An Augmented Lagrangian Approach to Constrained MAP Inference

André F. T. Martins[†‡] AFM@CS.CMU.EDU
Mário A. T. Figueiredo[‡] MARIO.FIGUEIREDO@LX.IT.PT
Pedro M. Q. Aguiar[♯] AGUIAR@ISR.IST.UTL.PT
Noah A. Smith[†] NASMITH@CS.CMU.EDU
Eric P. Xing[†] EPXING@CS.CMU.EDU
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[‡]Instituto de Telecomunicações / [♯]Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal

Abstract

We propose a new algorithm for approximate MAP inference on factor graphs, by combining augmented Lagrangian optimization with the dual decomposition method. Each slave subproblem is given a quadratic penalty, which pushes toward faster consensus than in previous subgradient approaches. Our algorithm is provably convergent, parallelizable, and suitable for fine decompositions of the graph. We show how it can efficiently handle problems with (possibly global) structural constraints via simple sort operations. Experiments on synthetic and real-world data show that our approach compares favorably with the state-of-the-art.

1. Introduction

Graphical models enable compact representations of probability distributions, being widely used in computer vision, natural language processing (NLP), and computational biology (Koller & Friedman, 2009). A prevalent problem is the one of inferring the most probable configuration, the so-called *maximum a posteriori* (MAP). Unfortunately, this problem is intractable, except for a limited class of models. This fact precludes computing the MAP exactly in many important models involving *non-local* features or requiring *structural constraints* to ensure valid predictions.

A significant body of research has thus been placed on *approximate* MAP inference, *e.g.*, via linear programming relaxations (Schlesinger, 1976). Several message-passing algorithms have been proposed that exploit the graph structure in these relaxations (Wainwright et al., 2005; Kolmogorov, 2006; Werner, 2007; Globerson & Jaakkola, 2008; Ravikumar et al., 2010). In the same line, Komodakis et al. (2007) proposed a method based on the classical *dual decomposition* technique (DD; Dantzig & Wolfe 1960; Everett III 1963; Shor 1985), which breaks the original problem into a set of smaller (slave) subproblems, splits the shared variables, and tackles the Lagrange dual with the *subgradient* algorithm. Initially applied in computer vision, DD has also been shown effective in NLP (Koo et al., 2010). The drawback is that the subgradient algorithm is very slow to converge when the number of slaves is large. This led Jojic et al. (2010) to propose an accelerated gradient method by smoothing the objective.

In this paper, we ally the simplicity of DD with the effectiveness of *augmented Lagrangian methods*, which have a long-standing history in optimization (Hestenes, 1969; Powell, 1969; Glowinski & Marroco, 1975; Gabay & Mercier, 1976; Boyd et al., 2011). The result is a novel algorithm for approximate MAP inference: *DD-ADMM* (*Dual Decomposition with the Alternating Direction Method of Multipliers*). Rather than placing all efforts in attempting progress in the dual, DD-ADMM looks for a saddle point of the Lagrangian function, which is augmented with a quadratic term to penalize slave disagreements. Key features of DD-ADMM are:

- It is suitable for heavy parallelization (many slaves);
- it is provably convergent, even when each slave subproblem is only solved approximately;
- consensus among slaves is fast, by virtue of the quadratic penalty term, hence it exhibits faster convergence in the *primal* than competing methods;
- in addition to providing an optimality certificate for the exact MAP, it also provides guarantees that the *LP-relaxed* solution has been found.

Appearing in *Proceedings of the* 28^{th} *International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

After providing the necessary background (Sect. 2) and introducing and analyzing DD-ADMM (Sect. 3), we turn to the slave subproblems (Sect. 4). Of particular concern to us are problems with *structural constraints*, which arise commonly in NLP, vision, and other structured prediction tasks. We show that, for several important constraints, each slave can be solved *exactly* and *efficiently* via sort operations. Experiments with pairwise MRFs and dependency parsing (Sect. 5) testify to the success of our approach.

2. Background

2.1. Problem Formulation

Let $\boldsymbol{X} \triangleq (X_1, \ldots, X_N) \in \mathcal{X}$ be a vector of discrete random variables, where each $X_i \in \mathcal{X}_i$, with $\mathcal{X}_i$ a finite set. We assume that $\boldsymbol{X}$ has a Gibbs distribution associated with a factor graph $\mathcal{G}$ (Kschischang et al., 2001), composed of a set of variable nodes $\{1, \ldots, N\}$ and a set of factor nodes $\mathcal{A}$, with each $a \in \mathcal{A}$ linked to a subset of variables $\mathcal{N}(a) \subseteq \{1, \ldots, N\}$:

$$P_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) \propto \exp\left(\sum_{i=1}^N \theta_i(x_i) + \sum_{a\in\mathcal{A}} \phi_a(\boldsymbol{x}_a)\right). \quad (1)$$

Above, $\boldsymbol{x}_a$ stands for the subvector indexed by the elements of $\mathcal{N}(a)$, and $\theta_i(.)$ and $\phi_a(.)$ are, respectively, unary and higher-order log-potential functions. To accommodate hard constraints, we allow these functions to take values in $\mathbb{R} \cup \{-\infty\}$. For simplicity, we write $\boldsymbol{\theta}_i \triangleq (\theta_i(x_i))_{x_i \in \mathcal{X}_i}$ and $\boldsymbol{\phi}_a \triangleq (\phi_a(\boldsymbol{x}_a))_{\boldsymbol{x}_a \in \mathcal{X}_a}$.

