Augmenting Dual Decomposition for MAP Inference

André F. T. Martins[*†] **Noah A. Smith**[*] **Eric P. Xing**[*]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Pedro M. Q. Aguiar[‡]
[‡]Instituto de Sistemas e Robótica,
Instituto Superior Técnico, Lisboa, Portugal

Mário A. T. Figueiredo[†]
[†]Instituto de Telecomunicações,
Instituto Superior Técnico, Lisboa, Portugal

Abstract

In this paper, we propose combining augmented Lagrangian optimization with the dual decomposition method to obtain a fast algorithm for approximate MAP (*maximum a posteriori*) inference on factor graphs. We also show how the proposed algorithm can efficiently handle problems with (possibly global) structural constraints. The experimental results reported testify for the state-of-the-art performance of the proposed approach.

1 Introduction

Finding the most probable (usually called MAP) configuration of a probabilistic graphical model (*e.g.*, a factor graph – FG [27]) is in general an NP-hard problem, a fact which has stimulated much work on approximate methods. The *dual decomposition* (DD) [3, 4] is one such approximate method, which has been recently used in computer vision [17] and natural language parsing [18]. In a nutshell, DD works by breaking the original hard problem into a set of smaller (slave) subproblems. This set of subproblems, together with the constraints that they should agree on the variables they share, yields a constrained optimization problem, the Lagrange dual of which is then usually attacked with a subgradient method. While a subgradient algorithm handles this dual problem efficiently for lightweight decompositions (*i.e.*, with few slaves), in the presence of a large number of slaves its performance degrades, making accelerated methods worthy of study [14]. Here, we propose to ally the strength of DD with the effectiveness of augmented Lagrangian (AL) methods, which have a long and successful history in optimization [13, 21], and which have recently been shown to be extremely competitive for some large scale problems [1, 7, 12]. Specifically, we use the *alternating direction method of multipliers* (ADMM) [8, 11, 6] to handle the dual of the constrained problem resulting from the DD, and show that the resulting method has state-of-the-art performance.

Being interested in problems with (possibly global) structural constraints (common in structured prediction [2]), we show that the proposed method can handle this class of problems efficiently. This efficiency is rooted in the fact that the slave problems associated with the hard factors enforcing some of these structural constraints can be solved with the help of sort operations. Each ADMM iteration thus has a cost comparable to that of an iteration of the message-passing algorithm or of the subgradient method applied to the DD. Moreover, the residual term in the AL can be used as a primal feasibility test and to obtain optimality certificates for the approximate MAP problem, an important advantage over other methods. Finally, our method is also heavily parallelizable.

2 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 FG $\mathcal{G}$, composed of variable nodes $\{1, \ldots, n\}$ and a set of factor nodes $\mathcal{A}$: $P_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) \propto \exp\left(\sum_{i=1}^{n} \theta_i(x_i) + \sum_{a \in \mathcal{A}} \phi_a(\boldsymbol{x}_a)\right)$,

where each factor $a \in \mathcal{A}$ is linked to a subset of variables $N(a) \subseteq \{1,\dots,n\}$, $\boldsymbol{x}_a$ stands for the subvector indexed by the elements of $N(a)$, and θ_i and ϕ_a are, respectively, unary and higher-order log-potentials. 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 $\boldsymbol{\phi}_a \triangleq (\phi_a(\boldsymbol{x}_a))_{\boldsymbol{x}_a \in \mathcal{X}_a}$.

