Technical Report

Department of Computer Science and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 12-007

MAP Inference on Million Node Graphical Models: KL-divergence based Alternating Directions Method

Arindam Banerjee, Huahua Wang, Qiang Fu, Stefan Liess, and Peter K. Snyder

February 27, 2012
Abstract

Motivated by a problem in large scale climate data analysis, we consider the problem of maximum a posteriori (MAP) inference in graphical models with millions of nodes. While progress has been made in recent years, existing MAP inference algorithms are inherently sequential and hence do not scale well. In this paper, we present a parallel MAP inference algorithm called KL-ADM based on two ideas: tree-decomposition of a graph, and the alternating directions method (ADM). However, unlike standard ADM, we use an inexact ADM augmented with a Kullback-Leibler (KL) divergence based regularization. The unusual modification leads to an efficient iterative algorithm while avoiding double-loops. We rigorously prove global convergence of KL-ADM. We illustrate the effectiveness of KL-ADM through extensive experiments on large synthetic and real datasets. The application on real world precipitation data finds all major droughts in the last century.

1 Introduction

Discrete graphical models have found applications in a wide variety of problems, including image analysis [19], speech recognition [22], bioinformatics [11] and error correcting codes [30]. Hidden Markov models (HMMs) [4], Markov random fields (MRFs) [42], and conditional random fields (CRFs) [27] 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 maximum a posteriori (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 [26]. For example, for HMMs, the most
likely sequence of latent states can be found efficiently using the Viterbi decoding algorithm [14]. For general graphs, however, the MAP inference problem is a computationally intractable integer program and is NP-hard [42]. 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 [31, 29]. 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 [36], and dual methods, which works on the dual variables, of which the dual decomposition approach is currently the most efficient [38].

The existing literature has, however, been applied to somewhat small graphs, consisting of few hundreds to a few thousands of nodes. In the context of large scale climate 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° × 0.5° over land. A 3-dimensional MRF (latitude × longitude × 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. Such large scale LPs are well beyond the capacity of commercially available LP-solvers. Even the recent advances in solving graph-structured LPs [20, 39, 25] get severely challenged at these scales. The key bottleneck, even in the advanced algorithms for solving graph-structured LPs, is that they are inherently sequential [36]. 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) [40].

In this paper, 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 [31] and the alternating directions method (ADM) [5]. 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 ADM. However, unlike standard ADM, we use a novel inexact ADM augmented by a Kullback-Leibler (KL) divergence regularization, which we refer to as KL-ADM. The unusual modification in KL-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 KL-ADM.

The proposed KL-ADM is extensively evaluated on both synthetic datasets and a large real precipitation dataset. The synthetic datasets have graphs of sizes ranging from 100,000 nodes to half a million nodes. The precipitation dataset consists of global precipitation information for over 100 years, ranging from 1901-2006, resulting in a graph of about 7 million nodes. In the synthetic datasets, we illustrate that sequential KL-ADM matches the performance of the state-of-the-art algorithms, whereas parallel KL-ADM gets almost linear speed-ups as more cores are used and is significantly faster than the state-of-the-art. On the precipitation dataset, the method finds the
major droughts in the past century efficiently. The parallel algorithm we implement using Open 
MPI solves the drought detection problem with more than 7 million variables in about 15 minutes.

The rest of the paper is organized as follows. We briefly review the MAP inference problem 
and the existing literature in Section 2. We introduce the KL-ADM algorithm in Section 3, and 
present rigorous analysis establishing its convergence. We discuss empirical evaluation in Section 
4 and conclude in Section 5.

2 Related Work

We start this section by introducing some basic background on Markov Random Fields. Then we 
briefly review the literature on MAP inference algorithms and outline the proximal algorithm which 
we use as a baseline in Section 4.

2.1 Markov Random Fields

An 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 \( \mathbf{X} = \{X_u | u \in V\} \in \mathcal{X}^n \). We assume that the distribution 
\( P \) of \( \mathbf{X} \) is a Markov Random Field \( [42] \), 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:

\[
P(\mathbf{x}) \propto \exp \left\{ \sum_{u \in V} f_u(x_u) + \sum_{(u, v) \in E} f_{uv}(x_u, x_v) \right\}.
\] (1)

An important problem in the context of MRF is that of maximum a posteriori (MAP) inference, 
which is to compute the configuration \( \mathbf{x}^* \) with the largest probability:

\[
\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}^n} \exp \left\{ \sum_{u \in V} f_u(x_u) + \sum_{(u, v) \in E} f_{uv}(x_u, x_v) \right\}.
\] (2)

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

\[
\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}^n} \left\{ \sum_{u \in V} f_u(x_u) + \sum_{(u, v) \in E} f_{uv}(x_u, x_v) \right\}.
\] (3)

The complexity of (3) 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 \([26]\). 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 this paper, we focus on the Linear Programming 
(LP) relaxation method \([11, 10]\). The LP relaxation of MAP inference problem is defined on a set
of pseudomarginals $\mu_u$ and $\mu_{uv}$, which are non-negative, normalized and locally consistent\cite{41,10}:

$$
\mu_u(x_u) \geq 0, \quad \forall u \in V ,
$$

(4)

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

(5)

$$
\sum_{x_u \in X_u} \mu_u(x_u) = 1, \quad \forall u \in V ,
$$

(6)

$$
\sum_{x_u \in X_u} \mu_{uv}(x_u, x_v) = \mu_v(x_v), \quad \forall (u, v) \in E .
$$

(7)

We denote the polytope defined by\cite{41-7} as $L(G)$ and the LP relaxation of MAP inference problem\cite{3} becomes solving the following LP (Problem I):

$$
\max_{\mu \in L(G)} \left\{ \langle \mu, f \rangle - \frac{1}{w} d_\phi(\mu || \mu^t) \right\},
$$

(8)

subject to the constraint that $\mu \in L(G)$. If the solution $\mu$ to (8) is an integer solution, it is guaranteed to be the optimal solution of\cite{3}. Otherwise, one can apply rounding schemes\cite{35,36} to round the fractional solution to an integer solution.

Although standard LP solvers can be used to solve the optimization problem \cite{8}, they are usually inefficient for the MAP inference problem \cite{41}, 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 or the dual problem of\cite{8}. In the following section, we briefly outline a convergent primal MAP inference algorithm (proximal algorithm) before introducing our new algorithm. We compare our algorithm with the proximal algorithm in Section 4.

Interestingly, the idea of ‘dual decomposition’ has been adopted to solve the dual of\cite{3}. The main idea behind the dual algorithms is that the dual problem decomposes into sub MAP problems, which can be solved exactly. It can also be shown that the dual of\cite{3} is equivalent to \cite{8}. For a detailed discussion on dual MAP algorithms, we refer the readers to\cite{38}. We also leave the comparison with the dual MAP algorithms for future work.

