PROXIMAL ALTERNATING-DIRECTION MESSAGE-PASSING FOR MAP LP RELAXATION

Guoqiang Zhang and Richard Heusdens

Department of Intelligent Systems
Delft University of Technology
Delft, the Netherlands
Email: {g.zhang-1, r.heusdens}@tudelft.nl

ABSTRACT

Linear programming (LP) relaxation for MAP inference over (factor) graphic models is one of the fundamental problems in machine learning. In this paper, we propose a new message-passing algorithm for the MAP LP-relaxation by using the proximal alternating-direction method of multipliers (PADMM). At each iteration, the new algorithm performs two layers of optimization, that is node-oriented optimization and factor-oriented optimization. On the other hand, the recently proposed augmented primal LP (APLP) algorithm, based on the ADMM, has to perform three layers of optimization. Our algorithm simplifies the APLP algorithm by removing one layer of optimization, thus reducing the computational complexities and further accelerating the convergence rate. We refer to our new algorithm as the proximal alternating-direction (PAD) algorithm. Experimental results confirm that the PAD algorithm indeed converges faster than the APLP method.

Index Terms— graphic models, ADMM, PADMM, message-passing, MAP, LP relaxation

1. INTRODUCTION

Graphic models are widely used to describe the statistics of a set of discrete random variables. Such formulation nicely captures the sparsity of the inter-dependencies of the variables, facilitating probabilistic inferences via local message-passing over the graph. One fundamental inference task is to find *maximum a-posteriori* (MAP) assignment. For tree-structured graphs, it is well known that the MAP solution can be easily computed by dynamic programming. On the other hand, for more general graphs, the computation of the MAP solution is known to be NP-hard.

In the literature, considerable attention has been devoted to approximate the MAP solution by solving a relaxed optimization problem. Due to its simplicity and effectiveness, linear programming (LP) relaxation has become one of the most popular approximation approaches [1, 2, 3, 4]). The research challenge is how to perform local updates (or message-passing) efficiently between neighboring nodes to reach the optimal solution of the LP relaxation.

In the past decade, various message-passing algorithms have been proposed for solving MAP LP-relaxation. Some algorithms perform block coordinate-descent operations, such as max-sum diffusion (MSD) [2] and max product linear-programming (MPLP) [5] algorithms. While these algorithms usually converge fast, they do not always reach the global optimal solution of the LP-relaxation [6]. To overcome the suboptimality issue, some algorithms perform variants of subgradient descent to the dual problem [7, 8]. Although these algorithms are guaranteed to converge globally, they typically converge slower than the block coordinate-descent ones.

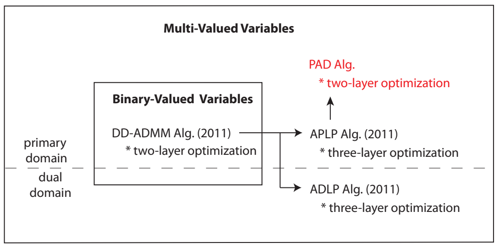


Fig. 1. ADMM-based algorithms for solving MAP LP-relaxation. The PAD algorithm is the new one we will present in this paper.

Recently, the *alternating direction method of multipliers* (ADMM) [9, 10, 11, 12] has been employed in designing more efficient and globally convergent message-passing algorithms for MAP LP-relaxation [13, 14]. The work in [13] considered the primal LP problem over graphs with binary-valued variables by using the ADMM. The proposed message-passing algorithm (referred to as DD-ADMM) in [13] performs two layers of optimization at each iteration, which are node-oriented optimization and factor-oriented optimization (see Fig. 1). In [14], Meshi and Globerson extended the work of [13] by considering graphic models with multi-valued variables. The authors proposed two algorithms: the augmented primal LP (APLP) method and the augmented dual LP (ADLP) method. The key point in designing the above two algorithms was to introduce a considerable number of auxiliary variables to enable the application of ADMM. Further, both algorithms have to perform three layers of optimization at each iteration (see Fig. 1). Compared with the DD-ADMM algorithm, the APLP and ADLP algorithms require one more layer of optimization due to the large number of auxiliary variables, increasing the computational complexities and further slowing down the convergence speed.