We are interested in the task of finding the most probable assignment (the MAP), $\hat{\boldsymbol{x}} \triangleq \arg\max_{\boldsymbol{x}\in\mathcal{X}} P_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$. This (in general NP-hard) combinatorial problem can be transformed into a linear program (LP) by introducing marginal variables $\boldsymbol{\mu} \triangleq (\boldsymbol{\mu}_i)_{i=1}^n$ and $\boldsymbol{\nu} \triangleq (\boldsymbol{\nu}_a)_{a\in\mathcal{A}}$, constrained to the *marginal polytope* of $\mathcal{G}$, *i.e.*, the set of realizable marginals (Wainwright & Jordan, 2008). Denoting this set by $\mathcal{M}(\mathcal{G})$, this yields

$$\text{OPT} \quad \triangleq \quad \max_{(\boldsymbol{\mu},\boldsymbol{\nu})\in\mathcal{M}(\mathcal{G})} \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_a \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \quad (2)$$

which always admits an integer solution. Unfortunately, $\mathcal{M}(\mathcal{G})$ often lacks a concise representation, which renders (2) intractable. A common workaround is to replace $\mathcal{M}(\mathcal{G})$ by the outer bound $\mathcal{L}(\mathcal{G}) \supseteq \mathcal{M}(\mathcal{G})$—the so-called *local polytope*, defined as

$$\mathcal{L}(\mathcal{G}) = \left\{ (\boldsymbol{\mu},\boldsymbol{\nu}) \;\middle|\; \begin{array}{l} \mathbf{1}^\top \boldsymbol{\mu}_i = 1, \forall i \wedge \\ \mathbf{H}_{ia}\boldsymbol{\nu}_a = \boldsymbol{\mu}_i, \forall a, i \in \mathcal{N}(a) \wedge \\ \boldsymbol{\nu}_a \geq 0, \forall a \end{array} \right\}, \quad (3)$$

where $\mathbf{H}_{ia}(x_i, \boldsymbol{x}_a) = 1$ if $[\boldsymbol{x}_a]_i = x_i$, and 0 otherwise. This yields the following LP relaxation of (2):

$$\text{OPT}' \quad \triangleq \quad \max_{(\boldsymbol{\mu},\boldsymbol{\nu})\in\mathcal{L}(\mathcal{G})} \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_a \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \quad (4)$$

which will be our main focus throughout. Obviously, $\text{OPT}' \geq \text{OPT}$, since $\mathcal{L}(\mathcal{G}) \supseteq \mathcal{M}(\mathcal{G})$.

2.2. Dual Decomposition

Several message passing algorithms (Wainwright et al., 2005; Kolmogorov, 2006; Globerson & Jaakkola, 2008) are derived via some reformulation of (4) followed by dualization. The DD method (Komodakis et al., 2007) reformulates (4) by adding new variables $\boldsymbol{\nu}_i^a$ (for each factor a and $i \in \mathcal{N}(a)$) that are local "replicas" of the marginals $\boldsymbol{\mu}_i$. Letting $\mathcal{N}(i) \triangleq \{a | i \in \mathcal{N}(a)\}$ and $d_i = |\mathcal{N}(i)|$ (the degree of node i), (4) is rewritten as

$$\begin{array}{ll} \max\limits_{\boldsymbol{\nu},\boldsymbol{\mu}} & \sum_a \left(\sum_{i\in\mathcal{N}(a)} d_i^{-1} \boldsymbol{\theta}_i^\top \boldsymbol{\nu}_i^a + \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a \right) \\ \text{s.t.} & (\boldsymbol{\nu}_{\mathcal{N}(a)}^a, \boldsymbol{\nu}_a) \in \mathcal{M}(\mathcal{G}_a), \quad \forall a \\ & \boldsymbol{\nu}_i^a = \boldsymbol{\mu}_i, \quad \forall a, i \in \mathcal{N}(a), \end{array} \quad (5)$$

where $\mathcal{G}_a$ is the subgraph of $\mathcal{G}$ comprised only of factor a and the variables in $\mathcal{N}(a)$, $\mathcal{M}(\mathcal{G}_a)$ is the corresponding marginal polytope, and we denote $\boldsymbol{\nu}_{\mathcal{N}(a)}^a \triangleq (\boldsymbol{\nu}_i^a)_{i\in\mathcal{N}(a)}$. (Note that by definition, $\mathcal{L}(\mathcal{G}) = \{(\boldsymbol{\mu},\boldsymbol{\nu}) \mid (\boldsymbol{\mu}_{N_a}, \boldsymbol{\nu}_a) \in \mathcal{M}(\mathcal{G}_a),\ \forall a \in \mathcal{A}\}$.) Problem (5) would be completely separable (over the factors) if it were not the "coupling" constraints $\boldsymbol{\nu}_i^a = \boldsymbol{\mu}_i$. Introducing Lagrange multipliers $\boldsymbol{\lambda}_i^a$ for these constraints, the dual problem (*master*) becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{\lambda}} & L(\boldsymbol{\lambda}) \triangleq \sum_a s_a \left(\left(d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a \right)_{i\in\mathcal{N}(a)}, \boldsymbol{\phi}_a \right) \\ \text{s.t.} & \boldsymbol{\lambda} \in \Lambda \triangleq \left\{ \boldsymbol{\lambda} \;\middle|\; \sum_{a\in\mathcal{N}(i)} \boldsymbol{\lambda}_i^a = 0,\ \forall i \right\}, \end{array} \quad (6)$$

where each s_a corresponds to a *slave* subproblem

$$s_a(\boldsymbol{\omega}_{\mathcal{N}(a)}^a, \boldsymbol{\phi}_a) \triangleq \max_{\substack{(\boldsymbol{\nu}_{\mathcal{N}(a)}^a, \boldsymbol{\nu}_a) \\ \in \mathcal{M}(\mathcal{G}_a)}} \sum_{i\in\mathcal{N}(a)} \boldsymbol{\omega}_i^\top \boldsymbol{\nu}_i^a + \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a. \quad (7)$$

Note that the slaves (7) are MAP problems of the same kind as (2), but *local* to each factor a. Denote by $(\hat{\boldsymbol{\nu}}_{\mathcal{N}(a)}^a, \hat{\boldsymbol{\nu}}_a) = \text{MAP}(\boldsymbol{\omega}_{\mathcal{N}(a)}^a, \boldsymbol{\phi}_a)$ the maximizer of (7). The master problem (6) can be addressed elegantly with a projected subgradient algorithm: note that a subgradient $\nabla_{\boldsymbol{\lambda}_i^a} L(\boldsymbol{\lambda})$ is readily available upon solving the ath slave, via $\nabla_{\boldsymbol{\lambda}_i^a} L(\boldsymbol{\lambda}) = \hat{\boldsymbol{\nu}}_i^a$. These slaves can be handled in parallel and then have their solutions gathered for computing a projection onto Λ, which is simply a centering operation. This results in Alg. 1.

