JMLR: Workshop and Conference Proceedings 29:48–61, 2013 ACML 2013

Linear Approximation to ADMM for MAP inference

Sholeh Forouzan SFOROUZA@ICS.UCI.EDU

Alexander Ihler IHLER@ICS.UCI.EDU

Department of Computer Science
University of California, Irvine
Irvine, CA, 92697

Editor: Cheng Soon Ong and Tu Bao Ho

Abstract

Maximum a *posteriori* (MAP) inference is one of the fundamental inference tasks in graphical models. MAP inference is in general NP-hard, making approximate methods of interest for many problems. One successful class of approximate inference algorithms is based on linear programming (LP) relaxations. The augmented Lagrangian method can be used to overcome a lack of strict convexity in LP relaxations, and the Alternating Direction Method of Multipliers (ADMM) provides an elegant algorithm for finding the saddle point of the augmented Lagrangian. Here we present an ADMM-based algorithm to solve the primal form of the MAP-LP whose closed form updates are based on a linear approximation technique. Our technique gives efficient, closed form updates that converge to the global optimum of the LP relaxation. We compare our algorithm to two existing ADMM-based MAP-LP methods, showing that our technique is faster on general, non-binary or non-pairwise models.

1. Introduction

Graphical models are powerful tools in machine learning to describe structure within a probability distribution and to organize computations to solve difficult problems in areas such as computer vision and computational biology. One of the fundamental inference tasks in graphical models is to find the most likely configuration of variables, called maximum a *posteriori* (MAP) inference, equivalent to the "most probable explanation" (MPE) task in Bayesian network literature. Unfortunately, MAP/MPE inference is in general an NP-hard problem, and cannot be solved exactly for many problems of interest.

One successful class of approximate inference algorithms are based on linear programming (LP) relaxations. These algorithms come in several different forms. One approach is to solve the dual of the LP using coordinate descent, employed by MPLP (Globerson and Jaakkola, 2007) and MSD (Werner, 2007). Although such methods show good empirical behavior, they are not guaranteed to reach the global optimum of the LP relaxation. Approaches based on variants of subgradient descent (Komodakis et al., 2011; Jojic et al., 2010) are guaranteed to converge globally but are typically slower than coordinate descent approaches in practice (Jojic et al., 2010).

Introduced by Gabay and Mercier (1976), the Alternating Direction Method of Multipliers (ADMM) has recently become popular as an easy-to-apply method for distributed convex optimization with good empirical performance on variety of problems (Lin et al., 2011; Boyd et al., 2011).

However, a direct application of ADMM to the MAP-LP relaxation involves solving a non-trivial quadratic program at each iteration of the algorithm.

To circumvent this difficulty, two different globally convergent algorithms based on ADMM have been proposed for MAP-LP relaxations: APLP/ADLP (Meshi and Globerson, 2011) and DD-ADMM (Martins et al., 2011).

Both of these methods avoid solving the non-trivial quadratic program by introducing additional auxiliary variables. In particular, Martins et al. (2011) gives a closed-form update for binary pairwise factors and special "logical constraint" factors; the resulting DD-ADMM algorithm works by converting a general model into this form before solving it. In contrast, the APLP and ADLP algorithms (which correspond to optimizing the primal and dual MAP-LP form, respectively) work on general graphs, but introduce multiple copies of the variables associated with each factor to provide a closed-form update. As a result, both the DD-ADMM and APLP/ADLP approaches increase the number of variables being updated at each iteration and their constraints, which can significantly slow convergence and increases the required memory.

In this work, we choose a different approach to overcome the difficulty of variable updates. Like DD-ADMM (Martins et al., 2011) we use the augmented Lagrangian of the primal MAP-LP problem. However, instead of binarizing the graph to obtain closed form updates, we replace the difficult quadratic local terms by their first order Taylor approximation, allowing closed form updates at each iteration. Such approximations have been previously applied to ADMM in problems such as Low-Rank Representation (Lin et al., 2011). We show that in practice our algorithm produces better bounds during optimization and improves convergence time for models with general (non-pairwise or non-binary) factors.

2. MAP and LP Relaxations

Let $\boldsymbol{x} \triangleq \{x_1, \ldots, x_N\}$ be a vector of discrete random variables, where each $x_i \in \mathcal{X}_i$, with $\mathcal{X}_i$ a finite set, and let

$$P(\boldsymbol{x}; \theta) \propto \exp\Big(\sum_{f \in F} \theta_f(\boldsymbol{x}_f) + \sum_{i \in V} \theta_i(x_i) \Big)$$

be a probability distribution over $\boldsymbol{x}$, expressed in terms of "factors" θ_f each defined over a subset of the variables, $\boldsymbol{x}_f$. We abuse notation to use $i \in f$ to indicate those variables x_i that are part of $\boldsymbol{x}_f$, and d_i to be the number of factors f in which variable i participates.

It is helpful to also use the overcomplete exponential family form of undirected graphical models (Wainwright and Jordan, 2008; Koller and Friedman, 2009), so that each factor $\theta_i(x_i)$ can be represented as a vector $\boldsymbol{\theta_i}^T \boldsymbol{\delta}_{x_i}$ where $\boldsymbol{\delta}_{x_i}$ is a binary indicator vector with one element, $\delta_{x_i;s}$, for each state $s \in \mathcal{X}_i$, and $\delta_{x_i;s}$ takes value one when $x_i = s$ and zero otherwise. We similarly define $\boldsymbol{\theta_f}$ and $\boldsymbol{\delta}_{\boldsymbol{x}_f}$ over the configurations of $\boldsymbol{x}_f$.

