Journal of Machine Learning Research ? (????) ?-?? Submitted 9/12; Published ??/??

AD^3: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models*

André F. T. Martins ATM@PRIBERAM.PT
Priberam Labs, Lisboa, Portugal
and
Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal

Mário A. T. Figueiredo MTF@LX.IT.PT
Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal

Pedro M. Q. Aguiar AGUIAR@ISR.IST.UTL.PT
Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal

Noah A. Smith NASMITH@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh, USA

Editor: ???

Abstract

We present AD^3, a new algorithm for approximate *maximum a posteriori* (MAP) inference on factor graphs, based on the alternating directions method of multipliers. Like other dual decomposition algorithms, AD^3 has a modular architecture, where local subproblems are solved independently, and their solutions are gathered to compute a global update. The key characteristic of AD^3 is that each local subproblem has a quadratic regularizer, leading to faster convergence, both theoretically and in practice. We provide closed-form solutions for these AD^3 subproblems for binary pairwise factors and factors imposing first-order logic constraints. For arbitrary factors (large or combinatorial), we introduce an active set method which requires only an oracle for computing a local MAP configuration, making AD^3 applicable to a wide range of problems. Experiments on synthetic and real-world problems show that AD^3 compares favorably with the state-of-the-art.

Keywords: MAP inference, graphical models, dual decomposition, alternating directions method of multipliers.

1. Introduction

Graphical models enable compact representations of probability distributions, being widely used in natural language processing (NLP), computer vision, signal processing, and computational biology (Pearl, 1988; Lauritzen, 1996; Koller and Friedman, 2009). When using these models, a central problem is that of inferring the most probable (a.k.a. *maximum a posteriori* – MAP) configuration. Unfortunately, exact MAP inference is an intractable problem for many graphical models of interest in applications, such as those involving non-local features and/or structural constraints. This fact has motivated a significant research effort on approximate techniques.

∗. An earlier version of this work appeared in Martins et al. (2011a).

A class of methods that proved effective for approximate inference is based on linear programming relaxations of the MAP problem (LP-MAP; Schlesinger 1976). Several message-passing and dual decomposition algorithms have been proposed to address the resulting LP problems, taking advantage of the underlying graph structure (Wainwright et al., 2005; Kolmogorov, 2006; Werner, 2007; Komodakis et al., 2007; Globerson and Jaakkola, 2008; Jojic et al., 2010). All these algorithms have a similar consensus-based architecture: they repeatedly perform certain "local" operations in the graph (as outlined in Table 1), until some form of local agreement is achieved. The simplest example is the *projected subgradient dual decomposition* (PSDD) algorithm of Komodakis et al. (2007), which has recently enjoyed great success in NLP applications (see Rush and Collins 2012 and references therein). The major drawback of PSDD is that it is too slow to achieve consensus in large problems, requiring $O(1/\epsilon^2)$ iterations for an ϵ-accurate solution. While block coordinate descent schemes are usually faster to make progress (Globerson and Jaakkola, 2008), they may get stuck in suboptimal solutions, due to the non-smoothness of the dual objective function. Smoothing-based approaches (Jojic et al., 2010; Hazan and Shashua, 2010) do not have these drawbacks, but in turn they typically involve adjusting a "temperature" parameter for trading off the desired precision level and the speed of convergence, and may suffer from numerical instabilities in the near-zero temperature regime.

In this paper, we present a new LP-MAP algorithm called AD^3 (*alternating directions dual decomposition*), which allies the modularity of dual decomposition with the effectiveness of augmented Lagrangian optimization, via the *alternating directions method of multipliers* (Glowinski and Marroco, 1975; Gabay and Mercier, 1976). AD^3 has an iteration bound of $O(1/\epsilon)$, an order of magnitude better than the PSDD algorithm. Like PSDD, AD^3 alternates between a *broadcast* operation, where subproblems are assigned to local workers, and a *gather* operation, where the local solutions are assembled by a controller, which produces an estimate of the global solution. The key difference is that AD^3 regularizes their local subproblems toward these global estimate, which has the effect of speeding up consensus. In many cases of interest, there are closed-form solutions or efficient procedures for solving the AD^3 local subproblems (which are quadratic). For factors lacking such a solution, we introduce an *active set method* which requires only a local MAP decoder (the same requirement as in PSDD). This paves the way for using AD^3 with dense or structured factors.

Our main contributions are:

- We derive AD^3 and establish its convergence properties, blending classical and newer results about ADMM (Eckstein and Bertsekas, 1992; Boyd et al., 2011; Wang and Banerjee, 2012). We show that the algorithm has the same form as the PSDD method of Komodakis et al. (2007), with the local MAP subproblems replaced by quadratic programs. We also show that AD^3 can be wrapped into a branch-and-bound procedure to retrieve the *exact* MAP.

- We show that these AD^3 subproblems can be solved exactly and efficiently in many cases of interest, including Ising models and a wide range of hard factors representing arbitrary constraints in first-order logic. Up to a logarithmic term, the asymptotic cost is the same as that of passing messages or doing local MAP inference.

- For factors lacking a closed-form solution of the AD^3 subproblems, we introduce a new *active set method*. Remarkably, our method requires only a black box that returns local MAP configurations for each factor (the same requirement of the PSDD algorithm). This paves the way for using AD^3 with large dense or structured factors, based on off-the-shelf combinatorial algorithms (*e.g.*, Viterbi or Chu-Liu-Edmonds).

Algorithm	Local Operation
TRW-S (Wainwright et al., 2005; Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Table 1: Several LP-MAP inference algorithms and the kind of the local operations they need to perform at the factors to pass messages and compute beliefs. Some of these operations are the same as the classic loopy BP algorithm, which needs marginals (sum-product variant) or max-marginals (max-product variant). In Section 6, we will see that the quadratic problems (QP) required by AD^3 can be solved as a sequence of local MAP problems.

AD^3 was originally introduced by Martins et al. (2010, 2011a) (then called DD-ADMM). In addition to a considerably more detailed presentation, this paper contains contributions that substantially extend that preliminary work in several directions: the $O(1/\epsilon)$ rate of convergence, the active set method for general factors, and the branch-and-bound procedure for exact MAP inference. It also reports a wider set of experiments and the release of open-source code (available at `http://www.ark.cs.cmu.edu/AD3`), which may be useful to other researchers in the field.

This paper is organized as follows. We start by providing background material: MAP inference in graphical models and its LP-MAP relaxation (Section 2); the PSDD algorithm of Komodakis et al. (2007) (Section 3). In Section 4, we derive AD^3 and analyze its convergence. The AD^3 local subproblems are addressed in Section 5, where closed-form solutions are derived for Ising models and several structural constraint factors. In Section 6, we introduce an active set method to solve the AD^3 subproblems for arbitrary factors. Experiments with synthetic models, as well as in protein design and dependency parsing (Section 7) testify for the success of our approach. Finally, a discussion of related work in presented in Section 8, and Section 9 concludes the paper.

2. Background

2.1 Factor Graphs

Let $Y_1, \ldots, Y_M$ be random variables describing a structured output, with each Y_i taking values in a finite set $\mathcal{Y}_i$. We follow the common assumption in structured prediction that some of these variables have strong statistical dependencies. In this chapter, we use *factor graphs* (Tanner, 1981; Kschischang et al., 2001), a convenient way of representing such dependencies that captures directly the factorization assumptions in a model.

Definition 1 (Factor graph) *A factor graph is a bipartite graph $G := (V, F, E)$, comprised of:*

- *a set of* variable nodes $V := \{1, \ldots, M\}$*, corresponding to the variables $Y_1, \ldots, Y_M$;*
- *a set of* factor nodes F *(disjoint from V);*
- *a set of* edges $E \subseteq V \times F$ *linking variable nodes to factor nodes.*

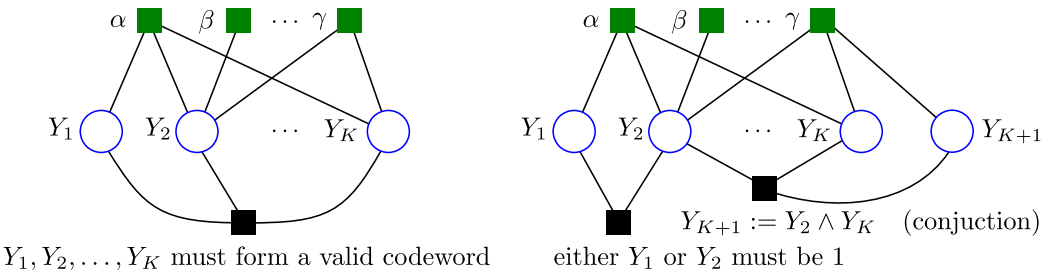

Figure 1: Constrained factor graphs, with soft factors shown as *green* squares above the variable nodes (circles) and hard constraint factors as *black* squares below the variable nodes. Left: a global factor that constrains the set of admissible outputs to a given codebook. Right: examples of declarative constraints; one of them is a factor connecting existing variables to an extra variable, allows scores depending on a logical functions of the former.

For notational convenience, we use Latin letters $(i, j, ...)$ and Greek letters $(\alpha, \beta, ...)$ to refer to variable and factor nodes, respectively. We denote by $\partial(\cdot)$ the *neighborhood set* of its node argument, whose cardinality is called the *degree* of the node. Formally, $\partial(i) := \{\alpha \in F \mid (i, \alpha) \in E\}$, for variable nodes, and $\partial(\alpha) := \{i \in V \mid (i, \alpha) \in E\}$ for factor nodes. We use the short notation $\boldsymbol{Y}_\alpha$ to refer to tuples of random variables, which take values on the product set $\mathcal{Y}_\alpha := \prod_{i \in \partial(\alpha)} \mathcal{Y}_i$.