Given a FG $\mathcal{G}$, a common task is to find the most probable assignment $\hat{\boldsymbol{x}} \triangleq \arg\max_{\boldsymbol{x} \in \mathcal{X}} P_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ (often termed MAP—*maximum a posteriori*—since $P_{\boldsymbol{\theta},\boldsymbol{\phi}}$ is usually the posterior distribution in some Bayesian inference problem). 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 $\boldsymbol{\nu} \triangleq (\boldsymbol{\nu}_a)_{a \in \mathcal{A}}$ and letting $\mathcal{M}(\mathcal{G})$ be the *marginal polytope* of $\mathcal{G}$ – i.e. the set of realizable marginals [27]. This yields

$$\text{OPT} \quad \triangleq \quad \max_{(\boldsymbol{\mu},\boldsymbol{\nu}) \in \mathcal{M}(\mathcal{G})} \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_a \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \tag{1}$$

which always admits an integer solution. Unfortunately, $\mathcal{M}(\mathcal{G})$ often lacks a concise representation (e.g., if $\mathcal{G}$ has loops), rendering (1) 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

$$\mathcal{L}(\mathcal{G}) = \left\{(\boldsymbol{\mu},\boldsymbol{\nu}) \mid \left(\mathbf{1}^\top \boldsymbol{\mu}_i = 1, \forall i\right) \wedge \left(\mathbf{H}_{ia}\boldsymbol{\nu}_a = \boldsymbol{\mu}_i, \forall a, i \in N(a)\right) \wedge \left(\boldsymbol{\nu}_a \geq 0, \forall a\right)\right\}, \tag{2}$$

where $\mathbf{H}_{ia}(x_i, \boldsymbol{x}_a) = 1$ if $\boldsymbol{x}_a$ agrees with x_i, 0 otherwise. This yields the following LP relaxation:

$$\text{OPT}' \quad \triangleq \quad \max_{(\boldsymbol{\mu},\boldsymbol{\nu}) \in \mathcal{L}(\mathcal{G})} \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{\mu}_i + \sum_a \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \tag{3}$$

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

3 Dual Decomposition

Several message passing algorithms [9, 15] are derived via some reformulation of (3) followed by dualization. The DD method [17] reformulates (3) by adding new variables $\boldsymbol{\nu}_i^a$ (for each factor a and $i \in N(a)$) that are local "replicas" of the marginals $\boldsymbol{\mu}_i$, and enforcing agreement among those variables. Letting $d_i = |\{a | i \in N(a)\}|$ be the degree of node i, (3) is rewritten as

$$\begin{aligned} \max_{\boldsymbol{\nu},\boldsymbol{\mu}} \quad & \sum_a \left(\sum_{i \in N(a)} d_i^{-1} \boldsymbol{\theta}_i^\top \boldsymbol{\nu}_i^a + \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a \right) \\ \text{s.t.} \quad & (\boldsymbol{\nu}_{N(a)}^a, \boldsymbol{\nu}_a) \in \mathcal{M}(\mathcal{G}_a), \quad \forall a; \qquad \boldsymbol{\nu}_i^a = \boldsymbol{\mu}_i, \quad \forall a, i \in N(a), \end{aligned} \tag{4}$$

where $\mathcal{G}_a$ is the subgraph of $\mathcal{G}$ comprised only of factor a and the variables in $N(a)$, $\mathcal{M}(\mathcal{G}_a)$ is the corresponding marginal polytope, and we denote $\boldsymbol{\nu}_{N(a)}^a \triangleq (\boldsymbol{\nu}_i^a)_{i \in N(a)}$. Note that problem (4) would be completely separable (over the factors) if it were not the "coupling" constraints $\boldsymbol{\nu}_i^a = \boldsymbol{\mu}_i$. With Lagrange multipliers $\boldsymbol{\lambda}_i^a$ for these constraints, the dual problem is $\min_{\boldsymbol{\lambda} \in \Lambda} L(\boldsymbol{\lambda})$, where:

$$L(\boldsymbol{\lambda}) \quad \triangleq \quad \sum_a \max_{(\boldsymbol{\nu}_{N(a)}^a, \boldsymbol{\nu}_a) \in \mathcal{M}(\mathcal{G}_a)} \sum_{i \in N(a)} \left(d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a\right)^\top \boldsymbol{\nu}_i^a + \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \tag{5}$$

$$\Lambda \quad \triangleq \quad \left\{ \boldsymbol{\lambda} \;\middle|\; \sum_{a | i \in N(a)} \boldsymbol{\lambda}_i^a = 0, \ \forall i \right\}. \tag{6}$$