Alg. 1 inherits the properties of subgradient algorithms, hence it converges to the optimal value of OPT' in (4) if the stepsize sequence $(\eta_t)_{t\in T}$ is diminishing and nonsummable (Bertsekas et al., 1999). In practice, convergence can be quite slow if the number of slaves is large. This is because it may be hard to reach a consensus on variables with many replicas.

Algorithm 1 DD-Subgradient

```
1: input: factor graph $\mathcal{G}$, parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, number of iterations $T$, stepsize sequence $(\eta_t)_{t=1}^T$
2: Initialize $\boldsymbol{\lambda} = \mathbf{0}$
3: for $t = 1$ to $T$ do
4:     for each factor $a \in \mathcal{A}$ do
5:         Set $\boldsymbol{\omega}_i^a = d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a$, for $i \in \mathcal{N}(a)$
6:         Compute $(\hat{\boldsymbol{\nu}}^a_{\mathcal{N}(a)}, \hat{\boldsymbol{\nu}}_a) = \mathrm{MAP}\big(\boldsymbol{\omega}^a_{\mathcal{N}(a)}, \boldsymbol{\phi}_a\big)$
7:     end for
8:     Compute average $\boldsymbol{\mu}_i = d_i{}^{-1}\sum_{a:i\in\mathcal{N}(a)} \hat{\boldsymbol{\nu}}_i^a$
9:     Update $\boldsymbol{\lambda}_i^a \leftarrow \boldsymbol{\lambda}_i^a - \eta_t\,(\hat{\boldsymbol{\nu}}_i^a - \boldsymbol{\mu}_i)$
10: end for
11: output: $\boldsymbol{\lambda}$
```

Algorithm 2 DD-ADMM

```
1: input: factor graph $\mathcal{G}$, parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, number of iterations $T$, sequence $(\eta_t)_{t=1}^T$, parameter $\tau$
2: Initialize $\boldsymbol{\mu}$ uniformly, $\boldsymbol{\lambda} = \mathbf{0}$
3: for $t = 1$ to $T$ do
4:     for each factor $a \in \mathcal{A}$ do
5:         Set $\boldsymbol{\omega}_i^a = d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a + \eta_t\boldsymbol{\mu}_i$, for $i \in \mathcal{N}(a)$
6:         Update $(\boldsymbol{\nu}^a_{\mathcal{N}(a)}, \boldsymbol{\nu}_a) \leftarrow \mathrm{QUAD}_{\eta_t}\big(\boldsymbol{\omega}^a_{\mathcal{N}(a)}, \boldsymbol{\phi}_a\big)$
7:     end for
8:     Update $\boldsymbol{\mu}_i \leftarrow d_i{}^{-1}\sum_{a:i\in\mathcal{N}(a)} \big(\boldsymbol{\nu}_i^a - \eta_t^{-1}\boldsymbol{\lambda}_i^a\big)$
9:     Update $\boldsymbol{\lambda}_i^a \leftarrow \boldsymbol{\lambda}_i^a - \tau\eta_t\,(\boldsymbol{\nu}_i^a - \boldsymbol{\mu}_i)$
10: end for
11: output: $\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\lambda}$
```

3. Augmented Lagrangian Method

In this section we introduce a faster method, DD-ADMM, which replaces the MAP computation at each factor a by a (usually simple) quadratic problem (QP); this method penalizes disagreements among slaves *more aggressively* than the subgradient method.

Given an optimization problem with equality constraints, the augmented Lagrangian (AL) function is the Lagrangian *augmented* with a quadratic constraint violation penalty. For the constraint problem (5), it is

$$A_\eta(\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\lambda}) \triangleq \sum_a \Big(\sum_{i\in\mathcal{N}(a)} \left(d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a\right)^\top \boldsymbol{\nu}_i^a + \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a \Big) - \sum_a \sum_{i\in\mathcal{N}(a)} \boldsymbol{\lambda}_i^{a\top}\boldsymbol{\mu}_i - \frac{\eta}{2}\sum_a\sum_{i\in\mathcal{N}(a)} \|\boldsymbol{\nu}_i^a - \boldsymbol{\mu}_i\|^2, \qquad (8)$$

where η controls the weight of the penalty. Applied to our problem, a traditional AL method (Hestenes, 1969; Powell, 1969) would alternate between the joint maximization of $A_\eta(\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\lambda})$ w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, and an update of the Lagrange multipliers $\boldsymbol{\lambda}$. Unfortunately, the quadratic term in (8) breaks the separability, making the joint maximization w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ unappealing.

We bypass this problem by using the *alternating direction method of multipliers* (ADMM; Gabay & Mercier 1976; Glowinski & Marroco 1975), in which the joint maximization is replaced by a single Gauss-Seidel step. This yields the following updates:

$$\boldsymbol{\nu}^{(t)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\nu}} A_{\eta_t}(\boldsymbol{\mu}^{(t-1)}, \boldsymbol{\nu}, \boldsymbol{\lambda}^{(t-1)}), \qquad (9)$$

$$\boldsymbol{\mu}^{(t)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\mu}} A_{\eta_t}(\boldsymbol{\mu}, \boldsymbol{\nu}^{(t)}, \boldsymbol{\lambda}^{(t-1)}), \qquad (10)$$

$$\boldsymbol{\lambda}_i^{a(t)} \leftarrow \boldsymbol{\lambda}_i^{a(t-1)} - \tau\eta_t \left(\boldsymbol{\nu}_i^{a(t)} - \boldsymbol{\mu}_i^{(t)}\right), \forall a, i \in \mathcal{N}(a). \qquad (11)$$

Crucially, the maximization w.r.t. $\boldsymbol{\mu}$ (10) has a closed form, while that w.r.t. $\boldsymbol{\nu}$ (9) can be carried out in parallel at each factor, as in Alg. 1. The only difference is that, instead of computing the MAP, each slave now needs to solve a QP of the form

$$\min_{(\boldsymbol{\nu}^a_{\mathcal{N}(a)},\boldsymbol{\nu}_a)\in\mathcal{M}(\mathcal{G}_a)} \frac{\eta_t}{2}\sum_{i\in\mathcal{N}(a)} \|\boldsymbol{\nu}_i^a - \eta_t^{-1}\boldsymbol{\omega}_i^a\|^2 - \boldsymbol{\phi}_a^\top\boldsymbol{\nu}_a. \qquad (12)$$