We say that the joint probability distribution of $Y_1, \ldots, Y_M$ factors according to the factor graph $G = (V, F, E)$ if it can be written as

$$\mathbb{P}(Y_1 = y_1, \ldots, Y_M = y_M) \;\propto\; \exp\left(\sum_{i \in V} \boldsymbol{\theta}_i(y_i) + \sum_{\alpha \in F} \boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha)\right), \tag{1}$$

where $\boldsymbol{\theta}_i(\cdot)$ and $\boldsymbol{\theta}_\alpha(\cdot)$ are called, respectively, the *unary* and *higher-order* log-potential functions.[1] To accommodate hard constraints, we allow these functions to take values in $\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}$, but we require them to be *proper* (*i.e.*, they cannot take the value $-\infty$ in their whole domain). Figure 1 shows examples of factor graphs with hard constraint factors (to be studied in detail in Section 5.2).

2.2 MAP Inference

Given a probability distribution specified as in Eq. 1, we are interested in finding an assignment with maximal probability (the so-called *MAP assignment/configuration*):

$$\widehat{\boldsymbol{y}}_1, \ldots, \widehat{\boldsymbol{y}}_M \;\in\; \arg\max_{y_1, \ldots, y_M} \sum_{i \in V} \boldsymbol{\theta}_i(y_i) + \sum_{\alpha \in F} \boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha). \tag{2}$$

In fact, this problem is not specific to probabilistic models: other models, *e.g.*, trained to maximize margin, also lead to maximizations of the form above. Unfortunately, for a general factor graph

1. Some authors omit the unary log-potentials, which do not increase generality since they can be absorbed into the higher-order ones. We explicitly state them here since they are frequently used in practice, and their presence highlights a certain symmetry between potentials and marginal variables that will appear in the sequel.

G, this combinatorial problem is NP-hard (Koller and Friedman, 2009), so one must resort to approximations. In this paper, we address a class of approximations based on linear programming relaxations, described formally in the next section.

Throughout the paper, we will make the following assumption:

Assumption 2 *The MAP problem Eq. 2 is feasible,* i.e.*, there is at least one assignment* $y_1, \ldots, y_M$ *such that* $\sum_{\alpha \in F} \boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha) + \sum_{\alpha \in F} \boldsymbol{\theta}_i(y_i) > -\infty$.

Note that Assumption 2 is substantially weaker than other assumptions made in the literature on graphical models, which sometimes require the solution of to be unique, or the log-potentials to be all finite. We will see in Section 4 that this is all we need for AD^3 to be globally convergent.

2.3 LP-MAP Inference

Schlesinger's linear relaxation (Schlesinger, 1976; Werner, 2007) is the building block for many popular approximate MAP inference algorithms. Let us start by representing the log-potential functions in vector notation, $\boldsymbol{\theta}_i := (\boldsymbol{\theta}_i(y_i))_{y_i \in \mathcal{Y}_i} \in \bar{\mathbb{R}}^{|\mathcal{Y}_i|}$ and $\boldsymbol{\theta}_\alpha := (\boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha))_{\boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha} \in \bar{\mathbb{R}}^{|\mathcal{Y}_\alpha|}$. We introduce "local" probability distributions over the variables and factors, represented as vectors of the same size:

$$\boldsymbol{p}_i \in \Delta^{|\mathcal{Y}_i|}, \ \forall i \in V \quad \text{and} \quad \boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}, \ \forall \alpha \in F, \tag{3}$$

where $\Delta^K := \{\boldsymbol{u} \in \mathbb{R}^K \mid \boldsymbol{u} \geq \boldsymbol{0}, \ \boldsymbol{1}^\top \boldsymbol{u} = 1\}$ denotes the K-dimensional probability simplex. We stack these distributions into vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, with dimensions $P := \sum_{i \in V} |\mathcal{Y}_i|$ and $Q := \sum_{\alpha \in F} |\mathcal{Y}_\alpha|$, respectively. If these local probability distributions are "valid" marginal probabilities (*i.e.*, marginals realizable by some global probability distribution $\mathbb{P}(Y_1, \ldots, Y_M)$), then a necessary (but not sufficient) condition is that they are *locally consistent*. In other words, they must satisfy the following *calibration equations*:

$$\sum_{\boldsymbol{y}_\alpha \sim y_i} q_\alpha(\boldsymbol{y}_\alpha) = p_i(y_i), \qquad \forall y_i \in \mathcal{Y}_i, \ \forall (i, \alpha) \in E, \tag{4}$$

where the notation $\sim$ means that the summation is over all configurations $\boldsymbol{y}_\alpha$ whose ith element equals y_i. Eq. 4 can be written in vector notation as $\mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha = \boldsymbol{p}_i, \ \forall (i, \alpha) \in E$, where we define *consistency matrices*

$$\mathbf{M}_{i\alpha}(y_i, \boldsymbol{y}_\alpha) = \begin{cases} 1, & \text{if } \boldsymbol{y}_\alpha \sim y_i \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The set of locally consistent distributions forms the *local polytope*:

$$\mathcal{L}(G) = \left\{ (\boldsymbol{p}, \boldsymbol{q}) \in \mathbb{R}^{P+Q} \;\middle|\; \begin{array}{ll} \boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}, & \forall \alpha \in F \\ \mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha = \boldsymbol{p}_i, & \forall (i, \alpha) \in E \end{array} \right\}. \tag{6}$$

We consider the following linear program (the *LP-MAP inference problem*):

$$\boxed{\begin{array}{lll} \textbf{LP-MAP:} & \text{maximize} & \displaystyle\sum_{\alpha \in F} \boldsymbol{\theta}_\alpha{}^\top \boldsymbol{q}_\alpha + \sum_{i \in V} \boldsymbol{\theta}_i{}^\top \boldsymbol{p}_i \\ & \text{with respect to} & (\boldsymbol{p}, \boldsymbol{q}) \in \mathcal{L}(G). \end{array}} \tag{7}$$

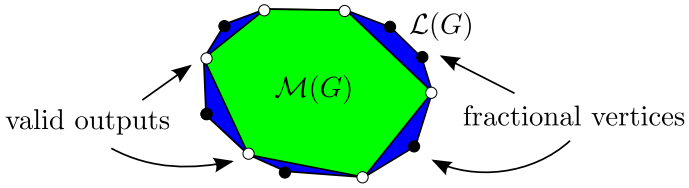

Figure 2: Marginal polytope (in green) and its outer aproximation, the local polytope (in blue). Each element of the marginal polytope corresponds to a joint distribution of $Y_1, \ldots, Y_M$, and each vertex corresponds to a configuration $\boldsymbol{y} \in \mathcal{Y}$, having coordinates in $\{0, 1\}$. The local polytope may have additional fractional vertices, with coordinates in $[0, 1]$.

If the solution $(\boldsymbol{p}^*, \boldsymbol{q}^*)$ of Eq. 7 happens to be integral, then each $\boldsymbol{p}_i^*$ and $\boldsymbol{q}_\alpha^*$ will be at corners of the simplex, *i.e.*, they will be indicator vectors of local configurations y_i^* and $\boldsymbol{y}_\alpha^*$, in which case the output $(y_i^*)_{i \in V}$ is guaranteed to be a solution of the MAP decoding problem (2). Under certain conditions—for example, when the factor graph G does not have cycles—Eq. 7 is guaranteed to have integral solutions. In general, however, the LP-MAP decoding problem (7) is a relaxation of Eq. 2. Geometrically, $\mathcal{L}(G)$ is an outer approximation of the *marginal polytope*, defined as the set of valid marginals (Wainwright and Jordan, 2008). This is illustrated in Figure 2.

2.4 LP-MAP Inference Algorithms

While any off-the-shelf LP solver can be used for solving Eq. 7, specialized algorithms have been designed to exploit the graph structure, achieving superior performance on several benchmarks (Yanover et al., 2006). Some of these algorithms are listed in Table 1. Most of these specialized algorithms belong to two classes: block (dual) coordinate descent, which take the form of *message-passing* algorithms, and projected subgradient algorithms, based on *dual decomposition*.

Block coordinate descent methods address the dual of Eq. 7 by alternately optimizing over blocks of coordinates. Examples are max-sum diffusion (Kovalevsky and Koval, 1975; Werner, 2007); max-product sequential tree-reweighted belief propagation (TRW-S, Wainwright et al. 2005; Kolmogorov 2006); and the max-product linear programming algorithm (MPLP; Globerson and Jaakkola 2008). These algorithms work by passing local messages (that require computing *max-marginals*) between factors and variables. Under certain conditions (more stringent than Assumption 2), one may obtain optimality certificates when the relaxation is tight. A disadvantage of coordinate descent algorithms is that they may get stuck at stationary suboptimal solutions, since the objective is non-smooth (Bertsekas et al. 1999, Section 6.3.4). An alternative is to optimize the dual with the projected subgradient method, which is globally convergent (Komodakis et al., 2007), and which requires computing *local MAP configurations* as its subproblems. Finally, smoothing-based approaches, such as the accelerated dual decomposition method of Jojic et al. (2010) and the norm-product algorithm of Hazan and Shashua (2010), smooth the dual objective with an entropic regularization term, leading to subproblems that involve computing *local marginals*.