Finding the most probable assignment (or MAP configuration),

$$\boldsymbol{x}^* \triangleq \arg\max_{\boldsymbol{x} \in \boldsymbol{\chi}} P(\boldsymbol{x}\,;\, \theta)$$

can then be framed as an integer linear program over the $\boldsymbol{\delta}_{x_i}, \boldsymbol{\delta}_{x_f}$. One of the methods most often used to tackle this NP-hard, combinatorial discrete optimization is linear programming. To perform this relaxation, we introduce the marginal variables $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_f$, which are constrained to the so-called *local polytope* $\mathcal{L}(\mathcal{G})$:

$$\mathcal{L}(\mathcal{G}) = \left\{ (\boldsymbol{\mu}_i, \boldsymbol{\mu}_f) \;\middle|\; \begin{array}{c} \sum_{\boldsymbol{x}_{f\setminus i}} \mu_f(\boldsymbol{x}_f) = \mu_i(\boldsymbol{x}_i) \\ \sum_{\boldsymbol{x}_i} \mu_i(\boldsymbol{x}_i) = 1 \\ \mu_i(\boldsymbol{x}_i) \geq 0 \wedge \mu_f(\boldsymbol{x}_f) \geq 0 \end{array} \right\}$$

Since every assignment to $\boldsymbol{\delta}_{\boldsymbol{x}_i}$ and $\boldsymbol{\delta}_{\boldsymbol{x}_f}$ in the integer problem also satisfies the constraints of the linear program, the resulting linear programming relaxation

$$\max_{(\boldsymbol{\mu}_i, \boldsymbol{\mu}_f) \in \mathcal{L}(\mathcal{G})} \sum_{f \in F} \boldsymbol{\theta}_f^T \boldsymbol{\mu}_f + \sum_{i \in V} \boldsymbol{\theta}_i^T \boldsymbol{\mu}_i \tag{1}$$

is an upper bound to the original integer problem. For a more thorough treatment, see Wainwright and Jordan (2008) or Koller and Friedman (2009).

There are a number of different approaches to solve the LP relaxation (1). One approach is to solve the dual of (1) using coordinate descent algorithms, as in MPLP (Globerson and Jaakkola, 2007; Werner, 2007). Another approach is based on variants of subgradient descent (Komodakis et al., 2011; Jojic et al., 2010); unlike the coordinate descent algorithms, these are guaranteed to converge to a global optimum. Subgradient descent algorithms such as dual decomposition (Komodakis et al., 2011) solve the dual by separating its objective into simple local problems, and using Lagrange multipliers to force the local solutions to agree on a consensus. In practice it usually takes many iterations to reach a consensus for variables that appear in many local problems and convergence can be slow.

Using the ADMM framework to solve the LP relaxation combines the advantages of both approaches, leading to an algorithm that can converge to the global optimum in a time comparable to coordinate descent algorithms. However, there are a number of different strategies to solve the LP relaxation using ADMM framework. In the next two sections, we give an overview of the ADMM framework for convex optimization, and describe its application to the MAP LP relaxation, including two existing algorithms.

3. ADMM

The Alternating Direction Method of Multipliers, or ADMM, has gained recent popularity as an easy-to-use and effective technique for convex optimization (Boyd et al., 2011).

Consider an optimization over convex functions f and g:

$$\min_{x,z} \; f(x) + g(z) \qquad s.t. \;\; Ax + Bz = c \tag{2}$$

The ADMM algorithm uses an augmented Lagrangian,

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|^2 \tag{3}$$

to enforce the linear constraints, where y is a vector of Lagrange multipliers and $\rho > 0$ is a quadratic penalty coefficient. The benefit of including the quadratic penalty is that the dual function can be shown to be differentiable under rather mild conditions (Boyd et al., 2011). The solution to (2) is then obtained as

$$\max_y \min_{x,z} \; L_\rho(x, z, y),$$

and the ADMM method provides an elegant algorithm for finding this saddle point.

ADMM performs iterative updates of the variables, using coordinate descent over the x and z and subgradient descent over the Lagrange multipliers y at each iteration t:

$$\begin{aligned} x^{(t+1)} &= \arg\min L_\rho(x, z^{(t)}, y^{(t)}) \\ z^{(t+1)} &= \arg\min L_\rho(x^{(t+1)}, z, y^{(t)}) \\ y^{(t+1)} &= y^t + \rho(Ax^{(t+1)} + Bz^{(t+1)} - c) \end{aligned}$$

Choosing the step size equal to the quadratic penalty ρ ensures the algorithm is monotonic.

4. ADMM for MAP-LP

The ADMM algorithm can be applied to the MAP-LP relaxation (1) in variety of ways, depending on how we define functions $f(x)$, $g(z)$ and how the constraints are enforced.

A simple and direct application of (3) to the LP (1) is:

$$\max_{(\boldsymbol{\mu}_i, \boldsymbol{\mu}_f) \in \mathcal{P}(\mu)} \sum_{f \in F} \boldsymbol{\theta}_f^T \boldsymbol{\mu}_f + \sum_{i \in V} \boldsymbol{\theta}_i^T \boldsymbol{\mu}_i + \sum_{\substack{f \in F \\ i \in f}} \lambda_{if}^T (\boldsymbol{A}_{if} \boldsymbol{\mu}_f - \boldsymbol{\mu}_i) - \sum_{\substack{f \in F \\ i \in f}} \frac{\rho}{2} \|\boldsymbol{A}_{if} \boldsymbol{\mu}_f - \boldsymbol{\mu}_i\|^2 \tag{5}$$

where $\boldsymbol{\mu}_f$, $\boldsymbol{\mu}_i$ take the roles of the x, z in ADMM, and we enforce that the $\boldsymbol{\mu}$ live in the probability simplex $\mathcal{P}(\boldsymbol{\mu}) = \{\boldsymbol{\mu} > 0 | \mathbf{1}^T \boldsymbol{\mu} = 1\}$. Lagrange multipliers enforce local consistency ($\mathcal{L}$) among the $\boldsymbol{\mu}$, and $\boldsymbol{A}_{if} \boldsymbol{\mu}_f$ marginalizes $\boldsymbol{\mu}_f$ with respect to variable i.

