, [i]_1^{c_1}[j]_1^{c_2}$ is the number of entries from the i^{th} ground truth block that are also in the j^{th} output block. The clustering accuracy is

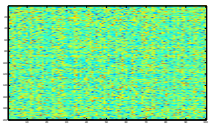
defined as :

$$\text{Clustering Accuracy} = \frac{\sum_{j=1}^{c2} \max_i a_{ij}}{\sum_{i=1}^{c1} \sum_{j=1}^{c2} a_{ij}} \ .$$

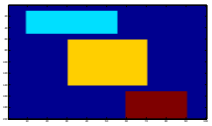
After the algorithms terminate, we re-permute the rows and columns back to the original order and see if the sub-block structure has been discovered. The results of the first two datasets for BOSC and STATPC are shown in Figure 4.2(c)-(d) and Figure 4.3(c)-(d). For D_1, the proposed algorithm can accurately find the three uniform blocks in the matrix; STATPC can approximately detect the three uniform blocks from the rows (where labels are provided) but dose not find the correct columns for each block. For D_2, our algorithm can still find the correct structure except for a few mistakes in the third co-cluster; however, STATPC, does not find meaningful block structures for most of the rows and almost all of the columns. We can see from Table 4.1 that MMM performs poorly, because it doesn't take the correlation between rows and columns into account, which also highlights the motivation of our work.



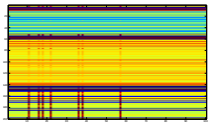
(a) Original



(b) After Permutation



(c) Results: BOSC

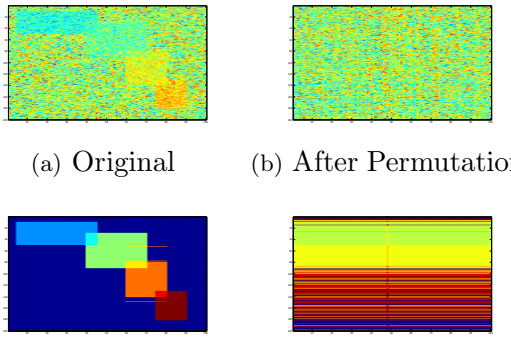


(d) Results: STATPC

Figure 4.2: Results on D_1: BOSC finds the correct structure along rows and columns, whereas STATPC finds a reasonable structure only along rows.

Accuracy	Dataset 1	Dataset 2	Dataset 3	Dataset 4
BOSC	**1**	**0.8959**	**0.6813**	**0.5723**
STATPC	0.7767	0.4786	0.2670	0.1286
MMM	0.5888	0.5287	0.4560	0.3549

Table 4.1: Clustering accuracy of BOSC, STATPC and MMM on four simulation datasets.



(a) Original (b) After Permutation

(c) Results: BOSC (d) Results: STATPC

Figure 4.3: Results on D_2: BOSC mostly finds the correct structure along rows and columns, whereas STATPC does not get a reasonable structure.

4.3.2 Microarray Gene Expression Dataset

The microarray gene expression dataset we use consists of 4062 genes and 215 experimental conditions [77]. We first select 1000 genes that have the highest variance of expressions over the 215 conditions. We run our algorithm on this 1000×215 matrix and want to find subsets of genes which highly co-express under subsets of conditions. The number of sub-blocks is set to be 30. The annealing parameters are set as follows: $T_{start} = 500$, $f_T = 0.67$, $I_T = 75$ and $N = 100$. As a strong baseline, we use the Plaid bi-clustering algorithm [65] which has been extensively used for gene-expression analysis. The Plaid algorithm finds overlapping dense regions in gene-expression datasets. We also compare our algorithm with a model-based overlapping co-clustering (MOC) algorithm [93] and the state-of-the-art Bayesian co-clustering (BOC) algorithm [94].

To evaluate whether the dense blocks identified are meaningful from a biological perspective, we check if the genes in each dense block show significant enrichment for known biological process annotations. We make use of Gene Ontology Term Finder[1] online tool, which searches for shared annotations given a set of genes and computes an associated p-value. The p-value measures the probability of observing a group of genes to be annotated with a certain annotation purely by chance. If genes in a dense block indeed correspond to known biological processes, we would expect a low p-value.

[1] http://www.yeastgenome.org/cgi-bin/GO/goTermFinder.pl

We consider an annotation to be significant if the p-value associated with it is less than 10^{-4}.

We initialize all algorithms by running kmeans on matrix entries. For co-clustering algorithms, we try different combinations of row/colum cluster numbers and report the best results. The BOC algorithm estimates for each row/column the probability of belonging to each row/column cluster. We consider that a row/column belongs to a row/column cluster if it has the highest probability on that row/column cluster.

The result is listed in Table 4.2. BOSC identifies 24 blocks, 13 are found to have significant enrichment. Most of the other blocks have p-values that are of the order of 10^{-4}. In contrast, of the 30 'layers' found by the Plaid model, only 8 have significant enrichment. Among the 30 co-clusters find by the model-based overlapping co-clustering algorithm, 10 have significant enrichment. The Bayesian co-clustering algorithm also finds 10 blocks with significant enrichment.

Algorithms	BOSC	Plaid	MOC	BOC
Number of Blocks with Significant Enrichment	13	8	10	10

Table 4.2: BOSC finds more dense blocks with significant enrichment.

4.3.3 MovieLens Dataset

MovieLens[2] is a movie rating dataset with 100,000 ratings from 943 users on 1682 movies. The ratings are on a 1-5 scale. We work with a subset with 568 users who submitted at least 50 ratings and 603 movies which have at least 50 ratings. The resulting matrix has 73544 ratings and 79% missing entries. Since different users may have different standards and ratings can be very personal, we z-score the ratings submitted by each user. The annealing parameters are the same as those in the gene expression experiment and we report results with $k = 20$.

The blocks inferred by BOSC are qualitatively meaningful, typically consisting of sub-groups of users with similar ratings on subsets of movies. Most blocks are coherent and contain either good or bad movies. For example, we find that a sub-block with high ratings contains popular movies *Dances with Wolves*, *Titanic*, *The Graduate*, *The*

[2] http://www.grouplens.org/taxonomy/term/14

Bridges of Madison County and *Speed*; another sub-block with movies which are not well received has low ratings from a group of users: *Chain Reaction*, *Waterworld* and *The Phantom*.

The model also finds significant overlapping structure on movies. Overall, 177 movies are assigned to 1 sub-block, 219 movies are assigned to 2 sub-blocks, 108 movies belong to 3 sub-blocks , 41 movies belong to 4 or more sub-blocks and 58 movies do not belong to any sub-block. The overlapping structure is generally consistent with the reputation of the movies. First, critically acclaimed movies usually belongs to sub-blocks with high ratings. For example, *Shawshank Redemption* and *The Silence of the Lambs* belong to 4 sub-blocks with very high rating. *The Godfather* and *Schindler's List* belong to 3 sub-blocks with high ratings. Secondly, movies which are generally not well-received by critics, like *Striptease* and *The Beautician and the Beast* are both placed in 4 sub-blocks with very low ratings. Last, some controversial movies are placed in different sub-blocks with either high or low ratings. For example, *Lost Highway*, which gains some critical recognition but is not a box office success, is in two dense blocks, one with very low ratings and the other with very high ratings.

If we treat each genre as a cluster, the MovieLens dataset has naturally overlapping cluster structure, because each movie may have several genres. Following the methodology in [93], we then cerate 2 subsets from the dataset we use above. The first dataset contains 220 movies from 3 genres : animation, children's and comedy. The second dataset contains 225 movies from 3 genres : thriller, action and adventure. We run the BOSC algorithm with $k = 20$ on these 2 datasets trying to discover genres based on the belief that similarity in the user ratings gives an indication about whether the movies are in related genres. Since the BOC algorithm [94] can handle datasets with missing entries, we use it as the baseline. We initialize two algorithms by running kmeans on the matrix entries. We compare precision, recall and F-measure over pairs of movies. Two movies are in a pair if they belong to the same cluster/genre. Precision, recall and

F-measure are calculated as follows:

$$\text{Precision} = \frac{\text{Number of Correctly Identified Pairs}}{\text{Number of Identified Pairs}},$$
$$\text{Recall} = \frac{\text{Number of Correctly Identified Pairs}}{\text{Number of Pairs in the Original Dataset}},$$
$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

For the BOC algorithm, we try different combinations of the number of row clusters and column clusters and report the best result. We do the experiment on each dataset three times. Since we get the same results, we only report one set of results, which is listed in Table 4.3 and 4.4. Two algorithms have comparable performance on precision, but BOSC has higher recall and F-score.

Dataset1	BOSC	BOC
Precision	0.7722	**0.7727**
Recall	**0.6050**	0.3335
F-measure	**0.6784**	0.4659

Table 4.3: The performance of BOSC and BOC on the first dataset with animation, children's and comedy movies.

Dataset2	BOSC	BOC
Precision	0.6496	**0.6643**
Recall	**1**	0.5567
F-measure	**0.7876**	0.6058

Table 4.4: The performance of BOSC and BOC on the second dataset with thriller, action and adventure movies.

Part II

MAP Inference on Discrete Graphical Models

Chapter 5

Motivation and Related Work

In this Chapter, we emphasis the emerging need of salable MAP inference algorithm in Section 5.1. In Section 5.2, we formally define Markov random fields and review the existing MAP inference algorithms.

5.1 Motivation

Discrete graphical models have found applications in a wide variety of problems, including image analysis [40], speech recognition [54], bioinformatics [30] and error correcting codes [72]. Hidden Markov models (HMMs) [14], Markov random fields (MRFs) [103], and conditional random fields (CRFs) [64] are popular examples of discrete graphical models which have found widespread usage in data analysis.

Given a discrete graphical model with known structure and parameters, the problem of finding the most likely configuration of the states is known as the MAP inference problem. For a tree-structured graph, the problem can be solved efficiently using a suitable dynamic programming algorithm such as the max-product algorithm [63]. For example, for HMMs, the most likely sequence of latent states can be found efficiently using the Viterbi decoding algorithm [32]. For general graphs, however, the MAP inference problem is a computationally intractable integer program and is NP-hard [103]. Existing approaches to solving the general case often consider a linear programming (LP) relaxation of the integer program. Over the past few years, several algorithms have been proposed to solve such graph-structured LPs [67, 61, 75, 71]. Such approaches

can be broadly classified into two groups: primal methods which work with the original variables, of which the proximal approach is among the most efficient [85], and dual methods, which works on the dual variables, of which the dual decomposition approach is currently the most efficient [96].

In the context of large scale data analysis, the ability to apply such methods efficiently to graphs of over millions or hundreds of millions of nodes is important and necessary. Consider the problem of detecting droughts from precipitation data of the past 100 years at a temporal resolution of a month and spatial resolution of $0.5^\circ \times 0.5^\circ$ over land. A 3-dimensional MRF (latitude $\times$ longitude $\times$ time) with neighborhood dependencies is a suitable model for such analysis since droughts have both spatial and temporal continuity. Assuming a boolean indicator variable of drought at each space-time location, the graph-structured LP relaxation of the MAP inference problem in this context has to work with approximately 7 million variables and about double that many constraints. The key bottleneck, even in the advanced algorithms for solving graph-structured LPs, is that they are inherently sequential [85]. Given that climate datasets are available at much higher resolutions, especially from climate model outputs used by the Intergovernmental Panel on Climate Change (IPCC) for future climate projections, we need algorithms for solving graph structured LPs which efficiently scale to problem sizes of millions or hundreds of millions of nodes. Further, due to the generality of the framework, the algorithms can find applications in other domains, such as community detection in large scale social networks, which can have millions to hundreds of millions of users (nodes) [98].

Driven by the emerging need for scalable MAP inference algorithms, we propose a parallel alternating directions algorithm for solving graph-structured LPs. The overall structure of the algorithm is based on two ideas: tree-based decomposition of a graph-structured LP [67] and the alternating directions method of multipliers (ADMM) [17]. The tree decomposition breaks the problem into small but overlapping parts, each involving small number of variables and constraints. The algorithm iterates between doing updates to variables in individual parts in parallel followed by suitable aggregation, all within the framework of ADMM. However, unlike standard ADMM, we use a novel inexact ADM augmented by a Bethe entropy regularization, which we refer to as Bethe-ADMM. The unusual modification in Bethe-ADM leads to an efficient projection

of partial solutions to subsets of constraints, leading to highly efficient iterations, and avoids a double-loop algorithm. Through rigorous analysis, we establish correctness and convergence of the proposed Bethe-ADMM. We also discuss its efficient parallel MPI implementation.

5.2 Related Work

5.2.1 Pairwise Markov Random Fields

A pairwise MRF is defined on an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set. Each node $u \in V$ has a random variable X_u associated with it, which can take value x_u in some discrete space $\mathcal{X} = \{1, \ldots, k\}$. Concatenating all the random variables X_u, $\forall u \in V$, we obtain an n dimensional random vector $\boldsymbol{X} = \{X_u | u \in V\} \in \mathcal{X}^n$. We assume that the distribution P of $\boldsymbol{X}$ is a Markov random field [103], meaning that it factors according to the structure of the undirected graph G as follows: With $f_u : \mathcal{X} \mapsto \mathbb{R}$, $\forall u \in V$ and $f_{uv} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, $\forall (u, v) \in E$ denoting nodewise and edgewise potential functions respectively, the distribution takes the form:

$$P(\boldsymbol{x}) \propto \exp\left\{ \sum_{u \in V} f_u(x_u) + \sum_{(u,v) \in E} f_{uv}(x_u, x_v) \right\} . \tag{5.1}$$

We can also view the distribution defined on an MRF as an exponential family distribution with sufficient statistics $\phi(\boldsymbol{x})$ and natural parameter $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle) , \tag{5.2}$$

where $Z(\boldsymbol{\theta})$ is a normalization constant or the partition function. In a pairwise MRF, the sufficient statistics are restricted to be only over the nodes and edges of G. In the most general form, they are indicator functions denoting local assignments to nodes and edges. To be more specific, for each node $u \in V$, we have

$$\mathbb{I}_{u;j}(x_u) = \begin{cases} 1 & \text{if } x_u = j \\ 0 & \text{Otherwise} \end{cases} , \tag{5.3}$$

and for each edge $(u, v) \in E$, we have

$$\mathbb{I}_{uv;jk}(x_u, x_v) = \begin{cases} 1 & \text{if } x_u = j \text{ and } x_v = k \\ 0 & \text{Otherwise} \end{cases} . \tag{5.4}$$

It is easy to see that the node-wise and edge-wise potentials $\mathbf{f}$ can be obtained by combining the sufficient statistics and the natural parameter.

An important problem in the context of MRF is that of MAP inference, which is to compute the configuration $\boldsymbol{x}^*$ with the largest probability:

$$\boldsymbol{x}^* \in \underset{\boldsymbol{x}\in\mathcal{X}^n}{\operatorname{argmax}} \exp\left\{\sum_{u\in V} f_u(x_u) + \sum_{(u,v)\in E} f_{uv}(x_u, x_v)\right\} . \tag{5.5}$$

The above optimization problem is equivalent to the following integer programming problem:

$$\boldsymbol{x}^* \in \underset{\boldsymbol{x}\in\mathcal{X}^n}{\operatorname{argmax}} \left\{\sum_{u\in V} f_u(x_u) + \sum_{(u,v)\in E} f_{uv}(x_u, x_v)\right\} . \tag{5.6}$$

The complexity of (5.6) depends critically on the structure of the underlying graph. When G is a tree structured graph, the MAP inference problem can be solved efficiently via the max-product algorithm [63]. However, for an arbitrary graph G, the MAP inference algorithm is usually computationally intractable. The intractability motivates the development of algorithms to solve the MAP inference problem approximately. In the upcoming sections, we briefly review the following approximate inference algorithms: belief propagation, linear programming (LP) relaxation [21, 67] and dual decomposition algorithms [96]. Other approximate algorithms which are not covered in this thesis include quadratic relaxation [86] and semi-definite relaxation [86].

We also want to emphasis that we mainly focus on MAP inference on pairwise MRFs and in principle, there is no loss of generality in restricting to pairwise interactions, since any higher-order MRF can be converted to pairwise MRFs by introducing auxiliary ramdom variabels [103]. Moreover, the techniques described in this thesis can all be generalized to apply directly to general MRFs.

5.2.2 Belief Propagation

Belief propagation (BP) is a heuristic for approximate inference that is simple to code and scales very well with problem size. BP is an example of a message-passing algorithm, and works by iteratively passing messages along edges of the graph. On tree-structured graphical models, BP can be seen to be equivalent to a dynamic programming algorithm and thus solves the inference problem exactly.

Although belief propagation is not guaranteed to give exact results on graphical models with cycles, it has often been observed empirically to give excellent results. Much research has been devoted to understanding its empirical success. For example, [105] shows that, for a graphical model that only has a single cycle, when BP converges (it might not) it gives exact results. [11] shows that BP gives exact results when applied to maximum weight bipartite matching problems that have a unique solution. However, in many problems, belief propagation may not even converge to a solution. [79] observe that convergence problems are highly correlated with BP not finding a good solution to the MAP problem. Intuitively the problem is that of "double counting", in which evidence is passed around the graph multiple times without being recognized as the same evidence already considered in previous belief updates. The problem is only compounded in graphs with short cycles. There have been several attempts to improve BPs accuracy, most notably by the generalized Belief Propagation (GBP) algorithm [112].

5.2.3 Linear Programming Relaxation

Marginal Polytope

In this subsection, we show how to formulate the integer MAP inference problem as an LP. Our analysis in the sequel is based on the marginal distributions defined by the indicator functions. In particular, taking expectations of these indicators with respect to the distribution (5.2) yields marginal probabilities for each node $u \in V$

$$\mu_{u,j} = E_{p(\boldsymbol{x}|\boldsymbol{\theta})}[\mathbb{I}_{u,j}(x_u)] = \sum_{\boldsymbol{x} \in \mathcal{X}^n} p(\boldsymbol{x}|\boldsymbol{\theta}) \mathbb{I}_{u,j}(x_u) \ , \tag{5.7}$$

and for each edge $(u, v) \in E$

$$\mu_{uv;jk} = E_{p(\boldsymbol{x}|\boldsymbol{\theta})}[\mathbb{I}_{uv;jk}(x_u, x_v)] = \sum_{\boldsymbol{x} \in \mathcal{X}^n} p(\boldsymbol{x}|\boldsymbol{\theta}) \mathbb{I}_{uv,jk}(x_u, x_v) \ . \tag{5.8}$$

Note that (5.7) and (5.8) define a d-dimensional marginal vector $\boldsymbol{\mu}$. We let $M(G)$ denote the set of all such marginals realizable in this way

$$M(G) = \{\boldsymbol{\mu} \in \mathbb{R}^d | \mu_{u,j} = E_{p(\boldsymbol{x}|\boldsymbol{\theta})}[\mathbb{I}_{u,j}(x_u)] \quad \text{and} \quad \mu_{uv;jk} = E_{p(\boldsymbol{x}|\boldsymbol{\theta})}[\mathbb{I}_{uv;jk}(x_u, x_v)]\} \ . \tag{5.9}$$

The conditions defining membership in $M(G)$ can be expressed more compactly in the equivalent vector form

$$\boldsymbol{\mu} = E_{p(\boldsymbol{x}|\boldsymbol{\theta})}[\phi(\boldsymbol{x})] = \sum_{\boldsymbol{x} \in \mathcal{X}^n} p(\boldsymbol{x}|\boldsymbol{\theta})\phi(\boldsymbol{x}) \ . \tag{5.10}$$

We refer to $M(G)$ as the marginal polytope associated with the graph G.

By construction, the marginal polytope is the convex hull of the $\phi(\boldsymbol{x})$ vectors, one for each assignment $\boldsymbol{x} \in \mathcal{X}^n$ to the variables of the Markov random fields. Now we formulate the MAP inference problem (5.6) as an LP. Since the optimal value of a linear program over a polytope can be shown to be obtained by one of its vertices and the marginal polytope $M(G)$ is the convex hull of the discrete set $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}^n\}$, the linear MAP inference problem is:

$$\max_{\boldsymbol{x} \in \mathcal{X}^n} \langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle = \max_{\boldsymbol{\mu} \in M(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \ . \tag{5.11}$$

However, the LP (5.11) is in general impractical to solve in practice, because the number of linear constraints required to characterize $M(G)$ grows rapidly in n for a general graph with cycles [103]. The LP relaxation focuses on a subset of the constraints that $\boldsymbol{\mu}$ must satisfy and the number of constraints specified in the LP relaxation is polynomial in n.

Local Polytope and LP relaxation

First, since the elements of $\boldsymbol{\mu}$ are marginal probabilities, we must have the non-negative constraints:

$$\mu_u(x_u) \geq 0, \quad \forall u \in V \ , \tag{5.12}$$

$$\mu_{uv}(x_u, x_v) \geq 0, \quad \forall (u, v) \in E \ . \tag{5.13}$$

Second, the node marginals must satisfy the normalization constraints

$$\sum_{x_u \in \mathcal{X}_u} \mu_u(x_u) = 1, \quad \forall u \in V \ . \tag{5.14}$$

Last, since the node marginals over x_u must be consistent with the joint marginal on (x_u, x_v), we also have the marginalization constraints:

$$\sum_{x_u \in \mathcal{X}_u} \mu_{uv}(x_u, x_v) = \mu_v(x_v), \quad \forall (u, v) \in E \ . \tag{5.15}$$

We denote the polytope defined by (5.12)-(5.15) as $L(G)$. By construction, $L(G)$ is an outer bound of $M(G)$, i.e., $M(G) \subseteq L(G)$. Moreover, in contrast to $M(G)$, it involves only a number of inequalities that is polynomial in n. Now the LP relaxation of MAP inference problem (5.6) becomes solving the following LP:

$$\max_{\boldsymbol{\mu} \in L(G)} \langle \boldsymbol{\mu}, \boldsymbol{f} \rangle = \sum_{u \in V} \sum_{x_u} \mu_u(x_u) f_u(x_u) + \sum_{(uv) \in E} \sum_{x_u, x_v} \mu_{uv}(x_u, x_v) f_{uv}(x_u, x_v) \ , \tag{5.16}$$

subject to the constraint that $\boldsymbol{\mu} \in L(G)$.