In Section 8, we discuss advantages and disadvantages of these and other LP-MAP inference methods with respect to AD^3.

3. Dual Decomposition with the Projected Subgradient Algorithm

We now describe the *projected subgradient dual decomposition* (PSDD) algorithm proposed by Komodakis et al. (2007). As we will see in Section 4, there is a strong affinity between PSDD and the main focus of this paper, AD^3.

Let us first reparametrize Eq 7 to express it as a consensus problem. For each edge $(i, \alpha) \in E$, we define a potential function $\boldsymbol{\theta}_{i\alpha} := (\boldsymbol{\theta}_{i\alpha}(y_i))_{y_i \in \mathcal{Y}_i}$ that satisfies $\sum_{\alpha \in \partial(i)} \boldsymbol{\theta}_{i\alpha} = \boldsymbol{\theta}_i$; a trivial choice is $\boldsymbol{\theta}_{i\alpha} = |\partial(i)|^{-1}\boldsymbol{\theta}_i$, which spreads the unary potentials evenly across the factors. Since we have a equality constraint $\boldsymbol{p}_i = \mathbf{M}_{i\alpha}\boldsymbol{q}_\alpha$, Eq. 7 is equivalent to the following *primal formulation*:

$$\boxed{\begin{array}{lrl} \textbf{LP-MAP-P:} & \text{maximize} & \displaystyle\sum_{\alpha \in F} \left(\boldsymbol{\theta}_\alpha + \sum_{i \in \partial(\alpha)} \mathbf{M}_{i\alpha}^\top \boldsymbol{\theta}_{i\alpha} \right)^\top \boldsymbol{q}_\alpha \\ & \text{with respect to} & \boldsymbol{p} \in \mathbb{R}^P, \quad \boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}, \forall \alpha \in F, \\ & \text{subject to} & \mathbf{M}_{i\alpha}\boldsymbol{q}_\alpha = \boldsymbol{p}_i, \ \forall (i,\alpha) \in E. \end{array}} \tag{8}$$

Note that, although the $\boldsymbol{p}$-variables do not appear in the objective of Eq. 8, they play a fundamental role through the constraints in the last line, which are necessary to ensure that the marginals encoded in the $\boldsymbol{q}$-variables are consistent on their overlaps. Indeed, it is this set of constraints that complicate the optimization problem, which would otherwise be separable into independent subproblems, one per factor. Introducing Lagrange multipliers $\boldsymbol{\lambda}_{i\alpha} := (\boldsymbol{\lambda}_{i\alpha}(y_i))_{y_i \in \mathcal{Y}_i}$ for each of these equality constraints leads to the Lagrangian function

$$L(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}) \ = \ \sum_{\alpha \in F} \left(\boldsymbol{\theta}_\alpha + \sum_{i \in \partial(\alpha)} \mathbf{M}_{i\alpha}^\top (\boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}) \right)^\top \boldsymbol{q}_\alpha - \sum_{(i,\alpha) \in E} \boldsymbol{\lambda}_{i\alpha}{}^\top \boldsymbol{p}_i, \tag{9}$$

the maximization of which w.r.t. $\boldsymbol{q}$ and $\boldsymbol{p}$ will yield the (Lagrangian) dual objective. Since the $\boldsymbol{p}$-variables are unconstrained, we have

$$\max_{\boldsymbol{q}, \boldsymbol{p}} L(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}) = \begin{cases} g(\boldsymbol{\lambda}) & \text{if } \boldsymbol{\lambda} \in \Lambda, \\ +\infty & \text{otherwise,} \end{cases} \tag{10}$$

and we arrive at the following *dual formulation*:

$$\boxed{\begin{array}{lll} \textbf{LP-MAP-D:} & \text{minimize} & g(\boldsymbol{\lambda}) := \displaystyle\sum_{\alpha \in F} g_\alpha(\boldsymbol{\lambda}) \\ & \text{with respect to} & \boldsymbol{\lambda} \in \Lambda, \end{array}} \tag{11}$$

where $\Lambda := \left\{ \boldsymbol{\lambda} \mid \sum_{\alpha \in \partial(i)} \boldsymbol{\lambda}_{i\alpha} = \mathbf{0}, \ \forall i \in V \right\}$ is a linear subspace, and each $g_\alpha(\boldsymbol{\lambda})$ is the solution of a *local subproblem*:

$$\begin{aligned} g_\alpha(\boldsymbol{\lambda}) \ &:= \ \max_{\boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}} \left(\boldsymbol{\theta}_\alpha + \sum_{i \in \partial(\alpha)} \mathbf{M}_{i\alpha}^\top (\boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}) \right)^\top \boldsymbol{q}_\alpha \\ &= \ \max_{\boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha} \left(\boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha) + \sum_{i \in \partial(\alpha)} (\boldsymbol{\theta}_{i\alpha}(y_i) + \boldsymbol{\lambda}_{i\alpha}(y_i)) \right); \end{aligned} \tag{12}$$

Algorithm 1 PSDD Algorithm (Komodakis et al., 2007)

1: **input:** graph G, parameters $\boldsymbol{\theta}$, maximum number of iterations T, stepsizes $(\eta_t)_{t=1}^T$
2: for each $(i,\alpha) \in E$, choose $\boldsymbol{\theta}_{i\alpha}$ such that $\sum_{\alpha\in\partial(i)} \boldsymbol{\theta}_{i\alpha} = \boldsymbol{\theta}_i$
3: initialize $\boldsymbol{\lambda} = \mathbf{0}$
4: **for** $t = 1$ **to** T **do**
5: **for each** factor $\alpha \in F$ **do**
6: set unary log-potentials $\boldsymbol{\xi}_{i\alpha} := \boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}$, for $i \in \partial(\alpha)$
7: set $\widehat{\boldsymbol{q}}_\alpha := \textsc{ComputeMap}(\boldsymbol{\theta}_\alpha + \sum_{i\in\partial(\alpha)} \mathbf{M}_{i\alpha}^\top \boldsymbol{\xi}_{i\alpha})$
8: set $\widehat{\boldsymbol{q}}_{i\alpha} := \mathbf{M}_{i\alpha}\widehat{\boldsymbol{q}}_\alpha$, for $i \in \partial(\alpha)$
9: **end for**
10: compute average $\boldsymbol{p}_i := |\partial(i)|^{-1} \sum_{\alpha\in\partial(i)} \widehat{\boldsymbol{q}}_{i\alpha}$ for each $i \in V$
11: update $\boldsymbol{\lambda}_{i\alpha} := \boldsymbol{\lambda}_{i\alpha} - \eta_t\,(\widehat{\boldsymbol{q}}_{i\alpha} - \boldsymbol{p}_i)$ for each $(i,\alpha) \in E$
12: **end for**
13: **output:** dual variable $\boldsymbol{\lambda}$ and upper bound $g(\boldsymbol{\lambda})$

the last equality is justified by the fact that maximizing a linear objective over the probability simplex gives the largest component of the score vector. Note that the local subproblem (12) can be solved by a COMPUTEMAP procedure, which receives unary potentials $\boldsymbol{\xi}_{i\alpha}(y_i) := \boldsymbol{\theta}_{i\alpha}(y_i) + \boldsymbol{\lambda}_{i\alpha}(y_i)$ and factor potentials $\boldsymbol{\theta}_\alpha(\boldsymbol{y}_\alpha)$ (eventually structured) and returns the MAP $\widehat{\boldsymbol{y}}_\alpha$.

Problem (11) is often referred to as the *master* or *controller*, and each local subproblem (12) as a *slave* or *worker*. The master problem (11) can be solved with a *projected subgradient algorithm*.[2] By Danskin's rule (Bertsekas et al., 1999, p. 717), a subgradient of g_α is readily given by

$$\frac{\partial g_\alpha(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}_{i\alpha}} = \mathbf{M}_{i\alpha}\widehat{\boldsymbol{q}}_\alpha, \quad \forall (i,\alpha) \in E; \tag{13}$$

and the projection onto Λ amounts to a centering operation. Putting these pieces together yields Algorithm 1. At each iteration, the algorithm broadcasts the current Lagrange multipliers to all the factors. Each factor adjusts its internal unary log-potentials (line 6) and invokes the COMPUTEMAP procedure (line 7).[3] The solutions achieved by each factor are then gathered and averaged (line 10), and the Lagrange multipliers are updated with step size η_t (line 11). The two following propositions establish the convergence properties of Algorithm 1.

Proposition 3 (Convergence rate) *If the non-negative step size sequence $(\eta_t)_{t\in\mathbb{N}}$ is diminishing and nonsummable ($\lim \eta_t = 0$ and $\sum_{t=1}^\infty \eta_t = \infty$), then Algorithm 1 converges to the solution $\boldsymbol{\lambda}^*$ of LP-MAP-D* (11)*. Furthermore, after $T = O(1/\epsilon^2)$ iterations, we have $g(\boldsymbol{\lambda}^{(T)}) - g(\boldsymbol{\lambda}^*) \leq \epsilon$.*

Proof: This is a property of projected subgradient algorithms (see, *e.g.*, Bertsekas et al. 1999). ■

Proposition 4 (Certificate of optimality) *If, at some iteration of Algorithm 1, all the local subproblems are in agreement (*i.e.*, if $\widehat{\boldsymbol{q}}_{i\alpha} = \boldsymbol{p}_i$ after line 10, for all $i \in V$), then: (i) $\boldsymbol{\lambda}$ is a solution of LP-MAP-D* (11)*; (ii) $\boldsymbol{p}$ is binary-valued and a solution of both LP-MAP-P and MAP.*

2. A slightly different formulation is presented by Sontag et al. (2011) which yields a subgradient algorithm with no projection.
3. Note that, if the factor log-potentials $\boldsymbol{\theta}_\alpha$ have special structure (*e.g.*, if the factor is itself combinatorial, such as a sequence or a tree model), then this structure is preserved since only the internal unary log-potentials are changed. Therefore, if evaluating $\textsc{ComputeMap}(\boldsymbol{\theta}_\alpha)$ is tractable, so is evaluating $\textsc{ComputeMap}(\boldsymbol{\theta}_\alpha + \sum_{i\in\partial(\alpha)} \mathbf{M}_{i\alpha}^\top \boldsymbol{\xi}_{i\alpha})$.