Unfortunately, (5) is difficult to optimize. Updating $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_f$ at each iteration involves solving the following kind of subproblems, where $h(x)$ is a linear function of x:

$$\min_{x \in \mathcal{P}(x)} h(x) + \frac{\rho}{2} \|Ax - w\|^2 \tag{6}$$

When fixing $\boldsymbol{\mu}_f$ variables, to update $\boldsymbol{\mu}_i$ variables, optimization (6) involves a quadratic term $\sum_f \|\boldsymbol{w}_{if} - \boldsymbol{\mu}_i\|^2$; its solution can be easily computed in closed form using a partitioning technique (Duchi et al., 2008), described for completeness in the Appendix. However when fixing $\boldsymbol{\mu}_i$, optimizing for $\boldsymbol{\mu}_f$ involves the quadratic term $\sum_i \|\boldsymbol{A}_{if} \boldsymbol{\mu}_f - \boldsymbol{w}_i\|^2$ under the constraint $\boldsymbol{\mu}_f \in \mathcal{P}(x)$, for which a general closed-form solution is not easy to compute.

4.1. APLP/ADLP

To overcome such difficulties, a common strategy is to introduce auxiliary variables and reformulate the optimization such that it involves solving only QPs with identity mappings at each step. The APLP algorithm (Meshi and Globerson, 2011) uses this strategy and formulates a primal MAP-LP relaxation with auxiliary variables. APLP keeps a copy $\boldsymbol{\mu}_{fi}$ of the factor marginals $\boldsymbol{\mu}_f$ for each variable $i \in f$ and enforces marginalization constraints over these copies:

$$\begin{aligned} \max_{\substack{(\boldsymbol{\mu}_i, \boldsymbol{\mu}_f) \in \mathcal{P}(\boldsymbol{\mu}) \\ \bar{\boldsymbol{\mu}}_{if})}} & \sum_{f \in F} \boldsymbol{\theta}_f^T \boldsymbol{\mu}_f + \sum_{i \in V} \boldsymbol{\theta}_i^T \boldsymbol{\mu}_i \;+\; \sum_{\substack{f \in F \\ i \in f}} \boldsymbol{y}_f^T (\boldsymbol{\mu}_f - \bar{\boldsymbol{\mu}}_{if}) - \sum_{\substack{f \in F \\ i \in f}} \frac{\rho}{2} \|\boldsymbol{\mu}_f - \bar{\boldsymbol{\mu}}_{if}\|^2 \\ & + \sum_{\substack{f \in F \\ i \in f}} \boldsymbol{\lambda}_{if}^T (\boldsymbol{\mu}_i - \boldsymbol{A}_{if} \bar{\boldsymbol{\mu}}_{if}) - \sum_{\substack{f \in F \\ i \in f}} \frac{\rho}{2} \|\boldsymbol{\mu}_i - \boldsymbol{A}_{if} \bar{\boldsymbol{\mu}}_{if}\|^2 \end{aligned}$$

Then, updating $\boldsymbol{\mu}_i$ or $\boldsymbol{\mu}_f$ involves solving an identity-mapping QP constrained to the probability simplex; this can be done efficiently via partitioning (see appendix). Moreover, $\bar{\boldsymbol{\mu}}_{if}$ can also be computed efficiently, since it requires inverting $Q = I + \boldsymbol{A}_{if}^T \boldsymbol{A}_{if}$, a block-diagonal binary matrix (its entries are zero or one) whose inverse can be computed efficiently in closed form. For more details see Meshi and Globerson (2011).

Although introducing such auxiliary variables makes each step of the algorithm more efficient, the increased number of variables in the optimization, and increased number of constraints to enforce, means that ADMM often needs more iterations to converge. This effect can be seen in our the experiments section.

Meshi and Globerson (2011) also introduced an ADLP algorithm that formulates the dual of the LP (1) as an ADMM optimization problem. Formulating the dual problem requires introducing fewer auxiliary variables; this tends to make ADLP significantly faster to converge than the primal APLP (Meshi and Globerson, 2011). Because of this, in our experiments we compare to ADLP rather than APLP.

4.2. DD-ADMM

Another approach to making each update of ADMM for the MAP-LP tractable is given by Martins et al. (2011) in the DD-ADMM algorithm. They formulate the primal LP as:

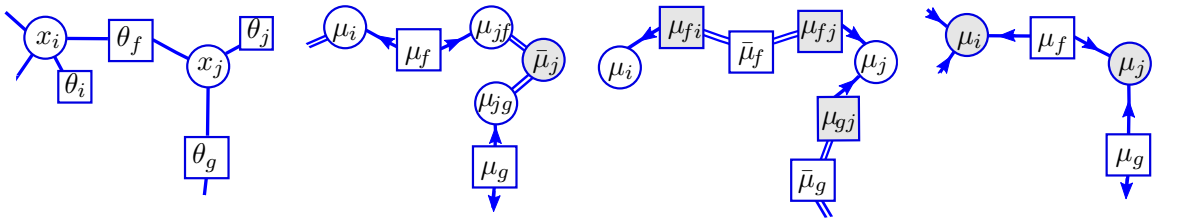
$$\max_{(\bar{\boldsymbol{\mu}}_i, \boldsymbol{\mu}_{if}, \boldsymbol{\mu}_f) \in \mathcal{L}(\mathcal{G})} \sum_{f \in F} \left(\boldsymbol{\theta}_f^T \boldsymbol{\mu}_f + \sum_{i \in V} \frac{1}{d_i} \boldsymbol{\theta}_i^T \boldsymbol{\mu}_{if} \right) \;+\; \sum_{\substack{f \in F \\ i \in f}} \lambda_{if}^T (\boldsymbol{\mu}_{if} - \bar{\boldsymbol{\mu}}_i) \;-\; \sum_{\substack{f \in F \\ i \in f}} \frac{\rho}{2} \|\boldsymbol{\mu}_{if} - \bar{\boldsymbol{\mu}}_i\|^2$$

They then note that the resulting quadratic forms can be solved in closed form for two specific cases corresponding to binary-valued x_i: when the factors are pairwise (involves only two variables), or when they take on specific logical constraints, such as enforcing an "exclusive-or". They propose to apply this to general graphical models by "binarizing" the model, creating a binary variable for each variable and state $x_i = s$, and for each clique state $x_f = (s_1 \dots s_{|f|})$. They argue that the overhead in terms of number of factors and time per update is minimal.

However, what is not obvious is that these many inter-related variables also create many more dependencies to be enforced. Consequently, although the time per iteration is similar to coordinate descent or subgradient methods, the number of iterations required to converge may increase. In our experiments, we find this effect can be significant.