Since $L(G)$ is an outer bound of $M(G)$ and we maximize over a larger space , we have

$$\max_{\boldsymbol{\mu} \in M(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \leq \max_{\boldsymbol{\mu} \in L(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \ , \tag{5.17}$$

i.e., the LP relaxation provides an upper bound on the value of the MAP assignment. In general, $L(G)$ has both integral and fractional vertices. In particular, all vertices of $M(G)$ are also vertices of $L(G)$. Thus, If the solution $\boldsymbol{\mu}$ to (5.16) is an integer solution, it is guaranteed to be the optimal solution of (5.6). Otherwise, one can apply rounding schemes [84, 85] to round the fractional solution to an integer solution.

Although standard LP solvers can be used to solve the optimization problem (5.16), they are usually inefficient for the MAP inference problem [111], mainly because they fail to take advantage of the underlying graph structure. Specially designed MAP inference algorithms usually exploit the structures of the dependency graphs and some of them solve either the primal LP (5.16) or the dual problem of (5.6).

Primal Based MAP Inference Algorithms

Although standard LP solvers can be used to solve the optimization problem (5.16), they are usually inefficient compared to the algorithms which exploit the graph structure [111]. In this section, we briefly introduce the proximal maximization algorithm [13] which can take advantage of the graph structure and is guaranteed to converge to the global maximizer of (5.16).

Instead of solving the constrained LP (5.16) directly, the proximal maximization methods solves a sequence of maximization problems:

$$\mu^{t+1} = \operatorname*{argmax}_{\mu \in L(G)} \left\{ \langle \mu, f \rangle - \frac{1}{w^t} D_h(\mu || \mu^t) \right\} \ , \tag{5.18}$$

where the subscript $t = 1, 2, \ldots$ denotes the iteration number, w^t is a positive constant and $D_h(\mu||\nu)$ is a Bregman divergence [18] between μ and ν induced by the strictly convex function h:

$$D_h(\mu||\nu) = h(\mu) - h(\nu) - \langle \nabla h(\nu), \mu - \nu \rangle \ .$$

Now we study the choice of h that will be used in the sequel. To take the advantage of the graph structure, in principle, we can decompose the graph into N (overlapping) parts and assign each part a strictly convex function $h_i, i = 1, \ldots, N$. Let μ_i denote the components of μ that belong to part i and $h = \sum_{i=1}^{N} h_i$. The Bregman divergence then becomes:

$$D_h(\mu||\nu) = \sum_{i=1}^{N} h_i(\mu_i) - h(\nu_i) - \langle \nabla h_i(\nu_i), \mu_i - \nu_i \rangle \ .$$

For the sake of simplicity, we focus on a straightforward decomposition: we decompose the graph into $|V|$ nodes and $|E|$ edges and the strictly convex function h_i is the negative entropy of the pseudomarginals:

$$h_u = \sum_{x_u} \mu_u(x_u) \log \mu_u(x_u), \forall u \in V,$$
$$h_{uv} = \sum_{x_u, x_v} \mu_{uv}(x_u, x_v) \log \mu_{uv}(x_u, x_v) \ , \forall (u, v) \in E.$$

Then the Bregman divergence is the Kullback-Leibler (KL) divergence across all the nodes and edges:

$$D_h(\mu||\nu) = \sum_{u \in V} D_{h_u}(\mu_u||\nu_u) + \sum_{(uv) \in E} D_{h_{uv}}(\mu_{uv}||\nu_{uv}) \ ,$$

where $D_{h_u}(\mu_u||\nu_u) = \sum_{x_u} \mu_u(x_u) \log \frac{\mu_u(x_u)}{\nu_u(x_u)} + \mu_u(x_u) - \nu_u(x_u)$. We note that this is also the setting of one of the proximal algorithms in [85].

We now show how to solve the optimization problem (5.18) by performing the Bregman projection [18]. We observe that (5.18) can be solved by first obtaining the solution $\mu^{t+1,0}$ to the unconstrained problem of (5.18) and then projecting $\mu^{t+1,0}$ to $L(G)$. To be more specific, we have:

$$\mu^{t+1,0} = \operatorname*{argmax}_{\mu} \left\{ \langle \mu, f \rangle - \frac{1}{w^t} D_h(\mu||\mu^t) \right\} \ , \tag{5.19}$$

$$\mu^{t+1} = \operatorname*{argmin}_{\mu \in L(G)} D_h(\mu||\mu^{t+1,0}) \ . \tag{5.20}$$

When h is the negative entropies on node and edge pseudomarginals, (5.19) has a closed form solution. Taking derivatives and setting them to zeros yields:

$$\mu_u^{t+1,0}(x_u) = \mu_u^t \exp(w^t f_u(x_u)) \ ,$$
$$\mu_{uv}^{t+1,0}(x_u, x_v) = \mu_{uv}^t \exp(w^t f_{uv}(x_u, x_v)) \ .$$

Unfortunately, (5.20) does not have a closed form solution and the projection is usually computed iteratively. In particular, we show that the projection can be obtained by performing a sequence of cyclic Bregman projections.

We note that the polytope $L(G)$ can be viewed as an intersection of the equality constraints, i.e., (5.14) and (5.15) (The inequality constraints (5.12) and (5.13) are taken care of automatically when h is the negative entropy function.). It is easy to see that the number of constraints is $NC = k|V| + 2k|E|$. Denote each equality constraint as $C_i, i = 1, \ldots, NC$ and define

$$\mu = \Pi_{C_i}(\nu) \tag{5.21}$$

as the operation of projecting ν onto the constraint C_i. The Bregman projection algorithm projects $\mu^{t+1,0}$ sequentially onto each constraint C_i of $L(G)$ in a cyclic manner, i.e., starting from $\nu = \mu^{t+1,0}$, we perform the following operation repeatedly until convergence:

$$\mu = \Pi_{C_i}(\nu),$$
$$\nu = \mu,$$

where i is the constraint index and $i = 1, \ldots, NC, 1, \ldots, NC, 1, \ldots$. It can be shown [18] that the above cyclic Bregman projection converges to the projection defined in (5.20). It is important to point out that when the constraint set includes inequality constraints, the Bregman projection has to be followed by a correction step [18]. We avoid this by choosing the KL divergence as the Bregman divergence.

We now show that the projection (5.21) onto each constraint has closed form solutions. We use the notation $\mu^{t+1,p}$ to denote the value of μ^{t+1} after $\mu^{t+1,0}$ has been projected to $L(G)$ p times according to the cyclic projection scheme. Consider the constraint on node $u : \sum_{x_u} \mu_u(x_u) = 1$ and let λ_u be the Lagrangian multiplier associated with the constraint. The KKT condition is:

$$\nabla_h(\mu_u^{t+1,p+1}(x_u)) = \nabla_h(\mu_u^{t+1,p}(x_u)) + \lambda_u^{t+1,p+1} \ .$$

Expanding the derivatives and performing some simple algebra yields:

$$\mu_u^{t+1,p+1}(x_u) = \mu_u^{t+1,p}(x_u)\exp(\lambda_u^{t+1,p+1}) \ ,$$
$$\exp(\lambda_u^{t+1,p+1}) = \frac{1}{\sum_{x_u}\mu_u^{t+1,p}(x_u)} \ .$$

Then it follows the normalization update for each node:

$$\mu_u^{t+1,p+1}(x_u) = \frac{\mu_u^{t+1,p}(x_u)}{\sum_{x_u}\mu_u^{t+1,p}(x_u)} \ .$$

Similarly, we can derive the update for each edge

$$\mu_{uv}^{t+1,p+1}(x_u,x_v) = \mu_{uv}^{t+1,p}(x_u,x_v)\sqrt{\frac{\mu_u^{t+1,p}(x_u)}{\sum_{x_v}\mu_{uv}^{t+1,p}(x_u,x_v)}} \ ,$$
$$\mu_u^{t+1,p+1}(x_u) = \sqrt{\mu_u^{t+1,p}(x_u)\sum_{x_v}\mu_{uv}^{t+1,p}(x_u,x_v)} \ .$$

In summary, the MAP inference algorithm (Algorithm 1) is a double loop algorithm: In the outer loop, the algorithm performs proximal maximization and in the inner loop, the algorithm performs cyclic Bregman projection. It is shown in [13, 85] that for appropriate choice of weight sequence $\{w^t\}$, Algorithm 1 has super-linear convergence rate.

5.2.4 Dual Decomposition

Recall the MAP inference problem solves the following integer programming problem:

$$MAP(\mathbf{f}) = \sum_{u\in V} f_u(x_u) + \sum_{(u,v)\in E} f_{uv}(x_u,x_v) \ . \tag{5.22}$$

The main idea of dual decomposition is: we construct a dual function $L(\delta)$ and $L(\delta) \geq MAP(\boldsymbol{\theta})$ holds for all dual variables δ. Thus, $L(\delta)$ is an upper bound on the value of the MAP assignment. We then seek a δ to make this upper bound as tight as possible. Before we describe the dual decomposition algorithm, we first specify the dual variables. To avoid the notation clusters, we denote e an edge $(u,v) \in E$ and $u \in e$ if u is an endpoint of e. For every $e \in E, u \in e$ and x_u, we have a dual variable $\delta_{e,u}(x_u)$.

Algorithm 1 MAP inference algorithm with proximal maximization and Bregman projection

Input: potential functions $\{f_u, f_{uv}\}$, weight sequence $\{w^t\}$ and the number of integer values k

Output: pseudomarginals μ

Initialization: $\mu_u^1(x_u) = \frac{1}{k}$ and $\mu_{uv}^1(x_u, x_v) = \frac{1}{k^2}$

Outer Loop: For $t = 1, 2, \ldots$ until convergence

Compute μ^{t+1} by the inner loop

1. Initialization:

$$\mu_u^{t+1,0}(x_u) = \mu_u^t(x_u) \exp(w^t f_u(x_u)) \ ,$$
$$\mu_{uv}^{t+1,0}(x_u, x_v) = \mu_{(uv)}^t(x_u, x_v) \exp(w^t f_{uv}(x_u, x_v)) \ .$$

2. Inner Loop : For $p = 0, 1, \ldots$ until convergence

For each node

$$\mu_u^{t+1,p+1}(x_u) = \frac{\mu_u^{t+1,p}(x_u)}{\sum_{x_u} \mu_u^{t+1,p}(x_u)} \ .$$

For each edge

$$\mu_{uv}^{t+1,p+1}(x_u, x_v) = \mu_{uv}^{t+1,p}(x_u, x_v) \sqrt{\frac{\mu_u^{t+1,p}(x_u)}{\sum_{x_v} \mu_{uv}^{t+1,p}(x_u, x_v)}} \ ,$$
$$\mu_u^{t+1,p+1}(x_u) = \sqrt{\mu_u^{t+1,p}(x_u) \sum_{x_v} \mu_{uv}^{t+1,p}(x_u, x_v)} \ .$$

We can interpret the dual variable as the message edge e sends to u regarding its state x_u.

Now we derive the dual optimization problem of (5.6). We first reformulate the integer programming problem by duplicate the $x_u$variables, once for each edge e if $u \in e$ and enforce all the copies are equal. We use x_u^e to denote the copy of x_u used by edge e. We also denote $\boldsymbol{x}^e$ the set of all variables $\{x_u^e\}, u \in e$ used by edge e and $\boldsymbol{x}^E$ the set

of all variable copies. We rewrite (5.6) as an equivalent integer programing problem:

$$\begin{aligned} \max \quad & \sum_{u \in V} f_u(x_u) + \sum_{e \in E} f_e(\boldsymbol{x}_e^e) \\ s.t. \quad & x_u^e = x_u, \quad \forall e, u \in e \ . \end{aligned} \tag{5.23}$$

We then introduce Lagrange multipliers (dual variables) δ and define the Lagrangian:

$$L(\delta, \boldsymbol{x}, \boldsymbol{x}^E) = \sum_{u \in V} f_u(x_u) + \sum_{e \in E} f_e(\boldsymbol{x}^e) + \sum_{e \in E} \sum_{u \in e} \sum_{\bar{x}_u} \delta_{e,u}(\bar{x}_u))(\mathbf{1}[x_u = \bar{x}_u] - \mathbf{1}[x_u^e = \bar{x}_u]) \ . \tag{5.24}$$

Note that maximizing (5.24) in conjunction with the consistency constraints in (5.23) is equivalent to (5.6):

$$\begin{aligned} \max_{\boldsymbol{x}, \boldsymbol{x}^E} \quad & L(\delta, \boldsymbol{x}, \boldsymbol{x}^E) \\ s.t. \quad & x_u^e = x_u, \quad \forall e, u \in e \ . \end{aligned} \tag{5.25}$$

Solving the integer programming problem (5.25) is as hard as solving the original MAP problem (5.6). To obtain a tractable problem, we omit the consistency constraints in (5.25) and define an upper bound $L(\delta)$ of (5.24):

$$L(\delta) = \max_{\boldsymbol{x}, \boldsymbol{x}^E} L(\delta, \boldsymbol{x}, \boldsymbol{x}^E) \tag{5.26}$$

$$= \sum_{u \in V} \max_{x_u} \left(f_u(x_u) + \sum_{e : u \in e} \delta_{e,u}(x_u) \right) + \sum_{e \in E} \max_{\boldsymbol{x}^e} \left(f_e(\boldsymbol{x}^e) - \sum_{u \in e} \delta_{e,u}(x_u^e) \right) \ . \tag{5.27}$$

The dual problem is to obtain a tightest upper bound by minimizing $L(\delta)$, i.e., $\min_\delta L(\delta)$.

It is important to point out that the LP relaxation is in fact equivalent to the dual decomposition approach discussed in this thesis. Standard duality transformations can be used to show the convex dual of the LP (5.16) is the Lagrangian relaxation (5.27). The connection between the LP relaxation and dual decomposition shows that these two types of approximate inference algorithms apply equally for the MAP problem.

We also want to mention that (5.27) is not the only dual relaxation. In recent years, a number of dual relaxation algorithms [96] have been proposed and these have been demonstrated to be useful tools for solving large MAP inference problems. In [97], these dual algorithms are placed under a common framework so that they can be understood as optimizing the same objective. The authors in [97] also demonstrate how to change from one dual representation to another in a monotone fashion relative to the common objective.

Block Coordinate Descent Algorithms

Block coordinate descent algorithms work by fixing the values of all dual variables except for a set of variables and then minimizing the objective as much as possible with respect to that set. In this subsection, we describe the MPLP algorithm [42], in which all variables δ except $\delta_{e,u}(x_u)$ for a specific edge e and both $u \in e$ are fixed. We would like to minimize $L(\delta)$ with respect to the free variables $\delta_{e,u}(x_u)$:

$$\bar{L}(\delta) = \sum_{u \in e} \max_{x_u} \left(f_u(x_u) + \sum_{\bar{e}: u \in \bar{e}} \delta_{\bar{e},u}(x_u) \right) + \max_{\boldsymbol{x}_e} \left(f_e(\boldsymbol{x}_e) - \sum_{u \in e} \delta_{e,u}(x_u) \right) \tag{5.28}$$

$$\geq \max_{\boldsymbol{x}_e} \left(f_e(\boldsymbol{x}_e) + \sum_{u \in e} \delta_u^{-e}(x_u) \right) \ , \tag{5.29}$$

where $\delta_u^{-e}(x_u) = f_u(x_u) + \sum_{\bar{e} \neq e} \delta_{\bar{e},u}(x_u)$. In [42], the authors derives the following update to achieve the above lower bound:

$$\delta_{e,u}(x_u) = -\delta_u^{-e}(x_u) + \frac{1}{2} \max_{\boldsymbol{x}_e \backslash u} \left(f_e(\boldsymbol{x}_e) + \sum_{v \in e} \delta_v^{-e}(x_v) \right) \ . \tag{5.30}$$

The update (5.30) is performed on all $u \in e$ and x_u simultaneously. When the algorithm converges, the MAP assignment can be obtained from the dual beliefs:

$$x_u^* \in \operatorname*{argmin}_{x_u} \left(f_u(x_u) + \sum_{e: u \in e} \delta_{e,u}(x_u) \right) \tag{5.31}$$

We summary the MPLP in Algorithm 2.

One key design choice to make when designing coordinate descent algorithms is which variables to update in each iteration. For example, in the Max-Sum algorithm [106], all of the dual variables except $\delta_{e,u}(x_u)$ for a specific e and u are fixed. In [42], the authors derive a star update, where the free dual variables are those associated with a given node u and its neighbors. Coordinate descent algorithms using blocks corresponding to tree-structured sub-graphs are derived in [97, 101]. For the empirical comparison of these block coordinate algorithms, we refer the readers to [96].

Although coordinate descent algorithms decrease the dual objective at every iteration, they are not generally guaranteed to converge to the dual optimum. The reason is that although the dual objective $L(\delta)$ is convex, it is not strictly convex. This implies that minimizing coordinate value may no be unique and thus convergence guarantees for coordinate descent algorithms do not hold.

Algorithm 2 MPLP Algorithm

Input: potential functions $\{f_u, f_{uv}\}$
Output: MAP assignment $x_1^*, \ldots, x_n^*$
Initialization: $\delta_{e,u}(x_u) = 0, \quad \forall e \in E, u \in e, x_u$
Perform the following update until the change of the value of $L(\delta)$ is small enough
For each $e \in E$, perform

$$\delta_{e,u}(x_u) = -\delta_u^{-e}(x_u) + \frac{1}{2}\max_{\boldsymbol{x}_e \backslash u}\left(f_e(\boldsymbol{x}_e) + \sum_{v \in e}\delta_v^{-e}(x_v)\right) ,$$

simultaneously for all $u \in E$ and x_u.
Compute MAP assignment: $x_u^* \in \operatorname{argmin}_{x_u}\left(f_u(x_u) + \sum_{e:u\in e}\delta_{e,u}(x_u)\right), \forall u \in V$

Subgradient Algorithms

In this subsection, we mainly describe the subgradient algorithm [62] to solve the approximate MAP inference problem. The subgradient algorithm works on a different dual optimization problem, which is based on rewriting the original MAP problem into a set of sub-problems on edges. To make sure the new optimization problem is equivalent to the original one, we impose consistency constraints on the nodes, i.e., if a node is shared by more than one edge, its value must be consistent among these edges. To be more specific, we have

$$\begin{aligned} \max_{\boldsymbol{x}_e, \boldsymbol{x}} \quad & \sum_{e\in E}\left(\sum_{u\in e}\bar{f}_u(x_u^e) + f_e(\boldsymbol{x}_e)\right) \\ s.t. \quad & x_u^e = x_u \ , \quad \forall u, e \in N(u) \end{aligned} \tag{5.32}$$

where $\sum_{e\in N(u)}\bar{f}_u(x_u^e) = f_u(x_u)$ and $N(u)$ denotes the set of edges that share u. A simple option is to set $\bar{f}_u(x_u^e) = \frac{1}{N(u)}f_u(x_u)$. It is clear that without the consistency constraints $x_u^e = x_u$, problem (5.32) would decouple into a series of smaller MAP inference problems on edges. Therefore, it is natural to relax these coupling constraints by introducing

Lagrange multipliers λ^e and form the dual function as

$$L(\lambda) = \max_{\boldsymbol{x}_e, \boldsymbol{x}} \sum_{e \in E} \left(\sum_{u \in e} \bar{f}_u(x_u^e) + f_e(\boldsymbol{x}_e) \right) + \sum_{e \in E} \sum_{u \in e} \lambda_u^e (x_u^e - x_u) \tag{5.33}$$

$$= \max_{\boldsymbol{x}_e, \boldsymbol{x}} \sum_{e \in E} \left(\sum_{u \in e} \bar{f}_u(x_u^e) + \lambda_u^e x_u^e + f_e(\boldsymbol{x}_e) \right) - \sum_{e \in E} \sum_{u \in e} \lambda_u^e x_u \ . \tag{5.34}$$

$$= \max_{\boldsymbol{x}_e} \sum_{e \in E} \left(\sum_{u \in e} \bar{f}_u(x_u^e) + \lambda_u^e x_u^e + f_e(\boldsymbol{x}_e) \right) \tag{5.35}$$

The last term in (5.34) can be eliminated by directly maximizing over $\boldsymbol{x}$, which simply imposes the feasibility constraints on λ:

$$\Lambda = \left\{ \lambda^e \Big| \sum_{e \in N(u)} \lambda_u^e = 0 \right\} \ . \tag{5.36}$$

Then, we can set up the dual problem:

$$\min_{\lambda \in \Lambda} L(\lambda) = \max_{\boldsymbol{x}_e} \sum_{e \in E} \left(\sum_{u \in e} \bar{f}_u(x_u^e) + \lambda_u^e x_u^e + f_e(\boldsymbol{x}_e) \right) \tag{5.37}$$

$$= \max_{\boldsymbol{x}_e} \sum_{e \in E} \widetilde{f}_e(\lambda^e) \ . \tag{5.38}$$

In (5.38), we write the dual problem as a summation over serval MAP inference problems on edges.