Proof: If all local subproblems are in agreement, then a vacuous update will occur in line 11, and no further changes will occur. Since the algorithm is guaranteed to converge, the current $\boldsymbol{\lambda}$ is optimal. Also, if all local subproblems are in agreement, the averaging in line 10 necessarily yields a binary vector $\boldsymbol{p}$. Since any binary solution of LP-MAP is also a solution of MAP, the result follows. ■

Propositions 3–4 imply that, if the LP-MAP relaxation is tight, then Algorithm 1 will eventually yield the exact MAP configuration along with a certificate of optimality. According to Proposition 3, even if the relaxation is not tight, Algorithm 1 still converges to a solution of LP-MAP. Unfortunately, in large graphs with many overlapping factors, it has been observed that convergence can be quite slow in practice (Martins et al., 2011b). This is not surprising, given that it attempts to reach a consensus among all overlapping components; the larger this number, the harder it is to achieve consensus. We describe in the next section another LP-MAP decoder (AD^3) with a faster convergence rate.

4. Alternating Directions Dual Decomposition (AD^3)

AD^3 avoids some of the weaknesses of PSDD by replacing the subgradient method with the *alternating directions method of multipliers* (ADMM). Before going into a formal derivation, let us go back to the PSDD algorithm to pinpoint the crux of their weaknesses. It resides on two aspects:

1. The dual objective function $g(\boldsymbol{\lambda})$ is *non-smooth*, this being why "subgradients" are used instead of "gradients." It is well-known that non-smooth optimization lacks some of the good properties of its smooth counterpart. Namely, there is no guarantee of monotonic improvement in the objective (see Bertsekas et al. 1999, p. 611). Ensuring convergence requires using a diminishing step size sequence, which leads to slow convergence rates. In fact, as stated in Proposition 3, $O(1/\epsilon^2)$ iterations are required to guarantee ϵ-accuracy.

2. A close look at Algorithm 1 reveals that the consensus is promoted solely by the Lagrange multipliers (line 6). These can be regarded as "price adjustments" that are made at each iteration and lead to a reallocation of resources. However, no "memory" exists about past allocations or adjustments, so the workers never know how far they are from consensus. One may suspect that a smarter use of these quantities may accelerate convergence.

The first of these aspects has been addressed by the accelerated dual decomposition method of Jojic et al. (2010), which improves the iteration bound to $O(1/\epsilon)$; we discuss that work further in Section 8. We will see that AD^3 also yields a $O(1/\epsilon)$ iteration bound with some additional advantages. The second aspect is addressed by AD^3 by broadcasting *the current global solution* in addition to the Lagrange multipliers, allowing the workers to regularize their subproblems toward that solution.

4.1 Augmented Lagrangians and the Alternating Directions Method of Multipliers

Let us start with a brief overview of augmented Lagrangian methods. Consider the following general convex optimization problem with equality constraints:

$$\begin{array}{ll} \text{maximize} & f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) \\ \text{with respect to} & \boldsymbol{q} \in \mathcal{Q}, \boldsymbol{p} \in \mathcal{P} \\ \text{subject to} & \mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} = \boldsymbol{c}, \end{array} \tag{14}$$

where $\mathcal{Q} \subseteq \mathbb{R}^P$ and $\mathcal{P} \subseteq \mathbb{R}^Q$ are convex sets and $f_1 : \mathcal{Q} \to \bar{\mathbb{R}}$ and $f_2 : \mathcal{P} \to \bar{\mathbb{R}}$ are concave functions. Note that the LP-MAP problem stated in Eq. 8 has this form. For any $\eta \geq 0$, consider the problem

$$\begin{array}{ll} \text{maximize} & f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) - \frac{\eta}{2}\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}\|^2 \\ \text{with respect to} & \boldsymbol{q} \in \mathcal{Q}, \boldsymbol{p} \in \mathcal{P} \\ \text{subject to} & \mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} = \boldsymbol{c}, \end{array} \tag{15}$$

which differs from (14) in the extra term penalizing violations of the equality constraints; since this term vanishes at feasibility, the two problems have the same solution. The Lagrangian of (15),

$$L_\eta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}) = f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) + \boldsymbol{\lambda}^\top(\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}) - \frac{\eta}{2}\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}\|^2, \tag{16}$$

is called the *η-augmented Lagrangian* of Eq. 14. The so-called *augmented Lagrangian* methods (Bertsekas et al., 1999, Section 4.2) address problem (14) by seeking a saddle point of L_{η_t}, for some sequence $(\eta_t)_{t\in\mathbb{N}}$. The earliest instance is the *method of multipliers* (Hestenes, 1969; Powell, 1969), which alternates between a joint update of $\boldsymbol{q}$ and $\boldsymbol{p}$ through

$$(\boldsymbol{q}^{t+1}, \boldsymbol{p}^{t+1}) \quad := \quad \arg\max_{\boldsymbol{q},\boldsymbol{p}} \{L_{\eta_t}(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}^t) \mid \boldsymbol{q} \in \mathcal{Q}, \boldsymbol{p} \in \mathcal{P}\} \tag{17}$$

and a gradient update of the Lagrange multiplier,

$$\boldsymbol{\lambda}^{t+1} \quad := \quad \boldsymbol{\lambda}^t - \eta_t(\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c}). \tag{18}$$

Under some conditions, this method is convergent, and even superlinear, if the sequence $(\eta_t)_{t\in\mathbb{N}}$ is increasing (Bertsekas et al. 1999, Section 4.2). A shortcoming of this method is that problem (17) may be difficult, since the penalty term of the augmented Lagrangian couples the variables $\boldsymbol{p}$ and $\boldsymbol{q}$. The *alternating directions method of multipliers* (ADMM) avoids this shortcoming, by replacing the joint optimization (17) by a single block Gauss-Seidel-type step:

$$\boldsymbol{q}^{t+1} := \arg\max_{\boldsymbol{q}\in\mathcal{Q}} L_{\eta_t}(\boldsymbol{q}, \boldsymbol{p}^t, \boldsymbol{\lambda}^t) = \arg\max_{\boldsymbol{q}\in\mathcal{Q}} f_1(\boldsymbol{q}) + (\mathbf{A}^\top\boldsymbol{\lambda}^t)^\top\boldsymbol{q} - \frac{\eta_t}{2}\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^t - \boldsymbol{c}\|^2, \tag{19}$$

$$\boldsymbol{p}^{t+1} := \arg\max_{\boldsymbol{p}\in\mathcal{P}} L_{\eta_t}(\boldsymbol{q}^{t+1}, \boldsymbol{p}, \boldsymbol{\lambda}^t) = \arg\max_{\boldsymbol{p}\in\mathcal{P}} f_2(\boldsymbol{p}) + (\mathbf{B}^\top\boldsymbol{\lambda}^t)^\top\boldsymbol{p} - \frac{\eta_t}{2}\|\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}\|^2. \tag{20}$$

In general, problems (19)–(20) are simpler than the joint maximization in Eq. 17. ADMM was proposed by Glowinski and Marroco (1975) and Gabay and Mercier (1976) and is related to other optimization methods, such as Douglas-Rachford splitting (Eckstein and Bertsekas, 1992) and proximal point methods (see Boyd et al. 2011 for an historical overview).

4.2 Derivation of AD^3

Our LP-MAP-P problem (8) can be cast into the form (14) by proceeding as follows:

- let $\mathcal{Q}$ in Eq. 14 be the Cartesian product of simplices, $\mathcal{Q} := \prod_{\alpha\in F} \Delta^{|\mathcal{Y}_\alpha|}$, and $\mathcal{P} := \mathbb{R}^P$;
- let $f_1(\boldsymbol{q}) := \sum_{\alpha\in F} \left(\boldsymbol{\theta}_\alpha + \sum_{i\in\partial(\alpha)} \mathbf{M}_{i\alpha}^\top \boldsymbol{\theta}_{i\alpha}\right)^\top \boldsymbol{q}_\alpha$ and $f_2 := \equiv 0$;
- let $\mathbf{A}$ in Eq. 14 be a $R \times Q$ block-diagonal matrix, where $R = \sum_{(i,\alpha)\in E} |\mathcal{Y}_i|$, with one block per factor, which is a vertical concatenation of the matrices $\{\mathbf{M}_{i\alpha}\}_{i\in\partial(\alpha)}$;

- let $-\mathbf{B}$ be a $R \times P$ matrix of grid-structured blocks, where the block in the (i, α)th row and the ith column is a negative identity matrix of size $|\mathcal{Y}_i|$, and all the other blocks are zero;
- let $\boldsymbol{c} := 0$.