In this paper, we reconsider solving the primal MAP LP-relaxation over the graphs with multi-valued variables. We apply the proximal ADMM (PADMM), which is a combination of ADMM and proximal point approach [15]. Our primary motivation is to design a more efficient message-passing algorithm with fewer number of auxiliary variables than that of [14].

Formally, in the design of the new PADMM-based algorithm, we introduce feedback from last iteration in computing a new estimate of the variables. The introduction of feedback signal makes it possible to introduce a small number of auxiliary variables compared to [14]. Our main contribution is the nice construction of the feedback signal, which leads to simple computation of the estimate per iteration. We refer to the new algorithm as the proximal alternating-direction (PAD) algorithm. The PAD algorithm only performs two layers of optimization per iteration due to the feedback signal (see Fig. 1). The convergence of the PAD algorithm follows directly from that of the PADMM [15, 12]. Experimental results show that the PAD algorithm indeed converges faster than the APLP algorithm.

2. PROBLEM FORMULATION

In this section, we first briefly describe the discrete MAP problem. After that, we present the LP relaxation to the discrete MAP problem.

2.1. Discrete MAP problem

Let $\boldsymbol{X} = \{X_i | i = 1, \ldots, n\}$ be a vector of discrete random variables, where each variable X_i takes its value from a discrete set $\mathcal{X}_i$. For each variable X_i, we denote its realization as $x_i \in \mathcal{X}_i$. More generally, for a subvector $\boldsymbol{X}_c$ of variables, where $c \subseteq \{1, \ldots, n\}$, we denote its realization as $\boldsymbol{x}_c \in \mathcal{X}_c$. When $c = \{1, \ldots, n\}$, we let $\boldsymbol{x}_c = \boldsymbol{x}$ and $\mathcal{X}_c = \mathcal{X}$ for simplicity. Suppose that the joint probability $P(\boldsymbol{x})$ of the variables admits a decomposition with respect to a factor graph $G = (V, C)$:

$$P(\boldsymbol{x}) \propto \exp\left(\sum_{i\in V} \theta_i(x_i) + \sum_{c\in C} \theta_c(\boldsymbol{x}_c)\right), \tag{1}$$

where C denotes a set of subsets of $\{1, \ldots, n\}$. Each element $c \in C$ is associated with a factor in the graph. We use $N(i)$ to denote the set of factors that include i, i.e., $N(i) = \{c | i \in c\}$. We use $|N(i)|$ to denote the number of factors in $N(i)$. Similarly, we let $|c|$ denote the number of variables in c. In the literature, $\theta_i(x_i)$ and $\theta_c(\boldsymbol{x}_c)$ are named as unary and higher-order log-potentials functions, respectively. The probability formulation (1) is termed as the Markov Random Field (MRF).

We are interested in finding the most probable assignment (MAP assignment) for the distribution $P(\boldsymbol{x})$ in (1)

$$\boldsymbol{x}^{MAP} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \left(\sum_{i\in V} \theta_i(x_i) + \sum_{c\in C} \theta_c(\boldsymbol{x}_c)\right). \tag{2}$$

As mentioned in the introduction, the above problem is NP-hard for general graphic models. As a result, the research attention has moved to the development of approximation approaches such as the MAP-LP relaxation. To simplify notations in the following, we use the vectors $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_c$ to represent the log-potential functions for any $i \in V$ and $c \in C$, respectively.

2.2. LP Relaxation

LP relaxation is one of the most popular approximation approaches for the MAP problem [1, 2]. The basic idea is to introduce two sets of auxiliary variables $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i, i \in V\}$ and $\boldsymbol{\nu} = \{\boldsymbol{\nu}_c, c \in C\}$ such that

$$\boldsymbol{\mu} \in \mathcal{L}_{\boldsymbol{\mu}} \triangleq \left\{\boldsymbol{\mu} | \boldsymbol{\mu}_i \geq 0, \mathbf{1}^\top \boldsymbol{\mu}_i = 1, \forall i \in V\right\}$$
$$\boldsymbol{\nu} \in \mathcal{L}_{\boldsymbol{\nu}} \triangleq \left\{\boldsymbol{\nu} | \boldsymbol{\nu}_c \geq 0, \mathbf{1}^\top \boldsymbol{\nu}_c = 1, \forall c \in C\right\}.$$

