SIMPLIFIED ALTERNATING-DIRECTION MESSAGE PASSING FOR DUAL 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

The approximate MAP inference over (factor) graphic models is of great importance in many applications. Due to its simplicity, linear-programming (LP) relaxation has become one of the most popular approaches to approximate MAP. In this paper, we propose a new message passing algorithm for the MAP LP-relaxation problem by using the alternating-direction method of multipliers (ADMM). At each iteration, the new algorithm performs two layers of optimization sequentially, that is node-oriented optimization and factor-oriented optimization. On the other hand, the recently proposed augmented dual LP (ADLP) algorithm, also based on the ADMM, has to perform three layers of optimization. We refer to our new algorithm as the simplified ADLP (SiADLP) algorithm. The design of the SiADLP algorithm stems from a new formulation for the dual LP problem. Experimental results show that the SiADLP algorithm outperforms the ADLP method.

Index Terms— graphic models, ADMM, 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 of the fundamental inference tasks is to find the *maximum a-posteriori* (MAP) assignment. It is known that for loopy-graphic models, the computation of the MAP solution is 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]). In the past few years, various message-passing algorithms have been proposed for solving MAP LP-relaxation. Some algorithms perform block coordinate-descent operations, such asmax-sum diffusion (MSD) [2] and max product linear-programming (MPLP) [5] algorithms. While these algorithms usually converge fast, they do not always reach the optimal solution of the LP-relaxation. To overcome the suboptimality issue, some algorithms perform variants of subgradient descent to the dual problem [6, 7]. Although these algorithms are guaranteed to converge globally, they typically converge slower than the block coordinate-descent ones.

Recently, the *alternating direction method of multipliers* (ADMM) [8, 9, 10] has been employed in designing more efficient and globally convergent message-passing algorithms for LP-relaxation [11, 12]. The work in [11] considered applying ADMM for graphs with binary-valued variables. The proposed DD-ADMM algorithm in [11] performs two layers of optimization at each iteration, which are node-oriented optimization and factor-oriented optimization (see Fig. 1). In [12], Meshi and Globerson extended the work of [11] by considering graph models with multi-valued variables. The authors proposed two algorithms: the augmented primal LP (APLP) method and the augmented dual LP (ADLP) method. However, both algorithms 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 optimization at each iteration, increasing the computational complexities and further slowing down the convergence speed.

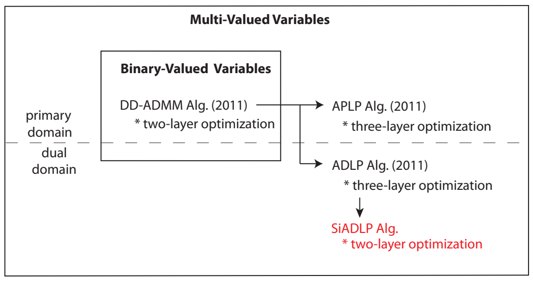

Fig. 1. ADMM-based algorithms for solving MAP LP-relaxation. The SiADLP algorithm is the new method we will present in this paper.

In this paper, we reconsider solving the dual MAP LP-relaxation problem by using ADMM. Our primary motivation is to design a new message passing algorithm that only performs two layers of optimization at each iteration. To achieve this goal, we first reformulate the dual MAP LP-relaxation problem in a more compact way than that of [12]. Specifically, we introduce fewer auxiliary variables than that of [12] and further add a set of constraints to bring the remaining variables more close (see Section 3 for detailed information).

The new algorithm follows directly by applying ADMM to the reformulated optimization problem. We refer to the new algorithm as the simplified ADLP (SiADLP) method (see Fig. 1). Experimental results show that the SiADLP algorithm converges faster than the ADLP 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 $X = \{X_i | i = 1, \ldots, n\}$ be a set 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 subset X_c of variables, where $c \subseteq \{1, \ldots, n\}$, we denote its realization as $x_c \in \mathcal{X}_c$. When $c = \{1, \ldots, n\}$, we let $x_c = x$ and $\mathcal{X}_c = \mathcal{X}$ for simplicity. Suppose that the joint probability $P(x)$ of the variables admits a decomposition with respect to a factor graph $G = (V, C)$:

$$P(x) \propto \exp\left(\sum_i \theta_i(x_i) + \sum_{c \in C} \theta_c(x_c)\right), \quad (1)$$

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 $\theta_c(x_c)$ 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 $P(x)$ in (1)

$$x^{MAP} = \arg\max_{x \in \mathcal{X}} \left(\sum_i \theta_i(x_i) + \sum_{c \in C} \theta_c(x_c)\right). \quad (2)$$

As mentioned in the introduction, the above problem is NP-hard for general graphic models. As a result, the research attention has been moved to the development of approximation approaches such as the MAP-LP relaxation.