The η-augmented Lagrangian associated with (8) is

$$L_\eta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}) = \sum_{\alpha \in F} \left(\boldsymbol{\theta}_\alpha + \sum_{i \in \partial(\alpha)} \mathbf{M}_{i\alpha}^\top (\boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}) \right)^\top \boldsymbol{q}_\alpha - \sum_{(i,\alpha) \in E} \boldsymbol{\lambda}_{i\alpha}{}^\top \boldsymbol{p}_i - \frac{\eta}{2} \sum_{(i,\alpha) \in E} \|\mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha - \boldsymbol{p}_i\|^2. \tag{21}$$

This is the standard Lagrangian (9) plus the Euclidean penalty term. The ADMM updates are

$$\textbf{Broadcast:} \quad \boldsymbol{q}^{(t)} := \arg\max_{\boldsymbol{q} \in \mathcal{Q}} L_{\eta_t}(\boldsymbol{q}, \boldsymbol{p}^{(t-1)}, \boldsymbol{\lambda}^{(t-1)}), \tag{22}$$

$$\textbf{Gather:} \quad \boldsymbol{p}^{(t)} := \arg\max_{\boldsymbol{p} \in \mathbb{R}^P} L_{\eta_t}(\boldsymbol{q}^{(t)}, \boldsymbol{p}, \boldsymbol{\lambda}^{(t-1)}), \tag{23}$$

$$\textbf{Multiplier update:} \quad \boldsymbol{\lambda}_{i\alpha}^{(t)} := \boldsymbol{\lambda}_{i\alpha}^{(t-1)} - \eta_t \left(\mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha^{(t)} - \boldsymbol{p}_i^{(t)} \right), \forall (i, \alpha) \in E. \tag{24}$$

We next analyze the broadcast and gather steps, and prove a proposition about the multiplier update.

Broadcast step. The maximization (22) can be carried out in parallel at the factors, as in PSDD. The only difference is that, instead of a local MAP computation, each worker now needs to solve a *quadratic program* of the form:

$$\boxed{\max_{\boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}} \left(\boldsymbol{\theta}_\alpha + \sum_{i \in \partial(\alpha)} \mathbf{M}_{i\alpha}^\top (\boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}) \right)^\top \boldsymbol{q}_\alpha - \frac{\eta}{2} \sum_{i \in \partial(\alpha)} \|\mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha - \boldsymbol{p}_i\|^2.} \tag{25}$$

This differs from the linear subproblem (12) of PSDD by the inclusion of an Euclidean penalty term, which penalizes deviations from the global consensus. In Sections 5 and 6, we will give procedures to solve these local subproblems.

Gather step. The solution of Eq. 23 has a closed form. Indeed, this problem is separable into independent optimizations, one for each $i \in V$; defining $\boldsymbol{q}_{i\alpha} := \mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha$,

$$\begin{aligned} \boldsymbol{p}_i^{(t)} &:= \arg\min_{\boldsymbol{p}_i \in \mathbb{R}^{|\mathcal{Y}_i|}} \sum_{\alpha \in \partial(i)} \left(\boldsymbol{p}_i - \left(\boldsymbol{q}_{i\alpha} - \eta_t^{-1} \boldsymbol{\lambda}_{i\alpha} \right) \right)^2 \\ &= |\partial(i)|^{-1} \sum_{\alpha \in \partial(i)} \left(\boldsymbol{q}_{i\alpha} - \eta_t^{-1} \boldsymbol{\lambda}_{i\alpha} \right) \\ &= \frac{1}{|\partial(i)|} \sum_{\alpha \in \partial(i)} \boldsymbol{q}_{i\alpha}. \end{aligned} \tag{26}$$

The equality in the last line is due to the following proposition:

Proposition 5 *The sequence $\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}, \ldots$ produced by the updates* (22)–(24) *is dual feasible, i.e., we have $\boldsymbol{\lambda}^{(t)} \in \Lambda$ for every t, with Λ as in Eq. 11.*

Algorithm 2 Alternating Directions Dual Decomposition (AD^3)

1: **input:** graph G, parameters $\boldsymbol{\theta}$, penalty constant η
2: initialize $\boldsymbol{p}$ uniformly (*i.e.*, $p_i(y_i) = 1/|\mathcal{Y}_i|, \forall i \in V, y_i \in \mathcal{Y}_i$)
3: initialize $\boldsymbol{\lambda} = \mathbf{0}$
4: **repeat**
5: **for each** factor $\alpha \in F$ **do**
6: set unary log-potentials $\boldsymbol{\xi}_{i\alpha} := \boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha}$, for $i \in \partial(\alpha)$
7: set $\widehat{\boldsymbol{q}}_\alpha := \textsc{SolveQP}\left(\boldsymbol{\theta}_\alpha + \sum_{i\in\partial(\alpha)} \mathbf{M}_{i\alpha}^\top \boldsymbol{\xi}_{i\alpha},\ (\boldsymbol{p}_i)_{i\in\partial(\alpha)}\right)$
8: set $\widehat{\boldsymbol{q}}_{i\alpha} := \mathbf{M}_{i\alpha}\widehat{\boldsymbol{q}}_\alpha$, for $i \in \partial(\alpha)$
9: **end for**
10: compute average $\boldsymbol{p}_i := |\partial(i)|^{-1} \sum_{\alpha\in\partial(i)} \widehat{\boldsymbol{q}}_{i\alpha}$ for each $i \in V$
11: update $\boldsymbol{\lambda}_{i\alpha} := \boldsymbol{\lambda}_{i\alpha} - \eta\,(\widehat{\boldsymbol{q}}_{i\alpha} - \boldsymbol{p}_i)$ for each $(i,\alpha) \in E$
12: **until** convergence
13: **output:** primal variables $\boldsymbol{p}$ and $\boldsymbol{q}$, dual variable $\boldsymbol{\lambda}$, upper bound $g(\boldsymbol{\lambda})$

Proof: We have:

$$\begin{aligned}\sum_{\alpha\in\partial(i)} \boldsymbol{\lambda}_{i\alpha}^{(t)} &= \sum_{\alpha\in\partial(i)} \boldsymbol{\lambda}_{i\alpha}^{(t-1)} - \eta_t \left(\sum_{\alpha\in\partial(i)} \boldsymbol{q}_{i\alpha}^{(t)} - |\partial(i)|\boldsymbol{p}_i^{(t)}\right) \\ &= \sum_{\alpha\in\partial(i)} \boldsymbol{\lambda}_{i\alpha}^{(t-1)} - \eta_t \left(\sum_{\alpha\in\partial(i)} \boldsymbol{q}_{i\alpha}^{(t)} - \sum_{\alpha\in\partial(i)} \left(\boldsymbol{q}_{i\alpha}^{(t)} - \eta_t^{-1}\boldsymbol{\lambda}_{i\alpha}^{(t-1)}\right)\right) = \mathbf{0}. \qquad (27)\end{aligned}$$

■

Assembling all these pieces together leads to AD^3 (Algorithm 2), where we use a fixed stepsize η. Notice that AD^3 retains the modular structure of PSDD (Algorithm 1). The key difference is that AD^3 also broadcasts the current global solution to the workers, allowing them to regularize their subproblems toward that solution, thus speeding up the consensus. This is embodied in the procedure SOLVEQP (line 7), which replaces COMPUTEMAP of Algorithm 1.

4.3 Convergence Analysis

Before proving the convergence of AD^3, we start with a basic result.

Proposition 6 (Existence of a Saddle Point) *Under Assumption 2, we have the following properties (regardless of the choice of log-potentials):*

1. *LP-MAP-P* (8) *is primal-feasible;*
2. *LP-MAP-D* (11) *is dual-feasible;*
3. *The Lagrangian function $L(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda})$ has a saddle point $(\boldsymbol{q}^*, \boldsymbol{p}^*, \boldsymbol{\lambda}^*) \in \mathcal{Q} \times \mathcal{P} \times \Lambda$, where $(\boldsymbol{q}^*, \boldsymbol{p}^*)$ is a solution of LP-MAP-P and $\boldsymbol{\lambda}^*$ is a solution of LP-MAP-D.*

Proof: Property 1 follows directly from Assumption 2 and the fact that LP-MAP is a relaxation of MAP. To prove properties 2–3, define first the set of structural constraints $\bar{\mathcal{Q}} := \prod_{\alpha \in F} \bar{\mathcal{Q}}_\alpha$, where $\bar{\mathcal{Q}}_\alpha := \{\boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|} \mid \boldsymbol{q}_\alpha(\boldsymbol{y}_\alpha) = 0, \forall \boldsymbol{y}_\alpha \text{ s.t. } \theta_\alpha(\boldsymbol{y}_\alpha) = -\infty\}$ are truncated probability simplices (hence convex). Since all log-potential functions are proper (due to Assumption 2), we have that each $\bar{\mathcal{Q}}_\alpha$ is non-empty, and therefore $\bar{\mathcal{Q}}$ has non-empty relative interior. As a consequence, the refined Slater's condition (Boyd and Vandenberghe, 2004, §5.2.3) holds; let $(q^*, p^*) \in \bar{\mathcal{Q}} \times \mathcal{P}$ be a primal feasible solution of LP-MAP-P, which exists by virtue of property 1. Then, the KKT optimality conditions imply the existence of a $\boldsymbol{\lambda}^*$ such that $(\boldsymbol{q}^*, \boldsymbol{p}^*, \boldsymbol{\lambda}^*)$ is a saddle point of the Lagrangian function L, *i.e.*, $L(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{q}^*, \boldsymbol{p}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{q}^*, \boldsymbol{p}^*, \boldsymbol{\lambda})$ holds for all $\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{\lambda}$. Naturally, we must have $\boldsymbol{\lambda}^* \in \Lambda$, otherwise $L(., ., \boldsymbol{\lambda}^*)$ would be unbounded with respect to $\boldsymbol{p}$. ■

We are now ready to show the convergence of AD^3, which follows directly from the general convergence properties of ADMM. Remarkably, unlike in PSDD, convergence is ensured with a fixed step size, therefore no annealing is required.