5. Linearized ADMM Algorithm

Ideally, since auxiliary variables increase the number of iterations required for convergence, we would prefer to solve the original, direct application of ADMM in (5). As discussed, the major obstacle in doing so is a difficult quadratic program when updating the $\boldsymbol{\mu}_f$. In this section, we will sidestep this difficulty using a proximal linearization technique, and derive a Linearized Augmented Primal LP (LAPLP) algorithm with fewer auxiliary variables and faster convergence.



(a) Factor graph (b) DD-ADMM (c) APLP (d) LAPLP (this work)

Figure 1: Auxiliary variables and updates in different frameworks. Double lines indicate enforced equality (an identity quadratic term); arrows indicate enforced marginal equality (a non-identity term for μ_f's update). ADMM alternates between updating all shaded nodes, then all unshaded nodes. (a) A portion of the original factor graph. (b) DD-ADMM binarizes each variable (not shown) and creates a copy $\bar{\mu}_i$ of variable marginals μ_i on which it enforces probability simplex constraints. (c) APLP creates a copy μ_{fi} of joint marginals μ_f for each variable $i \in f$, since a single outgoing non-identity marginalization constraint can be enforced in closed form. (d) Our linearized algorithm creates no copies, and linearizes the resulting non-trivial quadratic term on μ_f due to more than one outgoing marginalization constraint (arrow).

Consider again the primal MAP LP relaxation (5). This leads to the following updates at each iteration:

$$\begin{aligned}
\boldsymbol{\mu}_f^{(t+1)} &= \underset{\boldsymbol{\mu}_f \in \mathcal{P}(\boldsymbol{\mu})}{\arg\max}\ \boldsymbol{w}_f^{(t)T}\boldsymbol{\mu}_f - \frac{\rho}{2}\boldsymbol{\mu}_f^T \boldsymbol{Q}_f \boldsymbol{\mu}_f \qquad (7a)\\
\boldsymbol{w}_f^{(t)} &= \boldsymbol{\theta}_f + \sum_{f:i\in f} \boldsymbol{A}_{if}^T(\boldsymbol{\lambda}_{if}^{(t)} + \rho\boldsymbol{\mu}_i^{(t)})\\
\boldsymbol{Q}_f &= \sum_{i\in f} \boldsymbol{A}_{if}^T \boldsymbol{A}_{if}
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\mu}_i^{(t+1)} &= \underset{\boldsymbol{\mu}_i \in \mathcal{P}(\boldsymbol{\mu})}{\arg\max}\ \boldsymbol{w}_i^{(t+1)T}\boldsymbol{\mu}_i - \frac{\rho\, d_i}{2}\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i \qquad (7b)\\
\boldsymbol{w}_i^{(t+1)} &= \boldsymbol{\theta}_i + \sum_{f:i\in f} \left(-\boldsymbol{\lambda}_{if}^{(t)} + \rho \boldsymbol{A}_{if}\boldsymbol{\mu}_f^{(t+1)}\right)
\end{aligned}$$

$$\boldsymbol{\lambda}_{if}^{(t+1)} = \boldsymbol{\lambda}_{if}^{(t)} - \rho\left(\boldsymbol{A}_{if}\boldsymbol{\mu}_f^{(t+1)} - \boldsymbol{\mu}_i^{(t+1)}\right) \qquad (7c)$$

Optimization (7b) is a quadratic program with an identity mapping, constrained to the probability simplex, which can be solved efficiently via partitioning (see appendix). However, as discussed, computing a closed form solution to optimization (7a) with the non-identity mapping $\boldsymbol{Q}_f$ and linear constraints on $\boldsymbol{\mu}_f$ is not trivial. In order to find a closed form solution to optimization (7a) without

Algorithm 1 Linearized APLP

Input: factor graph $(\mathcal{G})$, penalty parameter ρ and maximum iterations T
Initialize $\boldsymbol{\lambda}_{if} = 0$ for all factors $f \in F$ and all $i \in f$
Initialize $\boldsymbol{\mu}_f = MAP(\boldsymbol{\theta}_f)$ for all factors $f \in F$
Initialize $\boldsymbol{\mu}_i = MAP(\boldsymbol{\theta}_i)$ for all variables $i \in V$
for $t = 1$ **to** T **do**
  for each $f \in F$ **do**
    Update $\boldsymbol{\mu}_f^{(t+1)} = Quad(\boldsymbol{\mu}_f)$ by solving (9)
    Update $\boldsymbol{\mu}_i^{(t+1)} = Quad(\boldsymbol{\mu}_i)$ by solving (7b)
    Update $\boldsymbol{\lambda}_{if}^{(t+1)} = \boldsymbol{\lambda}_{if}^{(t)} - \rho \left(\boldsymbol{A}_{if} \boldsymbol{\mu}_f^{(t+1)} - \boldsymbol{\mu}_i^{(t+1)} \right)$
  end for
end for

introducing auxiliary variables, we rewrite (7a) as:

$$\boldsymbol{\mu}_f^{(t+1)} = \underset{\boldsymbol{\mu}_f \in \mathcal{P}(\boldsymbol{\mu})}{\arg\min} - \boldsymbol{\theta}_f^T \boldsymbol{\mu}_f + \frac{\rho}{2} \sum_{i:i \in f} \| \boldsymbol{A}_{if} \boldsymbol{\mu}_f - \boldsymbol{\mu}_i^{(t)} - \frac{1}{\rho} \boldsymbol{\lambda}_{if}^{(t)} \|^2$$