### 2.2 Proximal Algorithms

One of the convergent algorithms to solve \cite{8} is the proximal algorithm \cite{3,36}. Instead of solving the constrained LP \cite{8} directly, the proximal maximization methods solves a sequence of maximization problems:

$$
\mu^{t+1} = \arg\max_{\mu \in L(G)} \left\{ \langle \mu, f \rangle - \frac{1}{w} d_\phi(\mu || \mu^t) \right\},
$$

(9)

where the subscript $t = 1, 2, \ldots$ denotes the iteration number, $w^t$ is a positive constant and $d_\phi(\mu || \nu)$ is a Bregman divergence\cite{8,7} between $\mu$ and $\nu$ induced by the strictly convex function $\phi$:

$$
d_\phi(\mu || \nu) = \phi(\mu) - \phi(\nu) - \langle \nabla \phi(\nu), \mu - \nu \rangle.
$$

(10)

To take the graph structure into account,\cite{36} studied three choices of the Bregman divergence $d_\phi$: the sum of quadratic norms, the sum of KL divergences across the nodes and edges and the divergence induced by the tree-reweighted Bethe entropy defined on a set of spanning trees of $G$. 
To solve (9), one can first obtain the solution $\mu^{t+1,0}$ to the unconstrained version of (9) and then project $\mu^{t+1,0}$ to $L(G)$. To be more specific, we have:

$$
\mu^{t+1,0} = \arg\max_{\mu} \left\{ \langle \mu, f \rangle - \frac{1}{w^{t}} d_{\phi}(\mu||\mu^{t}) \right\},
$$

(11)

$$
\mu^{t+1} = \arg\min_{\mu \in L(G)} d_{\phi}(\mu||\mu^{t+1,0}).
$$

(12)

(11) has a closed form solution for the Bregman divergences discussed in [36]. However, (12) does not have a closed form solution and [36] showed that the projection can be computed via the cyclic Bregman projection algorithm when the divergence is the sum of the square norms or the KL divergences across all the nodes and edges.

We note that the polytope $L(G)$ can be viewed as an intersection of linear constraints $C_i$, i.e., $L(G) = \cap_{i=1}^{M} C_i$, where $M$ is the number of linear constraints. Define

$$
\Pi_{C_i}(\nu) = \arg\min_{\mu \in C_i} d_{\phi}(\mu||\nu)
$$

(13)

as the operation of projecting $\nu$ onto the constraint $C_i$. The Bregman projection algorithm computes the projection (12) in a cyclic manner: by iteratively projecting onto each constraint $C_i$, i.e., starting from $\mu^{t+1,0}$, we perform the following operation repeatedly until convergence:

$$
\mu^{t+1,p+1} = \Pi_{C_{i(p)}}(\mu^{t+1,p})
$$

(14)

where $i(p)$ is the constraint index and $i(p) = p \text{Mod} M + 1$. It can be shown [7] that the above cyclic Bregman projection converges to the projection defined in (12). It is important to point out that when the constraint set includes inequality linear constraints, the Bregman projection has to be followed by a correction step [7]. [36] showed that the projection onto each individual constraint can be a simple message passing update. For a detailed discussion and derivation of the proximal algorithm, we refer the readers to [36].

Overall, the proximal algorithm 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. Since the Bregman projection algorithm is sequential, there is no easy way to parallelize the proximal algorithm, which may make it unsuitable for large datasets. To apply such double loop algorithms, one also has to decide the stopping criterion of the inner loop, e.g., we terminate the algorithm if the maximum change of the parameters between two consecutive iteration is smaller than a threshold. If we use a high threshold, we may have stopped the inner loop before it converges and the final results can be inaccurate. However, if we adopt a low threshold, the inner loop can be prohibitively slow for large graphs $G$ as $L(G)$ is defined on a large amount of constraints.

### 3 Algorithm

In this section, we first review the ADM algorithm and show how to solve convex optimization problems with equality constraints using ADM. We then introduce the proposed KL-ADM algorithm and analyze it convergence properties.
3.1 Alternating Directions Method

Since first introduced in [17], ADMs have been extensively explored in a broad spectrum of applications ranging from image processing [34, 13, 9] to applied statistics and machine learning [37, 1, 28, 2, 31, 29]. For further understanding of the ADM, we refer the readers to the recent review by Boyd et al. [5] and the references therein.

To facilitate the discussion in the sequel, we first outline the basic ADM updates. Suppose our optimization problem is as follows:

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{N} f_i(x), \tag{15}$$

and we assume that $f_i, i = 1, \ldots, N$ are convex functions. We can rewrite (15) with local variables $x_i \in \mathbb{R}^n$ and a global variable $z \in \mathbb{R}^n$:

$$\min_{x_i, z} \sum_{i=1}^{N} f_i(x_i) \tag{16}$$

subject to $x_i = z, \quad i = 1, \ldots, N. \tag{17}$

Note that (17) is a consensus constraint and we use it to make sure that the local variables and the global variable are of the same value.

Let $\lambda_i \in \mathbb{R}^n, i = 1, \ldots, N$ be the Lagrangian multipliers. The Lagrangian function of (16) is:

$$L(x_i, \lambda_i, z) = \sum_{i=1}^{N} (f_i(x_i) + \langle \lambda_i, x_i - z \rangle) \tag{18}$$

The ADM algorithm imposes a quadratic penalty on (18) and the augmented Lagrangian then becomes:

$$L(x_i, \lambda_i, z) = \sum_{i=1}^{N} \left( f_i(x_i) + \langle \lambda_i, x_i - z \rangle + \frac{\beta}{2} ||x_i - z||_2^2 \right) \tag{19}$$

where $\beta > 0$ is the positive penalty parameter. The problem (19) and (18) are equivalent because for any feasible $x_i$ and $z$, the penalty term is zero. The ADM algorithm minimizes (19) iteratively and consists of the following updates:

$$x_i^{t+1} = \arg\min_{x_i} \left( f_i(x_i) + \langle \lambda_i^t, x_i - z^t \rangle + \frac{\beta}{2} ||x_i - z^t||_2^2 \right), \tag{20}$$

$$z^{t+1} = \arg\min_{z} \left( -\sum_{i=1}^{N} \langle \lambda_i^{t+1}, z \rangle + \frac{\beta}{2} ||x_i^{t+1} - z||_2^2 \right), \tag{21}$$

$$y_i^{t+1} = y_i^t + \beta (x_i^{t+1} - z^{t+1}). \tag{22}$$

To use the ADM algorithm to solve the linear programming problem (8), 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 $\mu$ (global variable). Throughout the paper, we focus on tree-structured decompositions. To be more specific,
let $T = \{\{V_1, E_1\}, \ldots, \{V_T, E_T\}\}$ be a collection of subgraphs of $G$ which satisfies two criterion: (i) Each subgraph $\tau = \{V_\tau, E_\tau\}$ is a tree-structured graph. (ii) Each node $u \in V$ and each edge $(u, v) \in E$ is included in at least one subgraph $\tau \in T$.

We denote $\mu_\tau$ as the components of global variable $\mu$ that belong to subgraph $\tau$. Note that since $\mu \in L(G)$ and $\tau$ is a tree-structured subgraph of $G$, $\mu_\tau$ always lies in $L(\tau)$, which is the polytope defined on tree $\tau$:

$$
\mu_{\tau,u}(x_u) \geq 0, \quad \forall u \in V_\tau, \\
\mu_{\tau,uv}(x_u, x_v) \geq 0, \quad \forall (u, v) \in E_\tau, \\
\sum_{x_u \in X_u} \mu_{\tau,u}(x_u) = 1, \quad \forall u \in V_\tau, \\
\sum_{x_u \in X_u} \mu_{\tau,uv}(x_u, x_v) = \mu_v(x_v), \quad \forall (u, v) \in E_\tau.
$$

We also introduce local variable $m_\tau \in L(\tau)$ which is the pseudomarginal defined on each subgraph $\tau$. We use $\theta_\tau$ to denote the potentials on subgraph $\tau$. In the newly formulated optimization problem, we will impose consensus constraints to enforce that $\mu_\tau = m_\tau$ for each tree-structured subgraph.