Proposition 7 (Convergence of AD^3) *Let $(\boldsymbol{q}^{(t)}, \boldsymbol{p}^{(t)}, \boldsymbol{\lambda}^{(t)})_t$ be the sequence of iterates produced by Algorithm 2 with a fixed $\eta_t = \eta$. Then the following holds:*

1. *primal feasibility of LP-MAP-P* (8) *is achieved in the limit,* i.e.*,*
$$\|\mathbf{M}_{i\alpha} \boldsymbol{q}_\alpha^{(t)} - \boldsymbol{p}_i^{(t)}\| \to \mathbf{0}, \quad \forall (i, \alpha) \in E; \tag{28}$$
2. *the primal objective sequence $\left(\sum_{i \in V} {\boldsymbol{\theta}_i}^\top \boldsymbol{p}_i^{(t)} + \sum_{\alpha \in F} {\boldsymbol{\theta}_\alpha}^\top \boldsymbol{q}_\alpha^{(t)}\right)_{t \in \mathbb{N}}$ converges to the solution of LP-MAP-P* (8)*;*
3. *the dual sequence $(\boldsymbol{\lambda}^{(t)})_{t \in \mathbb{N}}$ converges to a solution of the dual LP-MAP-D* (11)*; moreover, this sequence is dual feasible,* i.e.*, it is contained in Λ. Thus, $g(\boldsymbol{\lambda}^{(t)})$ in* (11) *approaches the optimum from above.*

Proof: See Boyd et al. (2011, Appendix A) for a simple proof of the convergence of ADMM in the form Eq. 14, from which 1, 2, and the first part of 3 follow immediately. The two assumptions stated in Boyd et al. (2011, p.16) are met: denoting by $\iota_{\mathcal{Q}}$ the indicator function of the set $\mathcal{Q}$, which evaluates to zero in $\mathcal{Q}$ and to $-\infty$ outside $\mathcal{Q}$, we have that functions $f_1 + \iota_{\mathcal{Q}}$ and f_2 are closed proper convex (since the log-potential functions are proper and f_1 is closed proper convex), and the unaugmented Lagrangian has a saddle point (see property 3 in Proposition 6). Finally, the last part of statement 3 follows from Proposition 5. ■

The next proposition states the $O(1/\epsilon)$ iteration bound of AD^3, which is better than the $O(1/\epsilon^2)$ bound of PSDD.

Proposition 8 (Convergence rate of AD^3) *Assume the conditions of Proposition 7. Let $\boldsymbol{\lambda}^*$ be a solution of the dual problem* (11)*, $\bar{\boldsymbol{\lambda}}_T := \frac{1}{T} \sum_{t=1}^T \boldsymbol{\lambda}^{(t)}$ be the "averaged" Lagrange multipliers after T iterations of AD^3, and $g(\bar{\boldsymbol{\lambda}}_T)$ the corresponding estimate of the dual objective (an upper bound). Then, $g(\bar{\boldsymbol{\lambda}}_T) - g(\boldsymbol{\lambda}^*) \leq \epsilon$ after $T \leq C/\epsilon$ iterations, where C is a constant satisfying*

$$\begin{aligned} C &\leq \frac{5\eta}{2} \sum_{i \in V} |\partial(i)| \times (1 - |\mathcal{Y}_i|^{-1}) + \frac{5}{2\eta} \|\boldsymbol{\lambda}^*\|^2 \\ &\leq \frac{5\eta}{2} |E| + \frac{5}{2\eta} \|\boldsymbol{\lambda}^*\|^2. \end{aligned} \tag{29}$$

Proof: The proof (detailed in Appendix A) uses recent results of He and Yuan (2011) and Wang and Banerjee (2012), concerning convergence of ADMM in a variational inequality setting. ■

As expected, the bound (29) increases with the number of overlapping variables, quantified by the number of edges $|E|$, and the magnitude of the optimal dual vector $\boldsymbol{\lambda}^*$. Note that if there is a good estimate of $\|\boldsymbol{\lambda}^*\|$, then Eq. 29 can be used to choose a step size η that minimizes the bound—the optimal stepsize is $\eta = \|\boldsymbol{\lambda}^*\| \times |E|^{-1/2}$, which would lead to $T \leq 5\epsilon^{-1}|E|^{1/2}$. In fact, although Proposition 7 guarantees convergence for any choice of η, this parameter may strongly impact the behavior of the algorithm. In our experiments, we dynamically adjust η in earlier iterations using the heuristic described in Boyd et al. (2011, Section 3.4.1), and freeze it afterwards, not to compromise convergence.

4.4 Stopping Conditions and Implementation Details

4.4.1 PRIMAL AND DUAL RESIDUALS

Since the AD^3 iterates are dual feasible, it is also possible to check the conditions in Proposition 4 to obtain optimality certificates, as in PSDD. Moreover, even when the LP-MAP relaxation is not tight, AD^3 can provide stopping conditions by keeping track of primal and dual residuals, as described in Boyd et al. (2011, §3.3), based on which it is possible to obtain certificates, not only for the primal solution (if the relaxation is tight), but also to terminate when a near optimal relaxed primal solution has been found.[4] This is an important advantage over PSDD, which is unable to provide similar stopping conditions, and is usually stopped rather arbitrarily after a given number of iterations.

The *primal residual* $r_P^{(t)}$ is the amount by which the agreement constraints are violated,

$$r_P^{(t)} = \frac{\sum_{(i,\alpha)\in E} \|\mathbf{M}_{i\alpha}\boldsymbol{q}_\alpha^{(t)} - \boldsymbol{p}_i^{(t)}\|^2}{\sum_{(i,\alpha)\in E} |\mathcal{Y}_i|} \in [0,1], \tag{30}$$

where the constant in the denominator ensures that $r_P^{(t)} \in [0,1]$. The *dual residual* $r_D^{(t)}$,

$$r_D^{(t)} = \frac{\sum_{(i,\alpha)\in E} \|\boldsymbol{p}_i^{(t)} - \boldsymbol{p}_i^{(t-1)}\|^2}{\sum_{(i,\alpha)\in E} |\mathcal{Y}_i|} \in [0,1], \tag{31}$$

is the amount by which a dual optimality condition is violated (Boyd et al., 2011, §3.3). We adopt as stopping criterion that these two residuals fall below a threshold, *e.g.*, 10^{-6}.

4.4.2 APPROXIMATE SOLUTIONS OF THE LOCAL SUBPROBLEMS

The next proposition states that convergence may still hold if the local subproblems are only solved approximately. The importance of this result will be clear in Section 6, where we describe a general iterative algorithm for solving the local quadratic subproblems. Essentially, Proposition 9 allows these subproblems to be solved numerically up to some accuracy without compromising global convergence, as long as the accuracy of the solutions improves sufficiently fast over AD^3 iterations.

4. This is particularly useful if inference is embedded in learning, where it is more important to obtain a *fractional* solution of the relaxed primal than an approximate integer one (Kulesza and Pereira, 2007; Martins et al., 2009).

Proposition 9 (Eckstein and Bertsekas, 1992) *Let $\eta_t = \eta$, and for each iteration t, let $\widehat{\boldsymbol{q}}^{(t)}$ contain the exact solutions of Eq. 25, and $\tilde{\boldsymbol{q}}^{(t)}$ those produced by an approximate algorithm. Then Proposition 7 still holds, provided that the sequence of errors is summable,* i.e., $\sum_{t=1}^{\infty} \|\widehat{\boldsymbol{q}}^{(t)} - \tilde{\boldsymbol{q}}^{(t)}\| < \infty$.

4.4.3 RUNTIME AND CACHING STRATEGIES

In practice, considerable speed-ups can be achieved by caching the subproblems, a strategy which has also been proposed for the PSDD algorithm by Koo et al. (2010). After a few iterations, many variables $\boldsymbol{p}_i$ reach a consensus (*i.e.*, $\boldsymbol{p}_i^{(t)} = \boldsymbol{q}_{i\alpha}^{(t+1)}, \forall \alpha \in \partial(i)$) and enter an idle state: they are left unchanged by the $\boldsymbol{p}$-update (line 10), and so do the Lagrange variables $\boldsymbol{\lambda}_{i\alpha}^{(t+1)}$ (line 11). If at iteration t all variables in a subproblem at factor α are idle, then $\boldsymbol{q}_{\alpha}^{(t+1)} = \boldsymbol{q}_{\alpha}^{(t)}$, hence the corresponding subproblem does not need to be solved. Typically, many variables and subproblems enter this idle state after the first few rounds. We will show the practical benefits of caching in the experimental section (Section 7.5, Figure 9).

4.5 Exact Inference with Branch-and-Bound

Recall that AD^3, as just described, solves the LP-MAP *relaxation* of the actual problem. In some problems, this relaxation is tight (in which case a certificate of optimality will be obtained), but this is not always the case. When a fractional solution is obtained, it is desirable to have a strategy to recover the exact MAP solution.