The quadratic term can be approximated by a first order Taylor expansion around the current estimate, plus a proximal term (e.g., Martinet 1970, Rockafellar 1976), giving:

$$\boldsymbol{\mu}_f^{(t+1)} = \underset{\boldsymbol{\mu}_f \in \mathcal{P}(\boldsymbol{\mu})}{\arg\min} \; - \boldsymbol{\theta}_f^T \boldsymbol{\mu}_f \quad + \quad \sum_{i:i \in f} \langle \, \boldsymbol{\mu}_f - \boldsymbol{\mu}_f^{(t)} \, , \, \boldsymbol{A}_{if}^T (\rho (\boldsymbol{A}_{if} \boldsymbol{\mu}_f^{(t)} - \boldsymbol{\mu}_i^{(t)}) - \boldsymbol{\lambda}_{if}^{(t)}) \, \rangle + \sum_{i:i \in f} \frac{\rho \, \eta_{A_{if}}}{2} \| \boldsymbol{\mu}_f - \boldsymbol{\mu}_f^{(t)} \|^2 \quad (8)$$

where $\langle \, \rangle$ is the vector product and $\eta_{A_{if}} > 0$ is a proximal coefficient that will influence the convergence of the algorithm. Eq. (8) can be further simplified to

$$\begin{aligned} \boldsymbol{\mu}_f^{(t+1)} &= \underset{\boldsymbol{\mu}_f \in \mathcal{P}(\boldsymbol{\mu})}{\arg\min} \, \boldsymbol{w}_f^T \boldsymbol{\mu}_f + \frac{\rho \eta_A}{2} \| \boldsymbol{\mu}_f - \boldsymbol{\mu}_f^{(t)} \|^2 \\ \boldsymbol{w}_f &= - \boldsymbol{\theta}_f + \sum_{i:i \in f} \left(\rho (\boldsymbol{A}_{if} \boldsymbol{\mu}_f^{(t)} - \boldsymbol{\mu}_i^{(t)}) - \boldsymbol{\lambda}_{if}^{(t)} \right)^T \boldsymbol{A}_{if} \\ \eta_A &= \sum_{i:i \in f} \eta_{A_{if}} \end{aligned} \quad (9)$$

which is a QP with an identity mapping, which (as before) we can solve efficiently via partitioning. The above steps are summarized in Algorithm 1.

A similar linearization technique was used by Lin et al. (2011) to solve ADMM updates for a low-rank representation problem, a type of subspace clustering task. Linearization has significant advantages: it makes the auxiliary variables unnecessary, saving memory and avoiding updates to those variables. Moreover, without the extra constraints introduced by the auxiliary variables, the convergence (in terms of number of iterations) is also faster.

We illustrate the number of auxiliary variables introduced, along with the ADMM update pattern, for a small part of a factor graph in Figure 1. Figure 1(a) shows a factor graph, with variables

as circles and factors as squares. Figure 1(b)–(d) illustrate the dependence and updates of the DD-ADMM, APLP, and LAPLP algorithms. The alternating ADMM updates are shown using shaded and unshaded nodes; squares indicate marginals over clique configurations ($\boldsymbol{\mu}_f$) and circles indicate marginals over variable configurations (μ_i). Equality constraints are indicated using double lines, and marginalization constraints using arrows, pointing in the direction of the marginalization. The LAPLP update has significantly less variable duplication (some of the duplication of DD-ADMM is not visualized); its difficult quadratic update (7a) is visible as squares (e.g., $\boldsymbol{\mu}_f$) with more than one outgoing arrow.

6. Performance Analysis

6.1. Parameter Selection

Our linearized ADMM is guaranteed to converge to the global optimum if $\eta_{A_{if}} \geq \|\boldsymbol{A}_{if}\|^2$; see Lin et al. (2011). For this reason, we usually set η_A in (9) as

$$\eta_A = \|\boldsymbol{A}\|^2 = \sum_{i \in f} \|\boldsymbol{A}_{if}\|^2;$$

However, our experiments show that in practice and for the range of quadratic penalty terms ρ that are of interest, linearized ADMM converges to the global optimum even when η_A is set to smaller values. We compare the results for setting $\eta_A = \|\boldsymbol{A}\| = (\sum_{i \in f} \|\boldsymbol{A}_{if}\|^2)^{\frac{1}{2}}$ and $\eta_A = 2\|\boldsymbol{A}\|$ as well. Our experiments show that choosing smaller values for η_A results in faster convergence to global optimum. However, special care needs to be made when choosing the penalty ρ to make sure the algorithm converges to the global optimum.

Selecting the penalty parameter ρ, is an important issue when using any of the ADMM based algorithms. Setting ρ very small or very large makes ADMM based algorithms slow. In our experiments we studied the effect of choosing the penalty by cross validation. To do so, we run the ADMM based algorithms on a small number (e.g. k) istances in a problem class using a range of penalty terms ρ, and select the best penalty on those for the remaining instances in the same class. Figure 3 (bottom) and Figure 4 (bottom) compare the relative convergence times when using this selected penalty for all problems, compared to the case where the best value of penalty is chosen for each problem independently. As the results show, the relative convergence time of ADMM based algorithms does not change in the two different setting. However, this choice of ρ slows down ADMM based algorithms compared to MPLP.

6.2. Experimental Results

To evaluate our linearized augmented primal LP (LAPLP) algorithm, we compare it with the two other ADMM based algorithms for finding approximate MAP solutions, DD-ADMM by Martins et al. (2011) and ADLP by Meshi and Globerson (2011) (since it is faster than the more comparable APLP updates from the same work) as well as coordinate descent algorithm MPLP Globerson and Jaakkola (2007). For ADLP, we use the implementation provided in the Darwin C++ framework (Gould, 2012). For DD-ADMM, we use the code provided online by the authors[1].