The new optimization problem (Problem II) we formulate based on the graph decomposition is then as follows:

$$
\min_{m_\tau, \mu} \sum_{\tau=1}^{||T||} \rho_\tau \langle m_\tau, \theta_\tau \rangle
$$

subject to $m_\tau - \mu_\tau = 0, \quad \tau = 1, \ldots, ||T||$

$$
m_\tau \in L(\tau), \quad \tau = 1, \ldots, ||T||
$$

where $\rho_\tau$ is a constant associated with each subgraph. We use the consensus constraints (28) to make sure that the pseudomarginals agree with each other in the sharing components across all the subgraphs. Besides the consensus constraints, we also impose feasibility constraints (29), which guarantee that, for each subgraph, the local variable $m_\tau$ lies in $L(\tau)$. When all the constraints ((28) - (29)) are satisfied, the global variable $\mu$ is guaranteed to lie in $L(G)$.

To make sure that Problem I and II are equivalent, we also need to guarantee that

$$
\min_{m_\tau} \sum_{\tau=1}^{||T||} \rho_\tau \langle m_\tau, \theta_\tau \rangle = \max_{\mu} \langle \mu, f \rangle,
$$

assuming the constraints ((28) - (29)) are satisfied. Since the new optimization is a minimization problem, we achieve this goal by imposing equality constraints on $\theta_\tau$ such that

$$
\sum_{\tau=1}^{T} \rho_\tau \bar{\theta}_\tau = -f.
$$

The dimensionality of $\bar{\theta}_\tau$ is $k|V| + k^2|E|$, which is the same as that of $f$ and $\bar{\theta}_\tau$ can be constructed from $\theta_\tau$: $\bar{\theta}_\tau$ is augmented from $\theta_\tau$ and the components which do not belong to tree $\tau$ are set to zero.
It is easy to verify that, as long as (31) is satisfied, the choice of \(\rho_\tau\) and \(\theta_\tau\) do not change Problem \(\text{II}\). Let \(1,|\) be a binary indicator function and \(l = -f\), a straightforward approach to obtain the potential \(\theta_\tau\) can be:

\[
\theta_{\tau,u}(x_u) = \frac{l_u(x_u)}{\sum_{\tau'} \rho_{\tau'} 1_{u \in V_{\tau'}^T}}, \quad u \in V_\tau, \tag{32}
\]

\[
\theta_{\tau,uv}(x_u, x_v) = \frac{l_{uv}(x_u, x_v)}{\sum_{\tau'} \rho_{\tau'} 1_{(u, v) \in E_{\tau'}^T}}, \quad (u, v) \in E(\tau). \tag{33}
\]

The augmented Lagrangian of (27) is:

\[
L(m_\tau, \mu, \lambda) = \sum_{\tau=1}^{[T]} \left( \rho_\tau \langle m_\tau, \theta_\tau \rangle + \langle \lambda_\tau, m_\tau - \mu_\tau \rangle + \frac{\beta}{2} ||m_\tau - \mu_\tau||_2^2 \right), \tag{34}
\]

where \(\lambda\) is the dual variable and \(\beta\) is the penalty parameter. The following updates constitute a single iteration of the ADMM algorithm:

\[
m^{t+1}_\tau = \arg\min_{m_\tau \in L(\tau)} (\rho_\tau \langle m_\tau, \theta_\tau \rangle + \langle \lambda_\tau, m_\tau - \mu_\tau \rangle + \frac{\beta}{2} ||m^{t+1}_\tau - \mu_\tau||_2^2), \tag{35}
\]

\[
\mu^{t+1} = \arg\min_{\mu} \sum_{\tau=1}^{[T]} \left( -\langle \mu_\tau, \lambda_\tau^{t+1} \rangle + \frac{\beta}{2} ||m^{t+1}_\tau - \mu_\tau||_2^2 \right), \tag{36}
\]

\[
\lambda^{t+1}_\tau = \lambda^{t}_\tau + \beta (m^{t+1}_\tau - \mu^{t+1}_\tau). \tag{37}
\]

**Updating \(\mu\):** Since we have an unconstrained optimization problem \(\text{III}\) 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:

\[
\mu^{t+1}_u(x_u) = \frac{1}{|S_u|} \sum_{\tau \in S_u} (\beta m^{t+1}_{\tau,u}(x_u) + \lambda^{t}_{\tau,u}(x_u)) . \tag{38}
\]

(38) can be further simplified. With the average (over \(\tau \in S_u\)) of a scalar denoted with an overline, \(\overline{\cdot}\)

\[
\mu^{t+1}_u(x_u) = \overline{m^{t+1}_u(x_u)} + \frac{1}{\beta} \overline{\lambda^{t}_u(x_u)}. \tag{39}
\]

Similarly, for (37), we have:

\[
\bar{\lambda}^{t+1}_u(x_u) = \bar{\lambda}^{t}_u(x_u) + \beta (\overline{m^{t+1}_u(x_u)} - \mu^{t+1}_u(x_u)). \tag{40}
\]

Substituting (41) to (40) yields \(\bar{\lambda}^{t+1}_u(x_u) = 0\). Thus (38) can be simplified to:

\[
\mu^{t+1}_u(x_u) = \frac{1}{|S_u|} \sum_{\tau=1}^{T} m^{t+1}_{\tau,u}(x_u). \tag{41}
\]

Let \(S_{uv}\) be the subgraphs which contain edge \((u, v)\). Similarly, the update for the edge components is:

\[
\mu^{t+1}_{u,v}(x_u, x_v) = \frac{1}{|S_{uv}|} \sum_{\tau \in S_{uv}} m^{t+1}_{\tau,uv}(x_u, x_v). \tag{42}
\]
Updating $\mathbf{m}_\tau$: We need to solve a quadratic optimization problem for each tree-structured subgraph. Unfortunately, we do not have a close-form solution for (35) in general. One possible approach, similar to the proximal algorithm, is to first obtain the solution $\tilde{\mathbf{m}}_\tau$ to the unconstrained problem of (35):

$$\tilde{\mathbf{m}}_\tau = \mu^t_\tau - \frac{\rho_\tau \theta^t + \lambda^t}{\beta}_\tau$$

(43)

and then project $\tilde{\mathbf{m}}_\tau$ to $L(\tau)$:

$$\mathbf{m}_\tau = \arg\min_{m \in L(\tau)} ||\mathbf{m} - \tilde{\mathbf{m}}_\tau||^2_2.$$  

(44)

If we adopt the cyclic Bregman projection algorithm [7] to solve (44), the algorithm becomes a double-loop algorithm, i.e., the cyclic projection algorithm itself is a loop for each subgraph.

3.2 Inexact ADM with KL-divergence

In the above section, we outline the basic ADM updates to solve the relaxed MAP inference problem (8). However, solving the $\mathbf{m}_\tau$-minimization step (35) exactly requires an inner loop algorithm, which makes it unfavorable for large scale problems.

Instead of solving (35) exactly, a common way in inexact ADMs [43, 21, 12, 5, 23] is to linearize the objective function in (35), i.e., the first order Taylor expansion at $\mathbf{m}^t_\tau$, and add a new quadratic penalty term such that

$$\mathbf{m}^{t+1}_\tau = \arg\min_{\mathbf{m}_\tau \in L(\tau)} \langle \mathbf{y}^t_\tau, \mathbf{m}_\tau - \mathbf{m}^t_\tau \rangle + \alpha \frac{1}{2} ||\mathbf{m}_\tau - \mathbf{m}^t_\tau||^2,$$  

(45)

where $\alpha$ is a positive constant and

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

(46)

However, as discussed in the previous section, the quadratic problem (45) is generally difficult for a tree-structured graph and thus the conventional inexact ADM does not lead to an efficient update for $\mathbf{m}_\tau$.