The resulting algorithm is DD-ADMM (Alg. 2). Let $\mathrm{QUAD}_{\eta_t}(\boldsymbol{\omega}_{\mathcal{N}(a)}, \boldsymbol{\phi}_a)$ denote the solution of (12); as $\eta_t \to 0$, $\mathrm{QUAD}_{\eta_t}(\boldsymbol{\omega}_{\mathcal{N}(a)}, \boldsymbol{\phi}_a)$ approaches $\mathrm{MAP}(\boldsymbol{\omega}_{\mathcal{N}(a)}, \boldsymbol{\phi}_a)$, hence Alg. 2 approaches Alg. 1. However, DD-ADMM converges without the need of decreasing η_t:

Proposition 1 *Let $(\boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}, \boldsymbol{\lambda}^{(t)})_t$ be the sequence of iterates produced by Alg. 2 with a fixed $\eta_t = \eta$ and $0 < \tau \leq (\sqrt{5}+1)/2 \simeq 1.61$. Then the following holds:*

1. *Primal feasibility of (5) is achieved in the limit,* i.e., $\|\boldsymbol{\nu}_i^{a(t)} - \boldsymbol{\mu}_i^{(t)}\| \to 0, \forall a \in \mathcal{A}, i \in \mathcal{N}(a)$;
2. $(\boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)})$ *converges to an optimal solution of (4);*
3. $\boldsymbol{\lambda}^{(t)}$ *converges to an optimal solution of (6);*
4. $\boldsymbol{\lambda}^{(t)}$ *is always dual feasible; hence the objective of (6) evaluated at $\boldsymbol{\lambda}^{(t)}$ approaches* OPT' *from above.*

Proof: 1, 2, and 3 are general properties of ADMM algorithms (Glowinski & Le Tallec, 1989, Thm. 4.2). All necessary conditions are met: problem (5) is convex and the coupling constraint can be written in the form $(\boldsymbol{\nu}^a_{\mathcal{N}(a)})_{a\in\mathcal{A}} = \boldsymbol{M}\boldsymbol{\mu}$ where $\boldsymbol{M}$ has full column rank. To show 4, use induction: $\boldsymbol{\lambda}^{(0)} = \mathbf{0} \in \Lambda$; if $\boldsymbol{\lambda}^{(t-1)} \in \Lambda$, *i.e.*, $\sum_{a\in\mathcal{N}(i)} \boldsymbol{\lambda}_i^{a(t-1)} = 0$, $\forall i$, then, after line 9,

$$\sum_{a\in\mathcal{N}(i)} \boldsymbol{\lambda}_i^{a(t)} = \sum_{a\in\mathcal{N}(i)} \boldsymbol{\lambda}_i^{a(t-1)} - \tau\eta_t \left(\sum_{a\in\mathcal{N}(i)} \boldsymbol{\nu}_i^{a(t)} - d_i \boldsymbol{\mu}_i^{(t)} \right) = (1-\tau)\sum_{a\in\mathcal{N}(i)} \boldsymbol{\lambda}_i^{a(t-1)} = 0 \;\Rightarrow\; \boldsymbol{\lambda}^{(t)} \in \Lambda. \quad \blacksquare$$

Prop. 1 reveals yet another important feature of DD-ADMM: after a sufficient decrease of the residual term, say $\sum_a \sum_{i \in \mathcal{N}(a)} \|\boldsymbol{\nu}_i^a - \boldsymbol{\mu}_i\|^2 < \epsilon$, we have at hand a ϵ-feasible primal-dual pair. If, in addition, the duality gap (difference between (4) and (6)) is $< \delta$, then we are in possession of an (ϵ, δ)-optimality certificate for the LP relaxation. Such a certificate is not readily available in Alg. 1, unless the relaxation is tight.

The next proposition, based on results of Eckstein & Bertsekas (1992), states that convergence may still hold if (12) is solved approximately.

Proposition 2 *Let $\eta_t = \eta$ be fixed and $\tau = 1$. Consider the sequence of residuals $\boldsymbol{r}^{(t)} \triangleq (r_a^{(t)})_{a \in \mathcal{A}}$, where*

$$r_a^{(t)} \triangleq \|(\boldsymbol{\nu}_{\mathcal{N}(a)}^{a(t)}, \boldsymbol{\nu}_a^{(t)}) - \text{QUAD}_{\eta_t}(\boldsymbol{\omega}_{\mathcal{N}(a)}^{(t)}, \boldsymbol{\phi}_a)\|.$$

Then, Prop. 1 still holds provided $\sum_{t=1}^{\infty} \|\boldsymbol{r}^{(t)}\| < \infty$.

4. Solving the Slave Subproblems

We next show how to efficiently solve (12). Several cases admit closed-form solutions: binary pairwise factors, some factors expressing hard constraints, and factors linked to multi-valued variables, once binarized. In all cases, the asymptotic computational cost is the same as that of MAP computations, up to a logarithmic factor. Detailed proofs of the following facts are included as supplementary material.

4.1. Binary pairwise factors

If factor a is binary and pairwise ($|\mathcal{N}(a)| = 2$), problem (12) can be re-written as the minimization of $\frac{1}{2}(z_1 - c_1)^2 + \frac{1}{2}(z_2 - c_2)^2 - c_{12}z_{12}$, w.r.t. $(z_1, z_2, z_{12}) \in [0,1]^3$, under the constraints $z_{12} \le z_1$, $z_{12} \le z_2$, and $z_{12} \ge z_1 + z_2 - 1$, where c_1, c_2 and c_{12} are functions of $\boldsymbol{\omega}_i^a$ and $\boldsymbol{\phi}_a$. Considering $c_{12} \ge 0$, without loss of generality (if $c_{12} < 0$, we recover this case by redefining $c_1' = c_1 + c_{12}$, $c_2' = 1 - c_2$, $c_{12}' = -c_{12}$, $z_2' = 1 - z_2$, $z_{12}' = z_1 - z_{12}$), the lower bound constraints $z_{12} \ge z_1 + z_2 - 1$ and $z_{12} \ge 0$ are always innactive and can be ignored. By inspecting the KKT conditions we obtain the following closed-form solution: $z_{12}^* = \min\{z_1^*, z_2^*\}$ and

$$(z_1^*, z_2^*) = \begin{cases} ([c_1]_{\mathbb{U}}, [c_2 + c_{12}]_{\mathbb{U}}) & \text{if } c_1 > c_2 + c_{12} \\ ([c_1 + c_{12}]_{\mathbb{U}}, [c_2]_{\mathbb{U}}) & \text{if } c_2 > c_1 + c_{12} \\ ([(c_1 + c_2 + c_{12})/2]_{\mathbb{U}}\,, & \\ \;[(c_1 + c_2 + c_{12})/2]_{\mathbb{U}}) & \text{otherwise,} \end{cases}$$

where $[x]_{\mathbb{U}} = \min\{\max\{x, 0\}, 1\}$ denotes the projection (clipping) onto the unit interval.