Two observations are noteworthy. First, as we saw in Section 2.3, the optimal value of the relaxed problem LP-MAP provides an upper bound to the original problem MAP. In particular, any feasible dual point provides an upper bound to the original problem's optimal value. Second, during execution of the AD^3 algorithm, we always keep track of a sequence of feasible dual points (as guaranteed by Proposition 7, item 3. Therefore, each iteration constructs tighter and tighter upper bounds. In recent work (Das et al., 2012), we proposed a *branch-and-bound search* procedure that finds the exact solution of the ILP. The procedure works recursively as follows:

1. Initialize $L = -\infty$ (our best value so far).

2. Run Algorithm 2. If the solution $\boldsymbol{p}^*$ is integer, return $\boldsymbol{p}^*$ and set L to the objective value. If along the execution we obtain an upper bound less than L, then Algorithm 2 can be safely stopped and return "infeasible"—this is the *bound* part. Otherwise (if $\boldsymbol{p}^*$ is fractional) go to step 3.

3. Find the "most fractional" component of $\boldsymbol{p}^*$ (call it $p_j^*(.)$) and *branch*: for every $y_j \in \mathcal{Y}_j$, constrain $p_j(y_j) = 1$ and go to step 2, eventually obtaining an integer solution $\boldsymbol{p}^*|_{y_j}$ or infeasibility. Return the $\boldsymbol{p}^* \in \{\boldsymbol{p}^*|_{y_j}\}_{y_j \in \mathcal{Y}_j}$ that yields the largest objective value.

Although this procedure has worst-case exponential runtime, in many problems for which the relaxations are near-exact it is found empirically very effective. We will see one example in Section 7.4.

5. Local Subproblems in AD^3

This section shows how to solve the AD^3 local subproblems (25) exactly and efficiently, in several cases, including Ising models and logic constraint factors. These results will be complemented in

Section 6, where a new procedure to handle *arbitrary* factors widens the applicability of AD^3. By subtracting a constant, re-scaling, and flipping signs, Eq. 25 can be written more compactly as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{M}\boldsymbol{q}_\alpha - \boldsymbol{a}\|^2 - \boldsymbol{b}^\top \boldsymbol{q}_\alpha \\ \text{with respect to} \quad & \boldsymbol{q}_\alpha \in \mathbb{R}^{|\mathcal{Y}_\alpha|} \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{q}_\alpha = 1, \quad \boldsymbol{q}_\alpha \geq \mathbf{0}, \end{aligned} \tag{32}$$

where $\boldsymbol{a} := (\boldsymbol{a}_i)_{i \in \partial(\alpha)}$, with $\boldsymbol{a}_i := \boldsymbol{p}_i + \eta^{-1}(\boldsymbol{\theta}_{i\alpha} + \boldsymbol{\lambda}_{i\alpha})$; $\boldsymbol{b} := \eta^{-1}\boldsymbol{\theta}_\alpha$; and $\mathbf{M} := (\mathbf{M}_{i\alpha})_{i \in \partial(\alpha)}$ denotes a matrix with $\sum_i |\mathcal{Y}_i|$ rows and $|\mathcal{Y}_\alpha|$ columns.

We show that Eq. 32 has a closed-form solution or can be solved exactly and efficiently, in several cases; *e.g.*, for Ising models, for factor graphs imposing first-order logic (FOL) constraints, and for Potts models (after binarization). In these cases, AD^3 and the PSDD algorithm have (asymptotically) the same computational cost per iteration, up to a logarithmic factor.

5.1 Ising Models

Ising models are factor graphs containing only binary pairwise factors. A binary pairwise factor (say, α) is one connecting two binary variables (say, Y_1 and Y_2); thus $\mathcal{Y}_1 = \mathcal{Y}_2 = \{0, 1\}$ and $\mathcal{Y}_\alpha = \{00, 01, 10, 11\}$. Given that $\boldsymbol{q}_{1\alpha}, \boldsymbol{q}_{2\alpha} \in \Delta^2$, we can write $\boldsymbol{q}_{1\alpha} = (1 - z_1, z_1)$, $\boldsymbol{q}_{2\alpha} = (1 - z_2, z_2)$. Furthermore, since $\boldsymbol{q}_\alpha \in \Delta^4$ and marginalization requires that $q_\alpha(1,1) + q_\alpha(1,0) = z_1$ and $q_\alpha(0,1) + q_\alpha(1,1) = z_2$, we can also write $\boldsymbol{q}_\alpha = (1 - z_1 - z_2 + z_{12}, z_1 - z_{12}, z_2 - z_{12}, z_{12})$. Using this parametrization, problem (32) reduces to:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(z_1 - c_1)^2 + \frac{1}{2}(z_2 - c_2)^2 - c_{12} z_{12} \\ \text{with respect to} \quad & z_1, z_2, z_{12} \in [0, 1]^3 \\ \text{subject to} \quad & z_{12} \leq z_1, \quad z_{12} \leq z_2, \quad z_{12} \geq z_1 + z_2 - 1, \end{aligned} \tag{33}$$

where

$$c_1 = \frac{a_{1\alpha}(1) + 1 - a_{1\alpha}(0) - b_\alpha(0,0) + b_\alpha(1,0)}{2} \tag{34}$$

$$c_2 = \frac{a_{2\alpha}(1) + 1 - a_{2\alpha}(0) - b_\alpha(0,0) + b_\alpha(0,1)}{2} \tag{35}$$

$$c_{12} = \frac{b_\alpha(0,0) - b_\alpha(1,0) - b_\alpha(0,1) + b_\alpha(1,1)}{2}. \tag{36}$$

The next proposition (proved in Appendix B.1) establishes a closed form solution for this problem, which immediately translates into a procedure for SOLVEQP for binary pairwise factors.

Proposition 10 *Let* $[x]_{\mathbb{U}} := \min\{\max\{x, 0\}, 1\}$ *denote projection (clipping) onto the unit interval* $\mathbb{U} := [0, 1]$. *The solution* (z_1^*, z_2^*, z_{12}^*) *of problem* (33) *is the following. If* $c_{12} \geq 0$,

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

$$z_{12}^* = \min\{z_1^*, z_2^*\}; \tag{37}$$

otherwise ,

$$\begin{aligned}(z_1^*, z_2^*) &= \begin{cases} ([c_1 + c_{12}]_{\mathbb{U}}, & [c_2 + c_{12}]_{\mathbb{U}}), & \text{if } c_1 + c_2 + 2c_{12} > 1 \\ ([c_1]_{\mathbb{U}}, & [c_2]_{\mathbb{U}}), & \text{if } c_1 + c_2 < 1 \\ ([(c_1 + 1 - c_2)/2]_{\mathbb{U}}, & [(c_2 + 1 - c_1)/2]_{\mathbb{U}}), & \text{otherwise,} \end{cases} \\ z_{12}^* &= \max\{0, z_1^* + z_2^* - 1\}. \end{aligned} \tag{38}$$

5.2 Factor Graphs with First-Order Logic Constraints

Hard constraint factors allow specifying "forbidden" configurations, and have been used in error-correcting decoders (Richardson and Urbanke, 2008), bipartite graph matching (Duchi et al., 2007), computer vision (Nowozin and Lampert, 2009), and natural language processing (Smith and Eisner, 2008). In many applications, *declarative constraints* are useful for injecting domain knowledge, and first-order logic (FOL) provides a natural language to express such constraints. This is particularly useful in learning from scarce annotated data (Roth and Yih, 2004; Punyakanok et al., 2005; Richardson and Domingos, 2006; Chang et al., 2008; Poon and Domingos, 2009).

In this section, we consider hard constraint factors linked to binary variables, with log-potential functions of the form

$$\theta_\alpha(\boldsymbol{y}_\alpha) = \begin{cases} 0, & \text{if } \boldsymbol{y}_\alpha \in \mathcal{S}_\alpha \\ -\infty, & \text{otherwise,} \end{cases} \tag{39}$$

where $\mathcal{S}_\alpha \subseteq \{0,1\}^{|\partial(\alpha)|}$ is an *acceptance set*. These factors can be used for imposing FOL constraints, as we describe next. We define the *marginal polytope* $\mathcal{Z}_\alpha$ of a hard constraint factor α as the convex hull of its acceptance set,

$$\mathcal{Z}_\alpha = \operatorname{conv} \mathcal{S}_\alpha. \tag{40}$$

As shown in Appendix B.2, the AD^3 subproblem (32) associated with a hard constraint factor is equivalent to that of computing an Euclidean projection onto its marginal polytope:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{z} - \boldsymbol{z}_0\|^2 \\ \text{with respect to} \quad & \boldsymbol{z} \in \mathcal{Z}_\alpha, \end{aligned} \tag{41}$$

where $z_{0i} := (a_i(1) + 1 - a_i(0))/2$, for $i \in \partial(\alpha)$. We now show how to compute this projection for several hard constraint factors that are building blocks for writing FOL constraints. Each of these factors performs a logical function, and hence we represent them graphically as *logic gates* (Figure 3).

One-hot XOR (Uniqueness Quantification). The "one-hot XOR" factor linked to $K \geq 1$ binary variables is defined through the following potential function:

$$\theta_{\text{XOR}}(y_1, \ldots, y_K) := \begin{cases} 0 & \text{if } \exists! k \in \{1, \ldots, K\} \text{ s.t. } y_k = 1 \\ -\infty & \text{otherwise,} \end{cases} \tag{42}$$

where $\exists!$ denotes "there is one and only one." The name "one-hot XOR" stems from the following fact: for $K = 2$, $\exp(\theta_{\text{XOR}}(.))$ is the logic eXclusive-OR function; the prefix "one-hot" expresses that this generalization to $K > 2$ only accepts configurations with precisely one "active" input (not to be mistaken with other XOR generalizations commonly used for parity checks). The XOR factor can be used for binarizing a categorical variable, and to express a statement in FOL of the form $\exists! x : R(x)$.