Th dual problem is always convex, but non-differentiable and a projected subgradient algorithm is applicable for such problems. In each iteration in the algorithm, the following updates are performed: (i) for each edge, compute the best MAP assignment $\boldsymbol{x}_e^*$ (ii) update the dual variables $\lambda_{\mathbf{e}}$ according to $\boldsymbol{x}_e^*$ (iii) project λ to the feasible set Λ. The first update can be computed efficiently via the max-product algorithm. The second updates invokes computing the subgradient of $\widetilde{f}_e$ at λ^e. In [62], the authors shows that the subgradient $\nabla \widetilde{f}_e(\lambda^e)$ is simply $\boldsymbol{x}_e^*$ and the dual variables can be updated as

$$\lambda^{\mathbf{e}} \leftarrow \lambda^{\mathbf{e}} + \alpha \boldsymbol{x}_e^* \ , \tag{5.39}$$

where α is a positive step size. The third update amounts to a centering problem and

the projection reduces to subtracting the average vector from each λ_u^e:

$$\lambda_u^e \leftarrow \lambda_u^e + \alpha \boldsymbol{x}_u^{e*} - \frac{\sum_{e' \in N(u)} \lambda_u^{e'} + \alpha \boldsymbol{x}_u^{e'*}}{|N(u)|}$$

$$\lambda_u^e \leftarrow \lambda_u^e + \alpha \left(\boldsymbol{x}_u^{e*} - \frac{\sum_{e' \in N(u)} \boldsymbol{x}_u^{e'*}}{|N(u)|} \right) .$$

Chapter 6

Drought Detection of the Last Century: An MRF-based Approach

6.1 Motivation

Droughts are one of the most damaging climate-related hazards and the consequences are often abrupt, severe and potentially catastrophic to both society and the environment. Droughts may lead to reductions in water supply, diminished power generation, disturbed riparian habitats as well as a host of other associated economic, political and social activities [107]. A frequently cited example is the decades long Sahel drought [31, 53] starting in the late 1960s, which led to widespread famine, ecosystem degradation and dispersion of its inhabitants. Other examples include the Dust Bowl event in the central US [92] in the 1930s, which is marked by sudden reductions in precipitation.

In the climate science community, the cause of droughts has been extensively studied. For example, [89] identify the physical basis for long-term droughts and [29] analyze how the increase in deserts influences climate, e.g., a reduction in precipitation in a general circulation model (GCM). In addition to the general theory developed for droughts worldwide, some work has been focused on specific drought events. For instance, [92] perform model simulations to identify the mechanisms contributing to the

Dust Bowl. [31] show how the interaction between climate and vegetation prolongs the Sahel drought. [25] analyze the Sahel drought simulation with GCMs and projection what may happen in the future.

Even though the importance of understanding droughts cannot be overstated, there are few rigorous and systematic tools to detect them. The current standard of drought detection is the Palmer Severity Drought Index (PSDI) [46]. The PSDI is based on a supply-and-demand model of soil moisture. It uses a 0 as normal, and drought is shown in terms of negative numbers. However, the utility of the Palmer index is weakened by the arbitrary nature of Palmer's algorithms, including the technique used for standardization. In a recent paper, [80] analyze global historical rainfall observations and detect regions that have undergone large, sudden decreases in rainfall. Their algorithm identifies the potential regions of abrupt rainfall changes using a wavelet-based method. However, the algorithm is not fully automatic as it requires a manual inspection step to remove the potential droughts with low magnitude of rainfall change and short span of persistence.

In contrast, machine learning and data mining methods have been successfully applied to application domains, such as computer vision, natural language processing and others. With a significant increase in the number of climate datasets available, we believe that relevant machine learning and data mining algorithms can also be applicable to climate science. In this paper, we propose a drought detection algorithm based on the well studied Markov random field model [103].

We formulate the detection problem as the one of finding the most likely configuration of a binary MRF, where each node can only take two values: 1 means the node is in a drought state and 0 means a normal state. Since the gridded precipitation dataset we use is spatio-temporal, i.e., the dataset contains precipitation observations of the globe over a period of time, we construct a 3-dimensional grid graph as the underlying dependency graph for the MRF. More specifically, for a particular time, we model the dependency using a 4-nearest neighbor grid, where each node represents a location. The 3-dimensional grid can be viewed as a replication of 2-dimensional grids and the nodes representing the same location are connected together. We design the potential functions carefully from the climate datasets to ensure spatio-temporal consistency, i.e., the neighboring nodes in the 3-dimensional grid are encouraged to take the same value.

Our goal is to estimate the binary value each node takes based on the MRF and we consider that the nodes with value 1 are in drought states. However, in general the integer programming problem of finding the most likely configuration is computationally intractable and people often resort to relaxation and obtain approximate solutions [103]. Throughout the paper, we use the Linear Programming relaxation [67], which has been extensively studied in the MRF literature. Instead of solving the LP directly, we adopt the idea of proximal minimization [13] from the optimization literature and propose an efficient inference algorithm. After the algorithm terminates, we round the relaxed fractional solution and obtain the integer solution. We then identify major droughts which are spatially widespread over long duration based on the integer solution.

We apply our drought detection algorithm on the Climate Research Unit (CRU) precipitation dataset [76] over 106 years (1901-2006). The drought detection problem on this dataset is of large scale, since the underlying dependency graph has over 7 million nodes and the number of configurations is more than $2^{7,000,000}$. Our algorithm is fully automatic and solves the problem efficiently, i.e., the algorithm converges within one and a half hours in a Linux workstation. The empirical results show that the algorithm successfully detects the major droughts of the twentieth century, including the Dust Bowl in the 1930s and the drought in the Sahel starting in the late 1960s. We compare our algorithm with the drought detection algorithm in [80] and find that both algorithms detect similar droughts.

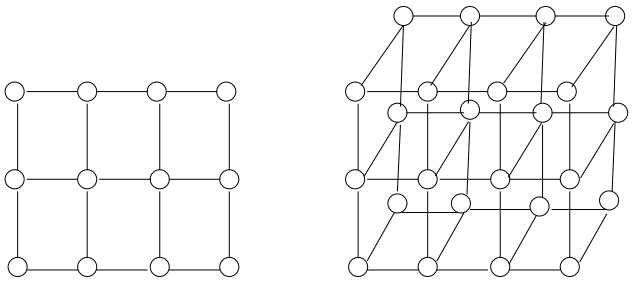
The rest of the chapter is organized as follows: we outline the MRF-based drought detection algorithm in Section 6.2. We show the experimental results on the CRU dataset in Section 6.3.

6.2 MRF-based Drought Detection Algorithm

In this section, we show how droughts are detected using the proximal MAP inference algorithm presented in Section 5.2.3.

6.2.1 Designing the Potential Functions

Climate datasets are usually spatio-temporal datasets in that they have climate variable observations over the globe for a period of time. Suppose a precipitation dataset has



(a) $m = 3, n = 4, T = 1$.

(b) $m = 3, n = 4, T = 2$.

Figure 6.1: The graph structure for climate datasets used in this paper.

yearly precipitation averages over a $m \times n$ global grid over T years, i.e., the resolution is $\frac{180}{m}$ degree latitude $\times$ $\frac{360}{n}$ degree longitude. To use the MRF model, the underlying graph structure has to be determined. To model the spatio nature of the dataset, we use a $m \times n$ 4-nearest neighbor grid for each year. To model the temporal nature, we construct a 3-dimensional grid by connecting the nodes representing the same location in the T 2-dimensional grids. Figure 6.1 shows the graph structure for $m = 3, n = 4, T = 2$.

To facilitate the discussion in the sequel, we introduce some notations first: We denote the precipitation observation at location u at time t as y_u^t. The nodewise potential of location u at time t is $f_u^t(x_u^t)$. For two neighboring locations at time t, we denote the pairwise potential as $f_{uv}^t(x_u^t, x_v^t)$. For the same location u at time $t-1$ and t, we denote the pairwise potential as $f_u^{t-1,t}(x_u^{t-1}, x_u^t)$.

We are now ready to define the potential functions based on the 3-dimensional grid. To detect drought regions, we set $k = 2$, where $x_u^t = 1$ means that location u at time t is in a drought state, i.e., abnormal state, and $x_u^t = 0$ means a normal state. To define the nodewise potential function for each location u, we partition the observations $y_u^t, t = 1, \ldots, T$ into two parts and we consider the observations below the $p\%$ percentile as abnormal and the rest as normal. We also compute $\mu_u^{abnormal}$ and μ_u^{normal}, the mean of the abnormal and normal observations respectively. Then the nodewise potential function comes from the log-likelihood of a Gaussian distribution

$$f_u^t(x_u^t = 1) = \log \mathcal{N}(y_u^t | \mu_u^{abnormal}, \sigma_u^2) \ ,$$
$$f_u^t(x_u^t = 0) = \log \mathcal{N}(y_u^t | \mu_u^{normal}, \sigma_u^2) \ ,$$

where σ_u is the standard deviation of the observations at location u.

We define the pairwise potential functions to encourage label consistency, i.e., the potential value is higher if neighboring nodes take same values. Specifically, we set the pairwise potential as follows:

$$f_{uv}^t(x_u^t, x_v^t) = \begin{cases} C_1 > 0, & \textit{if } x_u^t = x_v^t; \\ 0, & \textit{Otherwise.} \end{cases}$$

and

$$f_u^{t-1,t}(x_u^{t-1}, x_u^t) = \begin{cases} C_2 > 0, & \textit{if } x_u^{t-1} = x_u^t; \\ 0, & \textit{Otherwise.} \end{cases}$$

Intuitively, the higher C_1 is, the more likely the neighboring nodes in the 2-dimensional girds are to take same values. Similarly, the higher C_2 is, the nodes representing the same location at consecutive time intervals are to take same values.

6.2.2 Obtaining the Integer Solution from the Pseudomarginals

After the potential function is defined, we can run the MAP inference algorithm presented in Section 5.2.3 and compute the pseudomarginals μ. To find the drought regions, we need to decode the fractional solution μ and obtain the integer configuration $\boldsymbol{x}$. The value each node takes tells us whether the location is in a drought state or not.

We employ a simple node-based rounding scheme to interpret the pseudomarginals μ:

$$x_u = \underset{x' \in \mathcal{X}_u}{\operatorname{argmax}} \, \mu_u(x') \ . \tag{6.1}$$

We apply (6.1) to each node and obtain the corresponding integer solution.

Now each node has an integer value associated with it: If $x_u^t = 1$, we consider that the location u at time t is in a drought state, otherwise it is in a normal state.

6.2.3 Drought Detection from the Integer Solution

Once we have the integer solution, we can detect droughts based on it. Since a drought can be defined both spatially and temporally, we define it as a set of neighbouring nodes in the three-dimensional dependency graph (Figure 6.2) whose states are drought. Thus, the drought detection problem becomes one of finding sets of neighbouring nodes with

drought states. To accomplish this goal, we first construct a three-dimensional adjacency graph and calculate the connected components of this graph. We treat each connected component as a drought. In the adjacency graph, two nodes are connected if both nodes are in drought states, i.e., we simply remove the edges from the dependency graph if at least one node is not in the drought state. A connected component is defined as a subgraph of the adjacency graph in which any two vertices are connected to each other by paths and which is connected to no additional vertices. Since a connected component may only contain a few locations and may not be long in duration, we only pick the sizable components and consider them as major droughts. The details as how to select connected components are in Section 6.3.

6.2.4 Practical Issues

Algorithm 1 is a general MAP inference algorithm and can be applied to estimate the mostly likely configuration for any pairwise MRF. In practice, we find several ways to speed up the algorithm for our application.

The first option is that, instead of a uniform initialization on μ, we can initialize the pseudomarginals based on the precipitation value. To be more specific, we set $\mu_u^t = 1$, if the climate variable on node u at time t is below the $p\%$ percentile and $\mu_u^t = 0$, otherwise. We attribute the speed up to the conjecture that the value-based initialization is more close to the optimal solution than the uniform initialization.

The other option is to divide the globe into several disjoint parts, run the MAP inference algorithm on each part and combine the results. Since we can estimate the configuration for each part simultaneously, we can gain significant speed up due to parallel computing. Some climate datasets have disjoint parts in nature. For example, since the CRU only has precipitation over land, North America, South America and Australia are isolated from the rest of the world.

Finally, we choose the weight scalar $\{w^1, \ldots, w^T\}$ such that $w^T \to 0$. In this case, we observe a super-linear convergence rate if we terminate the inner loop when the change of μ between two consecutive iterations is less than 10^{-3}.

6.3 Experimental Results

In this section we show the drought regions detected by the algorithm using the CRU dataset, which has monthly precipitation from the years 1901-2006. The dataset (Figure 6.2) is of high gridded spatial resolution (0.5 degree latitude $\times$ 0.5 degree longitude) and only includes the precipitation over land (67420 locations with precipitation records). To eliminate the monthly variance, we convert the monthly dataset to a yearly dataset by calculating the average precipitation over 12 months for each year.

We first apply the drought detection algorithm over the United States and the Sahel region and show that our algorithm successfully discovers the dust bowl in the 1930s and the prolonged drought in the Sahel starting in the late 1960s. We then apply the algorithm to the global dataset and report the droughts that we identify.

For all the experiments, we set $C_1 = 0.5$, $C_2 = 3$, $w^1 = 1$ and $w^{t+1} = 0.8w^t$. We terminate both the inner and outer loop when the change of μ between two consecutive iterations is less than 10^{-3}.

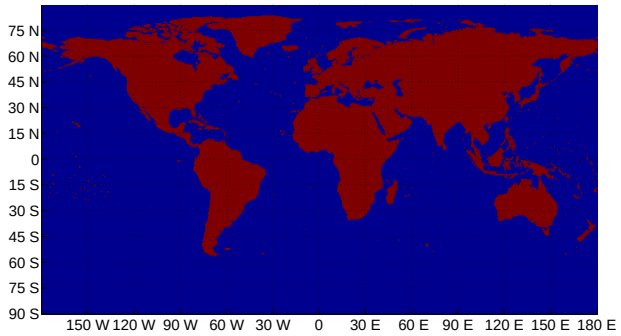


Figure 6.2: The CRU dataset is a highly gridded dataset containing precipitation for land locations only (red region).

6.3.1 The Dust Bowl

The Dust Bowl, in the 1930s, was one of the most devastating droughts of the past century in the Great Plains region of the United States. The severe drought affected almost two-thirds of the country and was infamous for the numerous dust storms that occurred [92].

In this subsection, we show how our approach can be used to detect the Dust Bowl drought from the CRU precipitation dataset. We first extract the precipitation of the United States and run the drought detection algorithm on this sub-dataset with $p = 15$. After the algorithm outputs the pseudomarginals, we compute the integer solution. Figure 6.3 shows the number of locations with drought state detected by our algorithm. We compare our drought detection algorithm with a simple threshold algorithm: If the precipitation in a location is lower than the $p\%$ percentile, we consider it in a drought state. Otherwise, it is in a normal state. It is very obvious from Figure 6.3 that our algorithm detects a drought in the US in the 1930s, while the threshold algorithm does not provide any meaningful results.

We then compute the connected components from the integer solution and find that the component starting from the year 1928 corresponds to the Dust Bowl region. This connected component spans over 13 years (1928-1940). The drought region we detect is shown in Figure 6.4. For each location in the drought region, we count the number of years it is included in the connected component and divide this number by the number of years that the component spans. Figure 6.4 shows the resulting ratio associated with each location in a color. We find that the Dust Bowl map is similar to the one in [92].

We also carefully examine the solution provided by the threshold algorithm. Since the threshold algorithm fails to capture the spatial and temporal consistency, we find that the nodes with drought states are isolated from each other and do not form sizable connected components. In contrast, the MRF model encourages neighboring nodes to be in same states and, as a result, the solution by the drought detection algorithm has less number of 'drought' locations, as shown in Figure 6.3.

To show the drought region detected by the algorithm is valid, we also draw a time series plot (Figure 6.5) of the average precipitation of the Dust Bowl region we identify. It is clear to see that a sudden reduction in precipitation occurred around 1930 and the drought lasts for about 10 years.

6.3.2 The Sahel Region

In this subsection, we show the results of detection of the 30-year drought in the Sahel starting in the late 1960s [31]. We extract the precipitation of the Sahel region and run the drought detection algorithm on this sub-dataset. We also use the 15%

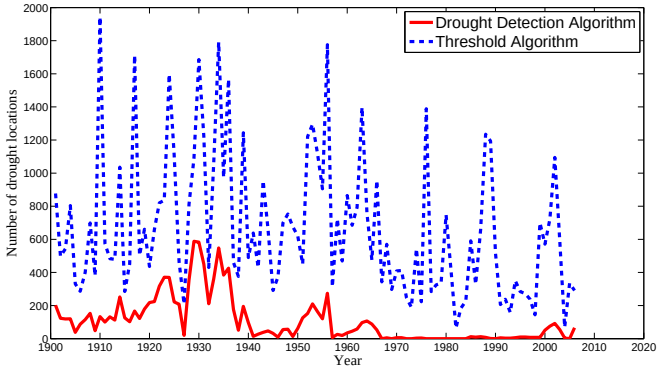


Figure 6.3: The number of locations with drought states in the United States detected by the drought detection and threshold algorithms.

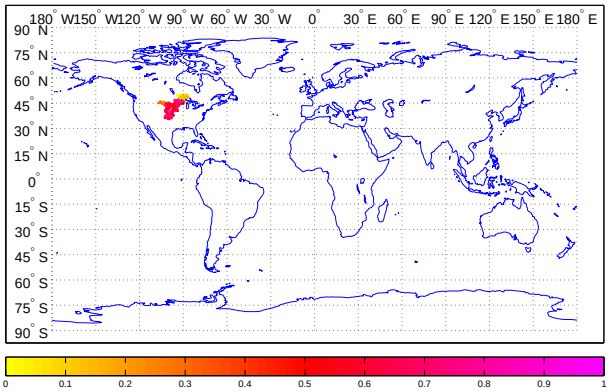


Figure 6.4: The dust bowl drought region, which corresponds to the connected component starting in 1928.

percentile. Figure 6.6 shows the number of locations with drought states detected by both algorithms. It is not surprising that the base line algorithm detects many more drought locations than our algorithm. We find that the largest component corresponds to the prolonged drought in the Sahel region starting in the late 1960s. The connected component spans 31 years (1968 to 1998). The drought region we detect is shown in Figure 6.7 and the color code is the same as the one used in Figure 6.4. We find that the drought map is similar to the one in [53]. The time series plot (Figure 6.8) shows the 30-year precipitation reduction in the area.

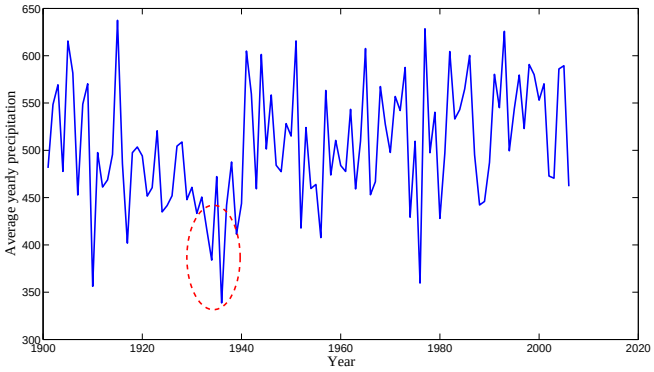


Figure 6.5: The time series plot of the average precipitation of the Dust Bowl region. The red ellipse shows the sudden reduction in precipitation in the 1930s.

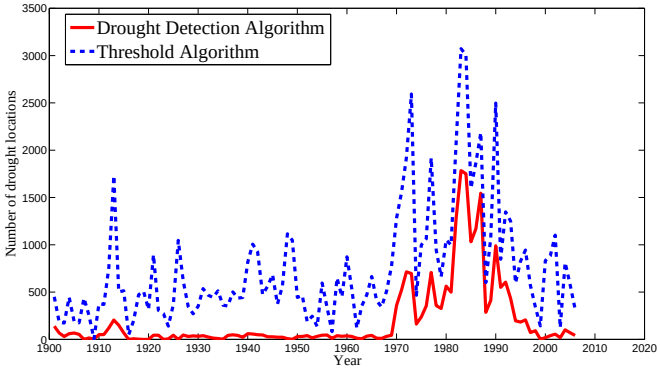


Figure 6.6: The number of locations with drought states in the Sahel region detected by the drought detection and threshold algorithms.

6.3.3 Global Data

Finally, we apply our drought detection algorithm to the entire CRU dataset. We want to emphasize that since the CRU dataset has a high resolution and the underlying 3-dimensional grid has 7,146,520 nodes (67,420 nodes per year $\times$ 106 years), the drought detection problem is of large scale.

Since the global precipitation exhibits large variance, a uniform percentile may not be sufficient. To take this into account, for each location, we compute the median precipitation over the 106 years and run the k-means clustering algorithm on the medians with $k = 9$. Figure 6.9 shows the 9 clusters over the globe. Intuitively, the locations in

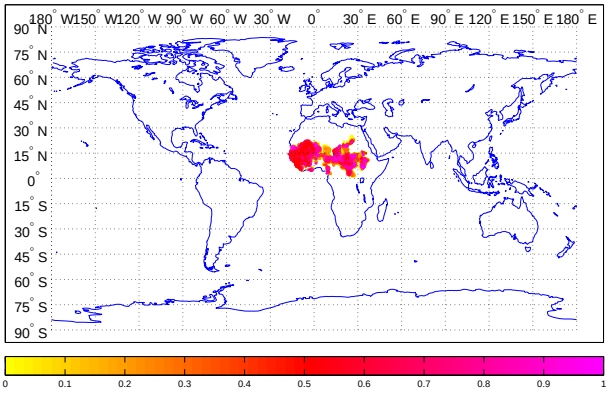


Figure 6.7: The prolonged drought in the Sahel region, which corresponds to the largest connected component starting in 1968.