We refer to $\boldsymbol{\mu}_i$ as the node variable for $i \in V$. Similarly, we refer to $\boldsymbol{\nu}_c$ as the factor variable for $c \in C$. The MAP problem can then be approximated by a tractable LP in terms of $(\boldsymbol{\mu}, \boldsymbol{\nu})$. Formally, the MAP-LP relaxation to (2) takes the form

$$(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*) = \arg\max_{\boldsymbol{\mu}\in\mathcal{L}_{\boldsymbol{\mu}}, \boldsymbol{\nu}\in\mathcal{L}_{\boldsymbol{\nu}}} \left(\sum_i \boldsymbol{\mu}_i^\top \boldsymbol{\theta}_i + \sum_c \boldsymbol{\nu}_c^\top \boldsymbol{\theta}_c\right), \tag{3}$$

subject to

$$\boldsymbol{\mu}_i = \boldsymbol{A}_{ci} \boldsymbol{\nu}_c \quad \forall c, i : i \in c, \tag{4}$$

where $\boldsymbol{A}_{ci}(x_i, \boldsymbol{x}_c) = 1$ if $[\boldsymbol{x}_c]_i = x_i$, and 0 otherwise. It is known that if the optimal solution $\boldsymbol{\mu}^*$ in (3) takes integer values, we obtain the exact MAP solution. On the other hand, if $\boldsymbol{\mu}^*$ takes non-integer values, a good approximation of the MAP solution can often be obtained by making hard-decision to $\boldsymbol{\mu}^*$. That is for each variable X_i, its MAP assignment k_i is approximated as $k_i = \arg\max(\boldsymbol{\mu}_i^*)$.

For the primal LP formulation (3), its dual LP takes the form of [14]

$$\min_{\delta} \left(\sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\right) + \sum_c \max_{\boldsymbol{x}_c} \left(\theta_c(\boldsymbol{x}_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\right)\right), \tag{5}$$

where $\delta_{ci} = \{\delta_{ci}(x_i) | x_i \in \mathcal{X}_i\}$ and $\delta = \{\delta_{ci} | c \in C, i : i \in c\}$. Note that the dual problem (5) has a small number of variables and no explicit constraints as compared to the primal problem (3), which makes it relatively easier to solve. In the literature, much progress has been obtained by considering the dual problem [5, 14]. In this paper, we will focus on the primal LP problem (3) instead.

3. PROXIMAL ALTERNATING-DIRECTION MESSAGE-PASSING

In this section we derive the PAD algorithm by applying the PADMM to the primal MAP-LP problem (3). The PADMM is an extension of the ADMM that introduces feedback from last iteration in constructing the proximal augmented Lagrangian function for next iteration [15, 12].

3.1. Functional construction

In this subsection, we study how to construct the proximal augmented Lagrangian function properly for (3) that leads to simple computation per-iteration. As will be shown below, the main difficulty sits in the estimation of $\boldsymbol{\nu}^*$ at each iteration.

Suppose that the estimate $(\hat{\boldsymbol{\mu}}^k, \hat{\boldsymbol{\nu}}^k)$ is obtained after the first k iterations, where $k \geq 0$. We would like to compute the new estimate $(\hat{\boldsymbol{\mu}}^{k+1}, \hat{\boldsymbol{\nu}}^{k+1})$ in next iteration. By following the PADMM framework [15, 12], we construct a proximal augmented Lagrangian function for the MAP-LP problem (3) as