In this section, we show that, by taking the tree structure into account, an inexact minimization of (35) augmented with a KL divergence leads to efficient update of $\mathbf{m}_\tau$. We prove that the new algorithm converges to the global optimizer of Problem II in the next section.

The basic idea in the new algorithm is that we replace the quadratic term in (45) with a Bregman-divergence term $d_\phi(\mathbf{m}_\tau||\mathbf{m}^t_\tau)$ such that

$$\mathbf{m}^{t+1}_\tau = \arg\min_{\mathbf{m}_\tau \in L(\tau)} \langle \mathbf{y}^t_\tau, \mathbf{m}_\tau - \mathbf{m}^t_\tau \rangle + \alpha d_\phi(\mathbf{m}_\tau||\mathbf{m}^t_\tau),$$  

(47)

is efficient to solve for tree $\tau$. For a tree-structured problem, what Bregman divergence should we choose? Expanding the Bregman divergence and removing the constants, we can rewrite (47) as

$$\mathbf{m}^{t+1}_\tau = \arg\min_{\mathbf{m}_\tau \in L(\tau)} \langle \mathbf{y}^t_\tau / \alpha - \nabla \phi(\mathbf{m}^t_\tau), \mathbf{m}_\tau \rangle + \phi(\mathbf{m}_\tau).$$  

(48)

It is well known that when $\phi$ is the negative Bethe entropy of $\mathbf{m}_\tau$, (48) can be solved efficiently in linear time via the sum-product algorithm [42, 26].
The Bethe entropy on a tree-structured distribution $m_\tau$ is defined as:

$$H_{\text{Bethe}}(x_\tau) = \sum_{u \in V_\tau} H_u(x_{\tau,u}) - \sum_{(u,v) \in E_\tau} I_{uv}(x_{\tau,uv}) ,$$

(49)

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

$$H_u(m_{\tau,u}) = -\sum_{x_u \in X} m_{\tau,u}(x_u) \log m_{\tau,u}(x_u) ,$$

(50)

$$I_{uv}(m_{\tau,uv}) = \sum_{(x_u,x_v) \in X \times X} m_{\tau,uv}(x_u,x_v) \log \frac{m_{\tau,uv}(x_u,x_v)}{m_{\tau,u}(x_u)m_{\tau,v}(x_v)} .$$

(51)

Then the derivative $\nabla H_{\text{Bethe}}(m_\tau)$ can be computed as:

$$\nabla H_{\text{Bethe}}(m_{\tau,u}(x_u)) = -\log m_{\tau,u}(x_u) - 1 , \forall s \in V_\tau ,$$

(52)

$$\nabla H_{\text{Bethe}}(m_{\tau,uv}(x_u,x_v)) = -\log \sum_{x_u'} m_{\tau,uv}(x_u',x_v) \sum_{x_v'} m_{\tau,uv}(x_u,x_v') + 1 , \forall u,v \in V_\tau .$$

(53)

The sum-product algorithm is a simple message passing algorithm and it is guaranteed to converge to the optimal solution when the underlying graph is a tree. We refer the readers to [42] regarding the details of the sum-product algorithm.

Recall that $m_\tau$ defines a tree-structured distribution on tree $\tau$ as follows:

$$p_{m_\tau}(x) = \prod_{s \in V(\tau)} m_{\tau,s}(x_s) \prod_{(s,t) \in E(\tau)} \frac{m_{\tau,st}(x_s,x_t)}{m_{\tau,s}(x_s)m_{\tau,t}(x_t)} .$$

(54)

It is easy to verify that $H_{\text{Bethe}}(m_\tau)$ is the entropy defined on the distribution $p_{m_\tau}$ and hence the Bregman divergence $d_\phi(m_\tau || m^t_\tau)$ in (47) is a KL-divergence between tree-structured distributions $P_{m_\tau}$ and $P_{m^t_\tau}$.

We call our new algorithm KL-ADM algorithm. For the sake of completeness, we summarize our algorithm as follows:

$$m^{t+1}_\tau = \arg\min_{m_\tau \in \mathcal{L}(\tau)} \langle y^t_\tau / \alpha - \nabla \phi(m^t_\tau), m_\tau \rangle + \phi(m_\tau) ,$$

(55)

$$\mu^{t+1}_\tau = \arg\min_{\mu} \sum_{\tau=1}^T \left(-\langle \lambda^t_\tau, \mu_\tau \rangle + \beta \frac{1}{2} || m^{t+1}_\tau - \mu_\tau ||^2 \right) ,$$

(56)

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

(57)

Note that, since in (55), the sum-product update on each $m_\tau$ is independent, the KL-ADM algorithm is inherently parallel.

### 3.3 Convergence

In this section, we prove that the KL-ADM algorithm converges to the global optimizer $\mu^*$ of Problem II. We first show that the value of the objective function (27) converges to $\langle \mu^*, \theta \rangle$ when
\[ t \to \infty. \text{ To be more specific, we show that the convergence of the objective function satisfies the following sublinear bound, i.e.,} \]

\[
\sum_{t=0}^{T} \left( \sum_{\tau=1}^{[\mathbb{T}]} \rho_{\tau}(\mu_{t,\tau}^{\pm} - \mu_{t,\tau}^\ast, \theta_{\tau}) \right) \leq o(T). \tag{58}
\]

However, (58) alone does not imply \( \mu^t \to \mu^\ast \), because \( \mu^t \) may not lie in \( L(G) \). To show that \( \mu^t \) is feasible, we further prove the equality constraints (28) are satisfied when \( t \to \infty \), i.e.,

\[
\sum_{t=0}^{T} \sum_{\tau=1}^{[\mathbb{T}]} \| m_{t,\tau}^{t+1} - \mu_{t,\tau}^{t+1} \|_2 + \| \mu_{t,\tau}^{t+1} - \mu_{t,\tau}^t \|_2 \leq o(T). \tag{59}
\]

Note that the constraint (29) is always satisfied because \( m_{t,\tau}^t \) is computed via the sum-product algorithm. Thus, (58), in conjunction with (59), guarantee that \( \mu^t \to \mu^\ast \).

We first need the following lemma to bound the Bregman divergence \( d_\phi \):

**Lemma 1** Let \( \mu_{\tau} \) and \( \nu_{\tau} \) be two concatenated vectors of the pseudomarginals on a tree \( \tau \) with \( n \) nodes. Let \( d_\phi(\mu_{\tau}||\nu_{\tau}) \) be the Bregman divergence induced by the negative Bethe entropy \( \phi \). Assume

\[
\alpha \geq \beta(2n - 1)^2, \tag{60}
\]

we have

\[
\alpha d_\phi(\mu_{\tau}||\nu_{\tau}) \geq \frac{\beta}{2} \| \mu_{\tau} - \nu_{\tau} \|_2^2. \tag{61}
\]

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

\[
\mu_{\tau,u}(x_u) = \sum_{-x_u} P(x_1, \ldots, x_u, \ldots, x_n). \tag{62}
\]

Thus, \( \mu_{u} \) is the marginal probability distribution for node \( u \).

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

\[
\mu_{\tau,uv}(x_u, x_v) = \sum_{-(x_u,x_v)} P(x_1, \ldots, x_u, \ldots, x_v, \ldots, x_n). \tag{63}
\]

Thus, \( \mu_{uv} \) is the marginal probability for nodes \( (u, v) \).