A closer look at (5) reveals a set of MAP subproblems of the same kind of (1), but now *local* to each factor a: the so-called *slave* problems. The *master* problem is the minimization of L w.r.t. $\boldsymbol{\lambda}$, which can be carried out elegantly via the projected subgradient algorithm: a subgradient is simply $\nabla_{\boldsymbol{\lambda}_i^a} L(\boldsymbol{\lambda}) = \hat{\boldsymbol{\nu}}_i^a$, where $(\hat{\boldsymbol{\nu}}_{N(a)}^a, \hat{\boldsymbol{\nu}}_a)$ is solution of the slave problem associated with a (these slaves can be handled in parallel); a projection onto Λ is simply a centering operation. The resulting algorithm is described in Alg. 1 (where we denote by $\text{MAP}(\boldsymbol{\omega}_{N(a)}, \boldsymbol{\phi}_a)$ the MAP solution for factor a, given unary log-potentials $\boldsymbol{\omega}_{N(a)} \triangleq (\boldsymbol{\omega}_i^a)_{i \in N(a)}$ and factor log-potentials $\boldsymbol{\phi}_a$). For adequate choices of the stepsize sequence $(\eta_t)_{t \in T}$, this method is guaranteed to converge; however, convergence can be slow if the number of slaves is large. In the next section we introduce a faster method which, instead of computing the MAP at each factor a, computes projections onto $\mathcal{M}(\mathcal{G}_a)$.

Algorithm 1 DD-Subgradient

input: factor graph $\mathcal{G}$, parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, number of iterations T, sequence $(\eta_t)_{t=1}^T$
Initialize $\boldsymbol{\lambda} = \mathbf{0}$
for $t = 1$ **to** T **do**
  for each factor $a \in \mathcal{A}$ **do**
    Set unary potentials $\boldsymbol{\omega}_i^a = d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a$, for $i \in N(a)$
    Compute $(\hat{\boldsymbol{\nu}}_{N(a)}^a, \hat{\boldsymbol{\nu}}_a) = \text{MAP}\big(\boldsymbol{\omega}_{N(a)}^a, \boldsymbol{\phi}_a\big)$
  end for
  Compute average $\bar{\boldsymbol{\nu}}_i = {d_i}^{-1} \sum_{a:i\in N(a)} \hat{\boldsymbol{\nu}}_i^a$
  Update $\boldsymbol{\lambda}_i^a \leftarrow \boldsymbol{\lambda}_i^a - \eta_t \left(\hat{\boldsymbol{\nu}}_i^a - \bar{\boldsymbol{\nu}}_i\right)$
end for
output: $\boldsymbol{\lambda}$

4 Augmented Lagrangian (AL) Method

Given an optimization problem with equality constraints, the AL function is the Lagrangian *augmented* with a quadratic constraint violation penalty. For the constraint problem (4), the AL is

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

where η controls the weight of the penalty. We want to maximize this function with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, subject to $(\boldsymbol{\nu}_{N(a)}^a, \boldsymbol{\nu}_a) \in \mathcal{M}(\mathcal{G}_a), \forall a$. Unfortunately, the quadratic term breaks the separability, making this maximization harder. In the ADMM approach, this difficulty is addressed by alternating between maximizing w.r.t. $\boldsymbol{\mu}$ and w.r.t. $\boldsymbol{\nu}$, followed by an adjustment of the Lagrange multipliers [8, 11, 6] (as shown in Alg. 2). Crucially, the maximization w.r.t. $\boldsymbol{\mu}$ has closed form, while that w.r.t. $\boldsymbol{\nu}$ can be carried out in parallel at each factor, as in Alg. 1; the only difference is that, instead of MAP, the slaves are now quadratic problems of the form