$$\begin{aligned} & L_\rho^{(k)}(\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\gamma}) \\ = \; & -\sum_i \boldsymbol{\mu}_i^\top \boldsymbol{\theta}_i - \sum_c \boldsymbol{\nu}_c^\top \boldsymbol{\theta}_c + \sum_c \sum_{i:i\in c} \boldsymbol{\gamma}_{ci}^\top (\boldsymbol{A}_{ci}\boldsymbol{\nu}_c - \boldsymbol{\mu}_i) \\ & + \frac{\rho}{2} \sum_c \sum_{i:i\in c} \|\boldsymbol{A}_{ci}\boldsymbol{\nu}_c - \boldsymbol{\mu}_i\|_2^2 \\ & + \frac{1}{2} \sum_{c\in C} (\boldsymbol{\nu}_c - \hat{\boldsymbol{\nu}}_c^k)^\top \boldsymbol{D}_c (\boldsymbol{\nu}_c - \hat{\boldsymbol{\nu}}_c^k), \end{aligned} \tag{6}$$

where $\boldsymbol{\gamma}$ is the Lagrangian multiplier, and $\rho > 0$ is the weighting factor for the quadratic penalty function. The last term in (6) captures the feedback $\hat{\boldsymbol{\nu}}^k$ in computing the new estimate. The positive semi-definite matrices $\{\boldsymbol{D}_c, c \in C\}$ in (6) control the amount of feedback, which are remained to be specified. By following the updating procedure of the PADMM, the new estimate $(\hat{\boldsymbol{\mu}}^{k+1}, \hat{\boldsymbol{\nu}}^{k+1}, \hat{\boldsymbol{\gamma}}^{k+1})$ can be computed as

$$\begin{aligned} \hat{\boldsymbol{\nu}}^{k+1} &= \arg\min_{\boldsymbol{\nu}\in\mathcal{L}_{\boldsymbol{\nu}}} L_\rho^{(k)}(\hat{\boldsymbol{\mu}}^k, \boldsymbol{\nu}, \hat{\boldsymbol{\gamma}}^k) && (7) \\ \hat{\boldsymbol{\mu}}^{k+1} &= \arg\min_{\boldsymbol{\mu}\in\mathcal{L}_{\boldsymbol{\mu}}} L_\rho^{(k)}(\boldsymbol{\mu}, \hat{\boldsymbol{\nu}}^{k+1}, \hat{\boldsymbol{\gamma}}^k) && (8) \\ \hat{\boldsymbol{\gamma}}_{ci}^{k+1} &= \hat{\boldsymbol{\gamma}}_{ci}^k + \rho(\boldsymbol{A}_{ci}\hat{\boldsymbol{\nu}}_c^{k+1} - \hat{\boldsymbol{\mu}}_i^{k+1}). \quad \forall c, i : i \in c && (9) \end{aligned}$$

We note that the computation of $\hat{\boldsymbol{\mu}}^{k+1}$ and $\hat{\boldsymbol{\gamma}}^{k+1}$ in (8)-(9) is straightforward (see next subsection). On the other hand, the computation of $\hat{\boldsymbol{\nu}}^{k+1}$ in (7) is rather difficult due to the constraint $\mathcal{L}_{\boldsymbol{\nu}}$ and the matrices $\{\boldsymbol{A}_{ci}\}$. Our motivation for introducing the matrices $\{\boldsymbol{D}_c\}$ is to make the resulting constrained quadratic optimization for $\boldsymbol{\nu}$ easily solvable. We explain in the following how to select the matrices $\{\boldsymbol{D}_c, c \in C\}$ in (6) to facilitate the computation of $\hat{\boldsymbol{\nu}}^{k+1}$.

We now focus on minimizing $L_\rho^{(k)}(\hat{\boldsymbol{\mu}}^k, \boldsymbol{\nu}, \hat{\boldsymbol{\gamma}}^k)$ with respect to $\boldsymbol{\nu}$. Note from (6) that the minimization of $L_\rho^{(k)}(\hat{\boldsymbol{\mu}}^k, \boldsymbol{\nu}, \hat{\boldsymbol{\gamma}}^k)$ can be broken into a set of independent subproblems. Further, each subproblem is in fact a constrained quadratic optimization problem in terms of a factor variable $\boldsymbol{\nu}_c$, $c \in C$:

$$\hat{\boldsymbol{\nu}}_c^{k+1} = \arg\min_{\boldsymbol{\nu}_c\in\mathcal{M}_c} \left(\frac{1}{2}\boldsymbol{\nu}_c^\top \boldsymbol{J}_c \boldsymbol{\nu}_c - \boldsymbol{h}_c^{(k)\top}\boldsymbol{\nu}_c \right), \qquad (10)$$