Let \( \mu_{\tau}, \nu_{\tau} \) be two pseudomarginals defined on tree \( \tau \) and \( P_{\mu_{\tau}}, P_{\nu_{\tau}} \) be the corresponding tree-structured distributions. Making use of (62), we have

\[
\| P_{\mu_{\tau}} - P_{\nu_{\tau}} \|_1 \geq \| \mu_{\tau,u} - \nu_{\tau,u} \|_1, \quad \forall u \in V_{\tau}. \tag{64}
\]

Similarly, for each edge, we have the following inequality because of (63)

\[
\| P_{\mu_{\tau}} - P_{\nu_{\tau}} \|_1 \geq \| \mu_{\tau,uv} - \nu_{\tau,uv} \|_1, \quad \forall (u, v) \in E_{\tau}. \tag{65}
\]
Adding them together gives

$$(2n-1)\|P_{\mu_r} - P_{\nu_r}\|_1 \geq \|\mu_r - \nu_r\|_1 \geq \|\mu_r - \nu_r\|_2.$$  \hspace{1cm} (66)

According to Pinsker’s inequality \cite{[3]}, we have

$$d_\phi(\mu_r\|\nu_r) = KL(P_{\mu_r}, P_{\nu_r}) \geq \frac{1}{2}\|P_{\mu_r} - P_{\nu_r}\|_1^2 \geq \frac{1}{2(2n-1)^2}\|\mu_r - \nu_r\|_2^2.$$  \hspace{1cm} (67)

Multiplying $\alpha$ on both sides and letting $\alpha \geq \beta(2n-1)^2$ completes the proof.

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

$$R_{t+1}^\tau = \rho_t(\mathbf{t}_{t+1}^\tau - \mathbf{u}_r^\star, \theta_r).$$  \hspace{1cm} (68)

We show that $R_{t+1}^\tau$ satisfies the following inequality:

**Lemma 2** Let $\{\mathbf{r}_\tau, \mu_r, \lambda_r\}$ be the sequences generated by (55), (56) and (57). Assume (60) holds. For any $\mathbf{m}_r^\star \in L(\tau)$, we have

$$R_{t+1}^\tau \leq \langle \mathbf{y}_r^t + \alpha(\nabla \phi(\mathbf{m}_r^{t+1}) - \nabla \phi(\mathbf{m}_r^t)), \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle \geq 0.$$  \hspace{1cm} (70)

where $R_{t+1}^\tau$ is defined in (68).

**Proof:** Recall that the optimum $\hat{x}$ of any constrained optimization problem $\min_{x \in S} g(x)$ is given by the stationary condition:

$$\langle \nabla g(\hat{x}), x - \hat{x} \rangle \geq 0,$$  \hspace{1cm} (71)

where $\nabla g$ is the gradient of $g$. Since $\mathbf{m}_r^{t+1}$ is the optimal solution for (55), for any $\mathbf{m}_r^\star \in L(\tau)$, we have the following inequality:

$$\langle \mathbf{y}_r^t + \alpha(\nabla \phi(\mathbf{m}_r^{t+1}) - \nabla \phi(\mathbf{m}_r^t)), \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle \geq 0.$$  \hspace{1cm} (72)

Substituting (56) into (71) and rearranging the terms, we have

$$R_{t+1}^\tau \leq \langle \lambda_r^t, \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle + \alpha\langle \nabla \phi(\mathbf{m}_r^{t+1}) - \nabla \phi(\mathbf{m}_r^t), \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle.$$  \hspace{1cm} (73)

The second term in the RHS of (72) is equivalent to

$$2\langle \mathbf{m}_r^t - \mathbf{m}_r^\star, \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle = \|\mathbf{m}_r^t - \mathbf{m}_r^{t+1}\|_2^2 + \|\mathbf{m}_r^\star - \mathbf{m}_r^t\|_2^2 - \|\mathbf{m}_r^\star - \mathbf{m}_r^{t+1}\|_2^2.$$  \hspace{1cm} (74)

The third term in the RHS of (72) can be rewritten as

$$\langle \nabla \phi(\mathbf{m}_r^{t+1}) - \nabla \phi(\mathbf{m}_r^t), \mathbf{m}_r^\star - \mathbf{m}_r^{t+1} \rangle = d_\phi(\mathbf{m}_r^\star\|\mathbf{m}_r^t) - d_\phi(\mathbf{m}_r^\star\|\mathbf{m}_r^{t+1}) - d_\phi(\mathbf{m}_r^{t+1}\|\mathbf{m}_r^t).$$  \hspace{1cm} (75)

Substituting (73) and (74) into (72) and using Lemma 1 complete the proof.

We next show that the first term in the RHS of (69) satisfies the following result:
Proposition 1 Let \( \{m_{\tau}, \mu_{\tau}, \lambda_{\tau}\} \) be the sequences generated by (53), (56) and (57). For any \( \mu^*_{\tau} \in L(\tau) \), we have

\[
\sum_{\tau=1}^{[T]} (\lambda^*_t, \mu^*_{\tau} - m^*_{\tau}) \leq \sum_{\tau=1}^{[T]} (\lambda^*_t, \mu^*_{\tau} - m^*_{\tau}) - \frac{\beta}{2} \| \mu^*_{\tau} - m^*_{\tau} \|_2^2
\]

\[
+ \frac{\beta}{2} \left( \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 - \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 \right) .
\]  \hspace{1cm} (75)

Proof: Let \( \mu_i \) be the \( i \)th component of \( \mu \). We augment \( \mu_{\tau}, m_{\tau} \) and \( \lambda_{\tau} \) in the following way: If \( \mu_i \) is not a component of \( \mu_{\tau} \), we set \( \mu_{\tau,i} = 0, m_{\tau,i} = 0 \) and \( \lambda_{\tau,i} = 0 \); otherwise, they are the corresponding components from \( \mu_{\tau}, m_{\tau} \) and \( \lambda_{\tau} \) respectively. We can then rewrite (56) in the following equivalent component-wise form:

\[
\mu^*_{\tau} = \arg\min_{\mu_{\tau}} \sum_{\tau=1}^{[T]} \left( \left( \lambda^*_t, m^*_{\tau,i} - \mu_{\tau,i} \right) + \frac{\beta}{2} \| m^*_{\tau,i} - \mu_{\tau,i} \|_2^2 \right) .
\]  \hspace{1cm} (76)

Since \( \mu^*_{\tau} \) is a minimizer of (76), for any \( \mu^*_{\tau} \in L(\tau) \), we have the following inequality:

\[
- \sum_{\tau=1}^{[T]} (\lambda^*_t, \beta(m^*_{\tau,i} - \mu^*_{\tau,i}), \mu^*_{\tau} - \mu^*_{\tau}) \geq 0 .
\]  \hspace{1cm} (77)

Combining all the components of \( \mu^*_{\tau} \), we can rewrite (77) in the following vector form:

\[
- \sum_{\tau=1}^{[T]} (\lambda^*_t, \beta(m^*_{\tau,i} - \mu^*_{\tau,i}), \mu^*_{\tau} - \mu^*_{\tau}) \geq 0 .
\]  \hspace{1cm} (78)