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 a set of variables and then approximate the MAP problem by a tractable LP in terms of the variables. The LP is designed such that the combinatorial constraints of the MAP problem (2) are relaxed to a set of linear constraints in terms of the variables. Formally, the MAP-LP relaxation to (2) takes the form [11]

$$\begin{aligned}(\mu^*, \nu^*) = \arg \max_{(\mu,\nu) \in L(G)} \Big(&\sum_i \sum_{x_i} \mu_i(x_i)\theta_i(x_i) \\ &+ \sum_c \sum_{x_c} \nu_c(x_c)\theta_c(x_c)\Big), \quad (3)\end{aligned}$$

where the local polytope $L(G)$ is defined as

$$L(G) = \left\{ (\mu, \nu) \left| \begin{array}{ll} \sum_{x_i} \mu_i(x_i) = 1, & \forall i \in V \\ \mu_i(x_i) = \sum_{x_{c\setminus i}} \nu_c(x_{c\setminus i}, x_i) & \forall i : i \in c, x_i \\ \nu_c(x_c) \geq 0 & c \in C, x_c \end{array} \right. \right\}.$$

It is known that if the optimal solution μ^* in (3) takes integer values, we obtain the exact MAP solution. On the other hand, if μ^* takes non-integer values, a good approximation of the MAP solution can be obtained by making hard-decision to μ^*. That is for each variable X_i, its MAP assignment k_i is approximated as $k_i = \arg\max(\mu_i^*)$.

For the primal LP formulation (3), its dual LP takes the form of [12]

$$\begin{aligned}\min_\delta \Bigg(&\sum_i \max_{x_i} \Big(\theta_i(x_i) + \sum_{c: i \in c} \delta_{ci}(x_i)\Big) \\ &+ \sum_c \max_{x_c} \Big(\theta_c(x_c) - \sum_{i: i \in c} \delta_{ci}(x_i)\Big)\Bigg), \quad (4)\end{aligned}$$

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 (4) 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, 12]. In this paper, we will also focus on the dual problem (4).

3. SIMPLIFIED ALTERNATING-DIRECTION MESSAGE PASSING

In this section, we derive the SiADLP algorithm by applying ADMM to the dual MAP-LP problem (4). The main step is to reformulate the dual problem properly to allow the application of ADMM.

3.1. Reformulation of dual LP-relaxation

In this subsection, we reformulate the dual MAP LP-relaxation problem (4). The new problem formulation will serve a starting point to derive the SiADLP algorithm.

We build a new form of the objective function of (4) in a similar manner as that of (3). That is we introduce an auxiliary variable $\lambda_c = \{\lambda_c(x_c), x_c \in \mathcal{X}_c\}$ for each factor $c \in C$, which corresponds to ν_c in (3). We denote the set of auxiliary variables as $\lambda = \{\lambda_c, c \in C\}$. With λ, the dual problem (4) can be rewritten as

$$\begin{aligned}\min_{\delta,\lambda} \Bigg(&\sum_i \max_{x_i} \Big(\theta_i(x_i) + \sum_{c: i \in c} \delta_{ci}(x_i)\Big) \\ &+ \sum_c \max_{x_c} \left(\theta_c(x_c) - \lambda_c(x_c)\right)\Bigg), \quad (5)\end{aligned}$$

where

$$\lambda_c(x_c) = \sum_{i: i \in c} \delta_{ci}(x_i) \quad \forall c, i : i \in c, x_c. \quad (6)$$

Our main purpose to introduce λ is to separate the node-oriented optimization and factor-oriented optimization in (4).