where

$$\begin{aligned} \boldsymbol{J}_c &= \rho \sum_{i\in c} \boldsymbol{A}_{ci}^\top \boldsymbol{A}_{ci} + \boldsymbol{D}_c \\ \boldsymbol{h}_c^{(k)} &= \boldsymbol{\theta}_c + \sum_{i\in c} \boldsymbol{A}_{ci}^\top(\rho\boldsymbol{\mu}_i^k - \boldsymbol{\gamma}_{ci}^k) + \boldsymbol{D}_c\boldsymbol{\nu}_c^k \\ \mathcal{M}_c &= \left\{ \boldsymbol{\nu}_c \geq 0 \middle| \boldsymbol{1}^\top\boldsymbol{\nu}_c = 1 \right\}, \end{aligned}$$

where the convex set $\mathcal{M}_c$ is in fact the simplex for $\boldsymbol{\nu}_c$.

In general, for an arbitrary positive semi-definite matrix $\boldsymbol{J}_c$, it is difficult to solve (10) directly. If, on the other hand, $\boldsymbol{J}_c$ takes a special structure, the problem (10) may have a simple solution. In our work, we choose a quadratic matrix $\boldsymbol{D}_c$ such that the resulting matrix $\boldsymbol{J}_c$ takes the form of

$$\boldsymbol{J}_c = \begin{pmatrix} \sigma_c & \tau_c & \cdots & \tau_c \\ \tau_c & \sigma_c & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tau_c \\ \tau_c & \cdots & \tau_c & \sigma_c \end{pmatrix}, \qquad (11)$$

where $\sigma_c > \tau_c \geq 0$. That is all the diagonal elements of $\boldsymbol{J}_c$ are identical and all the off-diagonal elements are identical. It is known from [16] that such symmetry of $\boldsymbol{J}_c$ enables us to solve (10) in a few number of steps. Specifically, the number of steps is in order of the dimensionality of $\boldsymbol{\nu}_c$.

Depending on the different values of τ_c in (11), we can choose different matrices for $\boldsymbol{D}_c$ accordingly. The basic principle is to make the matrix $\boldsymbol{D}_c$, $c \in C$, as small as possible (i.e., in a matrix-norm sense) in achieving the matrix form (11) for $\boldsymbol{J}_c$. By doing so, the solution $\hat{\boldsymbol{\nu}}_c^{k+1}$ would be more effective, accelerating the convergence speed. We note that $\{\boldsymbol{A}_{ci} | c \in C, i \in c\}$ is data-independent. Therefore, once $\boldsymbol{D}_c$ is chosen, it can be used in every iteration when implementing the corresponding message-passing algorithm.

We note that (7)-(9) only involves two layers of optimization. That is the node-oriented optimization for $\boldsymbol{\mu}$ and the factor-oriented optimization for $\boldsymbol{\nu}$. This is mainly because of the introduction of the feedback signal (the last term in (6)), which makes the individual minimization of $L_\rho^{(k)}(\boldsymbol{\nu}, \boldsymbol{\mu}, \boldsymbol{\gamma})$ over $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ easily solvable (see next subsection for details).

Remark 1. *It is worth noting that in [14], another set of auxiliary variables $\bar{\boldsymbol{\nu}} = \{\bar{\boldsymbol{\nu}}_c, c \in C\}$ (a copy of $\boldsymbol{\nu}$) has been introduced in designing the APLP algorithm. The set $\bar{\boldsymbol{\nu}}$ are used to bridge the connection between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ in (3). As a result, each iteration of the APLP algorithm involves three layers of optimization, where the third layer of optimization is for $\bar{\boldsymbol{\nu}}$.*

3.2. Computing new estimate

In this subsection, we compute the new estimate $(\hat{\boldsymbol{\mu}}^{k+1}, \hat{\boldsymbol{\nu}}^{k+1}, \hat{\boldsymbol{\gamma}}^{k+1})$ by following the PADMM updating procedure (7)-(9). For the computation of $\hat{\boldsymbol{\nu}}^{k+1}$, we make use of the special structure $\boldsymbol{J}_c$ in (11).