Rearranging the terms yields

\[
\sum_{\tau=1}^{[T]} (\lambda^*_t, \mu^*_{\tau} - \mu^*_{\tau}) \leq - \sum_{\tau=1}^{[T]} \beta(m^*_{\tau,i} - \mu^*_{\tau,i}, \mu^*_{\tau} - \mu^*_{\tau})
\]

\[
= \frac{\beta}{2} \sum_{\tau=1}^{[T]} \left( \| \mu^*_{\tau} - m^*_{\tau,i} \|_2^2 - \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 - \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 \right) .
\]  \hspace{1cm} (79)

Simple algebraic manipulation yields (75). \( \blacksquare \)

Summing over all tree-structured subgraphs, we have the following result:

Proposition 2 Let \( R^*_{\tau} \) be defined in (68). Then

\[
\sum_{\tau=1}^{[T]} R^*_{\tau} \leq \sum_{\tau=1}^{[T]} \frac{1}{2\beta} \left( \| \lambda^*_t \|_2^2 - \| \lambda^*_{\tau} \|_2^2 \right) + \frac{\beta}{2} \left( \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 - \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 \right)
\]

\[
+ \frac{\beta}{2} \left( \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 - \| \mu^*_{\tau} - m^*_{\tau} \|_2^2 \right) + \alpha (d_{\phi}(\mu^*_{\tau}, m^*_t) - d_{\phi}(\mu^*_{\tau}, m^*_{\tau})) - \frac{\beta}{2} \| m^*_{\tau} - \mu^*_{\tau} \|_2^2 .
\]  \hspace{1cm} (80)
Proof: Summing (69) over \( \tau \) from 1 to \( |T| \) and combining the result in (75), we have:

\[
\sum_{\tau=1}^{T} R_{\tau}^{t+1} \leq \sum_{\tau=1}^{T} (\lambda_{\tau}^{t+1} - m_{\tau}^{t+1}) - \frac{\beta}{2} (\|\mu_{\tau}^{t+1} - m_{\tau}^{t+1}\|_{2}^{2} + \|m_{\tau}^{t+1} - \mu_{\tau}^{t}\|_{2}^{2}) \\
+ \frac{\beta}{2} (\|\mu_{\tau}^{*} - \mu_{\tau}^{t}\|_{2}^{2} - \|\mu_{\tau}^{*} - \mu_{\tau}^{t+1}\|_{2}^{2}) + \frac{\beta}{2} (\|\mu_{\tau}^{*} - m_{\tau}^{t+1}\|_{2}^{2} - \|\mu_{\tau}^{*} - m_{\tau}^{t}\|_{2}^{2}) \\
+ \alpha (d_{\phi}(\mu_{\tau}^{*} || m_{\tau}^{t}) - d_{\phi}(\mu_{\tau}^{*} || m_{\tau}^{t+1})) .
\]  

(81)

Recall that \( \mu_{\tau}^{t+1} - m_{\tau}^{t+1} = \frac{1}{\beta} (\lambda_{\tau}^{t} - \lambda_{\tau}^{t+1}) \) in (57), we have

\[
\langle \lambda_{\tau}^{t}, \mu_{\tau}^{t+1} - m_{\tau}^{t+1} \rangle - \frac{\beta}{2} \|\mu_{\tau}^{t+1} - m_{\tau}^{t+1}\|_{2}^{2} = \frac{1}{2\beta}(\|\lambda_{\tau}^{t}\|_{2}^{2} - \|\lambda_{\tau}^{t+1}\|_{2}^{2}) .
\]  

(82)

Combining (81) and (82) completes the proof.

Now, we are ready to show the convergence of the objective.

**Theorem 1** Assume the following conditions hold: (1) \( m_{0}^{t} \) and \( \mu_{0}^{t} \) are uniform tree-structured distributions, \( \forall \tau = 1, \ldots, |T| \) \( \lambda_{0}^{t} = 0, \forall \tau = 1, \ldots, |T| \); (2) \( \lambda_{0}^{t} = 0, \forall \tau = 1, \ldots, |T| \); (3) \( \max_{\tau} d_{\phi}(\mu_{\tau}^{*} || m_{0}^{t}) = D_{\theta} \); (4) \( \max_{\tau} \|\rho_{\tau} \theta_{\tau}\|_{2}^{2} = D_{\theta} \); (5) \( \beta = \sqrt{T} \); (6) (60) holds. For any \( T \) and the optimal solution \( \mu^{*} \), we have

\[
\sum_{t=0}^{T} \left( \sum_{\tau=1}^{T} \rho_{\tau}(\mu_{\tau}^{t} - \mu_{\tau}^{*}, \theta_{\tau}) \right) \leq D_{\mu} \alpha |T| + \frac{|T| D_{\theta} \sqrt{T}}{2} = o(T) .
\]  

(83)

**Proof:** Using the inequality \( 2\langle x, y \rangle \leq \|x\|_{2}^{2} + \|y\|_{2}^{2} \), we have

\[
\rho_{\tau}(\mu_{\tau}^{t} - m_{\tau}^{t+1}, \theta_{\tau}) = \langle \sqrt{\beta}(\mu_{\tau}^{t} - m_{\tau}^{t+1}), \frac{\rho_{\tau}(\theta_{\tau})}{\sqrt{\beta}} \rangle \leq \frac{\beta}{2} \|m_{\tau}^{t+1} - \mu_{\tau}^{t}\|_{2}^{2} + \frac{1}{2\beta} \|\rho_{\tau} \theta_{\tau}\|_{2}^{2} .
\]  

(84)

Summing (81) over all tree-structured subgraphs and adding it to (80) yields

\[
\sum_{\tau=1}^{T} \rho_{\tau}(\mu_{\tau}^{t} - \mu_{\tau}^{*}, \theta_{\tau}) = \sum_{\tau=1}^{T} (R_{\tau}^{t+1} + \rho_{\tau}(\mu_{\tau}^{t} - m_{\tau}^{t+1}, \theta_{\tau})) \\
\leq \sum_{\tau=1}^{T} \frac{1}{2\beta}(\|\lambda_{\tau}^{t}\|_{2}^{2} - \|\lambda_{\tau}^{t+1}\|_{2}^{2}) + \frac{\beta}{2} (\|\mu_{\tau}^{*} - \mu_{\tau}^{t}\|_{2}^{2} - \|\mu_{\tau}^{*} - \mu_{\tau}^{t+1}\|_{2}^{2}) \\
+ \frac{\beta}{2} (\|\mu_{\tau}^{*} - m_{\tau}^{t+1}\|_{2}^{2} - \|\mu_{\tau}^{*} - m_{\tau}^{t}\|_{2}^{2}) + \alpha (d_{\phi}(\mu_{\tau}^{*} || m_{\tau}^{t}) - d_{\phi}(\mu_{\tau}^{*} || m_{\tau}^{t+1})) + \frac{1}{2\beta} \|\rho_{\tau} \theta_{\tau}\|_{2}^{2} .
\]  

(85)
Summing over $t$ from 1 to $T$, we have
\begin{align*}
\sum_{t=0}^T \sum_{r=1}^{[T]} \rho_r \langle \mu_r^t - \mu_r^*, \theta_r \rangle &\leq \sum_{r=1}^{[T]} \frac{1}{2\beta} (\|\lambda_r^0\|_2^2 - \|\lambda_r^T\|_2^2) + \frac{\beta}{2} (\|\mu_r^* - \mu_r^0\|_2^2 - \|\mu_r^* - \mu_r^T\|_2^2) \\
&\quad + \beta \left(\|\mu_r^* - m_r^T\|_2^2 - \|\mu_r^* - m_r^0\|_2^2\right) + \alpha \left(d_\phi(\mu_r^*||m_r^0) - d_\phi(\mu_r^*||m_r^T)\right) + \frac{T|\rho_r\theta_r|_2^2}{2\beta} \\
&\leq \sum_{r=1}^{[T]} \frac{\beta}{2} \|\mu_r^* - m_r^T\|_2^2 + \alpha \left(d_\phi(\mu_r^*||m_r^0) - d_\phi(\mu_r^*||m_r^T)\right) + \frac{\sqrt{T}||\rho_r\theta_r||_2^2}{2} \\
&\leq \sum_{r=1}^{[T]} \alpha d_\phi(\mu_r^*||m_r^0) + \frac{\sqrt{T}||\rho_r\theta_r||_2^2}{2} , \quad (86)
\end{align*}
where we use Lemma 1 to derive (86). Assuming $\max_r d_\phi(\mu_r^*||m_r^0) = D_\mu$ and $\max_r ||\rho_r\theta_r||_2^2 = D_\theta$ gets the desired bound.