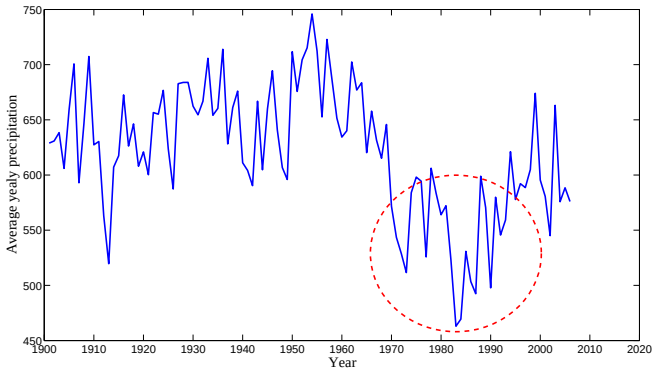


Figure 6.8: The time series plot of the average precipitation of the Sahel drought region. The red ellipse shows the decades long reduction in precipitation.

the clusters with low precipitation are more likely to experience droughts than those in the clusters with high precipitation. Thus, we sort the clusters according to the mean precipitation in descending order and set $p = 15$ for the first three clusters, $p = 10$ for the next three clusters and $p = 15$ for the rest.

After obtaining the integer solution from the pseudomarginals, we compute the connected components. Since a significant drought should be spatially widespread over a long duration, we first select the largest 200 connected components and then further pick among them the ones which last for more than 5 years. We consider the resulting connected components as major droughts.

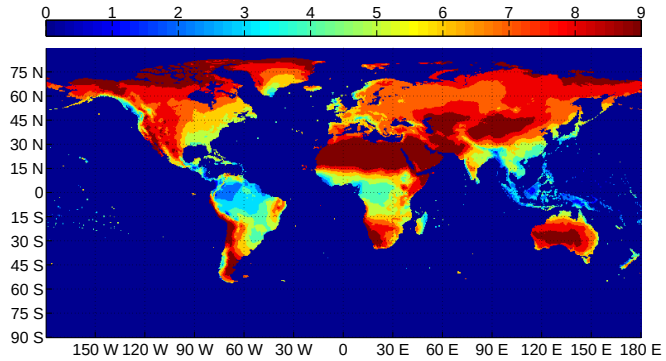
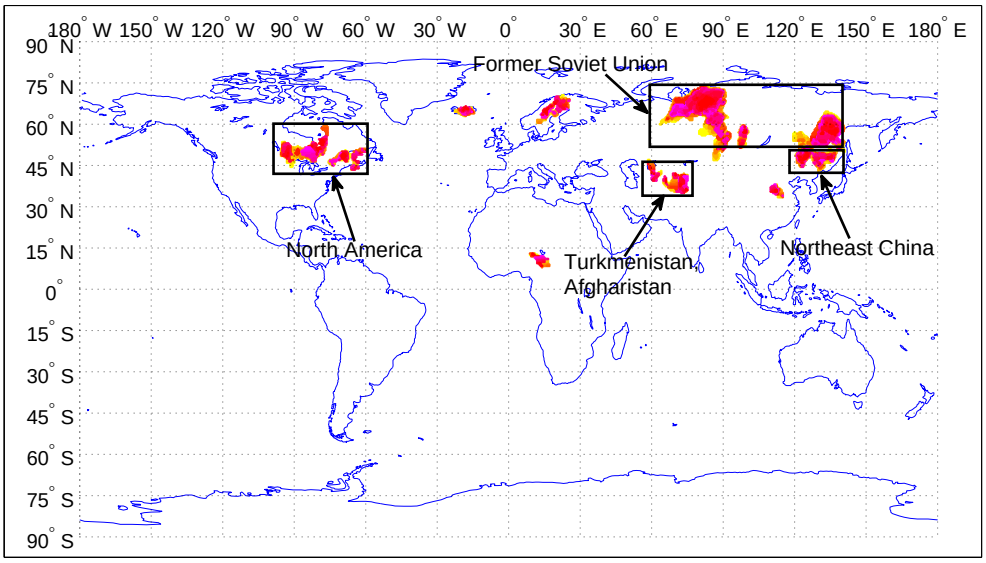


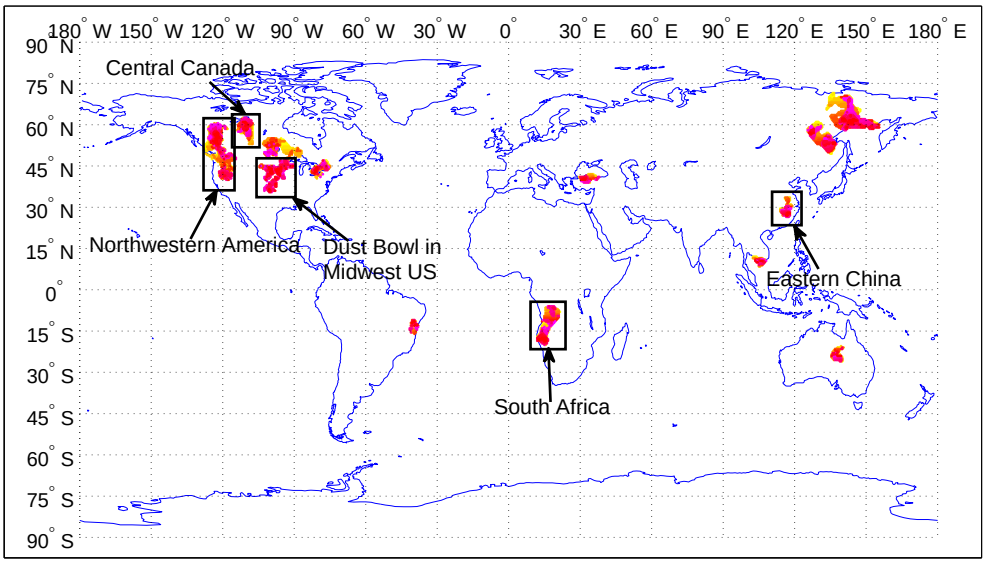
Figure 6.9: The k-means clustering on the medians with $k = 9$. Each color represents a different cluster. The cluster in dark red (cluster index 9) indicates the lowest precipitation while the blue cluster (cluster index 1) indicates the highest precipitation.

Our algorithm runs efficiently on the CRU dataset and converges within one and a half hours in a Linux workstation. The significant droughts are shown in Figure 6.10-6.13 and each sub-figure shows the droughts beginning in a particular decade. Besides the three-decade drought in the Sahel region starting in the late 1960s and the Dust Bowl in the 1930s, the algorithm also detects the drought in the southwest US and northern Mexico in the 1950s, the region's most severe drought of the 20th century [39]. Other detected strong droughts include: the drought in northeastern China in the 1920s, the drought in Kazakhstan in the 1930s, the drought in west Europe in the 1940s, the drought in Iran in the 1950s, the drought in eastern India and Bangladesh in the 1960s and the drought in southern Africa in the 1980s. We find that most of the droughts have a duration of at least 10 years. We also find that the drought regions are mostly located in the arid and semi-arid regions and this observation is consistent with many climate modeling studies [31, 90].

In [80], the authors list the top 30 regions of the world with abrupt decreases in rainfall during the 20th century. Our algorithm discovers all the droughts in the list, except for a drought in Ukraine and two droughts in Australia. The droughts found by both algorithms are shown by black rectangles in Figure 6.10 and 6.13.

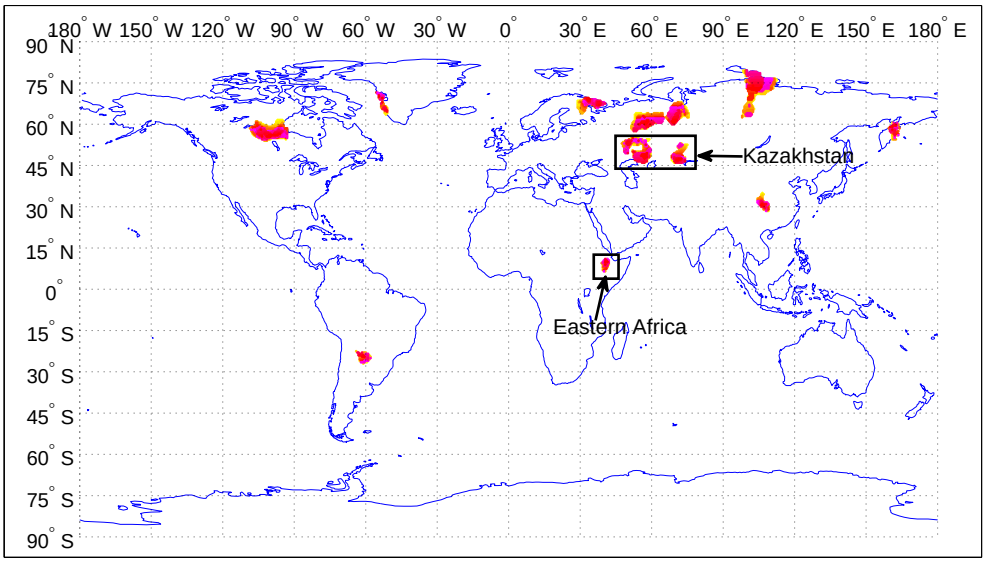


(a) Drought starting within the period1905-1920.

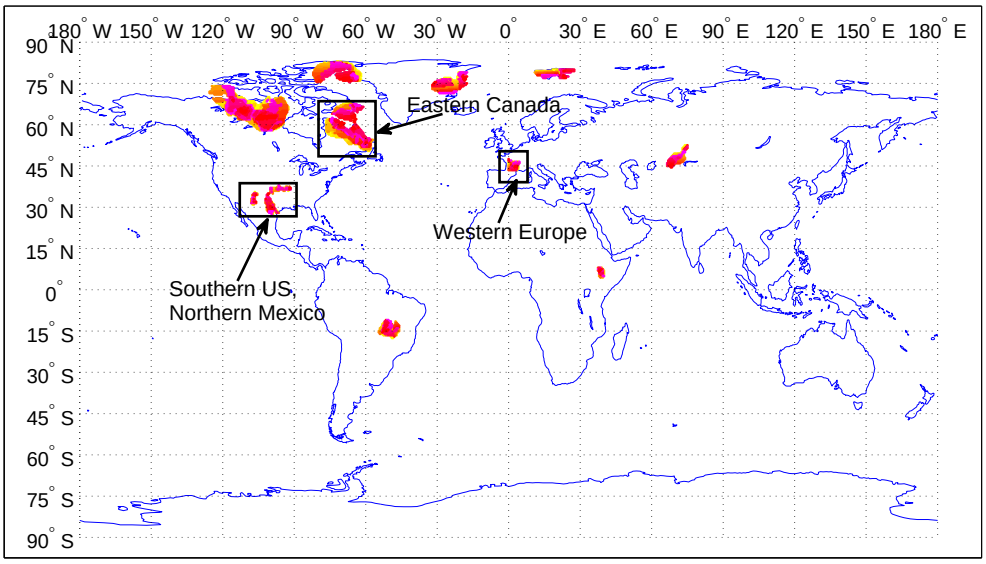


(b) Drought starting within the period 1921-1930.

Figure 6.10: The drought regions detected by our algorithm. Each figure shows the drought starting from a particular decade. The region in black rectangles indicate the common drought found by [80].

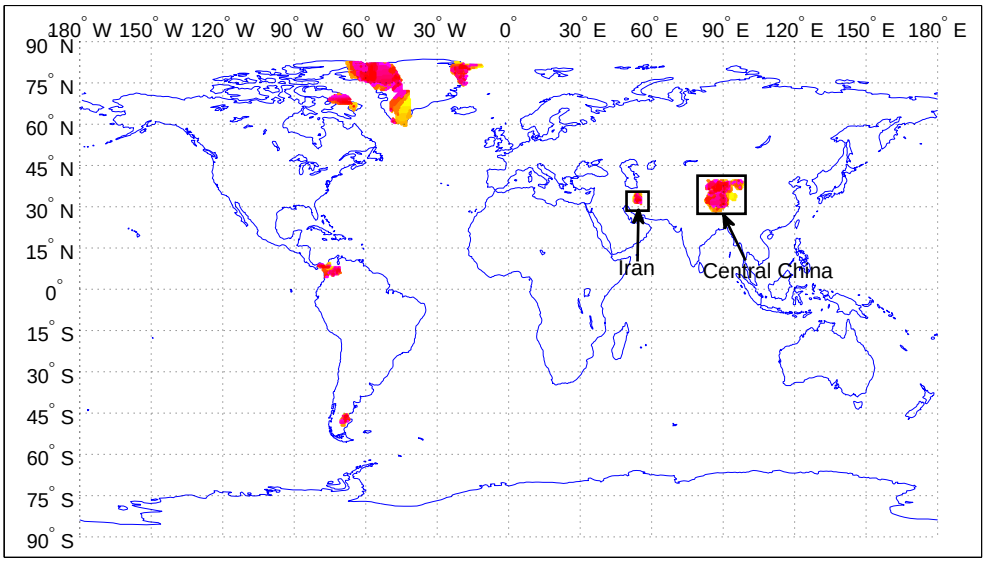


(a) Drought starting within the period 1931-1940.

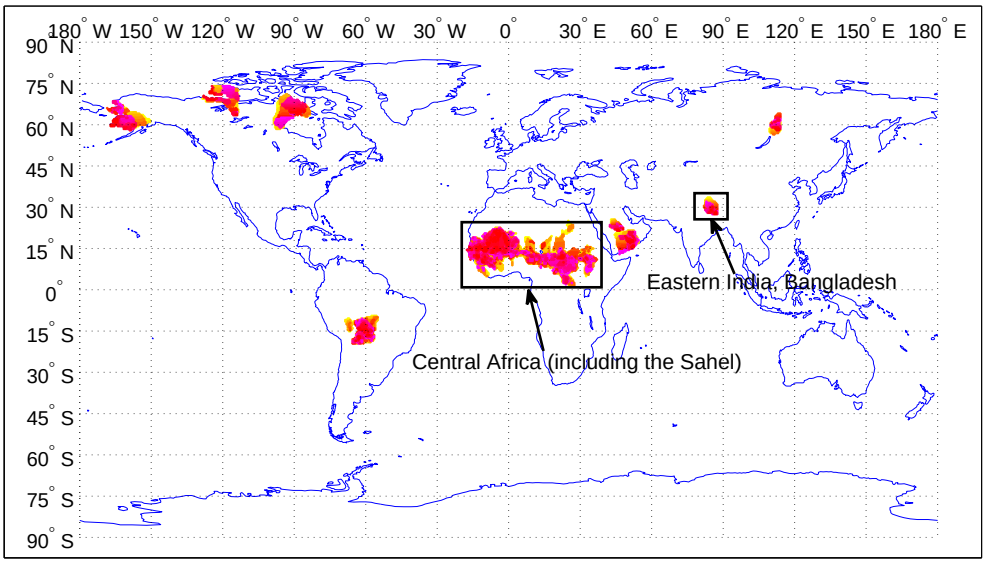


(b) Drought starting within the period 1941-1950.

Figure 6.11: The drought regions detected by our algorithm. Each figure shows the drought starting from a particular decade. The region in black rectangles indicate the common drought found by [80].

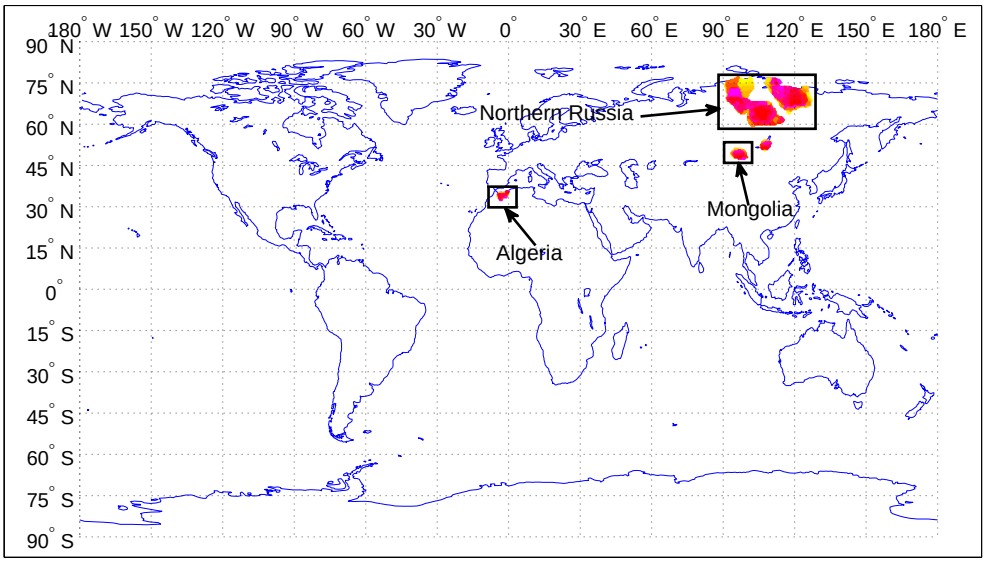


(a) Drought starting within the period 1951-1960.

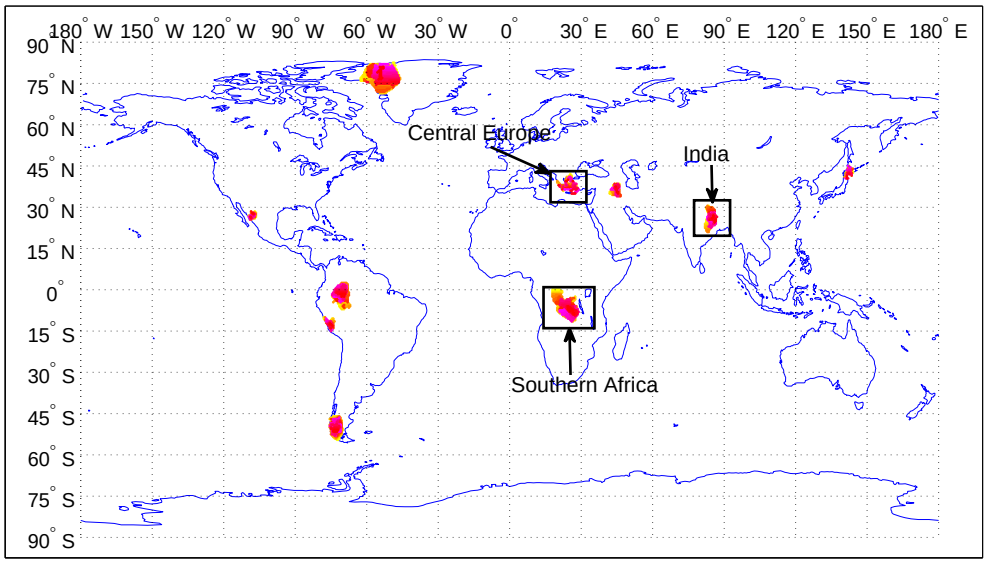


(b) Drought starting within the period 1961-1970.

Figure 6.12: The drought regions detected by our algorithm. Each figure shows the drought starting from a particular decade. The region in black rectangles indicate the common drought found by [80].



(a) Drought starting within the period 1971-1980.



(b) Drought starting within the period 1981-1995.

Figure 6.13: The drought regions detected by our algorithm. Each figure shows the drought starting from a particular decade. The region in black rectangles indicate the common drought found by [80].

Chapter 7

Bethe-ADMM for Tree Decomposition based Parallel MAP Inference

7.1 Motivation

Given a discrete graphical model with known structure and parameters, the problem of finding the most likely configuration of the states is known as the MAP inference problem [103]. Existing approaches to solving MAP inference problems on graphs with cycles often consider a graph-based LP relaxation of the integer program [21, 85, 67] .

To solve the graph-based LP relaxation problem, two main classes of algorithms have been proposed. The first class of algorithms are dual LP algorithms [42, 55, 62, 96, 97, 101], which uses the dual decomposition and solves the dual problem. The two main approaches to solving the dual problems are block coordinate descent [42] and sub-gradient algorithms [62]. The coordinate descent algorithms are empirically faster, however, they may not reach the dual optimum since the dual problem is not strictly convex. Recent advances in coordinate descent algorithms perform tree-block updates [97, 101]. The sub-gradient methods, which are guaranteed to converge to the global optimum, can be slow in practice. For a detailed discussion on dual MAP algorithms, we refer the readers to [96]. The second class of algorithms are primal LP

algorithms like the proximal algorithm [85]. The advantage of such algorithms is that it can choose different Bregman divergences as proximal functions which can take the graph structure into account. However, the proximal algorithms do not have a closed form update at each iteration in general and thus lead to double-loop algorithms.

As solving MAP inference in large scale graphical models is becoming increasingly important, in recent work, parallel MAP inference algorithms [71, 75] based on the ADMM [17] have been proposed. As a primal-dual algorithm, ADMM combines the advantage of dual decomposition and the method of multipliers, which is guaranteed to converge globally and at a rate of $O(1/T)$ even for non-smooth problems [104]. ADMM has also been successfully used to solve large scale problem in a distributed manner [17].