Note that the DD-ADMM code includes some basic "message scheduling" heuristics, updating only those variables whose neighbors have changed significantly at each iteration. Since the ADLP

1. http://www.ark.cs.cmu.edu/AD3

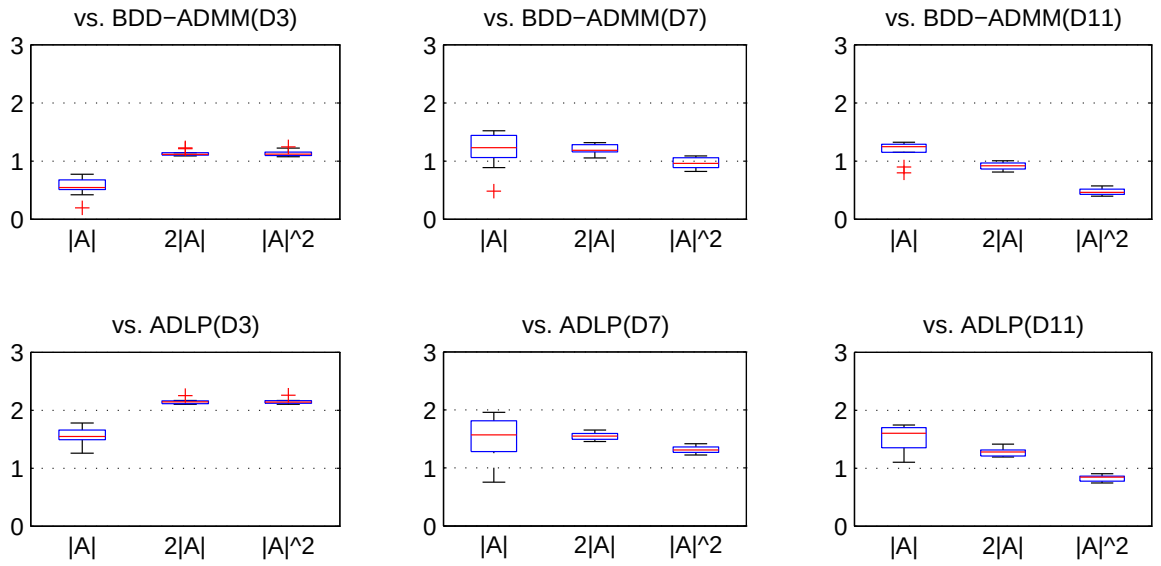


Figure 2: Comparison of convergence time of LAPLP with DD-ADMM(binarized) and ADLP for different Potts models. Log relative convergence time $-\log(t_c(\text{LAPLP})/t_c(\text{XLP}))$ is used for comparison, where $t_c(\text{LAPLP})$ is the convergence time of the LAPLP algorithm (tolerance=1e-4) and $t_c(\text{XLP})$ is the convergence time of XLP. Here XLP represents any of the algorithms DD-ADMM, ADLP, or MPLP. MPLP converges to local optimum in these experiments.

implementation does not perform scheduling, we disabled this aspect of DD-ADMM and did not include scheduling in our own implementation of LAPLP.

We evaluate the algorithms on different sets of problems including Potts model, pedigree trees and protein side-chain prediction. To compare different methods we use relative convergence time $-\log(t_c(\text{LAPLP})/t_c(\text{XLP}))$, where $t_c(\text{LAPLP})$ is the convergence time of LAPLP algorithm (tolerance=1e-4) and $t_c(\text{XLP})$ is the convergence time of XLP algorithm, where XLP can be replaced by any of DD-ADMM, ADLP and MPLP algorithms.

Potts Models To compare different algorithms on Potts models, we generated 20x20 Potts models with single node log-potentials chosen as $\theta_i(x_i) \sim \mathcal{U}[-1, 1]$ and edge log potentials as $\theta_{i,j}(x_i, x_j) \sim \mathcal{U}[-5, 5]$ if $x_i \neq x_j$ and 0 otherwise. We generated models with different variable cardinalities (3, 7 and 11) to study the effect of model size on different algorithms.

Figure 2 compares the convergence time of LAPLP to DD-ADMM (binarized) and ADLP, averaged over 10 models of same size (models with multi-valued variables with 3, 7 and 11 different values respectively). As shown LAPLP is faster than both ADLP and DD-ADMM. Its important to note that MPLP converges to local optima in these experiments while ADMM based algorithms are able to find the global optimum.

Pedigree Models We also compared the algorithms on pedigree models from UAI 2008 biological linkage analysis data. These models involve non-pairwise factors with variables that have cardinal-

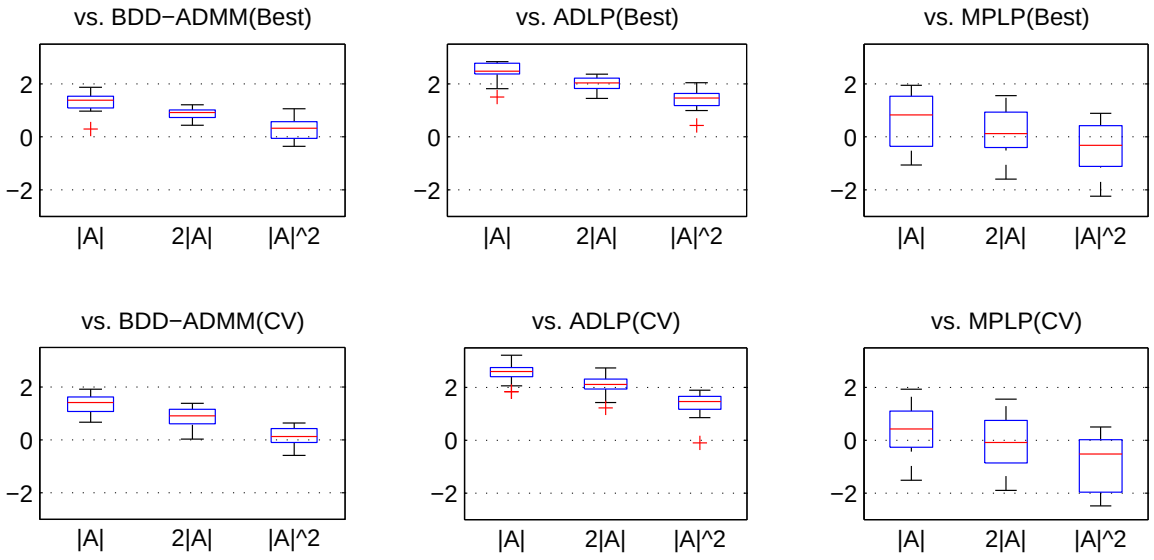


Figure 3: Comparison of convergence time of LAPLP with DD-ADMM(binarized) and ADLP on pedigree trees when using the best penalty ρ for each model (top) and choosing the penalty ρ using cross validation. Log relative convergence time $-\log(t_c(\text{LAPLP})/t_c(\text{XLP}))$ is used for comparison, where $t_c(\text{LAPLP})$ is the convergence time of the LAPLP algorithm (tolerance=1e-4) and $t_c(\text{XLP})$ is the convergence time of XLP. Here XLP represents any of the algorithms DD-ADMM, ADLP, or MPLP.

ities between 2 and 7. Of the total 19 pedigree models, MPLP converged to local optima in 3 experiments. Figure 3 (top) compares the convergence time of LAPLP to DD-ADMM, ADLP and MPLP. As shown here, MPLP can converge faster than ADMM based algorithms in some models but it has the potential to converge to local optima. Again, the linearized ADMM converges faster than existing globally convergent approaches. This shows that linearizing the quadratic term helps improve convergence time by avoiding the introduction of auxiliary variables and their corresponding constraints.