$$\min_{(\boldsymbol{\nu}_{N(a)}^a, \boldsymbol{\nu}_a)\in\mathcal{M}(\mathcal{G}_a)} \frac{\eta_t}{2} \sum_{i\in a} \|\boldsymbol{\nu}_i^a - \eta_t^{-1}\boldsymbol{\omega}_i^a\|^2 - \boldsymbol{\phi}_a^\top \boldsymbol{\nu}_a, \tag{7}$$

where $\eta_t > 0$ is the penalty parameter at iteration t. In Alg. 2, we denote by $\text{QUAD}_{\eta_t}(\boldsymbol{\omega}_{N(a)}, \boldsymbol{\phi}_a)$ the solution of (7); note that as $\eta_t \to 0$, $\text{QUAD}_{\eta_t}(\boldsymbol{\omega}_{N(a)}, \boldsymbol{\phi}_a)$ approaches $\text{MAP}(\boldsymbol{\omega}_{N(a)}, \boldsymbol{\phi}_a)$, thus Alg. 2 approaches Alg. 1. However, we will see that for a proper choice of η_t, Alg. 2 converges faster. In practice, it is common to use a fixed $\eta_t = \eta$ and $1 < \tau \leq (\sqrt{5}+1)/2 \simeq 1.61$ [10]. With these choices, and assuming that QUAD_{η_t} is computed exactly, convergence is guaranteed, since in (4), both the objective function (which is linear) and the feasible set are convex [10]. Under certain conditions, convergence can still be guaranteed with only approximate solutions of (7) [6].

Algorithm 2 DD-ADMM

input: factor graph $\mathcal{G}$, parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$, number of iterations T, sequence $(\eta_t)_{t=1}^T$, parameter τ
Initialize $\boldsymbol{\mu}$ uniformly, $\boldsymbol{\lambda} = \mathbf{0}$
for $t = 1$ **to** T **do**
  for each factor $a \in \mathcal{A}$ **do**
    Set unary potentials $\boldsymbol{\omega}_i^a = d_i^{-1}\boldsymbol{\theta}_i + \boldsymbol{\lambda}_i^a + \eta_t\boldsymbol{\mu}_i$, for $i \in N(a)$
    Update $(\boldsymbol{\nu}_{N(a)}^a, \boldsymbol{\nu}_a) \leftarrow \text{QUAD}_{\eta_t}\big(\boldsymbol{\omega}_{N(a)}^a, \boldsymbol{\phi}_a\big)$
  end for
  Update $\boldsymbol{\mu}_i \leftarrow {d_i}^{-1} \sum_{a:i\in N(a)} \left(\boldsymbol{\nu}_i^a - \eta_t^{-1}\boldsymbol{\lambda}_i^a\right)$
  Update $\boldsymbol{\lambda}_i^a \leftarrow \boldsymbol{\lambda}_i^a - \tau\eta_t \left(\boldsymbol{\nu}_i^a - \boldsymbol{\mu}_i\right)$
end for
output: $\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\lambda}$

The remaining challenge is to devise efficient algorithms for solving (7). We next show a closed form solution for the the case of binary pairwise factors, and an *exact and efficient* ($O(|N(a)|\log|N(a)|)$,

since it is based on sorting) algorithm for several hard constraint factors that arise in practice (see, e.g., [19]). Up to log terms, this cost is the same as that of computing the MAP for those factors.

Binary pairwise factors. If factor a is binary and pairwise ($|N(a)| = 2$), problem (7) 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 $z_{12} \leq z_1$, $z_{12} \leq z_2$, and $z_{12} \geq z_1 + z_2 - 1$, where c_1, c_2 and c_{12} are functions of $\boldsymbol{\omega}_i^a$ and $\boldsymbol{\phi}_a$. Considering $c_{12} \geq 0$, 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 $z_{12} \geq z_1 + z_2 - 1$ and $z_{12} \geq 0$ 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

$$(z_1^*, z_2^*) = \begin{cases} ([c_1]_{\mathbb{U}}, & [c_2 + c_{12}]_{\mathbb{U}}) & \text{if } c_1 > c_2 + c_{12} \\ ([c_1 + c_{12}]_{\mathbb{U}}, & [c_2]_{\mathbb{U}}) & \text{if } c_2 > c_1 + c_{12} \\ ([(c_1 + c_2 + c_{12})/2]_{\mathbb{U}}, & [(c_1 + c_2 + c_{12})/2]_{\mathbb{U}}) & \text{otherwise,} \end{cases} \tag{8}$$