Design of efficient parallel algorithms based on ADMM by problem decomposition has to consider a key tradeoff between the number of subproblems and the size of each subproblem. Having several simple subproblems makes solving each problem easy, but one has to maintain numerous dual variables to achieve consensus. On the other hand, having a few subproblems makes the number of constraints small, but each subproblem needs an elaborate often iterative algorithm, yielding a double-loop. Existing ADMM based algorithms for MAP inference [71, 75] decompose the problem into several simple subproblems, often based on single edges or local factors, so that the subproblems are easy to solve. However, to enforce consensus among the shared variables, such methods have to use dual variables proportional to the number of edges or local factors, which can make convergence slow on large graphs.

To overcome the limitations of existing ADMM methods for MAP inference, we propose a novel parallel algorithm based on tree decomposition. The individual trees need not be spanning and thus includes both edge decomposition and spanning tree decomposition as special cases. Compared to edge decomposition, tree decomposition has the flexibility of increasing the size of subproblems and reducing the number of subproblems by considering the graph structure. Compared to the tree block coordinate descent [97], which works with one tree at a time, our algorithm updates all trees in parallel. Note that the tree block coordinate descent algorithm in [101] updates disjoint trees within a forest in parallel, whereas our updates consider overlapping trees in parallel.

However, tree decomposition raises a new problem: the subproblems cannot be

solved efficiently in the ADMM framework and requires an iterative algorithm, yielding a double-loop algorithm [71, 85]. To efficiently solve the subproblem on a tree, we propose a novel inexact ADMM algorithm called Bethe-ADMM, which uses a Bregman divergence induced by Bethe entropy on a tree, instead of the standard quadratic divergence, as the proximal function. The resulting subproblems on each tree can be solved exactly in linear time using the sum-product algorithm [63]. However, the proof of convergence for the standard ADMM does not apply to Bethe-ADMM. We prove global convergence of Bethe-ADMM and establish a $O(1/T)$ convergence rate, which is the same as the standard ADMM [49, 104]. Overall, Bethe-ADMM overcomes the limitations of existing ADMM based MAP inference algorithms [71, 75] and provides the flexibility required in designing efficient parallel algorithm through: (i) Tree decomposition, which can take the graph structure into account and greatly reduce the number of variables participating in the consensus and (ii) the Bethe-ADMM algorithm, which yields efficient updates for each subproblem.

We compare the performance of Bethe-ADMM with existing methods on both synthetic and real datasets and illustrate three aspects. First, Bethe-ADMM is faster than existing primal LP methods in terms of convergence. Second, Bethe-ADMM is competitive with existing dual methods in terms of quality of solutions obtained. Third, in certain graphs, tree decomposition leads to faster convergence than edge decomposition for Bethe-ADMM.

The rest of the chapter is organized as follows: We review the ADMM-based MAP inference problem in Section 7.2. In Section 7.3, we introduce the Bethe-ADMM algorithm and prove its global convergence. We discuss empirical evaluation in Section 7.4.

7.2 Related Work

In recent years, ADMM [71, 75] has been used to solve large scale MAP inference problems. To solve (5.16) using ADMM, we need to split nodes or/and edges and introduce equality constraints to enforce consensus among the shared variables. The algorithm in [71] adopts edge decomposition and introduces equality constraints for shared nodes. Let d_i be the degree of node i. The number of equality constraints in [71] is $O(\sum_{i=1}^{|V|} d_i k)$, which is approximately equal to $O(|E|k)$. For binary pairwise MRFs,

the subproblems for the ADMM in [71] have closed-form solutions. For multi-valued MRFs, however, one has to first binarize the MRFs which introduces additional $|V|k$ variables for nodes and $2|E|k^2$ variables for edges. The binarization process increases the number of factors to $O(|V| + 2|E|k)$ and the complexity of solving each subproblem increases to $O(|E|k^2 \log k)$. We note that in a recent work [70], the active set method is employed to solve the quadratic problem for arbitrary factors. A generalized variant of [71] which does not require binarization is presented in [75]. We refer to this algorithm as Primal ADMM and use it as a baseline in Section 8.2. Although each subproblem in primal ADMM can be efficiently solved, the number of equality constraints and dual variables is $O(2|E|k+|E|k^2)$. In [75], ADMM is also used to solve the dual of (5.6). We refer to this algorithm as the Dual ADMM algorithm and use it as a baseline in Section 7.4. The dual ADMM works for multi-valued MRFs and has a linear time algorithm for each subproblem, but the number of equality constraint is $O(2|E|k + |E|k^2)$.

7.3 Algorithm and Analysis

We first show how to solve (5.16) using ADMM based on tree decomposition. The resulting algorithm can be a double-loop algorithm since some updates do not have closed form solutions. We then introduce the Bethe-ADMM algorithm where every subproblem can be solved exactly and efficiently, and analyze its convergence properties.

7.3.1 ADMM for MAP Inference

We first show how to decompose (5.16) into a series of subproblems. We can decompose the graph G into overlapping subgraphs and rewrite the optimization problem with consensus constraints to enforce the pseudomarginals on subgraphs (local variables) to agree with $\boldsymbol{\mu}$ (global variable). Throughout the paper, we focus on tree-structured decompositions. To be more specific, let $\mathbb{T} = \{(V_1, E_1), \ldots, (V_{|\mathbb{T}|}, E_{|\mathbb{T}|})\}$ be a collection of subgraphs of G which satisfies two criteria: (i) Each subgraph $\tau = (V_\tau, E_\tau)$ is a tree-structured graph and (ii) Each node $u \in V$ and each edge $(u, v) \in E$ is included in at least one subgraph $\tau \in \mathbb{T}$. We also introduce local variable $\boldsymbol{m}_\tau \in L(\tau)$ which is the pseudomarginal [21, 67] defined on each subgraph τ. We use $\boldsymbol{\theta}_\tau$ to denote the potentials on subgraph τ. We denote $\boldsymbol{\mu}_\tau$ as the components of global variable $\boldsymbol{\mu}$ that

belong to subgraph τ. Note that since $\boldsymbol{\mu} \in L(G)$ and τ is a tree-structured subgraph of G, $\boldsymbol{\mu}_\tau$ always lies in $L(\tau)$. In the newly formulated optimization problem, we will impose consensus constraints for shared nodes and edges. For the ease of exposition, we simply use the equality constraint $\boldsymbol{\mu}_\tau = \boldsymbol{m}_\tau$ to enforce the consensus.

The new optimization problem we formulate based on graph decomposition is then as follows:

$$\min_{\boldsymbol{m}_\tau, \boldsymbol{\mu}} \quad \sum_{\tau=1}^{|\mathbb{T}|} \rho_\tau \langle \boldsymbol{m}_\tau, \boldsymbol{\theta}_\tau \rangle \tag{7.1}$$

$$\text{subject to} \quad \boldsymbol{m}_\tau - \boldsymbol{\mu}_\tau = 0, \quad \tau = 1, \ldots, |\mathbb{T}| \tag{7.2}$$

$$\boldsymbol{m}_\tau \in L(\tau), \quad \tau = 1, \ldots, |\mathbb{T}| \tag{7.3}$$

where ρ_τ is a positive constant associated with each subgraph. We use the consensus constraints (7.2) to make sure that the pseudomarginals agree with each other in the shared components across all the tree-structured subgraphs. Besides the consensus constraints, we also impose feasibility constraints (7.3), which guarantee that, for each subgraph, the local variable $\boldsymbol{m}_\tau$ lies in $L(\tau)$. When the constraints (7.2) and (7.3) are satisfied, the global variable $\boldsymbol{\mu}$ is guaranteed to lie in $L(G)$.

To make sure that problem (5.16) and (7.1)-(7.3) are equivalent, we also need to guarantee that

$$\min_{\boldsymbol{m}_\tau} \sum_{\tau=1}^{|\mathbb{T}|} \rho_\tau \langle \boldsymbol{m}_\tau, \boldsymbol{\theta}_\tau \rangle = \max_{\boldsymbol{\mu}} \langle \boldsymbol{\mu}, \boldsymbol{f} \rangle \ , \tag{7.4}$$

assuming the constraints (7.2) and (7.3) are satisfied. It is easy to verify that, as long as (7.4) is satisfied, the specific choice of ρ_τ and $\boldsymbol{\theta}_\tau$ do not change the problem. Let $\mathbf{1}[.]$ be a binary indicator function and $\boldsymbol{l} = -\boldsymbol{f}$. For any positive $\rho_\tau, \forall \tau \in \mathbb{T}$, e.g., $\rho_\tau = 1$, a simple approach to obtaining the potential $\boldsymbol{\theta}_\tau$ can be:

$$\theta_{\tau,u}(x_u) = \frac{l_u(x_u)}{\sum_{\tau'} \rho_{\tau'} \mathbf{1}[u \in V_{\tau'}]}, u \in V_\tau \ ,$$

$$\theta_{\tau,uv}(x_u, x_v) = \frac{l_{uv}(x_u, x_v)}{\sum_{\tau'} \rho_{\tau'} \mathbf{1}[(u,v) \in E_{\tau'}]}, (u,v) \in E(\tau) \ .$$

Let $\boldsymbol{\lambda}_\tau$ be the dual variable and $\beta > 0$ be the penalty parameter. The following

updates constitute a single iteration of the ADMM [17]:

$$\boldsymbol{m}_\tau^{t+1} = \underset{\boldsymbol{m}_\tau \in L(\tau)}{\operatorname{argmin}} \langle \boldsymbol{m}_\tau, \rho_\tau \boldsymbol{\theta}_\tau + \boldsymbol{\lambda}_\tau^t \rangle + \frac{\beta}{2} ||\boldsymbol{m}_\tau - \boldsymbol{\mu}_\tau^t||_2^2 \ , \tag{7.5}$$

$$\boldsymbol{\mu}^{t+1} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{\tau=1}^{|\mathbb{T}|} \left(-\langle \boldsymbol{\mu}_\tau, \boldsymbol{\lambda}_\tau^t \rangle + \frac{\beta}{2} ||\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau||_2^2 \right) , \tag{7.6}$$

$$\boldsymbol{\lambda}_\tau^{t+1} = \boldsymbol{\lambda}_\tau^t + \beta(\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}) \ . \tag{7.7}$$

In the tree based ADMM (7.5)-(7.7), the equality constraints are only required for shared nodes and edges. Assume there are m shared nodes and the shared node v_i has C_i^v copies and there are n shared edges and the shared edge e_j has C_j^e copies. The total number of equality constraints is $O(\sum_{i=1}^m C_i^v k + \sum_{j=1}^n C_j^e k^2)$. A special case of tree decomposition is edge decomposition, where only nodes are shared. In edge decomposition, $n = 0$ and the number of equality constraints is $O(\sum_{i=1}^m C_i^v k)$, which is approximately equal to $O(|E|k)$ and similar to [71]. In general, the number of shared nodes and edges in tree decomposition is much smaller than that in edge decomposition. The smaller number of equality constraints usually lead to faster convergence in achieving consensus. Now, the problem turns to whether the updates (7.5) and (7.6) can be solved efficiently, which we analyze below:

Updating $\boldsymbol{\mu}$: Since we have an unconstrained optimization problem (7.6) and the objective function decomposes component-wisely, taking the derivatives and setting them to zero yield the solution. In particular, let S_u be the set of subgraphs which contain node u, for the node components, we have:

$$\mu_u^{t+1}(x_u) = \frac{1}{|S_u|\beta} \sum_{\tau \in S_u} \big(\beta m_{\tau,u}^{t+1}(x_u) + \lambda_{\tau,u}^t(x_u) \big) \ . \tag{7.8}$$

(7.8) can be further simplified by observing that $\sum_{\tau \in S_u} \lambda_{\tau,u}^t(x_u) = 0$ [17]:

$$\mu_u^{t+1}(x_u) = \frac{1}{|S_u|} \sum_{\tau=1}^{T} m_{\tau,u}^{t+1}(x_u) \ . \tag{7.9}$$

Let S_{uv} be the subgraphs which contain edge (u, v). The update for the edge components can be similarly derived as:

$$\mu_{u,v}^{t+1}(x_u, x_v) = \frac{1}{|S_{uv}|} \sum_{\tau \in S_{uv}} m_{\tau,uv}^{t+1}(x_u, x_v) \ . \tag{7.10}$$

Updating $\boldsymbol{m}_\tau$: For (7.5), we need to solve a quadratic optimization problem for each tree-structured subgraph. Unfortunately, we do not have a close-form solution for (7.5) in general. One possible approach, similar to the proximal algorithm, is to first obtain the solution $\tilde{\boldsymbol{m}}_\tau$ to the unconstrained problem of (7.5) and then project $\tilde{\boldsymbol{m}}_\tau$ to $L(\tau)$:

$$\boldsymbol{m}_\tau = \underset{\boldsymbol{m} \in L(\tau)}{\operatorname{argmin}} ||\boldsymbol{m} - \tilde{\boldsymbol{m}}_\tau||_2^2 \ . \tag{7.11}$$

If we adopt the cyclic Bregman projection algorithm [18] to solve (7.11), the algorithm becomes a double-loop algorithm, i.e., the cyclic projection algorithm projects the solution to each individual constraint of $L(\tau)$ until convergence and the projection algorithm itself is iterative. We refer to this algorithm as the Exact ADMM and use it as a baseline in Section 7.4.

7.3.2 Bethe-ADMM

Instead of solving (7.5) exactly, a common way in inexact ADMMs [57, 110] is to linearize the objective function in (7.5), i.e., the first order Taylor expansion at $\boldsymbol{m}_\tau^t$, and add a new quadratic penalty term such that

$$\boldsymbol{m}_\tau^{t+1} = \underset{\boldsymbol{m}_\tau \in L(\tau)}{\operatorname{argmin}} \langle \mathbf{y}_\tau^t, \boldsymbol{m}_\tau - \boldsymbol{m}_\tau^t \rangle + \frac{\alpha}{2} \|\boldsymbol{m}_\tau - \boldsymbol{m}_\tau^t\|_2^2 \ , \tag{7.12}$$

where α is a positive constant and

$$\mathbf{y}_\tau^t = \rho_\tau \boldsymbol{\theta}_\tau + \boldsymbol{\lambda}_\tau^t + \beta(\boldsymbol{m}_\tau^t - \boldsymbol{\mu}_\tau^t) \ . \tag{7.13}$$

However, as discussed in the previous section, the quadratic problem (7.12) is generally difficult for a tree-structured graph and thus the conventional inexact ADMM does not lead to an efficient update for $\boldsymbol{m}_\tau$. By taking the tree structure into account, we propose an inexact minimization of (7.5) augmented with a Bregman divergence induced by the Bethe entropy. We show that the resulting proximal problem can be solved exactly and efficiently using the sum-product algorithm [63]. We prove that the global convergence of the Bethe-ADMM algorithm in Section 7.3.3.

The basic idea in the new algorithm is that we replace the quadratic term in (7.12) with a Bregman-divergence term $d_\phi(\boldsymbol{m}_\tau || \boldsymbol{m}_\tau^t)$ such that

$$\boldsymbol{m}_\tau^{t+1} = \underset{\boldsymbol{m}_\tau \in L(\tau)}{\operatorname{argmin}} \langle \mathbf{y}_\tau^t, \boldsymbol{m}_\tau - \boldsymbol{m}_\tau^t \rangle + \alpha d_\phi(\boldsymbol{m}_\tau || \boldsymbol{m}_\tau^t) \ , \tag{7.14}$$

is efficient to solve for any tree τ. Expanding the Bregman divergence and removing the constants, we can rewrite (7.14) as

$$\boldsymbol{m}_\tau^{t+1} = \underset{\boldsymbol{m}_\tau \in L(\tau)}{\operatorname{argmin}} \langle \mathbf{y}_\tau^t/\alpha - \nabla\phi(\boldsymbol{m}_\tau^t), \boldsymbol{m}_\tau\rangle + \phi(\boldsymbol{m}_\tau). \tag{7.15}$$

For a tree-structured problem, what convex function $\phi(\boldsymbol{m}_\tau)$ should we choose? Recall that $\boldsymbol{m}_\tau$ defines the marginal distributions of a tree-structured distribution $p_{\boldsymbol{m}_\tau}$ over the nodes and edges:

$$\boldsymbol{m}_{\tau,u}(x_u) = \sum_{\neg x_u} p_{\boldsymbol{m}_\tau}(x_1, \ldots, x_u, \ldots, x_n),\ \forall u \in V_\tau\ ,$$
$$\boldsymbol{m}_{\tau,uv}(x_u, x_v) = \sum_{\neg x_u, \neg x_v} p_{\boldsymbol{m}_\tau}(x_1, \ldots, x_u, x_v, \ldots, x_n),\ \forall (uv) \in E_\tau.$$

It is well known that the sum-product algorithm [63] efficiently computes the marginal distributions for a tree structured graph. It can also be shown that the sum-product algorithm solves the following optimization problem [103] for tree τ for some constant $\boldsymbol{\eta}_\tau$:

$$\max_{\boldsymbol{m}_\tau \in L(\tau)} \langle \boldsymbol{m}_\tau, \boldsymbol{\eta}_\tau\rangle + H_{Bethe}(\boldsymbol{m}_\tau)\ , \tag{7.16}$$

where $H_{Bethe}(\boldsymbol{m}_\tau)$ is the Bethe entropy of $\boldsymbol{m}_\tau$ defined as:

$$H_{Bethe}(\boldsymbol{m}_\tau) = \sum_{u \in V_\tau} H_u(\boldsymbol{m}_{\tau,u}) - \sum_{(u,v)\in E_\tau} I_{uv}(\boldsymbol{m}_{\tau,uv})\ , \tag{7.17}$$

where $H_u(\boldsymbol{m}_{\tau,u})$ is the entropy function on each node $u \in V_\tau$ and $I_{uv}(\boldsymbol{m}_{\tau,uv})$ is the mutual information on each edge $(u, v) \in E_\tau$.

Combing (7.15) and (7.16), we set $\boldsymbol{\eta}_\tau = \nabla\phi(\boldsymbol{m}_\tau^t) - \mathbf{y}_\tau^t/\alpha$ and choose ϕ to be the negative Bethe entropy of $\boldsymbol{m}_\tau$ so that (7.15) can be solved efficiently in linear time via the sum-product algorithm.

For the sake of completeness, we summarize the Bethe-ADMM algorithm as follows :

$$\boldsymbol{m}_\tau^{t+1} = \underset{\boldsymbol{m}_\tau \in L(\tau)}{\operatorname{argmin}} \langle \mathbf{y}_\tau^t/\alpha - \nabla\phi(\boldsymbol{m}_\tau^t), \boldsymbol{m}_\tau\rangle + \phi(\boldsymbol{m}_\tau)\ , \tag{7.18}$$

$$\boldsymbol{\mu}^{t+1} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{\tau=1}^{T} \left(-\langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau\rangle + \frac{\beta}{2} ||\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau||_2^2 \right), \tag{7.19}$$

$$\boldsymbol{\lambda}_\tau^{t+1} = \boldsymbol{\lambda}_\tau^t + \beta(\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1})\ , \tag{7.20}$$

where $\mathbf{y}_\tau^t$ is defined in (7.13) and $-\phi$ is defined in (7.17).

7.3.3 Convergence

We prove the global convergence of the Bethe-ADMM algorithm. We first bound the Bregman divergence d_ϕ:

Lemma 1 *Let $\boldsymbol{\mu}_\tau$ and $\boldsymbol{\nu}_\tau$ be two concatenated vectors of the pseudomarginals on a tree τ with n_τ nodes. Let $d_\phi(\boldsymbol{\mu}_\tau||\boldsymbol{\nu}_\tau)$ be the Bregman divergence induced by the negative Bethe entropy ϕ. Assuming $\alpha \geq \max_\tau\{\beta(2n_\tau-1)^2\}$, we have*

$$\alpha d_\phi(\boldsymbol{\mu}_\tau||\boldsymbol{\nu}_\tau) \geq \frac{\beta}{2}\|\boldsymbol{\mu}_\tau - \boldsymbol{\nu}_\tau\|_2^2 \ . \tag{7.21}$$

Proof: Let $P_\tau(\boldsymbol{x})$ be a tree-structured distribution on a tree $\tau = (V_\tau, E_\tau)$, where $|V_\tau| = n_\tau$ and $|E_\tau| = n_\tau - 1$. The pseudomarginal $\boldsymbol{\mu}_\tau$ has a total of $2n_\tau - 1$ components, each being a marginal distribution. In particular, there are n_τ marginal distributions corresponding to each node $u \in V_\tau$, given by

$$\mu_{\tau,u}(x_u) = \sum_{\neg x_u} P_\tau(x_1, \ldots, x_u, \ldots, x_n) \quad . \tag{7.22}$$

Thus, $\boldsymbol{\mu}_u$ is the marginal probability for node u.

Further, there are $n_\tau - 1$ marginal components corresponding to each edge $(u, v) \in E_\tau$, given by

$$\mu_{\tau,uv}(x_u, x_v) = \sum_{\neg(x_u, x_v)} P(x_1, \ldots, x_u, \ldots, x_v, \ldots, x_n) \ . \tag{7.23}$$

Thus, $\boldsymbol{\mu}_{uv}$ is the marginal probability for nodes (u, v).