In addition to the convergence of the objective, we will show that the equality constraint is satisfied when $t \to \infty$. We need the following lemma:

**Lemma 3** Let $\{m_r, \mu_r, \lambda_r\}$ be the sequences generated by (55), (56) and (57). Then
\begin{align*}
\sum_{r=1}^{[T]} \|m_r^{t+1} - \mu_r^t\|_2^2 \geq \sum_{r=1}^{[T]} \|m_r^{t+1} - \mu_r^{t+1}\|_2^2 + \|\mu_r^{t+1} - \mu_r^t\|_2^2 . \quad (87)
\end{align*}

**Proof:** Recalling (57), (78) can be rewritten as
\begin{align*}
- \sum_{r=1}^{[T]} \langle \lambda_r^{t+1}, \mu_r^* - \mu_r^{t+1} \rangle \geq 0 , \quad (88)
\end{align*}
which holds for any $\mu_r^* \in L(\tau)$. Setting $\mu_r^* = \mu_r^t$ yields
\begin{align*}
- \sum_{r=1}^{[T]} \langle \lambda_r^{t+1}, \mu_r^t - \mu_r^{t+1} \rangle \geq 0 . \quad (89)
\end{align*}

Similarly, we have
\begin{align*}
- \sum_{r=1}^{[T]} \langle \lambda_r^t, \mu_r^{t+1} - \mu_r^t \rangle \geq 0 . \quad (90)
\end{align*}

Adding (89) and (90) together yields
\begin{align*}
\sum_{r=1}^{[T]} \langle \lambda_r^{t+1} - \lambda_r^t, \mu_r^{t+1} - \mu_r^t \rangle \geq 0 . \quad (91)
\end{align*}
Recall that $\lambda^{t+1}_r - \lambda^t_r = \beta (m^{t+1}_r - \mu^{t+1}_r)$ in (53), we have

$$0 \leq \sum_{\tau=1}^{T} \beta \langle m^{t+1}_r - \mu^{t+1}_r, \mu^{t+1}_r - \mu^{t}_r \rangle$$

$$= \frac{\beta}{2} \sum_{\tau=1}^{T} \|m^{t+1}_r - \mu^{t+1}_r\|_2^2 - \|m^{t+1}_r - \mu^{t+1}_r\|_2^2 - \|\mu^{t+1}_r - \mu^{t}_r\|_2^2 .$$

(92)

Rearranging the first two terms completes the proof.

Since both $m^{t+1}_r$ and $\mu^*_r$ lie in $L(\tau)$ and $\theta_\tau$ are bounded, without the loss of generality, we assume

$$\forall t, \sum_{\tau=1}^{T} R^{t+1}_\tau \geq -F ,$$

(93)

where $F$ is a constant. For the sake of simplicity, we also assume that each tree-structured graph has $n$ nodes. Our proof extends easily to the case where the tree-structured graphs are of different sizes. The following theorem gives a sublinear bound for the constraint violation.

**Theorem 2** Assume the conditions in Theorem 7 hold. Then

$$\sum_{t=0}^{T} \sum_{\tau=1}^{T} \| m^{t+1}_\tau - \mu^{t+1}_\tau \|_2^2 + \| \mu^{t+1}_\tau - \mu^*_\tau \|_2^2 \leq 2(2n - 1)^2 D_\mu |T| + 2F \sqrt{T} = o(T) ,$$

(94)

where $n$ is the number of nodes in each tree-structured subgraph.

**Proof:** Rearranging terms in (80) and dividing both sides by $\frac{\beta}{2}$, we have

$$\sum_{\tau=1}^{T} \| m^{t+1}_\tau - \mu^{t}_\tau \|_2^2 \leq \sum_{\tau=1}^{T} \frac{1}{\beta^2} (\| \lambda^t \|_2^2 - \| \lambda^{t+1}_\tau \|_2^2 + (\| \mu^*_\tau - \mu^t \|_2^2 - \| \mu^*_\tau - \mu^{t+1}_\tau \|_2^2)

+ (\| \mu^*_\tau - m^{t+1}_\tau \|_2^2 - \| \mu^*_\tau - m^t \|_2^2) + \frac{2\alpha}{\beta} (\varphi_\tau (m^t_\tau) - \varphi_\tau (m^{t+1}_\tau)) + \frac{2F}{\beta} ,$$

(95)

where $F$ is defined in (93).

Summing over $t$ from 1 to $T$, we have

$$\sum_{t=0}^{T} \sum_{\tau=1}^{T} \| m^{t+1}_\tau - \mu^{t}_\tau \|_2^2 \leq \sum_{\tau=1}^{T} \frac{1}{\beta^2} (\| \lambda^0 \|_2^2 - \| \lambda^T \|_2^2 + (\| \mu^*_\tau - \mu^0 \|_2^2 - \| \mu^*_\tau - \mu^T \|_2^2)

+ (\| \mu^*_\tau - m^T \|_2^2 - \| \mu^*_\tau - m^0 \|_2^2) + \frac{2\alpha}{\beta} (\varphi_\tau (m^0_\tau) - \varphi_\tau (m^T_\tau)) + \frac{2FT}{\beta}$$

$$\leq \sum_{\tau=1}^{T} \frac{2\alpha}{\beta} \varphi_\tau (m^0_\tau) + \frac{2FT}{\beta} .$$

(96)

The last inequality is because of Lemma 11. Using Lemma 3 yields the desired bound.

\[\square\]
Figure 1: (a) A $3 \times 4$ four nearest neighbor grid. (b) Two replications of the 2-dimensional grid, where the replicated nodes are connected by an edge.

4 Experimental Results

We show experimental results in this section. We implement both the proximal \[36\] and KL-ADM algorithm in C. We also implement a parallel version of the KL-ADM algorithm using Open MPI \[18\]. We use ‘edge decomposition’ throughout this section: each $\tau$ is simply an edge of the graph and $|T| = |E|$. For the proximal algorithm, we terminate the inner loop if the maximum change between two consecutive iteration is less than a given threshold $\epsilon$ and we choose the Bregman divergence as the sum of KL-divergences across all the nodes and edges for the proximal algorithm.

4.1 Simulation Datasets

We show the experimental results on three simulation datasets with varying graph sizes and $k$. The underlying graph of each dataset is a three dimensional grid and it can be viewed as a replication of a four nearest neighbor grid, where the replicated nodes are connected by edges. We denote the number of rows and columns of the two dimensional grid as $m$ and $n$ respectively and the number of replications as $t$. Figure 1 illustrate a dependency graph with $m = 3, n = 4$ and $t = 2$.