4.2. Hard constraint factors

Many applications, *e.g.*, in error-correcting coding (Richardson & Urbanke, 2008) and NLP (Smith & Eisner, 2008; Martins et al., 2010; Tarlow et al., 2010) involve *hard constraint factors*—these are factors with indicator log-potential functions: $\boldsymbol{\phi}_a(\boldsymbol{x}_a) = 0$, if $\boldsymbol{x}_a \in \mathcal{S}_a$, and $-\infty$ otherwise, where $\mathcal{S}_a$ is an *acceptance set.* For binary variables, these factors impose logical constraints; *e.g.*,

- the one-hot XOR factor, for which $\mathcal{S}_{\text{XOR}} = \{(x_1, \ldots, x_n) \in \{0,1\}^n \,|\, \sum_{i=1}^n x_i = 1\}$,
- the OR factor, for which $\mathcal{S}_{\text{OR}} = \{(x_1, \ldots, x_n) \in \{0,1\}^n \,|\, \bigvee_{i=1}^n x_i = 1\}$,
- the OR-WITH-OUTPUT factor, for which $\mathcal{S}_{\text{OR-OUT}} = \{(x_1, \ldots, x_n) \in \{0,1\}^n \,|\, \bigvee_{i=1}^{n-1} x_i = x_n\}$.

Variants of these factors (*e.g.*, with negated inputs/outputs) allow computing a wide range of other logical functions. It can be shown that the marginal polytope of a hard factor with binary variables and acceptance set $\mathcal{S}_a$ is defined by $\boldsymbol{z} \in \operatorname{conv} \mathcal{S}_a$, where $\boldsymbol{z} \triangleq (\mu_1(1), \ldots, \mu_n(1))$ and conv denotes the convex hull. Letting $\boldsymbol{c} \triangleq (\omega_a^i(1) + 1 - \omega_a^i(0))_{i \in \mathcal{N}(a)}$, problem (12) is that of minimizing $\|\boldsymbol{z} - \boldsymbol{c}\|^2$ s.t. $\boldsymbol{z} \in \operatorname{conv} \mathcal{S}_a$, which is a Euclidean projection onto a polyhedron:

- $\operatorname{conv} \mathcal{S}_{\text{XOR}}$ is the probability simplex; the projection is efficiently obtained via a sort (Duchi et al., 2008).
- $\operatorname{conv} \mathcal{S}_{\text{OR}}$ is a "faulty" hypercube with a vertex removed, and $\operatorname{conv} \mathcal{S}_{\text{OR-OUT}}$ is a pyramid whose base is a hypercube with a vertex removed; both projections can be efficiently computed with sort operations.

The algorithms and proofs of correctness are provided as supplementary material. In all cases, complexity is $O(|\mathcal{N}(a)| \log |\mathcal{N}(a)|)$ and can be improved to $O(|\mathcal{N}(a)|)$ using a technique similar to Duchi et al. (2008).

4.3. Larger slaves and multi-valued variables

For general factors, a closed-form solution of problem (12) is not readily available. One possible strategy (exploiting Prop. 2) is to use an *inexact* algorithm that becomes sufficiently accurate as Alg. 2 proceeds; this can be achieved by warm-starting with the solution obtained in the previous iteration. This strategy can be useful for handling coarser decompositions, in which each factor is a subgraph such as a chain or a tree. However, unlike the MAP problem in DD-subgradient, in which dynamic programming can be used to compute an exact solution for these special structures, that does not seem possible in QUAD.

Yet, there is an alternative strategy for handling multi-valued variables, which is to *binarize* the graph and make use of the results established in Sect. 4.2 for hard constraint factors. We illustrate this procedure for pairwise MRFs (but the idea carries over when higher order potentials are used): let $X_1, \ldots, X_N$ be the variables of the original graph, and $\mathcal{E} \subseteq \{1, \ldots, N\}^2$ be

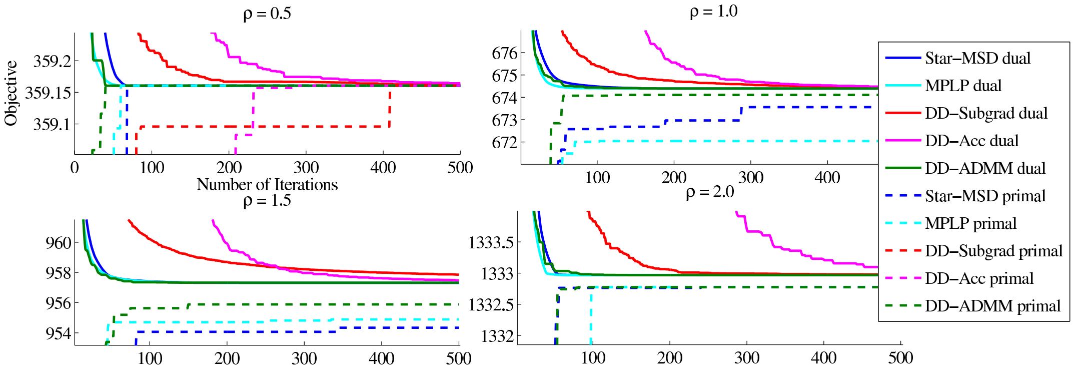

Figure 1. Results for 30×30 random Ising models with several edge couplings. We plot the dual objectives and the best primal feasible solution at each iteration. For DD-subgradient, we set $\eta_t = \eta_0/t$, with η_0 yielding the maximum dual improvement in 10 iterations, with halving steps (those iterations are not plotted). For DD-Acc we plot the most favorable $\epsilon \in \{0.1, 1, 10, 100\}$. For DD-ADMM, we set $\eta = 5.0$ and $\tau = 1.0$. All decompositions are edge-based.