We note that the optimal solution of (5) is not unique. Suppose $(\delta^* = \{\delta^*_{ci} | c \in C, i : i \in c\}, \lambda^* = \{\lambda^*_c | c \in C\})$ is a particular optimal solution. One can add a constant $a_{ci} \in \mathbb{R}$ to each solution δ^*_{ci} to obtain $\bar{\delta}^*_{ci} = \delta^*_{ci} + a_{ci} e_i$ and let $\bar{\lambda}^*_c = \lambda^*_c + \sum_{i: i \in c} a_{ci} e_c$, where e_i and e_c are all-one vectors. The resulting solution $(\bar{\delta}^*, \bar{\lambda}^*)$ is also optimal for (5). In other words, each variable δ_{ci} in (5) has one degree of freedom. In order to make the problem (5) more strict, we introduce a set of constraints to (δ, λ) as

$$\lambda \in \left\{\sum_{x_c} \lambda_c(x_c) = 0, \forall c \in C\right\} \quad (7)$$

$$\delta \in \left\{\sum_{x_i} \delta_{ci}(x_i) = 0, \forall c \in C, i \in V\right\}. \qquad (8)$$

Equations (6)-(8) together specify the constraints for the dual LP problem.

To briefly summarize, in the process of reformulating the dual problem (4), we introduce a set λ of auxiliary variables and also add a set of constraints to (δ, λ). The new problem specified by (5)-(8) is equivalent to (4). In next subsection, we consider the new problem (5)-(8) instead of solving (4) directly.

3.2. Message updating-expressions

In this subsection, we derive the message updating-expressions for the SiADLP algorithm. The basic idea is to construct an augmented Lagrangian function for (5), allowing the usage of the ADMM. We will show that at each iteration, the SiADLP algorithm only involves two layers of optimization (see Fig. 1).

By following the ADMM framework [10], we construct the augmented Lagrangian function for (5) as

$$\begin{aligned} & L_\rho(\delta, \lambda, \gamma) \\ = & \sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\right) \\ & + \sum_c \max_{x_c} \left(\theta_c(x_c) - \lambda_c(x_c)\right) \\ & + \sum_c \sum_{x_c} \gamma_c(x_c) \left(\lambda_c(x_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\right) \\ & + \frac{\rho}{2} \sum_c \sum_{x_c} \left(\lambda_c(x_c) - \sum_{i:i\in c} \delta_{ci}(x_i)\right)^2, \end{aligned} \qquad (9)$$

where the variables (δ, λ) satisfy the constraints (6)-(8). The quadratic penalty function in (9) accounts for the equality constraint (6). By following the updating procedure of the ADMM [13], the new estimate $(\hat{\delta}^{(k+1)}, \hat{\lambda}^{(k+1)})$ can be computed as

$$\hat{\delta}^{(k+1)} = \arg\min_{\delta} L_\rho(\delta, \hat{\lambda}^{(k)}, \hat{\gamma}^{(k)}) \qquad (10)$$

$$\hat{\lambda}^{(k+1)} = \arg\min_{\lambda} L_\rho(\hat{\delta}^{(k+1)}, \lambda, \hat{\gamma}^{(k)}). \qquad (11)$$

We note the computations for $\hat{\delta}^{(k+1)}$ and $\hat{\lambda}^{(k+1)}$ are realized by performing coordinate-descent operation.

We are now in a position to derive the explicit updating-expressions for (δ, λ). Suppose that the estimate $(\hat{\delta}^{(k)}, \hat{\lambda}^{(k)}, \hat{\gamma}^{(k)})$ is obtained after the first k iterations, where $k \geq 0$. We first consider the minimization of $L_\rho(\delta, \hat{\lambda}^{(k)}, \hat{\gamma}^{(k)})$ over δ. Without loss of generality, we focus on the vectors $\{\delta_{ci}|c : i \in c\}$ that are related with node $i \in V$. The function for $\{\delta_{ci}|c : i \in c\}$ can be extracted from (9) as

$$\begin{aligned} & \sum_{c:i\in c} \sum_{x_i} \left(\frac{\rho|\mathcal{X}_{c\backslash i}|}{2} \delta_{ci}(x_i)^2 - q_{ci}^{(k)}(x_i)\delta_{ci}(x_i)\right) \\ & + \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\right) + \sum_{c:i\in c} \rho \left(\sum_{x_i} \delta_{ci}(x_i)\right) \\ & \cdot \left[\sum_{j\in c, j\neq i} |\mathcal{X}_{c\backslash\{i,j\}}| \left(\sum_{x_j} \delta_{cj}(x_j)\right)\right], \end{aligned} \qquad (12)$$

where for each $c, i \in c$ and $x_i \in \mathcal{X}_i$

$$q_{ci}^{(k)}(x_i) = \sum_{x_{c\backslash i}} \left(\hat{\gamma}_c^{(k)}(x_{c\backslash i}, x_i) + \rho \hat{\lambda}_c^{(k)}(x_{c\backslash i}, x_i)\right). \qquad (13)$$

The last term in (12) captures the interactions between $\{\delta_{ci}|c : i \in c\}$ and those of the nodes involved in the factors that include i.