We generate the potentials as follows: We set the nodewise potentials as random numbers from $[-\alpha, \alpha]$, where $\alpha > 0$. We set the edgewise potentials according to the Potts model:

$$
\theta_{uv}(x_u, x_v) = \begin{cases} 
\beta_{uv}, & \text{if } x_u = x_v; \\
0, & \text{Otherwise.}
\end{cases}
$$

We chose $\beta_{uv}$ randomly from $[-1, 1]$. The edgewise potentials penalize disagreement if $\beta_{st} > 0$ and penalize agreement if $\beta_{st} < 0$. The detailed information about the simulation datasets are in Table 1.

| Dataset | $m$ | $n$ | $t$ | $k$ | $\alpha$ | $|V|$ | $|E|$ |
|---------|-----|-----|-----|-----|----------|------|------|
| 1       | 50  | 40  | 50  | 3   | 2        | 100,000 | 293500 |
| 2       | 100 | 100 | 20  | 4   | 3        | 200,000 | 586000 |
| 3       | 100 | 100 | 50  | 5   | 4        | 500,000 | 1480000 |

Table 1: The basic information about the simulation datasets.

We run both the proximal and KL-ADM algorithm on the simulation datasets for 500 iterations. Since (8) is a constrained optimization problem, we evaluate both algorithms based on objective
Figure 2: 500-iteration run of the proximal and KL-ADM algorithm on dataset 2. For each iteration, both the objective function value and the maximum constraint violation are calculated.

Figure 3: 500-iteration run of both the proximal and KL-ADM algorithm on dataset 3. For each iteration, both the objective function value and the maximum constraint violation are calculated.

function value and constraint violation: In each iteration, we calculate the value of the objective function and measure the largest violation of the constrains that define $L(G)$. The results for dataset 2 and 3 are shown in Figure 2 and 3.

We can make several interesting observations based on the plots. First, for the KL-ADM algorithm, there is a trade off between convergence and constraint satisfaction. When we increase $\beta$, the penalty of constraint violation will increase and, as a result, the algorithm tends to find solutions with small amount of constrain violation. However, this leads to more iterations before the algorithm converges to the optimal value. Second, different from the proximal algorithm, the KL-ADM algorithm reduces the constraint violation gradually as it progresses. Finally, for the proximal algorithm, to make sure the constraints are satisfied, we have to choose lower thresholds $\epsilon$, i.e., 0.0001, which increases the running time by a large margin. On the contrary, the KL-ADM algorithm is a single loop algorithm and there is no need to choose any threshold.

We also have a look at the running time of both algorithms and we find that our KL-ADM algorithm runs much faster than the proximal algorithm. For example, on simulation dataset 2, the running time of the KL-ADM algorithm is about 2400 seconds. However, the proximal algorithm
is almost 5 times slower and it takes about 12600 seconds to finish the 500 iterations. Since the KL-ADM algorithm is an inherently parallel algorithm, it can be made even faster by running it on multiple cores. Thus for each simulation dataset, we run the MPI code using different number of cores. In our MPI parallel implementation, to achieve load balancing, we schedule almost the same amount of tree-structured subgraphs to each core. According to (11) and (12), the update of the global variable $\mu$ only requires information from the relevant local variables $m_{\tau}$. Hence, to reduce the communication overhead, we avoid storing a central copy $\mu$ and distribute $\mu$ among the cores: each core only maintains a copy of the components of $\mu$ that are relevant to the trees scheduled to that core. As a result, two cores only communicate with each other regarding the sharing components of $\mu$. Figure 4 shows the speedup we obtain. When we increase the number of cores to 8, the algorithm runs 6-7 times faster.

![Figure 4: Speedup on the three simulation datasets. We get almost linear speedup in the number of cores used. When the number of cores is 8, we get significant speedup.](image)

### 4.2 CRU Precipitation Dataset

The dataset used in this section is the Climate Research Unit (CRU) precipitation dataset [32], which has monthly precipitation from the years 1901-2006. The dataset is of high gridded spatial resolution ($360 \times 720$, i.e., 0.5 degree latitude $\times$ 0.5 degree longitude) and only includes the precipitation over land.

Our goal is to detect major droughts of the last century based on precipitation. We formulate the drought detection problem as the one of estimating the most likely configuration of a binary hidden MRF. In the underlying graph, each node represents a location and it can be in two possible states: dry and normal. We use a four nearest neighbor grid ($m=360, n=720$) to model the global dependency and replicate it 106 times. The resulting graph is similar to the ones used in the previous section and the structure respects the CRU dataset, i.e, it only has the nodes that correspond to the locations with precipitation record. Overall, the three dimensional grid has 7,146,520 nodes.

We design the potential functions carefully from the CRU datasets to enforce label consistency, i.e., neighboring nodes should take same values. We refer the readers to [15] regarding the details on designing potential functions. We run the KL-ADM algorithm with different penalty parameter on the CRU dataset for 500 iterations. Figure 5 plots the objective function value and the constraint violation.

We obtain the integer solution after rounding the node pseudomarginals and we can detect droughts based on it. Since a drought can be defined both spatially and temporally, we define
it as a set of neighboring nodes in the three-dimensional underlying grid graph whose states are drought. Thus, the drought detection problem becomes one of finding sets of neighboring 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 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.

In [33], the authors list the top 30 regions of the world with abrupt decreases in rainfall during the 20th century. We note that the algorithm in [33] requires human inspection, however our algorithm is automatic. The KL-ADM algorithm discovers all the droughts in [33], except for a drought in Ukraine and two droughts in Australia. Figure 6 and 7 illustrate some major droughts we identify and the droughts found by both the KL-ADM and [33] are shown by black rectangles. 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 [16]. 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.

Since the drought detection problem on the CRU dataset is of large scale, we want to examine how the KL-ADM algorithm scales. We run the MPI code with different number of cores and Figure 8 shows the speedup. Using the same MPI implementation used in Section 4.1 we manage to obtain almost linear speedup in the number of cores. When we use 10 computing cores, we manage to obtain significant speedup and the optimization problem with more than 7 million nodes is solved in about 15 minutes.
Figure 6: Major droughts starting within the period 1921-1930. The droughts found by the KL-ADM and [33] are shown by black rectangles, which include the Dust Bowl in the 1930s, the drought in central Canada, northwestern America, South Africa and eastern China in the 1920s.

Figure 7: Major droughts starting within the period 1961-1970. The droughts found by the KL-ADM and [33] are shown by black rectangles, which include the three decade long Sahel drought and the drought in eastern India in the 1960s.
Figure 8: Speedup on the CRU dataset. We obtain almost linear speedup in the number of cores used. When we use 10 cores, we can solve the problem in about 15 minutes.

5 Conclusions

We propose a provably convergent MAP inference algorithm for large scale graphs with over millions of nodes. The algorithm is based on the ‘tree decomposition’ idea from the MAP inference algorithm literature and the alternating directions 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 the proximal algorithm in terms of running time. We also apply the algorithm to identify the droughts of the last century based on a precipitation dataset. The experimental results demonstrate that the algorithm scales well with the problem size and the drought detection problem with more than 7 million variables can be solved in about 15 minutes when we use 10 computing cores.

Acknowledgements: This research was supported in part by NSF CAREER Grant IIS-0953274, and NSF Grants IIS-1029711, IIS-0916750, and IIS-0812183. The authors are grateful for the technical support from the University of Minnesota Supercomputing Institute (MSI).

References