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

Hard constraint factors. Hard factors have 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. This type of factors has several applications, such as error-correcting decoders [23], named entity resolution [26], and dependency parsing [19]. For binary variables, hard 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 $\boldsymbol{c} = (\omega_a^i(1) + 1 - \omega_a^i(0))_{i \in N(a)}$, problem (7) becomes that of minimizing $\|\boldsymbol{z} - \boldsymbol{c}\|^2$ subject to $\boldsymbol{z} \in \operatorname{conv} \mathcal{S}_a$, which is a Euclidean projection onto a polyhedron:

- $\operatorname{conv} \mathcal{S}_{\text{XOR}}$ is the probability simplex; the projection can be computed efficiently using a sort [5].
- $\operatorname{conv} \mathcal{S}_{\text{OR}}$ is a 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; in both cases, the projections can be efficiently computed using one or two sorts; due to lack of space, we omit details.

Larger slaves. Finally, note that it is simple to address general, larger subgraphs using algorithms with fast convergence guarantees: e.g., a primal-dual vanilla scheme, like the one proposed in [22], resulting in cyclic projection algorithms. These can be useful to tackle coarser decompositions, in which each subgraph is a chain or a tree. Even if each projection is not computed exactly, convergence of ADMM may still be guaranteed under certain conditions [6]; this deserves further study.

5 Experiments

We compare DD-ADMM (Alg. 2) with two other approximate MAP inference algorithms: DD-subgradient ([17], Alg. 1); Star-MSD (max-sum diffusion with star updates, [20]), which performs dual block coordinate descent message-passing. Fig. 1 shows a typical plot for an Ising model (binary pairwise MRF) on a random grid. 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 (we have only considered naive decompositions; coarser ones, *i.e.*, fewer slave problems, are the topic of future work). Star-MSD performs better but it is outperformed by DD-ADMM, which manages to approach a near optimal primal-dual solution in a few tens of iterations.

A second 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 [18]. Fig. 2 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

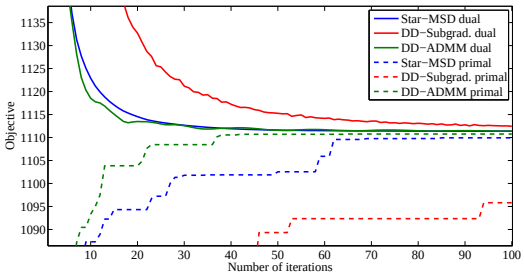

Figure 1: Results for a randomly generated 30×30 Ising model. Shown are the dual objectives and the best primal feasible solution at each iteration. For DD-subgradient, we set $\eta_t = \eta_0/t$ and picked the η_0 yielding maximum dual improvement in 10 iterations, with halving steps (those iterations are not plotted). For DD-ADMM, we set $\eta = 5.0$ and $\tau = 1.5$. All decompositions are edge-based.