In order to minimize the function in (12) over $\{\delta_{ci}|c : i \in c\}$, we first have to take care of the last term of the function. Fortunately, we can totally remove the last term in (12) by using the constraint (8). That is (12) can be reformulated as

$$\begin{aligned} & \sum_{c:i\in c} \left(\frac{\rho|\mathcal{X}_{c\backslash i}|}{2} \|\delta_{ci}\|^2 - q_{ci}^{(k),\top} \delta_{ci}\right) \\ & + \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i)\right), \end{aligned} \qquad (14)$$

where $\{\delta_{ci}|c : i \in c\}$ satisfy the constraints

$$\sum_{x_i} \delta_{ci}(x_i) = 0 \quad \forall c : i \in c. \qquad (15)$$

The function in (14) can be easily minimized with respect to $\{\delta_{ci}|c : i \in c\}$ by inspecting the KKT conditions. The updating procedure is summarized in **Algorithm 1**. In particular, in **Algorithm 1**, the subroutine $w = \text{TRIM}(v, d)$ serves to clip the values in the vector v at some threshold t (i.e., $w_i = \min(v_i, t)$) such that the sum of removed parts equals $d > 0$ (i.e., $\sum_i v_i - w_i = d$). We point out that the subroutine TRIM was originally developed in [12] for the design of the ADLP algorithm.

To briefly summarize from the above analysis, the update for δ eventually boils down to solving a set of independent subproblems. Each subproblem is a constrained quadratic optimization problem of the form (14)-(15). We refer to the computation of δ as *node-oriented optimization*.

Next we consider the minimization of $L_\rho(\hat{\delta}^{(k+1)}, \lambda, \hat{\gamma}^{(k)})$ over λ. Similarly, the minimization problem can be broken into a set of independent subproblems. Each subproblem is a constrained quadratic optimization problem in terms of a factor variable λ_c, $c \in C$:

$$\min_{\lambda_c} \frac{1}{2}\|\lambda_c\|^2 - p_c^{(k),\top}\lambda_c + \frac{1}{\rho}\max_{x_c}(\theta_c(x_c) - \lambda_c(x_c)), \qquad (16)$$

where

$$p_c^{(k),\top}\lambda_c = \sum_{x_c} \left(\sum_{i:i\in c} \hat{\delta}_{ci}^{(k+1)}(x_i) - \frac{1}{\rho}\hat{\gamma}_c^{(k)}(x_c)\right)\lambda_c(x_c) \qquad (17)$$

$$0 = \sum_{x_c} \lambda_c(x_c). \qquad (18)$$

The updating procedure for λ_c in (16) can be derived in a similar manner as for δ, which is also presented in **Algorithm 1**. We refer to the computation of λ as *factor-oriented optimization*.

Finally, we consider computing $\hat{\gamma}^{(k+1)}$. According to the updating procedure of the ADMM [10], the computation of $\hat{\gamma}^{(k+1)}$ is realized by performing gradient-descent operation. See **Algorithm 1** for the final updating expression of γ.

We note that at iteration k, the updates for $\left(\hat{\delta}^{(k+1)}, \hat{\lambda}^{(k+1)}\right)$ correspond to two layers of optimization. The update for $\hat{\delta}^{(k+1)}$ is computed by performing node-oriented optimization. On the other hand,