We first consider the minimization of $L_\rho^{(k)}(\hat{\boldsymbol{\mu}}^k, \boldsymbol{\nu}, \hat{\boldsymbol{\gamma}}^k)$ over $\boldsymbol{\nu}$. From the analysis in subsection 3.1, it is clear that the new estimate $\hat{\boldsymbol{\nu}}^{k+1} = \{\hat{\boldsymbol{\nu}}_c^{k+1} | c \in C\}$ can be computed by solving (10) for each factor $c \in C$. We suppose that the matrix $\boldsymbol{J}_c$ in (10) takes the form of (11) due to careful construction of $\boldsymbol{D}_c$. In [16], an efficient algorithm has been proposed for solving (10)-(11). We refer to the algorithm as Subroutine "projectSimplex" in our work. The updating procedure for $\hat{\boldsymbol{\nu}}^{k+1}$ is summarized in **Algorithm 1**.

Next we consider the minimization of $L_\rho^{(k)}(\boldsymbol{\mu}, \hat{\boldsymbol{\nu}}^{k+1}, \hat{\boldsymbol{\gamma}}^k)$ over $\boldsymbol{\mu}$. Again, the minimization problem can be broken into a set of independent subproblems. Each subproblem is a constrained quadratic optimization problem in terms of a node variable $\boldsymbol{\mu}_i$, $i \in V$:

$$\hat{\boldsymbol{\mu}}_i^{k+1} = \arg\min_{\boldsymbol{\mu}_i\in\mathcal{M}_i} \left(\frac{\rho|N(i)|}{2} \|\boldsymbol{\mu}_i\|_2^2 - \boldsymbol{h}_i^{(k)\top}\boldsymbol{\mu}_i \right), \qquad (12)$$

where

$$\begin{aligned} \boldsymbol{h}_i^{(k)} &= \boldsymbol{\theta}_i + \sum_{c:i\in c} (\boldsymbol{A}_{ci}\rho\boldsymbol{\nu}_c^{k+1} + \boldsymbol{\gamma}_{ci}^k) \\ \mathcal{M}_i &= \left\{ \boldsymbol{\mu}_i \geq 0 \middle| \boldsymbol{1}^\top\boldsymbol{\mu}_i = 1 \right\}. \end{aligned}$$

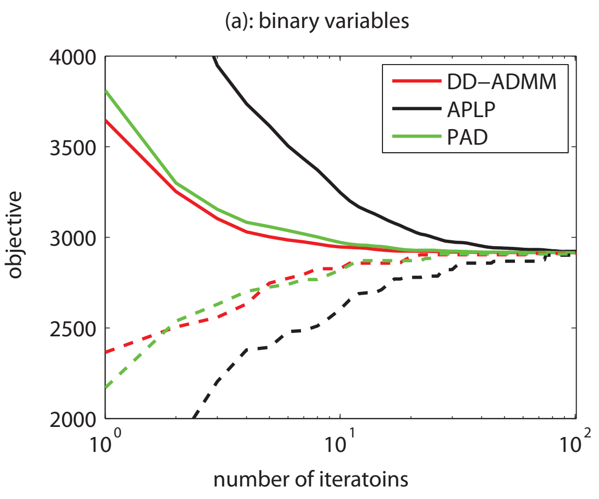
We note that the quadratic matrix $\boldsymbol{J}_i$ in (12) is actually an identity matrix multiplied by a scalar. Thus, we can also apply the subroutine "projectSimplex" to compute the optimal solution $\hat{\boldsymbol{\mu}}_i^{k+1}$ efficiently, as demonstrated in **Algorithm 1**.

Finally, the computation for $\hat{\boldsymbol{\gamma}}^{k+1}$ is straightforward. Equ. (9) fully specifies the updating expression for $\hat{\boldsymbol{\gamma}}^{k+1}$. We note that it is in fact a gradient-descent operation. On the other hand, the computation for $(\hat{\boldsymbol{\mu}}^{k+1}, \hat{\boldsymbol{\nu}}^{k+1})$ are coordinate-descent operations.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the PAD algorithm by experiment. Besides the PAD algorithm, we also implement the DD-ADMM and the APLP algorithms for performance comparison. Our primary interest in the experiment is to find out if the PAD algorithm converges faster than the APLP algorithm.