the set of edges. Let $M = |\mathcal{E}|$ be the number of edges and $L = |\mathcal{X}_i|, \forall i$ the number of labels. Then:

- For each node i, define binary variables U_{ik} for each possible $k \in \{1, \ldots, L\}$ of X_i. Link these variables to a XOR factor, imposing $\sum_{k=1}^{L} \mu_i(k) = 1$, $\forall i$.
- For each edge $(i, j) \in \mathcal{E}$, define binary variables $U_{ijkk'}$ for each value pair $(k, k') \in \{1, \ldots, L\}^2$. Link variables $\{U_{ijkk'}\}_{k'=1}^{L}$ and $\neg U_{ik}$ to a XOR factor, for each $k \in \{1, \ldots, L\}$; and link variables $\{U_{ijkk'}\}_{k=1}^{L}$ and $\neg U_{jk'}$ to a XOR factor for each $k' \in \{1, \ldots, L\}$. These impose constraints $\mu_i(k) = \sum_{k'=1}^{L} \mu_{ij}(k, k')$, $\forall k$, and $\mu_j(k') = \sum_{k=1}^{l} \mu_{ij}(k, k')$, $\forall k'$.

The constraints above define a local polytope which is equivalent to the one of the original graph, hence problem (4) is the same for both graphs. This process increases the number of factors to $N + 2ML$, where each is a XOR of size L or $L + 1$. However, solving QUAD for each of these factors only costs $O(L \log L)$ (see Sect. 4.2), hence the overall cost per iteration of Alg. 2 is $O(ML^2 \log L)$ if the graph is connected. Up to a log factor, this is the same as message-passing algorithms or DD-subgradient when run in the original graph, which have $O(ML^2)$ cost per iteration.

5. Experiments

We compare DD-ADMM (Alg. 2) with four other approximate MAP inference algorithms:

- Star-MSD (Sontag et al., 2011), an acceleration of the max-sum diffusion algorithm (Kovalevsky & Koval, 1975; Werner, 2007) based on star updates;
- Generalized MPLP (Globerson & Jaakkola, 2008);
- DD-subgradient (Komodakis et al. 2007, Alg. 1);
- DD-Acc (accelerated DD, Jojic et al. 2010).

All these algorithms address problem (4) with the same cost per iteration: the first two use message-passing, performing block coordinate descent in the dual; DD-Acc uses a smoothed dual objective by adding an entropic term to each subproblem, and then applies optimal first-order methods (Nesterov, 1983), yielding $O(1/\epsilon)$ complexity. The primal and dual objectives are the same for all algorithms.

5.1. Binary Pairwise MRFs

Fig. 1 shows typical plots for an Ising model (binary pairwise MRF) on a random grid, with single node log-potentials chosen as $\theta_i(1) - \theta_i(0) \sim \mathcal{U}[-1, 1]$ and mixed edge couplings in $\mathcal{U}[-\rho, \rho]$, where $\rho \in \{0.5, 1, 1.5, 2\}$. Decompositions are edge-based for all methods. For MPLP and Star-MSD, primal feasible solutions $(\hat{x}_i)_{i=1}^{N}$ were obtained by decoding the single node messages, as in Globerson & Jaakkola (2008); for the DD methods we use $\hat{x}_i = \operatorname{argmax}_{x_i} \mu_i(x_i)$.

We observe that DD-subgradient is the slowest, taking a long time to find a "good" primal feasible solution, arguably due to the large number of slave subproblems. Surprisingly, DD-Acc is also not competitive in this setting, as it consumes many iterations before it reaches a near-optimal region.[1] MPLP performs slightly better than Star-MSD and both are comparable to DD-ADMM in terms of convergence of the dual objective. However, DD-ADMM outperforms all competitors at obtaining a "good" feasible primal solution in early iterations (it retrieved the true MAP in all cases, in ≤ 200 iterations). We conjecture that this rapid progress in the primal is due to the penalty term in the AL (8), which is very effective at pushing

[1] It is conceivable that the early iterations of DD-Acc could make faster progress by annealing ϵ. Here we have just used the variant described by Jojic et al. (2010).

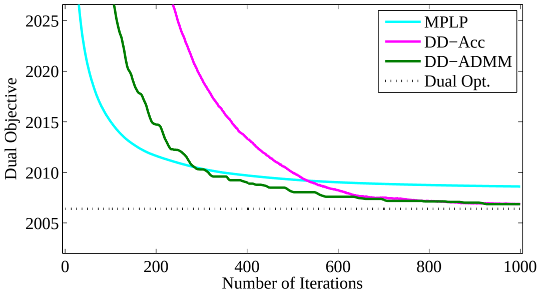

Figure 2. Results for a 20×20 random Potts model with 8-valued nodes and coupling factor $\rho = 10$. We plot the dual objectives, and the value of the true dual optimum. For DD-Acc, $\epsilon = 1.0$; for DD-ADMM, $\eta = 0.5$, $\tau = 1.0$.

for a feasible primal solution of the relaxed LP.

5.2. Multi-valued Pairwise MRFs

To assess the effectiveness of DD-ADMM in the non-binary case, we evaluated it against DD-Acc and MPLP in a Potts model (multi-valued pairwise MRF) with single node log-potentials chosen as $\theta_i(x_i) \sim \mathcal{U}[-1, 1]$ and edge log-potentials as $\theta_{ij}(x_i, x_j) \sim \mathcal{U}[-10, 10]$ if $x_i = x_j$ and 0 otherwise. For DD-Acc and MPLP, we used the same edge decomposition as before, since they can handle multi-valued variables; for DD-ADMM we binarized the graph as described in Sect. 4.3. Fig. 2 shows the best dual solutions obtained at each iteration for the three algorithms. We observe that MPLP decreases the objective very rapidly in the beginning and then slows down. DD-Acc manages to converge faster, but it is relatively slow to take off. DD-ADMM has the best features of both methods.