Protein Side-chain Prediction Finally we evaluate the algorithms on protein side-chain prediction problems from Yanover and Weiss (2003) and Yanover et al. (2006)[2]. We use the set of "large" model instances, containing 20 problems of between $300 - 1000$ amino acids (variables), each with $2 - 81$ possible states (average cardinality ≈ 20) and pairwise potential functions. These results show that ADLP's convergence time is less affected by model size compared to DD-ADMM (binarized) and LAPLP and convergence time of the three algorithms are comparable in half of the experiments. MPLP convergence time is much faster when it finds the global optimum. The results of these experiments are summarized in Figure 4 (top).

Figure 5 compares the behavior of LAPLP, ADLP and DD-ADMM across the three sets of problems. Since different models have different energy values and times to convergence, to plot average performance we compute the normalized energy for each algorithm, consisting of the percentage in-

2. http://cyanover.fhcrc.org/proteinMRFs.html

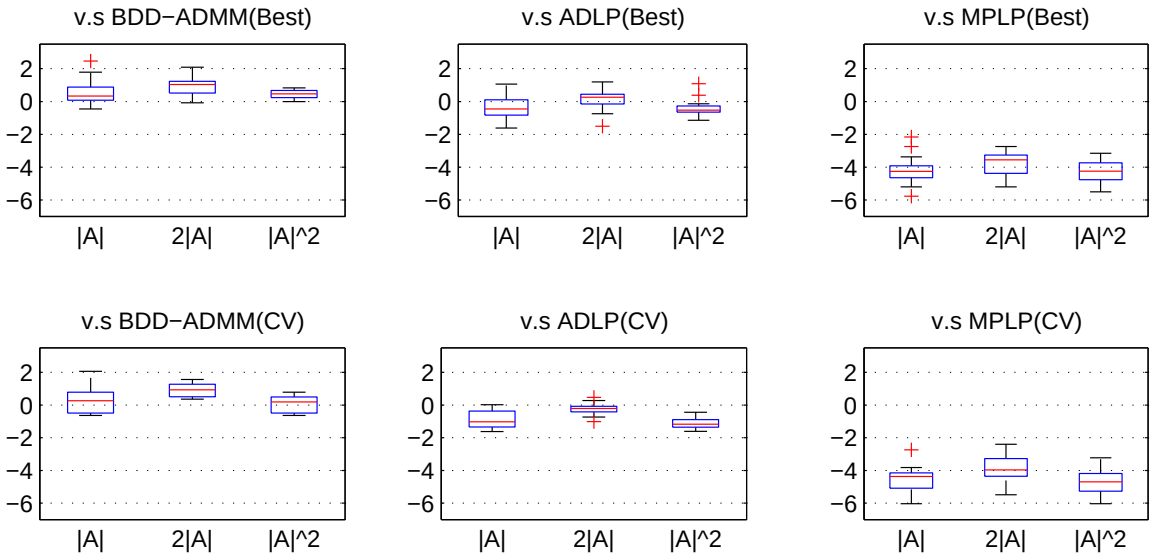


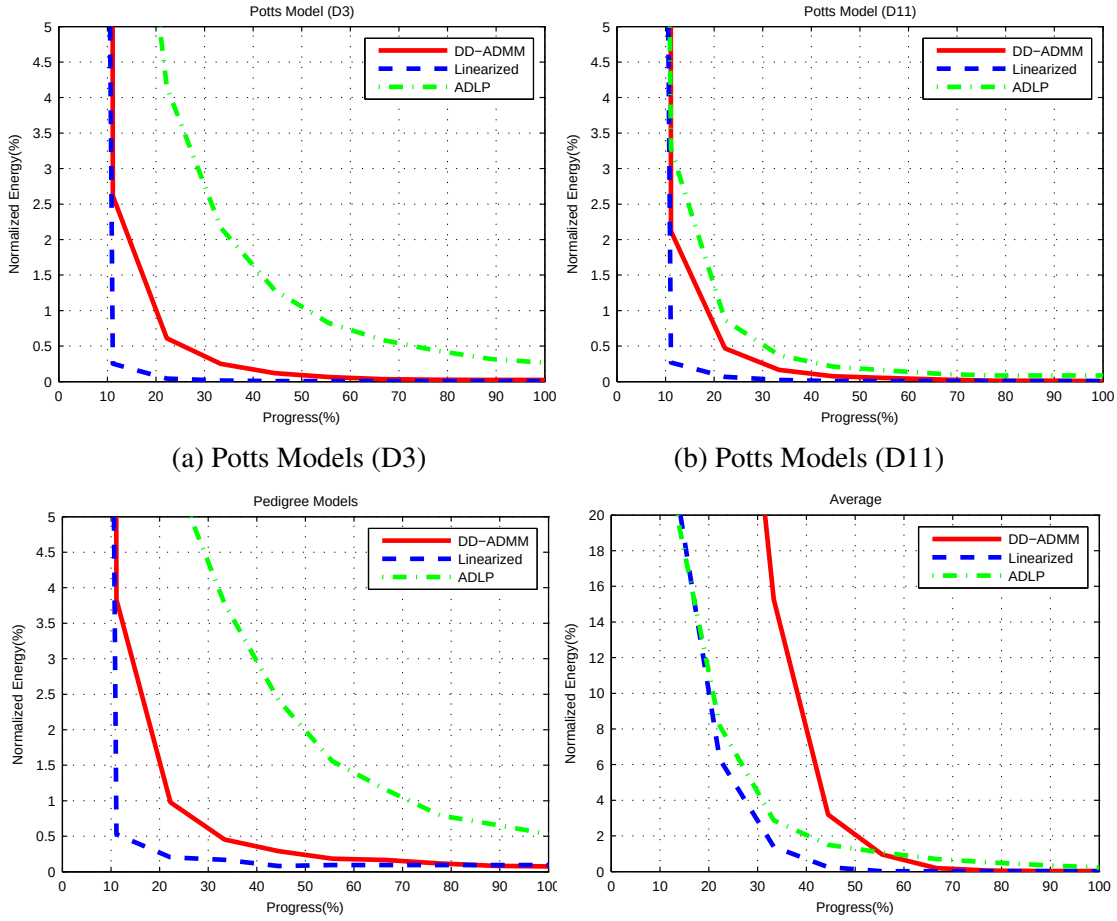
Figure 4: Comparison of convergence time of LAPLP with DD-ADMM(binarized) and ADLP on protein side-chain prediction when using the best penalty ρ for each model (top) and choosing the penalty ρ using cross validation. Log relative convergence time $-\log(t_c(\text{LAPLP})/t_c(\text{XLP}))$ is used for comparison, where $t_c(\text{LAPLP})$ is the convergence time of the LAPLP algorithm (tolerance=1e-4) and $t_c(\text{XLP})$ is the convergence time of XLP. Here XLP represents any of the algorithms DD-ADMM, ADLP, or MPLP.