We conducted two experiments on a grid of size 20×20 with pairwise nearest neighbor interactions. The first experiment was for



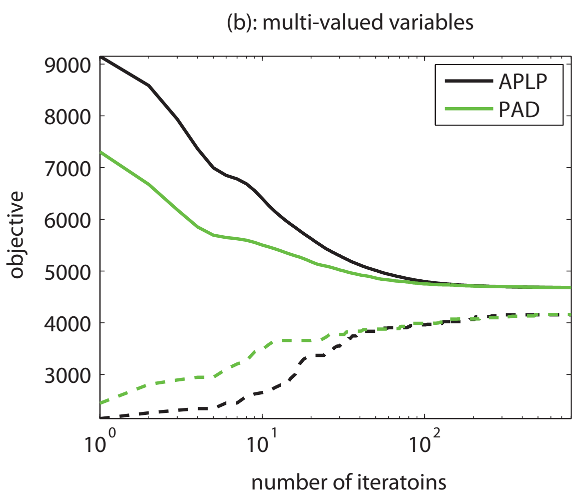


Fig. 2. The graphic model in the two experiments is a grid of 20×20 with pairwise nearest neighbor interactions. Each node may have two, three or four neighbors. The solid curve for each algorithm denotes the dual objective of (5). On the other hand, the dot-dashed curve denote the value of the best decoded primal solution.

```
Algorithm 1: The PAD Algorithm
Input: Parameters $\{\boldsymbol{\theta}_i, i \in V\}$, $\{\boldsymbol{\theta}_c, c \in C\}$, number of
        iterations $T$, parameter $\rho$ and $\{\boldsymbol{D}_c, c \in C\}$
Initialization: $\boldsymbol{\gamma}^0 = 0$, $(\boldsymbol{\mu}^0, \boldsymbol{\nu}^0)$ are set to be uniformly
        distributed.
for $k = 0$ to $T - 1$ do
    Update $\boldsymbol{\nu}$: for all $c \in C$
        Set $\boldsymbol{h}_c^{(k)} = \boldsymbol{\theta}_c + \sum_{i\in c} \boldsymbol{A}_{ci}^\top(\rho\hat{\boldsymbol{\mu}}_i^k - \hat{\boldsymbol{\gamma}}_{ci}^k) + \boldsymbol{D}_c\hat{\boldsymbol{\nu}}_c^k$
        $\hat{\boldsymbol{\nu}}_c^{k+1}$ = projectSimplex$(\boldsymbol{h}_c^{(k)}, \sigma_c, \tau_c)$
    Update $\boldsymbol{\mu}$: for all $i = 1, \ldots, n$
        Set $\boldsymbol{h}_i^{(k)} = \boldsymbol{\theta}_i + \sum_{c:i\in c}(\boldsymbol{A}_{ci}\rho\hat{\boldsymbol{\nu}}_c^{k+1} + \hat{\boldsymbol{\gamma}}_{ci}^k)$
        $\hat{\boldsymbol{\mu}}_i^{k+1}$ = projectSimplex$\left(\boldsymbol{h}_i^{(k)}, \rho|N(i)|, 0\right)$
    Update $\boldsymbol{\gamma}$
        $\hat{\boldsymbol{\gamma}}_{ci}^{k+1} = \hat{\boldsymbol{\gamma}}_{ci}^k + \rho\left(\boldsymbol{A}_{ci}\hat{\boldsymbol{\nu}}_c^{k+1} - \hat{\boldsymbol{\mu}}_i^{k+1}\right) \quad \forall c \in C, i : i \in c$
end for
```

the binary-valued case (see Fig. 2:(a)). The unary and pairwise potentials were sampled from a Gaussian distribution with variance 16. In the implementation of the three algorithms, the parameter ρ was set as $\rho = 5$. For the PAD algorithm, the parameters (τ_c, σ_c) in $\boldsymbol{J}_c$ were set as $(\tau_c, \sigma_c) = (1, 3)$. The matrices $\{\boldsymbol{D}_c, c \in C\}$ were then determined accordingly.