5.3. Dependency Parsing

A third set of experiments aims at assessing the ability of DD-ADMM for handling problems with heavily constrained outputs. The task is *non-projective dependency parsing* of natural language sentences, to which DD approaches have recently been applied (Koo et al., 2010). Fig. 3 depicts an example of a sentence (the input) and its dependency tree (the output to be predicted). Second-order models are state-of-the-art for this task: they include scores for each possible arc and for certain pairs of arcs (*e.g.*, siblings and grandparents); the goal is to find a directed spanning tree maximizing the overall score.

We experimented with two factor graphs that represent this problem (see Fig. 3). Both models have hard constraints on top of a binary pairwise MRF whose nodes represent arc candidates and whose edges link pairs of arcs which bear a sibling or grandparent relationship. The "tree" model has a TREE hard constraint factor connected to all nodes enforcing the overall assignment to be a directed spanning tree (Smith & Eisner, 2008). Unfortunately, handling this factor poses difficulties for all methods except DD-subgradient.[2]

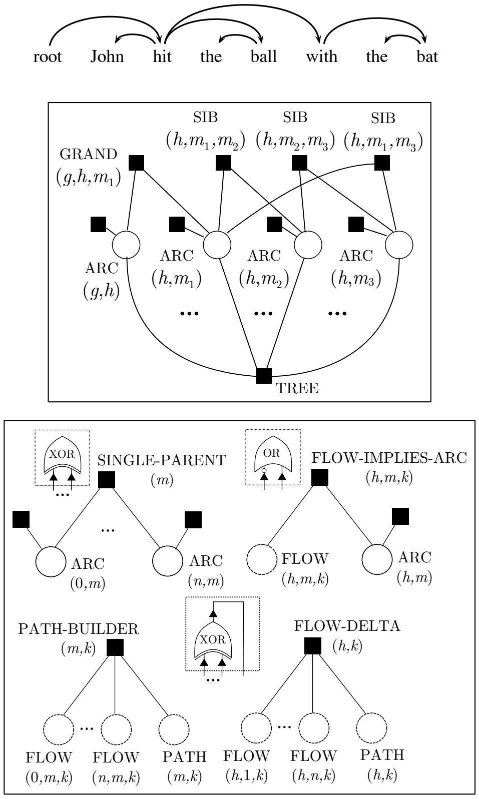

Figure 3. Top: a dependency parse tree, where each arc (h, m) links a *head* word h to a *modifier* word m. Middle: tree-based factor graph corresponding to a second-order dependency parsing model with sibling and grandparent features, including a TREE hard constraint factor. Bottom: the flow-based factor graph is an alternative representation for the same model, in which extra flow and path variables are added, and the TREE factor is replaced by smaller XOR, OR and OR-OUT. See Martins et al. (2010) for details.

[2] Further information about the combinatorial TREE factor can be found in Martins et al. (2010) and references therein. Briefly, solving the MAP subproblem (necessary for DD-subgradient) corresponds to finding a maximum weighted arborescence, which can be done in $O(n^2)$ time, where n is the number of words in the sentence (Tarjan, 1977). Computing all posterior marginals (necessary for DD-Acc) can be done in $O(n^3)$ time invoking the matrix-tree theorem (Smith & Smith, 2007; Koo et al., 2007; McDonald & Satta, 2007). Unfortunately, this procedure suffers from severe numerical problems in the low-temperature setting, which prevents its use in DD-Acc where the temperature must be set as $O(\epsilon/(n \log n))$. Finally, no efficient algorithm is currently known for the simultaneous computation of all max-marginals in the TREE factor (which

The "flow" model imposes the same global constraint by adding extra variables and several XOR, OR and OR-OUT factors (Martins et al., 2010). The two models have the same expressive power, and differ from the one in Koo et al. (2010), which combines a tree constraint with factors that emulate head automata (instead of *all* pairs of arcs) and has fewer slaves. In our case, both factor graphs yield $O(n^3)$ slaves (n is the number of words in the sentence), degrading the performance of standard DD methods.

This is illustrated in Fig. 4, which shows that DD-subgradient converges slowly, even with the more favorable tree-based factor graph (both for synthetic and real-world data). For this problem, MPLP and Star-MSD also have poor performance, while DD-ADMM manages to converge to a near-optimal solution very fast (note the sharp decrease in relative error on the bottom plot, compared with DD-subgradient). This has an impact in final accuracy: setting a maximum of 1000 iterations for all algorithms gave DD-ADMM an advantage of $> 1.5\%$ in unlabeled attachment score (fraction of words with the correct head attached), in the Penn Treebank dataset.

6. Related Work and Final Remarks

DD-subgradient was first proposed for image segmentation using pairwise (Komodakis et al., 2007) and higher order factor graphs (Komodakis & Paragios, 2009). It was recently adopted for NLP (Koo et al., 2010), with only a few slave subproblems handled with dynamic programming or combinatorial algorithms.

Johnson et al. (2007) proposed smoothing the objective by adding an *entropic* term to each subproblem, with the goal of enabling gradient-based optimization; each subproblem becomes that of computing marginals at a particular temperature. Jojic et al. (2010) combined this with optimal first order methods to accelerate DD to $O(1/\epsilon)$ complexity. This rate, however, relies on the ability to compute low-temperature marginals with arbitrary precision, which poses problems for some of the hard constraint factors considered in this paper. In contrast, DD-ADMM uses *exact* solutions of the corresponding slave subproblems, efficiently computed using sort operations (see Sect. 4.2).

We point out that replacing the quadratic penalty of the AL (8) by an entropic one would *not* lead to the same subproblems as in Jojic et al. (2010): it would lead to the problem of minimizing non-strictly convex free energies with different counting numbers. Although extensions of ADMM to Bregman penalties have been considered in the literature, convergence has been shown only for quadratic penalties.