Let $\boldsymbol{\mu}_\tau, \boldsymbol{\nu}_\tau$ be two pseudomarginals defined on tree τ and $P_{\boldsymbol{\mu}_\tau}, P_{\boldsymbol{\nu}_\tau}$ be the corresponding tree-structured distributions. Making use of (7.22), we have

$$\|P_{\boldsymbol{\mu}_\tau} - P_{\boldsymbol{\nu}_\tau}\|_1 \geq \|\boldsymbol{\mu}_{\tau,u} - \boldsymbol{\nu}_{\tau,u}\|_1, \quad \forall u \in V_\tau \ . \tag{7.24}$$

Similarly, for each edge, we have the following inequality because of (7.23)

$$\|P_{\boldsymbol{\mu}_\tau} - P_{\boldsymbol{\nu}_\tau}\|_1 \geq \|\boldsymbol{\mu}_{\tau,uv} - \boldsymbol{\nu}_{\tau,uv}\|_1, \ \forall (u, v) \in E_\tau \ . \tag{7.25}$$

Adding them together gives

$$(2n_\tau - 1)\|P_{\boldsymbol{\mu}_\tau} - P_{\boldsymbol{\nu}_\tau}\|_1 \geq \|\boldsymbol{\mu}_\tau - \boldsymbol{\nu}_\tau\|_1 \geq \|\boldsymbol{\mu}_\tau - \boldsymbol{\nu}_\tau\|_2 \ . \tag{7.26}$$

According to Pinsker's inequality [19], we have

$$d_\phi(\boldsymbol{\mu}_\tau||\boldsymbol{\nu}_\tau) = KL(P_{\boldsymbol{\mu}_\tau}, P_{\boldsymbol{\nu}_\tau}) \geq \frac{1}{2}\|P_{\boldsymbol{\mu}_\tau} - P_{\boldsymbol{\nu}_\tau}\|_1^2 \geq \frac{1}{2(2n_\tau - 1)^2}\|\boldsymbol{\mu}_\tau - \boldsymbol{\nu}_\tau\|_2^2 \ . \quad (7.27)$$

Multiplying α on both sides and letting $\alpha \geq \beta(2n_\tau - 1)^2$ complete the proof. ■

To prove the convergence of the objective function, we define a residual term R_τ^{t+1} as

$$R_\tau^{t+1} = \rho_\tau \langle \boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^*, \boldsymbol{\theta}_\tau \rangle \ , \quad (7.28)$$

where $\boldsymbol{\mu}_\tau^*$ is the optimal solution for tree τ. We show that R_τ^{t+1} satisfies the following inequality:

Lemma 2 *Let $\{\boldsymbol{m}_\tau, \boldsymbol{\mu}_\tau, \boldsymbol{\lambda}_\tau\}$ be the sequences generated by Bethe-ADMM. Assume $\alpha \geq \max_\tau\{\beta(2n_\tau - 1)^2\}$. For any $\boldsymbol{\mu}_\tau^* \in L(\tau)$, we have*

$$\begin{aligned} R_\tau^{t+1} \leq & \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle + \alpha\big(d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^t) - d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^{t+1})\big) \\ & + \frac{\beta}{2}\big(\|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^t\|_2^2 - \|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^t\|_2^2 - \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2\big) \ , \end{aligned} \quad (7.29)$$

where R_τ^{t+1} is defined in (7.28).

Proof: Since $\boldsymbol{m}_\tau^{t+1}$ is the optimal solution for (7.18), for any $\boldsymbol{\mu}_\tau^* \in L(\tau)$, we have the following inequality:

$$\langle \mathbf{y}_r^t + \alpha(\nabla\phi(\boldsymbol{m}_\tau^{t+1}) - \nabla\phi(\boldsymbol{m}_\tau^t)), \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle \geq 0 \ . \quad (7.30)$$

Substituting (7.13) into (7.30) and rearranging the terms, we have

$$\begin{aligned} R_\tau^{t+1} \leq & \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle + \beta \langle \boldsymbol{m}_\tau^t - \boldsymbol{\mu}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle \\ & + \alpha \langle \nabla\phi(\boldsymbol{m}_\tau^{t+1}) - \nabla\phi(\boldsymbol{m}_\tau^t), \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle \ . \end{aligned} \quad (7.31)$$

The second term in the RHS of (7.31) is equivalent to

$$\begin{aligned} 2\langle \boldsymbol{m}_\tau^t - \boldsymbol{\mu}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1} \rangle = & \|\boldsymbol{m}_\tau^t - \boldsymbol{m}_\tau^{t+1}\|_2^2 \\ + \|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^t\|_2^2 - \|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^t\|_2^2 - & \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2 . \end{aligned} \quad (7.32)$$

The third term in the RHS of (7.31) can be rewritten as

$$\begin{aligned}&\langle \nabla\phi(\boldsymbol{m}_\tau^{t+1}) - \nabla\phi(\boldsymbol{m}_\tau^{t}), \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\rangle \\ &= d_\phi(\boldsymbol{\mu}_\tau^* || \boldsymbol{m}_\tau^t) - d_\phi(\boldsymbol{\mu}_\tau^* || \boldsymbol{m}_\tau^{t+1}) - d_\phi(\boldsymbol{m}_\tau^{t+1} || \boldsymbol{m}_\tau^t). \end{aligned} \tag{7.33}$$

Substituting (7.32) and (7.33) into (7.31) and using Lemma 1 complete the proof. ∎

We next show that the first term in the RHS of (7.29) satisfies the following result:

Lemma 3 *Let $\{\boldsymbol{m}_\tau, \boldsymbol{\mu}_\tau, \boldsymbol{\lambda}_\tau\}$ be the sequences generated by Bethe-ADMM. For any $\boldsymbol{\mu}_\tau^* \in L(\tau)$, we have*

$$\sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\rangle \le \frac{1}{2\beta}(\|\boldsymbol{\lambda}_\tau^t\|_2^2 - \|\boldsymbol{\lambda}_\tau^{t+1}\|_2^2) + \frac{\beta}{2}\left(\|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\|_2^2 - \|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1}\|_2^2\right) .$$

Proof: Let μ_i be the ith component of $\boldsymbol{\mu}$. We augment $\boldsymbol{\mu}_\tau, \boldsymbol{m}_\tau$ and $\boldsymbol{\lambda}_\tau$ in the following way: If $\boldsymbol{\mu}_i$ is not a component of $\boldsymbol{\mu}_\tau$, we set $\boldsymbol{\mu}_{\tau,i} = 0, \boldsymbol{m}_{\tau,i} = 0$ and $\boldsymbol{\lambda}_{\tau,i} = 0$; otherwise, they are the corresponding components from $\boldsymbol{\mu}_\tau, \boldsymbol{m}_\tau$ and $\boldsymbol{\lambda}_\tau$ respectively. We can then rewrite (7.19) in the following equivalent component-wise form:

$$\mu_i^{t+1} = \underset{\mu_i}{\operatorname{argmin}} \sum_{\tau=1}^{|\mathbb{T}|} \left(\langle \lambda_{\tau,i}^t, m_{\tau,i}^{t+1} - \mu_{\tau,i}\rangle + \frac{\beta}{2} ||m_{\tau,i}^{t+1} - \mu_{\tau,i}||_2^2 \right).$$

For any $\boldsymbol{\mu}_\tau^* \in L(\tau)$, we have the following optimality condition:

$$-\sum_{\tau=1}^{|\mathbb{T}|} \langle \lambda_{\tau,i}^t + \beta(m_{\tau,i}^{t+1} - \mu_{\tau,i}^{t+1}), \mu_{\tau,i}^* - \mu_{\tau,i}^{t+1}\rangle \ge 0 . \tag{7.34}$$

Combining all the components of $\boldsymbol{\mu}^{t+1}$, we can rewrite (7.34) in the following vector form:

$$-\sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t + \beta(\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}), \boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1}\rangle \ge 0 . \tag{7.35}$$

Rearranging the terms yields

$$\sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\rangle$$

$$\leq \sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1} \rangle - \sum_{\tau=1}^{|\mathbb{T}|} \beta \langle \boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}, \boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1} \rangle$$
$$= \sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1} \rangle + \frac{\beta}{2} \sum_{\tau=1}^{|\mathbb{T}|} \big(\|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\|_2^2$$
$$- \|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1}\|_2^2 - \|\boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1}\|_2^2 \big) \ . \tag{7.36}$$

Recall $\boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1} = \frac{1}{\beta}(\boldsymbol{\lambda}_\tau^t - \boldsymbol{\lambda}_\tau^{t+1})$ in (7.20), then

$$\langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1} \rangle - \frac{\beta}{2} \|\boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{m}_\tau^{t+1}\|_2^2 = \frac{1}{2\beta} (\|\boldsymbol{\lambda}_\tau^t\|_2^2 - \|\boldsymbol{\lambda}_\tau^{t+1}\|_2^2) \ . \tag{7.37}$$

Plugging (7.37) into (7.36) completes the proof. ∎

We also need the following lemma:

Lemma 4 *Let $\{\boldsymbol{m}_\tau, \boldsymbol{\mu}_\tau, \boldsymbol{\lambda}_\tau\}$ be the sequences generated by Bethe-ADMM. Then*

$$\sum_{\tau=1}^{|\mathbb{T}|} \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2 \geq \sum_{\tau=1}^{|\mathbb{T}|} \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}\|_2^2 + \|\boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2 \ .$$

Proof: According to (7.20), (7.35) can be rewritten as

$$- \sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^{t+1}, \boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1} \rangle \geq 0 \ , \tag{7.38}$$

which holds for any $\boldsymbol{\mu}_\tau^* \in L(\tau)$. Setting $\boldsymbol{\mu}_\tau^* = \boldsymbol{\mu}_\tau^t$ yields

$$- \sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^{t+1}, \boldsymbol{\mu}_\tau^t - \boldsymbol{\mu}_\tau^{t+1} \rangle \geq 0 \ . \tag{7.39}$$

Similarly, we have

$$- \sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t \rangle \geq 0 \ . \tag{7.40}$$

Adding (7.39) and (7.40) them together yields

$$\sum_{\tau=1}^{|\mathbb{T}|} \langle \boldsymbol{\lambda}_\tau^{t+1} - \boldsymbol{\lambda}_\tau^t, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t \rangle \geq 0 \ . \tag{7.41}$$

Recall that $\boldsymbol{\lambda}_\tau^{t+1} - \boldsymbol{\lambda}_\tau^t = \beta(\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1})$ in (7.20), we have

$$\begin{aligned}0 \leq & \sum_{\tau=1}^{|\mathbb{T}|} \beta \langle \boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}, \boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t \rangle \\ = & \frac{\beta}{2} \sum_{\tau=1}^{|\mathbb{T}|} \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2 - \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^{t+1}\|_2^2 - \|\boldsymbol{\mu}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2.\end{aligned}$$

Rearranging the first two terms completes the proof. ∎

Theorem 1 *Assume the following hold: (1)* $\boldsymbol{m}_\tau^0$ *and* $\boldsymbol{\mu}_\tau^0$ *are uniform tree-structured distributions,* $\forall \tau = 1, \ldots, |\mathbb{T}|$ *(2)* $\boldsymbol{\lambda}_\tau^0 = \mathbf{0}, \forall \tau = 1, \ldots, |\mathbb{T}|$*; (3)* $\max_\tau d_\phi(\boldsymbol{\mu}_\tau^* || \boldsymbol{m}_\tau^0) = D_{\boldsymbol{\mu}}$*; (4)* $\alpha \geq \max_\tau \{\beta(2n_\tau - 1)^2\}$ *holds. Denote* $\bar{\boldsymbol{m}}_\tau^T = \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{m}_\tau^t$ *and* $\bar{\boldsymbol{\mu}}_\tau^T = \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\mu}_\tau^t$*. For any* T *and the optimal solution* $\boldsymbol{\mu}^*$*, we have*

$$\sum_{\tau=1}^{|\mathbb{T}|} \left(\rho_\tau \langle \bar{\boldsymbol{m}}_\tau^T - \boldsymbol{\mu}_\tau^*, \theta_\tau \rangle + \frac{\beta}{2} \|\bar{\boldsymbol{m}}_\tau^T - \bar{\boldsymbol{\mu}}_\tau^T\|_2^2 \right) \leq \frac{D_{\boldsymbol{\mu}} \alpha |\mathbb{T}|}{T} \ .$$

Proof: Summing (7.29) over τ from 1 to $|\mathbb{T}|$ and using Lemma 3, we have:

$$\begin{aligned}& \sum_{\tau=1}^{|\mathbb{T}|} \left(R_\tau^{t+1} + \frac{\beta}{2} \|\boldsymbol{m}_\tau^{t+1} - \boldsymbol{\mu}_\tau^t\|_2^2 \right) \\ \leq & \sum_{\tau=1}^{|\mathbb{T}|} \frac{1}{2\beta} (\|\boldsymbol{\lambda}_\tau^t\|_2^2 - \|\boldsymbol{\lambda}_\tau^{t+1}\|_2^2) + \frac{\beta}{2} \left(\|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^t\|_2^2 - \|\boldsymbol{\mu}_\tau^* - \boldsymbol{\mu}_\tau^{t+1}\|_2^2 \right) \\ & + \frac{\beta}{2} \left(\|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^{t+1}\|_2^2 - \|\boldsymbol{\mu}_\tau^* - \boldsymbol{m}_\tau^t\|_2^2 \right) + \alpha \left(d_\phi(\boldsymbol{\mu}_\tau^* || \boldsymbol{m}_\tau^t) - d_\phi(\boldsymbol{\mu}_\tau^* || \boldsymbol{m}_\tau^{t+1}) \right) \ . \end{aligned} \tag{7.42}$$

Summing over the above from $t = 0$ to $T - 1$, we have

$$\begin{aligned}
&\sum_{t=0}^{T-1}\sum_{\tau=1}^{|\mathbb{T}|}\left(R_\tau^{t+1}+\frac{\beta}{2}\|\boldsymbol{m}_\tau^{t+1}-\boldsymbol{\mu}_\tau^t\|_2^2\right)\\
&\leq\sum_{\tau=1}^{|\mathbb{T}|}\frac{1}{2\beta}(\|\boldsymbol{\lambda}_\tau^0\|_2^2-\|\boldsymbol{\lambda}_\tau^T\|_2^2)+\frac{\beta}{2}\big(\|\boldsymbol{\mu}_\tau^*-\boldsymbol{\mu}_\tau^0\|_2^2-\|\boldsymbol{\mu}_\tau^*-\boldsymbol{\mu}_\tau^T\|_2^2\big)\\
&+\frac{\beta}{2}\left(\|\boldsymbol{\mu}_\tau^*-\boldsymbol{m}_\tau^T\|_2^2-\|\boldsymbol{\mu}_\tau^*-\boldsymbol{m}_\tau^0\|_2^2\right)+\alpha\left(d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^0)-d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^T)\right)\\
&\leq\sum_{\tau=1}^{|\mathbb{T}|}\frac{\beta}{2}\|\boldsymbol{\mu}_\tau^*-\boldsymbol{m}_\tau^T\|_2^2+\alpha\left(d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^0)-d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^T)\right)\\
&\leq\sum_{\tau=1}^{|\mathbb{T}|}\alpha d_\phi(\boldsymbol{\mu}_\tau^*||\boldsymbol{m}_\tau^0)\ , \qquad (7.43)
\end{aligned}$$

where we use Lemma 1 to derive (7.43). Applying Lemma 4 and Jensen's inequality yield the desired bound. ∎

Theorem 1 establishes the $O(1/T)$ convergence rate for the Bethe-ADMM in ergodic sense. As $T \to \infty$, the objective value $\sum_{\tau=1}^{|\mathbb{T}|}\rho_\tau\langle\bar{\boldsymbol{m}}_\tau^T,\theta_\tau\rangle$ converges to the optimal value and the equality constraints are also satisfied.

7.3.4 Extension to MRFs with General Factors

Although we present Bethe-ADMM in the context of pairwise MRFs, it can be easily generalized to handle MRFs with general factors. For a general MRF, we can view the dependency graph as a factor graph [63], a bipartite graph $G = (V \cup F, E)$, where V and F are disjoint set of variable nodes and factor nodes and E is a set of edges, each connecting a variable node and a factor node. The distribution $P(\boldsymbol{x})$ takes the form: $P(\boldsymbol{x}) \propto \exp\left\{\sum_{u\in V} f_u(x_u) + \sum_{\alpha\in F} f_\alpha(\boldsymbol{x}_\alpha)\right\}$. The relaxed LP for general MRFs can be constructed in a similar fashion with that for pairwise MRFs.

We can then decompose the relaxed LP to subproblems defined on factor trees and impose equality constraints to enforce consistency on the shared variables among the subproblems. Each subproblem can be solved efficiently using the sum-product algorithm for factor trees and the Bethe-ADMM algorithm for general MRFs bears similar structure with that for pairwise MRFs.

7.4 Experimental Results

We compare the Bethe-ADMM algorithm with several other state-of-the-art MAP inference algorithms. We show the comparison results with primal based MAP inference algorithms in Section 7.4.1 and dual based MAP inference algorithm in Section 7.4.2 respectively. We also show in Section 7.4.3 how tree decomposition benefits the performance of Bethe-ADMM. We run experiments in Section 7.4.1-7.4.3 using sequential updates.

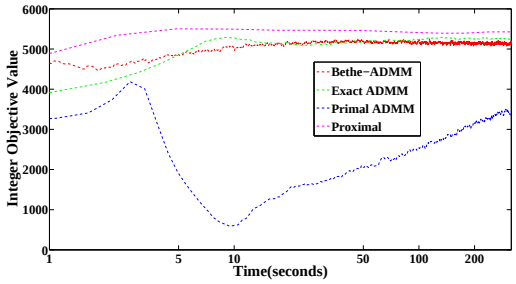
7.4.1 Comparison with Primal based Algorithms

We compare the Bethe-ADMM algorithm with the proximal algorithm [85], Exact ADMM algorithm and Primal ADMM algorithm [75]. For the proximal algorithm, we choose the Bregman divergence as the sum of KL-divergences across all node and edge distributions. Following the methodology in [85], we terminate the inner loop if the maximum constraint violation of $L(G)$ is less than 10^{-3} and set $w^t = t$. Similarly, in applying the Exact ADMM algorithm, we terminate the loop for solving $\boldsymbol{m}_\tau$ if the maximum constraint violation of $L(\tau)$ is less than 10^{-3}. For the Exact ADMM and Bethe-ADMM algorithm, we use 'edge decomposition': each τ is simply an edge of the graph and $|\mathbb{T}| = |E|$. To obtain the integer solution, we use node-based rounding: $x_u^* = \text{argmax}_{x_u} \mu_u(x_u)$.

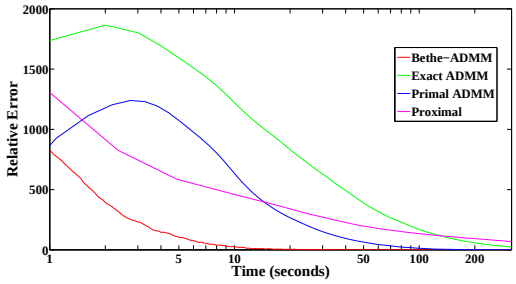
We show experimental results on two synthetic datasets. The underlying graph of each dataset is a three dimensional $m \times n \times t$ grid. We generate the potentials as follows: We set the nodewise potentials as random numbers from $[-a, a]$, where $a > 0$. We set the edgewise potentials according to the Potts model, i.e., $\theta_{uv}(x_u, x_v) = b_{uv}$ if $x_u = x_v$ and 0 otherwise. We choose b_{uv} randomly from $[-1, 1]$. The edgewise potentials penalize disagreement if $b_{uv} > 0$ and penalize agreement if $b_{uv} < 0$. We generate datasets using $m = 20, n = 20, t = 16, k = 6$ with varying a.

Figure 7.1(a) shows the plots of (5.6) on one synthetic dataset and we find that the algorithms have similar performances on other simulation datasets. We observe that all algorithms converge to the optimal value $\langle \boldsymbol{\mu}^*, \boldsymbol{f} \rangle$ of (5.16) and we plot the relative error with respect to the optimal value $|\langle \boldsymbol{\mu}^* - \boldsymbol{\mu}_t, \boldsymbol{f} \rangle|$ on the two datasets in Figure 8.2(a) and 8.2(b).

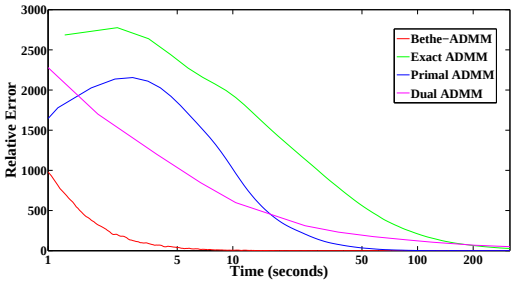
Overall, the Bethe-ADMM algorithm converges faster than other primal algorithms. We observe that the proximal algorithm and Exact ADMM algorithm are the slowest, due to the sequential projection step. In terms of the decoded integer solution, the Bethe-ADMM, Exact ADMM and proximal algorithm have similar performances. We also note that a higher objective function value does not necessarily lead to a better decoded integer solution.



(a) Rounded solution with $a = 0.5$.



(b) Relative error with $a = 0.5$.



(c) Relative error with $a = 1$.

Figure 7.1: Results of Bethe-ADMM, Exact ADMM, Primal ADMM and proximal algorithms on two simulation datasets. Figure 7.1(a) plots the value of the decoded integer solution as a function of runtime (seconds). Figure 8.2(a) and 8.2(b) plot the relative error with respect to the optimal LP objective as a function of runtime (seconds). For Bethe-ADMM, we set $\alpha = \beta = 0.05$. For Exact ADMM, we set $\beta = 0.05$. For Primal ADMM, we set $\beta = 0.5$. Bethe-ADMM converges faster than other primal based algorithms.

7.4.2 Comparison with Dual based Algorithms

In this section, we compare the Bethe-ADMM algorithm with the MPLP algorithm [42] and the Dual ADMM algorithm [75]. We conduct experiments on protein design problems [111]. In these problems, we are given a 3D structure and the goal is to find a sequence of amino-acids that is the most stable for that structure. The problems are modeled by nodewise and pairwise factors and can be posed as finding a MAP assignment for the given model. This is a demanding setting in which each problem may have hundreds of variables with 100 possible states on average.