The second experiment was for the multi-valued case (see Fig. 2:(b)). Each random variable ranged over eight states. Similarly, the unary and pairwise potentials were sampled from a Gaussian distribution with variance 16. In this case, we only tested the PAD and the APLP algorithms. The parameter ρ for the two algorithms was set as $\rho = 5$. The parameters (τ_c, σ_c) for the PAD algorithm were set as $(\tau_c, \sigma_c) = (0, 16)$.

The experimental results are displayed in Fig. 2. It is seen that the PAD algorithm converges significantly faster than the APLP algorithm in both experiments. As explained in Subsection 3.1, this may be due to the fact that the PAD algorithm involves fewer number of auxiliary variables than that of the APLP algorithm, resulting in more efficient information flow on the graph. It is also worth noting that the performance of the DD-ADMM and the PAD algorithms is comparable. This is because the amount of feedback signal is quite small, only bringing little effect to the performance of the PAD algorithm.

5. CONCLUSION

In this paper, we have proposed the PAD algorithm for the primal MAP LP-relaxation problem by using the PADMM. In the design of the PAD algorithm, we have introduced feedback signal to reduce the number of auxiliary variables as required by the APLP algorithm. As a result, the PAD algorithm only performs two layers of optimization at each iteration as compared to the APLP algorithm which has to perform three layers of optimization. Therefore, the PAD algorithm has lower computational complexity and converges faster than the APLP algorithm. Experimental results confirms that the PAD algorithm indeed outperforms that the APLP algorithm.

6. REFERENCES

[1] T. Jaakkola M. Wainwright and A. Willsky, “MAP estimation via agreement on (hyper)trees: message passing and linear programming approaches,” *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

[2] T. Werner, “A Linear Programming Approach to Max-Sum Problem: A review,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 7, pp. 1165–1179, 2007.

[3] M. Collins A. M. Rush, D. Sontag and T. Jaakkola, “On dual decomposition and linear programming relaxations for natural language processing,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010.

[4] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, “A linear programming formulation and approximation algorithms for the

metric labeling problem," *SIAM Journal on Discrete Mathematics*, vol. 18, pp. 608–625, 2005.

[5] A. Globerson and T. Jaakkola, "Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations," *Advances in Neural Information Processing Systems 21*.

[6] D. Sontag, A. Globerson, and T. Jaakkola, "Introduction to Dual Decomposition for Inference," in *Optimization for Machine Learning*. 2011, MIT Press.

[7] S. Gould V. Jojic and D. Koller, "Fast and smooth: Accelerated dual decomposition for MAP inference," in *Proceedings of International Conference on Machine Learning,*, 2010.

[8] N. Paragios N. Komodakis and G. Tziritas, "Mrf energy minimization and beyond via dual decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, pp. 531–552, 2011.

[9] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Computers and Mathematics with Applications*, vol. 2, pp. 17–40, 1976.

[10] R. Glowinski and A. Marrocco, "Sur lapproximation, par elements finis dordre un, et la resolution, par penalisation-dualite, dune classe de problems de dirichlet non lineares," *Revue Franccaise d'Automatique, Informatique, et Recherche Operationelle*, vol. 9, pp. 41–76, 1975.

[11] R. Glowinski and P. Le Tallec, "Augmented Lagrangian and operator-splitting methods in nonlinear mechanics," *Society for Industrial Mathematics*, 1989.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *In Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[13] A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing, "An Augmented Lagrangian Approach to Constrained MAP Inference," in *Proc. Int. Conf. Machine Learning*, 2011.

[14] O. Meshi and A. Globerson, "An Alternating Direction Method for Dual MAP LP Relaxation," in *ECML*, 2011.

[15] J. Eckstein, "Some saddle-function splitting methods for convex programming," *Optimization Methods and Software*, vol. 4, no. 1, pp. 75–83, 1994.

[16] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projecxtions onto the L1-ball for learning in high dimensions," in *ICML*, 2008.