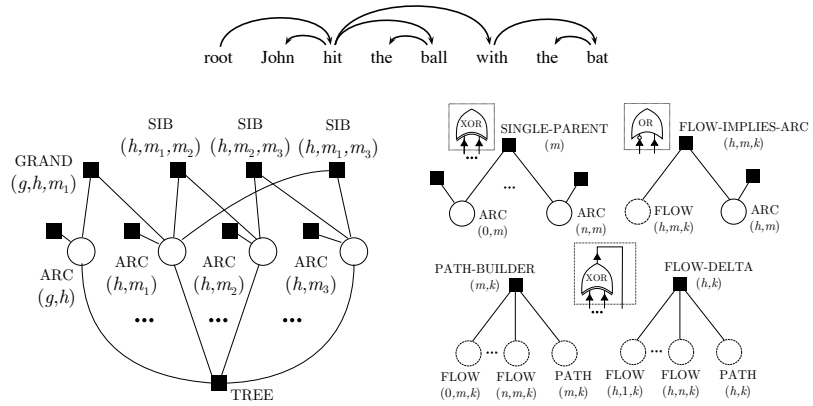

Figure 2: Top: Example of a dependency parse tree. Each arc (h, m) links a *head* word h to a *modifier* word m. Bottom left: Tree-based factor graph corresponding to a second-order dependency parsing model with sibling and grandparent features [25]. Note the TREE hard constraint factor, which enforces the overall variable assignment to encode a valid parse tree. Bottom right: 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 factors to impose connectivity. See [19] for further information.

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. 2). Our model is different from the one in Koo et al. [18], which combines a tree constraint with factors that emulate head automata (instead of pairs of arcs); this essentially amounts to combining only two or three slaves. In our case, both factor graphs yield $O(n^3)$ slaves (n being the number of words in the sentence), which degrades the performance of standard DD methods. This is illustrated in Fig. 3, which shows that DD-subgradient is slow to converge, even for the more favorable tree-based factor graph (both with synthetic and real-world data). For this problem, Star-MSD also has poor performance, while DD-ADMM manages to converge to a near-optimal solution very fast (note the sharp decrease in relative error on the righthand plot, compared with DD-subgradient).

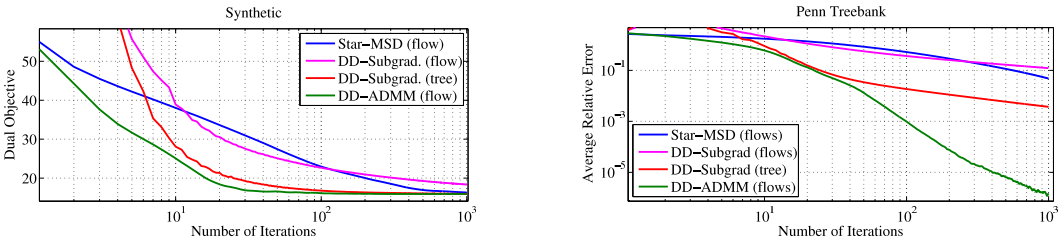

Figure 3: Dependency parsing with second-order models. For DD-subgradient, we considered both tree-based and flow-based factor graphs, and set η_0 as in Fig. 1. DD-ADMM ($\eta = 0.05$ and $\tau = 1.5$) and Star-MSD ran on a flow-based factor graph. Left: synthetic model for a sentence with 10 words; we randomly generated (unary) arc log-potentials from $\mathcal{N}(0, 1)$, and (pairwise) grandparent and sibling log-potentials from $\mathcal{N}(0, 0.1)$. Right: results on §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.

6 Related Work and Final Remarks

The DD method for MAP inference was first proposed for image segmentation using pairwise [17] and higher order factor graphs [16]. It was recently adopted for natural language parsing [24, 18], with only a couple of slave subproblems handled with dynamic programming. Accelerated DD were first considered in [14], where the approach is to *individually* smooth each slave subproblem with the addition of an *entropic* term, making the objective differentiable, thus adequate for gradient-based methods. Although there are variants of ADMM that use entropic (rather than quadratic) penalties, that does *not* lead to the same problem as in [14]. Quadratic problems (as in (7)) were also recently considered in a sequential algorithm [22]; however, that algorithm tackles the *primal* formulation and only pairwise models are considered.

The proposed DD-ADMM algorithm is dual decomposable, hence the slaves can all be solved in parallel, making DD-ADMM particularly suitable for multi-core architectures, offering important speed-ups. Finally, note that a significant amount of computation can be saved by *caching* and *warm-starting* the subproblems, which tend to become more and more similar in later iterations. In future work, we plan to experiment with larger slaves and approximate ADMM steps.