We run the algorithms on two problems with different sizes [111], i.e., 1jo8 (58 nodes and 981 edges) and 1or7 (180 nodes and 3005 edges). For the MPLP and Dual ADMM algorithm, we plot the value of the integer programming problem (5.6) and its dual.. For Bethe-ADMM algorithm, we plot the value of dual LP of (5.16) and the integer programming problem (5.6). Note that although Bethe-ADMM and Dual ADMM have different duals, their optimal values are the same. We run the Bethe-ADMM based on edge decomposition. Figure 7.3 shows the result.

We observe that the MPLP algorithm usually converges faster, but since it is a coordinate ascent algorithm, it can stop prematurely and yield suboptimal solutions. Figure 7.2 shows that on the 1fpo dataset, the MPLP algorithm converges to a suboptimal solution. We note that the convergence time of the Bethe-ADM and Dual ADM are similar. The three algorithms have similar performance in terms of the decoded integer solution.

7.4.3 Edge based vs Tree based

In the previous experiments, we use 'edge decomposition' for the Bethe-ADMM algorithm. Since our algorithm can work for any tree-structured graph decomposition, we want to empirically study how the decomposition affects the performance of the Bethe-ADMM algorithm. In the following experiments, we show that if we can utilize the graph structure when decomposing the graph, the Bethe-ADMM algorithm will have better performance compared to simply using 'edge decomposition', which does not take the graph structure into account.

We conduct experiments on synthetic datasets. We generate MRFs whose dependency graphs consist of several tree-structured graphs and cross-tree edges to introduce cycles. To be more specific, we first generate m binary tree structured MRFs each with s nodes. Then for each ordered pair of tree-structured MRFs $(i, j), 1 \leq i, j \leq m, i \neq j$, we uniformly sample n nodes from MRF i with replacement and uniformly sample n ($n \leq s$) nodes from MRF j without replacement, resulting in two node sets S_{ij} and D_{ij}. We then connect the nodes in S_{ij} and D_{ij}, denoting them as E_{ij}. We repeat this process for every pair of trees. By construction, the graph consisting of tree i, nodes in D_{ij} and edges in E_{ij}, $\forall j \neq i$ is still a tree. We will use these m augmented trees as the tree-structured subgraphs for the Bethe-ADMM algorithm. Figure 7.4 illustrates the graph generation and tree decomposition process. A simple calculation shows that for this particular tree decomposition, $O(m^2nk)$ equality constraints are maintained, while for edge decomposition, $O(msk+m^2nk)$ are maintained. When the graph has dominant tree structure, tree decomposition leads to much less number of equality constraints.

For the experiments, we run the Bethe-ADMM algorithm based on tree and edge decomposition with different values of s, keeping m and n fixed. It is easy to see that the tree structure becomes more dominant when s becomes larger. Since we observe that both algorithms first converge to the optimal value of (5.16) and then the equality constraints are gradually satisfied, we evaluate the performance by computing the maximum constraint violation of $L(G)$ at each iteration for both algorithms. The faster the constraints are satisfied, the better the algorithm is. The results are shown in Figure 7.5. When the tree structure is not obvious, the two algorithms have similar performances. As we increase s and the tree structure becomes more dominant, the difference between the two algorithms is more pronounced. We attribute the superior performance to the fact that for the tree decomposition case, much fewer number of equality constraints are imposed and each subproblem on tree can be solved efficiently using the sum-product algorithm.

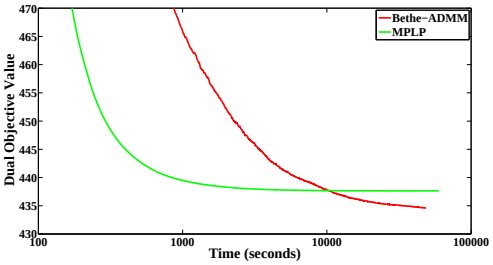
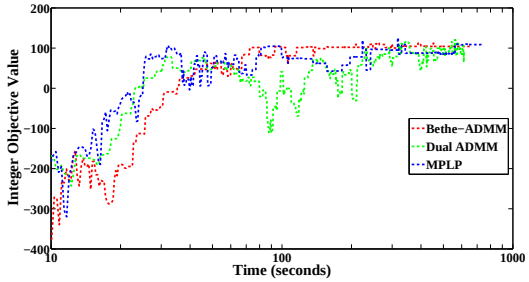
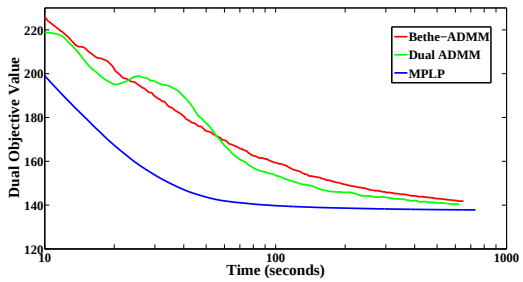


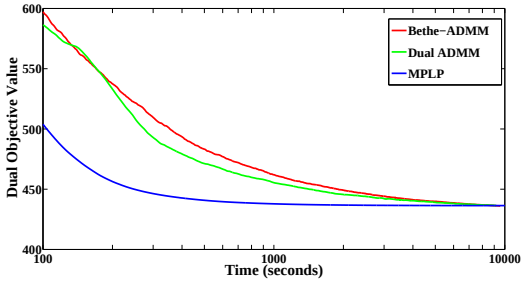
Figure 7.2: Both Bethe-ADMM and MPLP are run for sufficiently long, i.e., 50000 iterations. The dual objective value is plotted as a function of runtime (seconds). The MPLP algorithm gets stuck and does not reach the global optimum.



(a) Rounded integer solution on 1jo8.

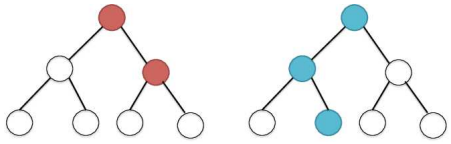


(b) Dual value on 1jo8.

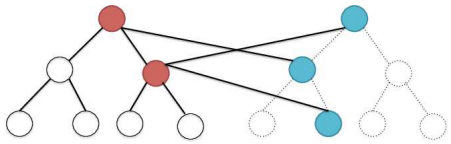


(c) Dual value on 1or7.

Figure 7.3: Results of Bethe-ADMM, MPLP and Dual ADMM algorithms on two protein design datasets. Figure 7.3(a) plots the the value of the decoded integer solution as a function of runtime (seconds). Figure 7.3(b) and 7.3(c) plot the dual value as a function of runtime (seconds). For Dual ADMM, we set $\beta = 0.05$. For Bethe-ADMM, we set $\alpha = \beta = 0.1$. Bethe-ADMM and Dual ADMM have similar performance in terms of convergence. All three methods have comparable performances for the decoded integer solution.

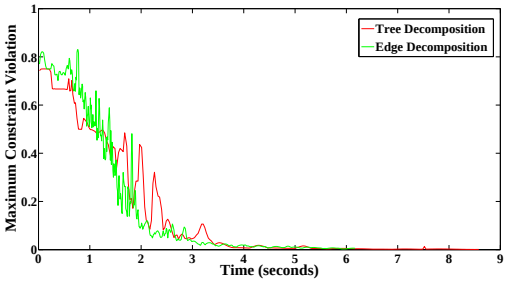


(a)

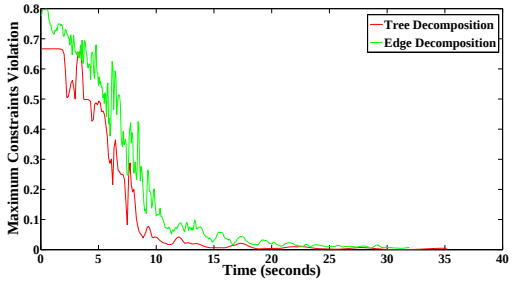


(b)

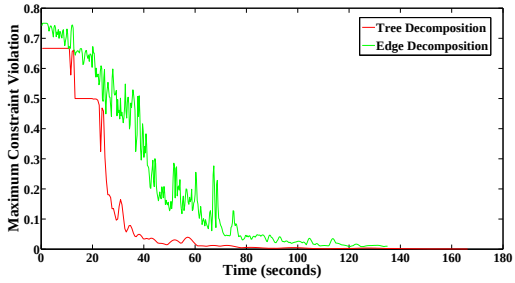
Figure 7.4: A simulation dataset with $m = 2$, $s = 7$ and $n = 3$. In 7.4(a), the red nodes (S_{12}) are sampled from tree 1 and the blue nodes (D_{12}) are sampled from tree 2. In 7.4(b) , sampled nodes are connected by cross-tree edges (E_{12}). Tree 1 with nodes in D_{12} and edges in E_{12} still form a tree, denoted by solid lines. This augmented tree is a tree-structured subgraph for Bethe-ADMM.



(a) $s = 1023$.



(b) $s = 4095$.



(c) $s = 16383$.

Figure 7.5: Results of Bethe-ADMM algorithms based on tree and edge decomposition on three simulation datasets with $m = 10, n = 20$. The maximum constraint violation in $L(G)$ is plotted as a function of runtime (seconds). For both algorithms, we set $\alpha = \beta = 0.05$. The tree based Bethe-ADMM algorithm has better performance than that of the edge based Bethe-ADMM when the tree structure is more dominant in G.

Chapter 8

Efficient MPI Implementation of the Bethe-ADMM algorithm

To illustrate the efficiency of the Bethe-ADMM algorithm, we implement it using message passing interface (MPI), which is a natural fit for the parallel algorithm given its flexible message passing mechanism, along with its portability and wide adoption in distributed and high performance clusters. The other advantage of using MPI is that its I/O interface is optimized for a wide variety of underlying parallel file systems (PFS) and sustains high I/O bandwidth. We evaluate our algorithms on a simulation and a real precipitation dataset, which are both of large scale. The empirical results show that we manage to obtain almost linear speedup in the number of cores used. The rest of this chapter is organized as follows: we discuss the MPI implementation in detail in Section 8.1 and present the experimental results in Section 8.2.

8.1 Parallel Implementation

In this section, we explain the key components of our MPI implementation in detail. Our goal is to run the Bethe-ADMM algorithm on modern high performance computers with thousands of cores and it requires us to adopt the best parallelization practice. To achieve this goal, we carefully design our MPI implementation so that the underlying parallel computing architecture can be fully utilized.

Since the update of $\boldsymbol{m}_\tau$ in (7.18) for each tree is independent, the Bethe-ADMM

algorithm is inherently parallel. In the parallel Bethe-ADMM algorithm, each process only maintains the information of a subset of trees in $\mathbb{T}$ and $\boldsymbol{m}_\tau$ is updated simultaneously. According to (7.19), the update of variable $\boldsymbol{\mu}$ involves averaging over $\boldsymbol{m}_\tau$ from the relevant trees. If these trees belong to different processes, the value of $\boldsymbol{m}_\tau$ needs to be exchanged among the processes so that $\boldsymbol{\mu}$ can be computed correctly. Because of the communication occurred among the processes, the message passing framework is a good fit for our parallel implementation. Hence, we implement the Bethe-ADMM algorithm using MPI. We also make the following implementation assumptions: (i) The MRF dependency graph is a regular grid shaped graph, e.g., two dimensional four nearest neighbor grid. (ii) Each tree structured subgraph is simply an edge of G. (iii)The input to the MAP inference algorithm is some data file, which has the potential and graph structure information.

An efficient parallel implementation is more challenging than an efficient sequential implementation. To fully utilize the computing power provided by the underlying parallel architecture, we need to address the following issues:

- How to design an efficient I/O scheme to load the data files, i.e., node potentials, edge potentials and graph structure?
- How to decompose the graph so that the work load on each process is balanced?
- How to efficiently figure out, for each process, what 'messages' it needs to exchange with other processes?

We illustrate in Fig 8.1 the key components of our MPI implementation. We take advantage of the PFS so that processes can access the data file (input.nc) in parallel. We also design a simple heuristic to partition the graph to achieve load balancing. Making use of the graph structure information, we deploy a decentralized algorithm to figure out, for each MPI process, the information it needs to exchange with other processes. After each process reads the data file in parallel to fetch the relevant nodewise and edgewise potentials, it computes the local variables $\boldsymbol{m}_\tau$, communicate with other processes and update the global variable $\boldsymbol{\mu}$.

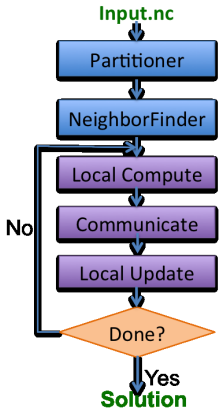


Figure 8.1: Bethe-ADMM parallel implementation.

8.1.1 Parallel File Loading

The data file used as input to the MAP inference algorithm contains the nodewise and edgewise potentials and the graph structure information. We represent the graph as a set of edges with two node ids. (Figure 8.2(a) shows an example on a simple grid graph.) A naive way to load the data file is to have a master process read the entire data file and send to other slave processes the information they need. This approach is clearly not efficient because a slave process remains idle when other slave processes receive data from the master process. Our approach is to take advantage of the PFS, which stripes a file across multiple storage devices and enables parallel access to the data file.

To be more specific, we adopt the Pnetcdf [66] file format for parallel data file loading. The Pnetcdf is suitable for our implementation because the potential data and graph structure information can be easily stored as Pnetcdf multi-dimensional arrays. A Pnetcdf file also provides a rich suite of APIs that allow users to define metadata which describe datasets in details, such as the number of nodes and edges of a given graph, the type of graphs and the dimensions of the datasets. Moreover, it integrates tightly with MPI-IO and the underlying PFS so that our algorithm can achieve high degree of parallelism in terms of I/O operations.

8.1.2 Graph Partitioning

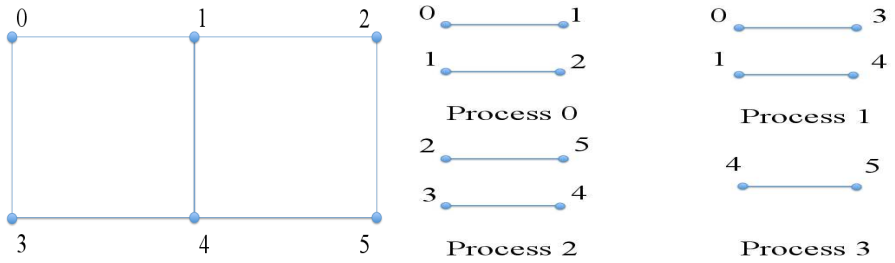
To take advantage of the parallel architecture, the work load should be split evenly among the processes and the partition should also minimize the intercommunication among the processes. This problem is usually NP hard and most practical solutions are based on heuristics. For example, in Pregel [69] and Giraph [1], the solution is to use node-centric partition, where assignment of a node to a partition depends solely on the node id. The simplest implementation is to calculate the hash value of each node id and modulus by N, where N is the number of partitions. However this simple heuristic comes with a cost that neighboring nodes are likely to be distributed on different processes and thus incur high communication overhead.

In our implementation, we adopt edge-centric partition, where we evenly divide the edges among all the processes. (Figure 8.2(b) shows the partition on a 2×3 grid.) Since the underlying dependency graph is a regular shaped grid graph, edge partition is empirically a good choice, as shown by the experimental results in Section 8.2.

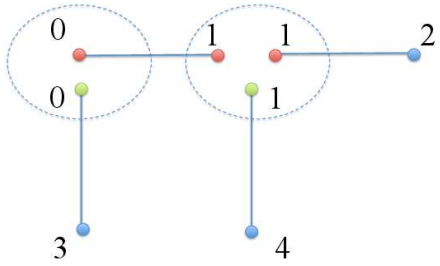
8.1.3 Inter-process communication

After the graph decomposition step, each process reads from the input Pnetcdf file, retrieve the nodewise and edgewise potentials and compute $\boldsymbol{m}_\tau$. To compute $\boldsymbol{\mu}$, a simple solution is to have a master process collect the value of $\boldsymbol{m}_\tau$ from the slave processes and compute $\boldsymbol{\mu}$ according to (7.19). After $\boldsymbol{\mu}$ is updated, the master process has to send $\boldsymbol{\mu}$ back to each slave process so that $\boldsymbol{m}_\tau$ can be computed in the next iteration. This approach is clearly not efficient and we adopt a fully distributed algorithm: each process maintains a copy of the relevant elements of $\boldsymbol{\mu}$, receive $\boldsymbol{m}_\tau$ from other processes and update $\boldsymbol{\mu}$ according to (7.19).

To apply the above distributed algorithm, each process needs to figure out the neighbor processes with which it exchanges the value of $\boldsymbol{m}_\tau$. This can be done by comparing the node ids of each process and a pair of processes need to communicate with each other if they have sharing nodes. To be more specific, we compactly represent the node list of a process as a list of pairs $\{v_i, l_i\}$, where l_i is the length of continuous ids starting from v_i. (Figure 8.2(c) illustrates the compact representation of node lists on two processes.) Each process then gathers $\{(v_i, l_i)\}$ from all other processes, compare



(a) A grid with 6 nodes and 7 edges. (b) Edge-centric partition with 4 processes.



(c) Process 0 and 1 share node 0 and 1.

Figure 8.2: 8.2(a): We label the nodes row by row and represent the graph structure as a set of edges: (0, 1), (1, 2), (0, 3), (1, 4), (2, 5), (3, 4), (4, 5). 8.2(b): We use 4 MPI processes and apply edge-centric partition. 8.2(c): The node list of process 0 can be represented as: {{0, 3}} and the node list of process 1 can be represented as: {{0, 2}, {3, 2}}. The processes share node 1 and 0. The degree of node 1 is 3. The process 0 has partial degree of 2 (red nodes) and the process 1 has degree of 1 (green node). The degree of node 0 is 2 and both processes have local degree of 0.

the lists with its own node list and decide what processes it communicates with. Beside deciding the neighbor processes, each process also needs to figure out the degrees of the sharing nodes. The degree (count) information will be used when the averaging operation is performed according to (7.19). As a result, the neighbor process also exchanges the local partial degree of the sharing nodes and compute the full degree accordingly. Algorithm 3 summaries the above procedure.

Algorithm 4 shows the details on the communication occurred among the processes: we reduce our communication cost by exchanging the partial sum of $\boldsymbol{m}_\tau$ rather than individual $\boldsymbol{m}_\tau$. We use asynchronous MPI APIs, which allows messages to be send or received asynchronously while not blocking the following operations.

Algorithm 3 NeighborFinder

```
1: idList = getNodeId()
2: pairCount = idList.size()
3: MPI_Allgather(
4:       pairCount, 1, MPI_INT,
5:       countArr, 1, MPI_INT, comm)
6: Copy idList to sendBuf
7: Construct displacementArr from countArr
8: MPI_Allgatherv(
9:       sendBuf, 2 * pairCount, MPI_INT,
10:      recvBuf, 2 * countArr, displacementArr,
11:      MPI_INT, comm)
12: Compute neighbor processes by comparing idLists
13: Count partial degree of sharing nodes
14: Exchange partial degree with neighbor processes
15: Compute full degree of sharing nodes
```

8.2 Experimental Results

In this section, we present experimental results on a simulation dataset and a precipitation dataset. Our experiments are conducted on Hopper [2], the Cray XE6 parallel machine at the National Energy Research Scientific Computing Center. Hopper is a 6384 compute node cluster where each compute node consists of two twelve-core AMD MagnyCours processors with a theoretical peak performance of 8.4 GFlop/sec per core. 6000 compute nodes have 32 GB DD3 memory each and the rest have 64 GB memory each. Hopper runs "Cray Linux Environment" (CLE) operating system which is restricted low-overhead and optimized for high performance computing. The PFS is Lustre with 156 I/O servers (OSTs). The measured peak write performance on Hopper is 35 GB per second. To maximize the possible read bandwidth, we stripe our input file across 128 stripes and the file stripe size is set to 1 MB.

8.2.1 Simulation Dataset

We show experimental results on a simulation dataset. The underlying graph is a 2 dimensional 1,000 $\times$ 10,000 grid with $k = 3$ and the potentials are random numbers in [0, 1]. The resulting MRF has 10 million nodes and approximately 20 million edges.

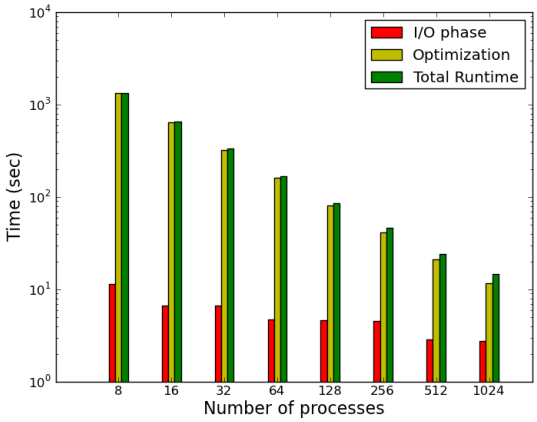
Algorithm 4 Exchange $\boldsymbol{m}_\tau$ among neighbor processes

```
1: for Each node u
2:       partial_sum[u] = 0
3: end for
4: for Each edge τ (u, v)
5:       partial_sum[u] += m_τ[u]
6:       partial_sum[v] += m_τ[v]
7: end for
8: idx = 0
9: MPI_Request request[neighbors.size() * 2]
10: for i in neighbors
11:       sharing_node = getSharingNode(i)
12:       copy partial_sum[sharing_node] to sendBuf
13:        MPI_ISend(sendBuf, k * sharing_node.size(),
14:             MPI_FLOAT, i, rank,
15:             comm, &request[idx++])
16:        MPI_IRecv(recvBuf, k * sharing_node.size(),
17:             MPI_FLOAT, i, i,
18:             comm, &requests[idx++])
19: end for
```

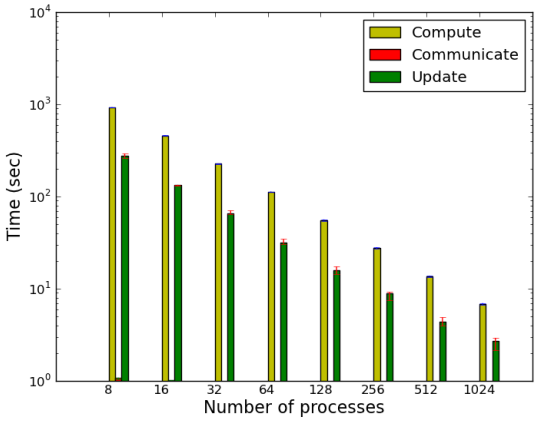
We apply the edge-centric partitioning and run the Bethe-ADMM algorithm for 100 iterations.