crease in energy over the optimal value of the LP at convergence, and plot it against the percentage of time used compared to the convergence time for our LAPLP method on that model.

7. Discussion

In this paper, we presented an algorithm based on the Alternating Direction Method of Multipliers (ADMM) for approximate MAP inference using its linear programming relaxation. Our algorithm is based on augmenting the primal MAP-LP with a quadratic term that enforces strict convexity of the Lagrangian, and solving this quadratic form by linearization with an additional proximal term. Importantly, we find that performing such approximate solutions does not significantly affect the convergence time of the ADMM algorithm. We compared our algorithm with two existing ADMM-based algorithms on Potts models, pedigree trees and protein side-chain prediction problems for approximate MAP inference, showing that our linearized primal MAP-LP algorithm can solve MAP inference faster than methods based on auxiliary variables in models with non-binary variables. We also showed that a cross validation procedure can be used to choose the penalty term ρ for a problem class.

Several practical improvements can be considered over our basic algorithm. One is to use an adaptive penalty parameter ρ, which may improve convergence in practice. However, the theoretical convergence guarantees of ADMM may no longer hold. Another potential improvement is to use a scheduling method (Elidan et al., 2006; Tarlow et al., 2011) to select which sub-problems to solve



(a) Potts Models (D3)

(b) Potts Models (D11)

(c) Pedigree Models

(d) Protein Models

Figure 5: Comparing average run time of different ADMM algorithms. We show the percent increase in energy over the optimal LP value, relative to the percentage of time to our LAPLP algorithms convergence, averaged across problem instances. Here LAPLP is fastest, closely followed by ADLP, with the binarized DD-ADMM slower for much of the runtime but catching up near the end.

during each iteration of ADMM. As a simple example, we need only solve each local sub-problem $\boldsymbol{\mu}_f$ if some neighboring consensus variable $\boldsymbol{\mu}_i$ has been changed at the previous iteration, since otherwise the previous results can be simply re-used.

Acknowledgments

This research was supported in part by NSF grants IIS-1065618, IIS-1254071, and DMS-0928427.

References

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

John Duchi, Shai S. Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 272–279, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390191. URL http://dx.doi.org/10.1145/1390156.1390191.

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, pages 165–173, Boston, Massachussetts, 2006.

Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1): 17 – 40, 1976. ISSN 0898-1221. doi: 10.1016/0898-1221(76)90003-1. URL http://www.sciencedirect.com/science/article/pii/0898122176900031.

Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems*, 2007.

Stephen Gould. Darwin, 2012. http://mloss.org/software/view/362/.

V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):531 –552, march 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.108.

Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 612–620. 2011.

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Revue Française dInformatique et de Recherche Opérationelle*, 4:154–158, 1970.

André L. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. An augmented Lagrangian approach to constrained MAP inference. In *ICML*, pages 169–176, 2011.

Ofer Meshi and Amir Globerson. An alternating direction method for dual MAP LP relaxation. In *ECML/PKDD (2)*, pages 470–483, 2011.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877, 1976.

Daniel Tarlow, Dhruv Batra, Pushmeet Kohli, and Vladimir Kolmogorov. Dynamic tree block coordinate ascent. In *ICML*, pages 113–120, 2011.

Martin Wainwright and Michael Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

T. Werner. A linear programming approach to max-sum problem: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):1165 –1179, july 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1036.

Chen Yanover and Yair Weiss. Approximate inference and protein-folding. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1457–1464. MIT Press, Cambridge, MA, 2003.

Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation - an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

Appendix A

Several components of APLP and our LAPLP require optimizing a quadratic form with identity matrix over a probability simplex, i.e., a Euclidean projection onto the simplex. Duchi et al. (2008) describe an efficient algorithm for this projection, which can be more formally described as:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}-\mathbf{v}\|_2^2 \quad s.t. \quad w_i \geq 0, \quad \sum_{i=1}^{n} w_i = z \tag{10}$$

with $z = 1$ for the probability simplex. The solution to (10) can be found using Algorithm 2.

Algorithm 2 Efficient projection on to the l_1 ball

Input: A vector $\mathbf{v} \in \mathbb{R}^n$ and a scalar $z > 0$

Sort $\mathbf{v}$ into $\nu : \nu_1 \geq \nu_2 \geq \ldots \geq \nu_p$

Find J, the largest j such that $\nu_j - \frac{1}{j}\Big(\sum_{r=1}^{j} \nu_r - z\Big) > 0$

Define $S = \frac{1}{J}\Big(\sum_{i=1}^{J} \nu_i - z\Big)$

Output: $\mathbf{w}$ $s.t.$ $w_i = \max\{v_i - S, 0\}$