Acknowledgments

A. M. was supported by FCT/ICTI through the CMU-Portugal Program, and also by Priberam Informática. N. S. was supported by NSF IIS-0915187 and an IBM faculty award. E. X. was supported by AFOSR FA9550010247, ONR N000140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship. This work was partially funded by the FET programme (EU FP7), under the SIMBAD project (contract 213250), and by a FCT grant PTDC/EEA-TEL/72572/2006.

References

[1] Afonso, M., Bioucas-Dias, J., and Figueiredo, M. (2010). Fast image recovery using variable splitting and constrained optimization. *IEEE Trans. on Image Processing*, 19:2345–2356.
[2] Bakır, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., and Vishwanathan, S. (2007). *Predicting Structured Data*. The MIT Press.
[3] Bertsekas, D., Hager, W., and Mangasarian, O. (1999). *Nonlinear programming*. Athena Scientific.
[4] Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations research*, 8(1):101–111.
[5] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the L1-ball for learning in high dimensions. In *ICML*.
[6] Eckstein, J. and Bertsekas, D. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318.
[7] Figueiredo, M. and Bioucas-Dias, J. (2010). Restoration of Poissonian images using alternating direction optimization. *IEEE Trans. on Image Processing*. to appear.
[8] Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40.
[9] Globerson, A. and Jaakkola, T. (2008). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *NIPS*, 20.
[10] Glowinski, R. (1984). *Numerical methods for nonlinear variational problems*. Springer-Verlag.
[11] Glowinski, R. and Marroco, A. (1975). Sur l'approximation, par elements finis dordre un, et la resolution, par penalisation-dualite dune classe de problemes de Dirichlet non lineares. *Revue Française dAutomatique, Informatique et Recherche Opérationelle*, 9:41–76.
[12] Goldfarb, D., Ma, S., and Scheinberg, K. (2010). Fast alternating linearization methods for minimizing the sum of two convex functions. Technical Report 10-02, UCLA CAM.
[13] Hestenes, M. (1969). Multiplier and gradient methods. *Jour. Optim. Theory and Applic.*, 4:302–320.
[14] Jojic, V., Gould, S., and Koller, D. (2010). Accelerated dual decomposition for MAP inference. In *Proc. of ICML*.
[15] Kolmogorov, V. and Wainwright, M. (2005). On the optimality of tree-reweighted max-product message passing. In *UAI*, volume 21. Citeseer.
[16] Komodakis, N. and Paragios, N. (2009). Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proc. of CVPR 2009*, pages 2985–2992.
[17] Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *International Conference on Computer Vision*. Citeseer.
[18] Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.
[19] Martins, A. F. T., Smith, N. A., Xing, E. P., Figueiredo, M. A. T., and Aguiar, P. M. Q. (2010). Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.
[20] Meshi, O., Sontag, D., Jaakkola, T., and Globerson, A. (2010). Learning Efficiently with Approximate Inference via Dual Losses. In *Proc. ICML*. Citeseer.
[21] Powel, M. (2009). A method for nonlinear constraints in minimization problems. In Fletcher, R., editor, *Optimization*, pages 283–298. Academic Press.
[22] Ravikumar, P., Agarwal, A., and Wainwright, M. (2010). Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *JMLR*, 11:1043–1080.
[23] Richardson, T. and Urbanke, R. (2008). *Modern coding theory*. Cambridge Univ Pr.
[24] Rush, A. M., Sontag, D., Collins, M., and Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of EMNLP*.
[25] Smith, D. A. and Eisner, J. (2008). Dependency parsing by belief propagation. In *Proc. of EMNLP*.
[26] Tarlow, D., Givoni, I. E., and Zemel, R. S. (2010). HOP-MAP: Efficient message passing with high order potentials. In *Proc. of AISTATS*.
[27] Wainwright, M. J. and Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference.* Now Publishers.