Figure 8.3(a) shows the run time performance using 8 to 1024 MPI processes. The algorithm runs about half an hour on 8 process, and dramatically reduces to 16 seconds on 1024 processes. The input file size is close to 1GB and data loading only takes 1.2 seconds. We attribute the speedup to our adoption of PNetCDF as well as stripping the input file across 128 OSTs.

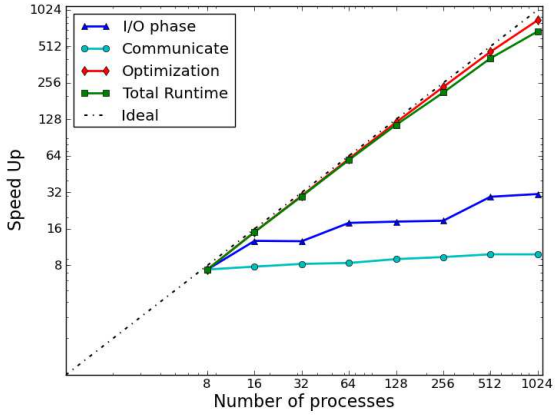
Figure 8.3(b) illustrates the average time it takes per process to compute $\boldsymbol{m}_\tau$, update $\boldsymbol{\mu}$ and communicate with neighbor processes respectively and the error bars show the minimum and maximum time spent on these three steps across all the processes. Since we evenly distribute the edges to the processes, the time spent on computing $\boldsymbol{m}_\tau$ has little fluctuation among the processes. The time to update $\boldsymbol{\mu}$, however, also depends on the number of neighbor processes and the number of shared nodes, hence the fluctuation between the min and max time among all the processes becomes more obvious as the number of processes increases. The plot shows the communication cost incurred by the



(a) Time spent on the I/O phase and the Bethe-ADMM optimization. The I/O cost is low.



(b) Time spent on the three steps of the Bethe-ADMM optimization. The communication overhead can be negligible.



(c) Almost linear speedup in the number of MPI processes

Figure 8.3: Results on the simulation dataset with 10 million nodes and 20 million edges using 8-1024 MPI processes. The I/O and communication cost is relatively low. Overall, the MPI implementation achieves almost linear speedup in the number of processes.

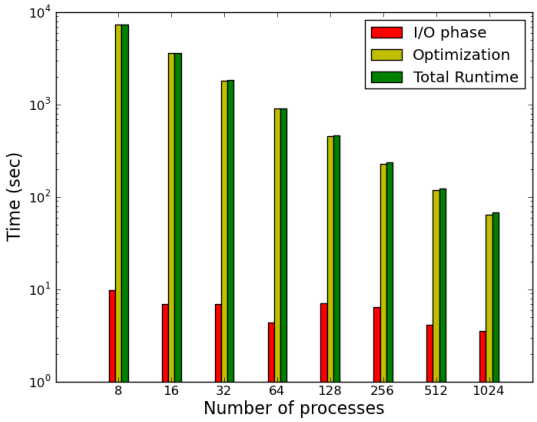
edge-centric partition is negligible. The main reason that the communication cost is so small is because when we partition the grid, we sweep row edges and column edges from top to bottom, which essentially behaves as row partitioning where each process has at most 2 neighbors and only the boundary data are exchanged.

Figure 8.3(c) shows that the Bethe-ADMM algorithm implementation achieves almost linear speedup while the speedup of the entire implementation (I/O phase + Bethe-ADMM optimization) starts to deviate from the ideal case after 256 processes. This is because as the number of MPI processes increases, each process has less work load and optimization part becomes less dominating compared with the I/O part.

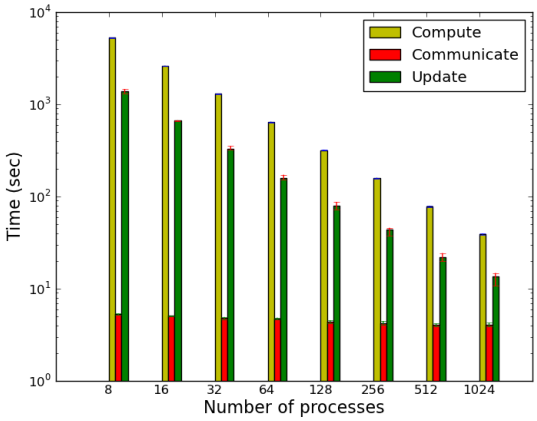
8.2.2 CRU Precipitation Dataset

In this section, we adopt Bethe-ADMM to solve the drought detection problem presented in Chapter 6 and the precipitation dataset we use is still the CRU dataset. We run the Bethe-ADMM algorithm on the CRU dataset for 500 iterations with edge-centric partitioning. The input PNetcdf file is around 530 MB. The runtime performance, as shown in Figure 8.4(a) exhibits the nice decreasing trend as it does on the simulation data. The algorithm takes less than 2 minutes to complete with 1024 MPI processes which would run more than two hours with 8 processes. The amount of time saved by our implementation is tremendous.

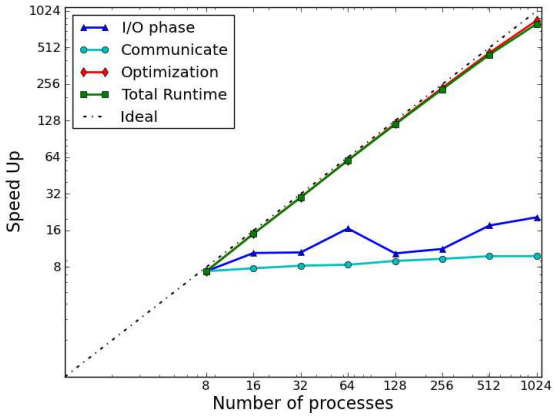
Figure 8.4(b) illustrates the average time per process to compute $\boldsymbol{m}_\tau$, communicate with neighbors and update $\boldsymbol{\mu}$ respectively. The error bars mark the minimum and maximum time spent on these three steps across all the processes. The communication cost on the CRU dataset is no longer negligible anymore. This is because the underlying 3 dimensional grid has missing nodes (CRU only has precipitation over land) and when we apply edge-centric partitioning, each process may have more than two neighbors. Hence as the number of processes increases, the number of neighbors for each process is more dynamic and the communication pattern becomes more complicated. Figure 8.4(c) plots the almost linear speedup on the CRU dataset. It also shows the trend that our implementation is scalable beyond 1024 processes. This is understandable because I/O time is only 2% of the total execution time, even at 1024 MPI processes.



(a) Time spent on the I/O phase and Bethe-ADMM optimization. The I/O cost is low.



(b) Time spent on the three steps of the Bethe-ADMM optimization. The communication overhead is low.



(c) Almost linear speedup in the number of MPI processes

Figure 8.4: Results on the CRU dataset with 7,146,520 nodes and 20,777,480 edges using 8-1024 MPI processes. The I/O and communication cost is relatively low. Overall, the MPI implementation achieves almost linear speedup in the number of processes.

Chapter 9

Conclusion and Discussion

Wh have presented our research on two important machine learning problems: overlapping clustering and MAP inference in discrete graphical models.

In Chpater 3, we have presented an overlapping clustering approach based on MMMs. The proposed MMM inherently assumes that each point is generated from a product of a subset of the component distributions. When each component distribution in a MMM is from an exponential family, we show that there is an efficient alternating maximization algorithm that converges to a (local) maxima of the joint likelihood of the observations and their assignments. We also show that when each component in a MMM is a multivariate Gaussian, we can use kernel techniques to get non-linear separators and obtain better clustering quality. In practice, the algorithms are accurate, fast, and scale to large datasets.

In Chapter 4, we have presented a systematic hierarchical generative model and corresponding algorithms for discovering uniform sub-blocks in a given data matrix. Preliminary empirical evidence goes significantly in favor of the proposed algorithm. Perhaps more importantly, the BOSC model introduces a substantially novel way to approach the problem. There are at least two important future research directions. First, the BOSC model assumes that the number of co-clusters k is known, which is typically not the case in several problems. We would like to investigate nonparametric priors, such as the Indian Buffet Process [6], to relax this assumption. Second, in spite of the effectiveness of the proposed algorithm, it is computationally slow for large datasets. In future, we would like to investigate faster approximate inference algorithms for the

BOSC model.

Due to the importance of understanding droughts, in Chapter 6, we consider the problem of their detection and develop a fully automic drought detection algorithm. We formulate the problem as the one of estimating the most likely configuration of a binary MRF, where each node can be in either a drought or normal state. We adopt the proximal maximization and Bregman cyclic projection scheme for the MAP inference task. To maintain spatio-temporal consistency, we design the potential functions to encourage the neighbouring nodes to take same values. We run the algorithm on the high resolutiuon CRU precipitation dataset and it efficiently solves this large scale problem with over 7 million variables. The empirical results shows that we successfully identify some well-documented drought regions of the last century in different parts of the world. We want to emphasize that even though we mainly run our algorithm on a precipitation dataset, the methodology is applicable to other climate variables as well. We plan to extend the current algorithm to handle multi-variate climate datasets in the future, e.g., incorporating the soil moisture variable to the model. We are also interested in applying the algorithm to the model output datasets.

In Chapter 7, we propose a provably convergent MAP inference algorithm for large scale MRFs. The algorithm is based on the tree decomposition idea from the MAP inference literature and the alternating direction method from the optimization literature. Our algorithm solves the tree structured subproblems efficiently via the sum-product algorithm and is inherently parallel. The empirical results show that the new algorithm, in its sequential version, compares favorably to other existing approximate MAP inference algorithm in terms of running time and accuracy. The experimental results on large datasets demonstrate that the parallel version scales almost linearly with the number of cores in the multi-core setting.

In Chapter 8, we discuss the MPI implementation of the Bethe-ADMM algorithm and the experimental results show that our implementation scales almost linearly with the number of MPI processes for grid-structured graphs.

References

[1] `http://giraph.apache.org`.

[2] `http://www.nersc.gov/users/computational-systems/hopper`.

[3] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. M. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the ACM International Conference on Management of Data*, 1999.

[4] C. C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the ACM International Conference on Management of Data*, 2000.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM International Conference on Management of Data*, 1998.

[6] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, January 2003.

[7] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.

[8] A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, and J. Ghosh. Model-based overlapping clustering. In *Proceedings of the 11th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 532–537, 2005.

[9] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 2002.

[10] A. Battle, E. Segal, and D. Koller. Probabilistic disocvery of overlapping cellular processes and their regulation. *Journal of Computational Biology*, 12(7):909–927, 2005.

[11] M. Bayati, D. Shah, and M. Sharma. Max-product for maximum weight matching: Convergence, correctness, and lp duality. *IEEE Transaction on Information Theory*, 54(3):1241–1251, 2008.

[12] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the 6th Annual International Conference on Computational Biology*, 2002.

[13] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, Boston, MA, 1997.

[14] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-02, University of Berkeley, 1997.

[15] D. Blei and M. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006.

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

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[18] Y. Censor and S. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.

[19] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[20] C-C. Chang and C-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[21] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):608–625, March 2005.

[22] C-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[23] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceeding of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

[24] H. Cho, I. Dhillon, Y. Yuan, and S.Sra. Minimum sum squared residue co-clustering of gene expression data. In *SIAM International Conference on Data Mining*, 2004.

[25] K. H. Cook. Climate science: The mysteries of sahel drought. *Nature Geoscience*, 1(10):647–648, 2008.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[27] M. Deodhar, H. Cho, G. Gupta, J. Ghosh, and I. Dhillon. Bregman bubble co-clustering, 2007.

[28] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[29] P. A. Dirmeyer and J. Shukla. The effect on regional and global climate of expansion of the world's deserts. *Quarterly Journal of the Royal Meteorological Society*, 122(530):451–482, 1996.

[30] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 2006.

[31] J. A. Foley, M. T. Coe, M. Scheffer, and G. Wang. Regime Shifts in the Sahara and Sahel: Interactions between Ecological and Climatic Systems in Northern Africa. *Ecosystems*, 6:524–532, 2003.

[32] G. D. Fornay. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[33] Q. Fu and A. Banerjee. Multiplicative mixture models for overlapping clustering. In *Proceedings of the IEEE International Conference on Data Mining*, 2008.

[34] Q. Fu and A. Banerjee. Bayesian overlapping subspace clustering. In *Proceedings of the IEEE International Conference on Data Mining*, 2009.

[35] Q. Fu, A. Banerjee, S. Liess, and P. K. Snyder. Drought detection of the last century: An MRF-based approach. In *Proceedings of the SIAM International Conference on Data Mining*, 2012.

[36] Q. Fu, C. Jin, H. Wang, A. Agrawal, W. Hendrix, W-K. Liao, M. Patwary, A. Banerjee, and A. Choudhary. Solving combinatorial optimization problems using relaxed linear programming: A high performance computing perspective. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2013.

[37] Q. Fu, H. Wang, and A. Banerjee. Bethe-admm for tree decomposition based parallel map inference. In *Proceedings of the 29th Conference in Uncertainty in Artificial Intelligence*, 2013.

[38] Q. Fu, H. Wang, A. Banerjee, S. Liess, and P. K. Snyder. MAP inference on million node graphical models: KL-divergence based alternating directions method. Technical report, Computer Science and Engineering Department, University of Minesota, 2012.

[39] F. K. Fye, D. W. Stahle, and E. R. Cook. Paleoclimatic analogs to twentieth-century moisture regimes across the United States. *Bulletin of the American Meteorological Society*, 84(7):901–909, 2003.

[40] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[41] Z. Ghahramani. Factorial learning and the em algorithm. In *Proceedings of the 8th Annual Conference on Neural Information Processing Systems*, 1995.

[42] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, 2007.

[43] S. Goil, H. Nagesh, and A. Choudhary. MAFIA : Efficient and scalable subspace clustering for very large data sets. Technical Report CPDC-TR-9906-010, 1999.

[44] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical Report GCNU TR 2005-001, Gatsby Computational Neuroscience Unit, 2005.

[45] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedinsg of the National Academy of Science*, 101(1):5228–5225, 2004.

[46] N. B. Guttman. Comparing the palmer drought index and the standardized precipitation index1. *JAWRA Journal of the American Water Resources Association*, 34(1):113–121, 1998.

[47] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[48] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[49] B. He and X. Yuan. On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[50] K. A. Heller and Z. Ghahramani. A nonparametric Bayesian approach to modeling overlapping clusters. In *Proceedings of the 11th International Conference on AI and Statistics*, 2007.

[51] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14, 2002.

[52] G. Hinton and R. Zemel. Autoencoders, minimum description length, and contrastive divergence. In *Proceedings of the 7th Annual Conference on Neural Information Processing Systems*, 1994.

[53] M. Hoerling, J. Hurrell, J. Eischeid, and A. Phillips. Detection and Attribution of Twentieth-Century Northern and Southern African Rainfall Change. *Journal of Climate*, 19(16):3989–4008, August 2006.

[54] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, 2001.

[55] V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of the twenty-Seventh International Conference on Machine Learning*, 2010.

[56] K. Kailing, H-P. Kriegel, and P. Kröeger. Density-connected subsapce clustering for high-dimensional data. In *SIAM International Conference on Data Mining*, 2004.

[57] S. P. Kasiviswanathan, P. Melville, A. Banerjee, and V. Sindhwani. Emerging topic detection using dictionary learning. In *Proceedings of the Twentieth ACM international conference on Information and knowledge management*, 2011.

[58] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[59] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectural biclustering of micraorray data : coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

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

[61] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, oct. 2006.

[62] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531 –552, march 2011.

[63] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[64] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[65] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[66] J. Li, W-K. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale. Parallel netcdf: A high-performance scientific I/O interface. In *Proceedings of ACM/IEEE conference on Supercomputing*, 2003.

[67] T. S. Jaakkola M. J. Wainwright and A. S. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions of Information Theory*, 51(11), 2005.

[68] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[69] G. Malewicz, M. H. Austern, A. J.C Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM International Conference on Management of Data*, 2010.

[70] A. F. Martins. *The Geometry of Constrained Structured Prediction: Applications to Inference and Learning of Natural Language Syntax*. PhD thesis, Carnegie Mellon University, 2012.

[71] A. F. Martins, P. M. Aguiar, M. A. Figueiredo, N. A. Smith, and E. P. Xing. An augmented Lagrangian approach to constrained MAP inference. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, 2011.

[72] R. J. Mceliece, D. J. C. Mackay, and J. F. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communications*, 16:140–152, 1998.

[73] G. Mclachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2000.

[74] G. J. McLachlan and T. Krishnan. *The EM algorithm and Extensions*. Wiley-Interscience, 1996.

[75] O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011.

[76] T. D. Mitchell, T. R. Carter, P. D. Jones, M. Hulme, and M. New. *A comprehensive set of high-resolution grids of monthly climate for Europe and the globe: the observed record (1901-2000) and 16 scenarios (2001-2100)*. Tyndall Centre for Climate Change Research, 2004.

[77] S. Mnaimneh, A. P. Davierwala, J. Haynes, J. Moffat, W-T. Peng, W. Zhang, X. Yang, J. Pootoolal, G. Chua, and A. Lopez. Exploration of essential gene functions via titratable promoter alleles. *Cell*, 118(1):31–44, 2004.

[78] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[79] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 1999.

[80] G. T. Narisma, J. A. Foley, R. Licker, and N. Ramankutty. Abrupt changes in rainfall during the twentieth century. *Geophysical Research Letters*, 34:L06710, March 2007.

[81] A. Ng, M. Jordan, and Y Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[82] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[83] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM International Conference on Management of Data*, 2002.

[84] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[85] P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, 2010.

[86] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *ICML*, pages 737–744, 2006.

[87] R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics. Princeton University Press, 1970.

[88] M. Sahami, M. Hearst, and E. Saund. Applying the multiple cause mixture model to text categorization. In *Proceedings of the 13th International Conference on Machine Learning*, 1996.

[89] M. Scheffer, S. Carpenter, J. A. Foley, C. Folke, and B. Walker. Catastrophic shifts in ecosystems. *Nature*, 413(6856):591–596, October 2001.

[90] M. Scheffer, M. Holmgren, V. Brovkin, and M. Claussen. Synergy between small- and large-scale feedbacks of vegetation on the water cycle. *Global Change Biology*, 11(7):1003–1012, 2005.

[91] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

[92] S. D. Schubert, M. J. Suarez, P. J. Pegion, R. D. Koster, and J. T. Bacmeister. On the cause of the 1930s dust bowl. *Science*, 303:1855–1859, 2004.

[93] M. Shafie and E. Milios. Model-based overlapping co-clustering. In *Proceedings of the Fourth Workshop on Text Mining, Sixth SIAM International Conference on Data Mining*, 2006.

[94] H. Shan and A. Banerjee. Bayesian co-clustering. In *Proceedings of the IEEE International Conference on Data Mining*, 2008.

[95] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[96] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

[97] D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*.

[98] M. Szell, R. Lambiotte, and S. Thurner. Multirelational organization of large-scale social networks. *Proceedings of the National Academy of Sciences USA*, 107(31):13636–13641, 2010.

[99] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–144, 2002.

[100] C. Tang, L. Zhang, M. Ramanathan, and A. Zhang. Interralated two-way clustering : An supervised approach for gene expression data analysis. In *Proc. of 2nd IEEE symposium on Bioinformatics and BioEngineering*, 2001.

[101] D. Tarlow, D. Batra, P. Kohli, and V. Kolmogorov. Dynamic tree block coordinate ascent. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, 2011.

[102] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[103] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[104] H. Wang and A. Banerjee. Online alternating direction method. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, 2012.

[105] Y. Weiss. Belief propagation and revision in networks with loops. Technical report, 1997.

[106] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

[107] C. A. Woodhouse and J. T. Overpeck. 2000 years of drought variability in the central United States. *Bulletin of the American Meteorological Society*, 79:2693–2714, 1998.

[108] P. Wu, Q. Fu, and F. Tang. Social community detection from photo collections using bayesian overlapping subspace clustering. In *Proceedings of the 17th international Conference on Advances in Multimedia Modeling*, 2011.

[109] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proc. of 3rd IEEE symposium on Bioinformatics and BioEngineering*, 2003.

[110] J. Yang and Y. Zhang. Alternating direction algorithms for l1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.

[111] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation: an empirical study. *Journmal of Machine Learning Research*, 7:1887–1907, 2006.

[112] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[113] K. Y. Yip, D. W. Cheung, and M. K. Ng. Harp: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1387–1397, 2004.