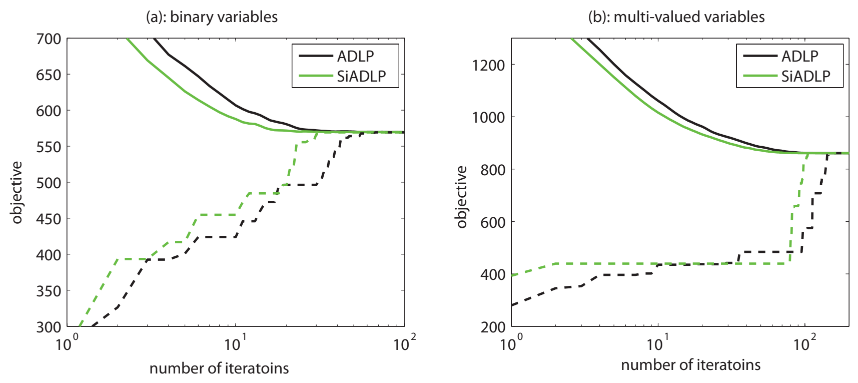

Fig. 2. The graphic model in the two experiments is a grid of 10×10 with pairwise nearest neighbor interactions. Each node may have two, three or four neighbors. The solid curve for each algorithm denotes the dual objective of (4). On the other hand, the dot-dashed curve denotes the objective value of the best decoded primal solution.

```
Algorithm 1 The SiADLP Algorithm
Input: Parameters $\{\theta_i, i \in V\}$, $\{\theta_c, c \in C\}$,
        number of iterations $T$.
Initialization: $\gamma = 0$, $(\lambda, \delta) = (0, 0)$
for $t = 0$ to $T - 1$ do
  Update $\delta$: for all $i = 1, \ldots, n$
    Set $q_i^{(t)} = \frac{\theta_i}{|N(i)|} \sum_{c:i \in c} \rho |\mathcal{X}_{c \setminus i}| + \sum_{c:i \in c} q_{ci}^{(t)}$
    $\bar{q}_i^{(t)} = \text{TRIM}(q_i^{(t)}, |N(i)|)$
    $h_i^{(t)} = (q_i^{(t)} - \bar{q}_i^{(t)})/|N(i)|$
    Update $\delta_{ci}^{(t+1)} = \frac{(q_{ci}^{(t)} - h_i^{(t)}) - \text{mean}(q_{ci}^{(t)} - h_i^{(t)}) e_i}{\rho |\mathcal{X}_{c \setminus i}|}$
  Update $\lambda$: for all $c \in C$
    Set $h_c^{(t)} = \theta_c - p_c^{(t)}$
    $\bar{h}_c^{(t)} = \text{TRIM}(h_c^{(t)}, \frac{1}{\rho})$
    Update $\lambda_c^{(t+1)} = \theta_c - \bar{h}_c^{(t)} - \text{mean}(\theta_c - \bar{h}_c^{(t)}) e_c$
  Update the multipliers
    Update $\gamma_c^{(t+1)}(x_c) = \gamma_c^{(t)}(x_c) + \rho\Big(\lambda_c^{(t+1)}(x_c)$
                                   $- \sum_{i:i \in c} \delta_{ci}^{(t+1)}(x_i)\Big)$
end for
```

the update for $\hat{\lambda}^{(k+1)}$ is computed by performing factor-oriented optimization. The above operation is similar to that of the DD-ADMM algorithm designed for binary-valued graph [11].

Remark 1. *It is worth noting that in [12], the authors introduced another set of auxiliary variables (copies of δ) in designing the ADLP algorithm. The auxiliary variables are used to bridge the connection between δ and λ in (5). As a result, each iteration of the ADLP algorithm involves three layers of optimization, where the third layer of optimization is for the auxiliary variables.*

4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the SiADLP algorithm by experiment. Besides the SiADLP algorithm, we also implement the ADLP algorithms for performance comparison. Our primary interest in the experiment is to find out if the SiADLP algorithm converges faster than the ADLP algorithm.

We conducted two experiments on a grid of size 10×10 with pairwise nearest neighbor interactions. The first experiment was for the binary-valued case (see Fig. 2:(a)). The second experiment was for the multi-valued case (see Fig. 2:(b)). The unary and pairwise potentials were sampled from a Gaussian distribution with variance 16. In the implementation of the two algorithms, the parameter ρ was set as $\rho = 0.5$. In the second experiment, each variable takes four states, i.e., $|\mathcal{X}_i| = 4$, $i \in V$.

The experimental results are displayed in Fig. 2. It is seen that for each algorithm, the objective value of (4) (as denoted by a solid curve) decreases along with the number of iterations, confirming the convergence of ADMM. The SiADLP algorithm converges faster than the ADLP algorithm in both experiments. This may be due to the fact that the SiADLP algorithm performs two layers of optimization per-iteration. On the other hand, the ADLP algorithm performs three layers of optimization per-iteration.

5. CONCLUSION

In this paper, we have proposed the SiADLP algorithm for the dual MAP LP-relaxation problem by using ADMM. The SiADLP algorithm only performs two layers of optimization at each iteration as compared to the ADLP algorithm which has to perform three layers of optimization. As a result, the SiADLP algorithm has lower computational complexity and converges faster than the ADLP algorithm. Experimental results demonstrate that the SiADLP algorithm indeed converges faster than the ADLP 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 Confer-*

ence 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] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.