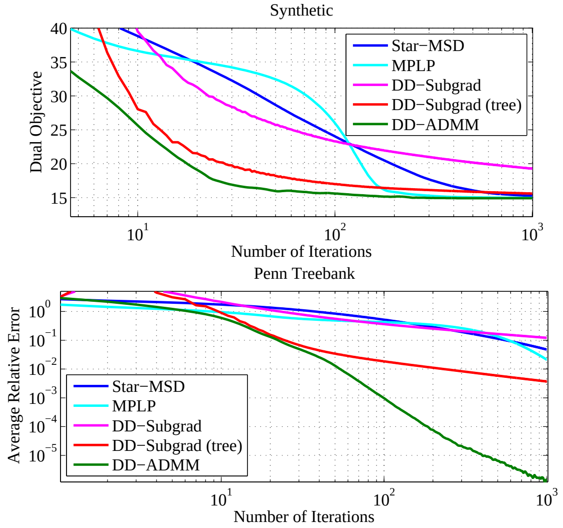

Figure 4. Dependency parsing with 2nd-order models. For DD-subgradient, we consider both tree-based and flow-based factor graphs, and η_0 as in Fig. 1. DD-ADMM ($\eta = 0.05$, $\tau = 1.5$), MPLP, and Star-MSD ran only on the flow-based factor graph (see footnote 2). DD-Acc is not shown due to numerical problems when computing some low-temperature marginals. Top: synthetic 10-word sentences; we randomly generated (unary) arc log-potentials from $\mathcal{N}(0, 1)$ and (pairwise) grandparent and sibling log-potentials from $\mathcal{N}(0, 0.1)$. Bottom: §23 of the Penn Treebank; the plot shows relative errors per iteration w.r.t. the dual optimum, averaged over the 2,399 test sentences.

Quadratic problems were also recently considered in a sequential algorithm (Ravikumar et al., 2010); however, that algorithm tackles the *primal* formulation and only pairwise models are considered. A similar cyclic projection can be adopted in DD-ADMM to approximately solve QUAD for larger slaves.

DD-ADMM is dual decomposable, hence the slaves can be solved in parallel, making it suitable for multi-core architectures with obvious speed-ups. A significant amount of computation can be saved by *caching* and *warm-starting* the subproblems, which tend to become more and more similar across later iterations.

In the future, we plan to experiment with larger slaves, by using approximate ADMM steps as enabled by Prop. 2. The encouraging results of DD-ADMM for solving LP relaxations suggest that it can also be useful for *tightening* these relaxations towards the true MAP, as the MPLP algorithm in Sontag et al. (2008).

would be necessary for MPLP and Star-MSD); or for computing an Euclidean projection onto the arborescence polytope (which would be necessary for DD-ADMM).

Acknowledgments

A. M. was supported by a FCT/ICTI grant through the CMU-Portugal Program, and also by Priberam. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250), and by a FCT grant PTDC/EEA-TEL/72572/2006. N. S. was supported by NSF CAREER IIS-1054319. E. X. was supported by AFOSR FA9550010247, ONR N000140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship.

References

Bertsekas, D., Hager, W., and Mangasarian, O. *Nonlinear programming*. Athena Scientific, 1999.

Bertsekas, D.P., Nedic, A., and Ozdaglar, A.E. *Convex analysis and optimization*. Athena Scientific, 2003.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers (to appear)*. Now Publishers, 2011.

Boyle, J.P. and Dykstra, R.L. A method for finding projections onto the intersections of convex sets in Hilbert spaces. In *Advances in order restricted statistical inference*, pp. 28–47. Springer Verlag, 1986.

Dantzig, G. and Wolfe, P. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the L1-ball for learning in high dimensions. In *ICML*, 2008.

Eckstein, J. and Bertsekas, D. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992.

Everett III, H. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

Globerson, A. and Jaakkola, T. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *NIPS*, 20, 2008.

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

Glowinski, R. and Marroco, A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, 9:41–76, 1975.

Hestenes, M. Multiplier and gradient methods. *Jour. Optim. Theory and Applic.*, 4:302–320, 1969.

Johnson, J.K., Malioutov, D.M., and Willsky, A.S. Lagrangian relaxation for MAP estimation in graphical models. In *45th Annual Allerton Conference on Communication, Control and Computing*, 2007.

Jojic, V., Gould, S., and Koller, D. Accelerated dual decomposition for MAP inference. In *ICML*, 2010.

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

Kolmogorov, V. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28:1568–1583, 2006.

Komodakis, N. and Paragios, N. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.

Komodakis, N., Paragios, N., and Tziritas, G. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

Koo, T., Globerson, A., Carreras, X., and Collins, M. Structured prediction models via the matrix-tree theorem. In *EMNLP*, 2007.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. Dual decomposition for parsing with non-projective head automata. In *EMNLP*, 2010.

Kovalevsky, V.A. and Koval, V.K. A diffusion algorithm for decreasing energy of max-sum labeling problem. Glushkov Institute of Cybernetics, Kiev, USSR, 1975.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. Factor graphs and the sum-product algorithm. *IEEE Trans. Information Theory*, 47, 2001.

Martins, A., Smith, N., Xing, E., Figueiredo, M., and Aguiar, P. Turbo parsers: Dependency parsing by approximate variational inference. In *EMNLP*, 2010.

McDonald, R. and Satta, G. On the complexity of non-projective data-driven dependency parsing. In *IWPT*, 2007.

Nesterov, Y. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Doklady*, 27:372–376, 1983.

Powell, M. A method for nonlinear constraints in minimization problems. In Fletcher, R. (ed.), *Optimization*, pp. 283–298. Academic Press, 1969.

Ravikumar, P., Agarwal, A., and Wainwright, M. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *JMLR*, 11:1043–1080, 2010.

Richardson, T.J. and Urbanke, R.L. *Modern coding theory*. Cambridge Univ Pr, 2008.

Schlesinger, M. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976.

Shor, N. *Minimization methods for non-differentiable functions*. Springer, 1985.

Smith, D. and Eisner, J. Dependency parsing by belief propagation. In *EMNLP*, 2008.

Smith, D. and Smith, N. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*, 2007.

Sontag, D., Meltzer, T., Globerson, A., Weiss, Y., and Jaakkola, T. Tightening LP relaxations for MAP using message-passing. In *UAI*, 2008.

Sontag, D., Globerson, A., and Jaakkola, T. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press, 2011.

Tarjan, R.E. Finding optimum branchings. *Networks*, 7 (1):25–36, 1977.

Tarlow, D., Givoni, I. E., and Zemel, R. S. HOP-MAP: Efficient message passing with high order potentials. In *AISTATS*, 2010.

Wainwright, M. and Jordan, M. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers, 2008.

Wainwright, M., Jaakkola, T., and Willsky, A. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. Information Theory*, 51(11):3697–3717, 2005.

Werner, T. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:1165–1179, 2007.