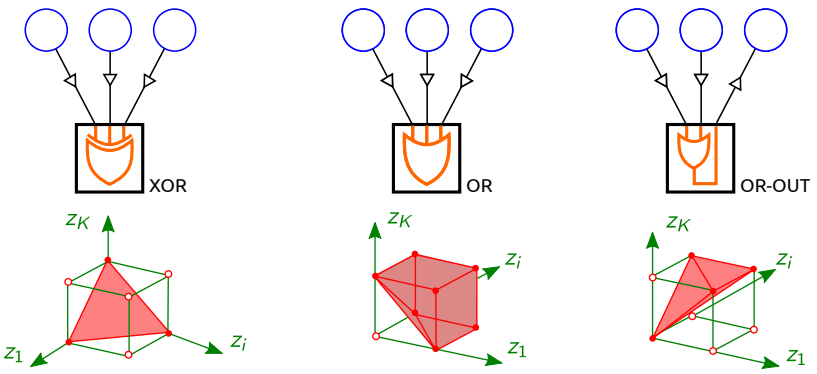

Figure 3: Logic factors and their marginal polytopes; the AD^3 subproblems Eq. 41 are projections onto these polytopes. Left: the one-hot XOR factor (its marginal polytope is the probability simplex). Middle: the OR factor. Right: the OR-with-output factor.

From Eq. 40, the marginal polytope associated with the one-hot XOR factor is

$$\mathcal{Z}_{\text{XOR}} = \text{conv}\left\{\boldsymbol{y} \in \{0,1\}^K \mid \exists! k \in \{1,\dots,K\} \text{ s.t. } y_k = 1\right\} = \Delta^K \tag{43}$$

as illustrated in Figure 3. Therefore, the AD^3 subproblem for the XOR factor consists in projecting onto the probability simplex, a problem well studied in the literature (Brucker, 1984; Michelot, 1986; Duchi et al., 2008). In Appendix B.3, we describe a simple $O(K \log K)$ algorithm. Note that there are $O(K)$ algorithms for this problem which are slightly more involved.

OR (Existential Quantification). This factor represents a disjunction of $K \geq 1$ binary variables,

$$\theta_{\text{OR}}(y_1,\dots,y_K) := \begin{cases} 0 & \text{if } \exists k \in \{1,\dots,K\} \text{ s.t. } y_k = 1 \\ -\infty & \text{otherwise,} \end{cases} \tag{44}$$

The OR factor can be used to represent a statement in FOL of the form $\exists x : R(x)$.

From Proposition 16, the marginal polytope associated with the OR factor is:

$$\mathcal{Z}_{\text{OR}} = \text{conv}\left\{\boldsymbol{y} \in \{0,1\}^K \mid \exists k \in \{1,\dots,K\} \text{ s.t. } y_k = 1\right\} \tag{45}$$

$$= \left\{\boldsymbol{z} \in [0,1]^K \;\middle|\; \sum_{k=1}^K z_k \geq 1\right\}; \tag{46}$$

geometrically, it is a "truncated" hypercube, as depicted in Figure 3. We derive a $O(K \log K)$ algorithm for projecting onto $\mathcal{Z}_{\text{OR}}$, using a sifting technique and a sort operation (see Appendix B.4).

Logical Variable Assignments: OR-with-output. The two factors above define a constraint on a group of existing variables. Alternatively, we may want to define a new variable (say, y_{K+1}) which is the result of an operation involving other variables (say, $y_1,\dots,y_K$). Among other things, this will allow dealing with "soft constraints," *i.e.*, constraints that can be violated but whose violation will decrease the score by some penalty. An example is the OR-with-output factor:

$$\theta_{\text{OR}-\text{out}}(y_1,\dots,y_K,y_{K+1}) := \begin{cases} 1 & \text{if } y_{K+1} = y_1 \vee \cdots \vee y_K \\ 0 & \text{otherwise.} \end{cases} \tag{47}$$

This factor constrains the variable y_{K+1} to indicate the existence of one or more active variables among $y_1, \ldots, y_K$. It can be used to express the following statement in FOL: $T(x) := \exists z : R(x, z)$.

The marginal polytope associated with the OR-with-output factor (also depicted in Figure 3):

$$\mathcal{Z}_{\mathrm{OR-out}} = \mathrm{conv}\left\{ \boldsymbol{y} \in \{0,1\}^{K+1} \;\middle|\; y_{K+1} = y_1 \vee \cdots \vee y_K \right\} \tag{48}$$

$$= \left\{ \boldsymbol{z} \in [0,1]^{K+1} \;\middle|\; \sum_{k=1}^{K} z_k \geq z_{K+1},\ z_k \leq z_{K+1}, \forall k \in \{1, \ldots, K\} \right\}. \tag{49}$$

Although projecting onto $\mathcal{Z}_{\mathrm{OR-out}}$ is slightly more complicated than the previous cases, in Appendix B.5, we propose (and prove correctness of) an $O(K \log K)$ algorithm for this task.

Negations, De Morgan's laws, and AND-with-output. The three factors just presented can be extended to accommodate *negated* inputs, thus adding flexibility. Solving the corresponding AD^3 subproblems can be easily done by reusing the methods that solve the original problems. For example, it is straightforward to handle negated conjunctions (NAND),

$$\theta_{\mathrm{NAND}}(y_1, \ldots, y_K) := \begin{cases} -\infty & \text{if } y_k = 1, \forall k \in \{1, \ldots, K\} \\ 0 & \text{otherwise,} \end{cases}$$
$$= \theta_{\mathrm{OR}}(\neg y_1, \ldots, \neg y_K), \tag{50}$$

as well as implications (IMPLY),

$$\theta_{\mathrm{IMPLY}}(y_1, \ldots, y_K, y_{K+1}) := \begin{cases} 0 & \text{if } (y_1 \wedge \cdots \wedge y_K) \Rightarrow y_{K+1} \\ -\infty & \text{otherwise} \end{cases}$$
$$= \theta_{\mathrm{OR}}(\neg y_1, \ldots, \neg y_K, y_{K+1}). \tag{51}$$

In fact, from De Morgan's laws, $\neg\left(Q_1(x) \wedge \cdots \wedge Q_K(x)\right)$ is equivalent to $\neg Q_1(x) \vee \cdots \vee \neg Q_K(x)$, and $(Q_1(x) \wedge \cdots \wedge Q_K(x)) \Rightarrow R(x)$ is equivalent to $(\neg Q_1(x) \vee \cdots \vee \neg Q_K(x)) \vee R(x)$. Another example is the AND-with-output factor,

$$\theta_{\mathrm{AND-out}}(y_1, \ldots, y_K, y_{K+1}) := \begin{cases} 0 & \text{if } y_{K+1} = y_1 \wedge \cdots \wedge y_K \\ -\infty & \text{otherwise} \end{cases}$$
$$= \theta_{\mathrm{OR-out}}(\neg y_1, \ldots, \neg y_K, \neg y_{K+1}), \tag{52}$$

which can be used to impose FOL statements of the form $T(x) := \forall z : R(x, z)$.

Let α be a binary constraint factor with marginal polytope $\mathcal{Z}_\alpha$, and β a factor obtained from α by negating the kth variable. For notational convenience, let $\mathrm{sym}_k : [0,1]^K \to [0,1]^K$ be defined as $(\mathrm{sym}_k(\boldsymbol{z}))_k = 1 - z_k$ and $(\mathrm{sym}_k(\boldsymbol{z}))_i = z_i$, for $i \neq k$. Then, the marginal polytope $\mathcal{Z}_\beta$ is a symmetric transformation of $\mathcal{Z}_\alpha$,

$$\mathcal{Z}_\beta = \left\{ \boldsymbol{z} \in [0,1]^K \;\middle|\; \mathrm{sym}_k(\boldsymbol{z}) \in \mathcal{Z}_\alpha \right\}, \tag{53}$$

and, if $\mathrm{proj}_{\mathcal{Z}_\alpha}$ denotes the projection operator onto $\mathcal{Z}_\alpha$,

$$\mathrm{proj}_{\mathcal{Z}_\beta}(\boldsymbol{z}) = \mathrm{sym}_k\left(\mathrm{proj}_{\mathcal{Z}_\alpha}(\mathrm{sym}_k(\boldsymbol{z}))\right). \tag{54}$$

Naturally, $\mathrm{proj}_{\mathcal{Z}_\beta}$ can be computed as efficiently as $\mathrm{proj}_{\mathcal{Z}_\alpha}$ and, by induction, this procedure can be generalized to an arbitrary number of negated variables.

5.3 Potts Models and Graph Binarization

Although general factors lack closed-form solutions of the corresponding AD^3 subproblem (32), it is possible to *binarize* the graph, *i.e.*, to convert it into an equivalent one that only contains binary variables and XOR factors. The procedure is as follows:

- For each variable node $i \in V$, define binary variables $U_{i,y_i} \in \{0,1\}$, for each state $y_i \in \mathcal{Y}_i$; link these variables to a XOR factor, imposing $\sum_{y_i \in \mathcal{Y}_i} p_i(y_i) = 1$.
- For each factor $\alpha \in F$, define binary variables $U_{\alpha,\boldsymbol{y}_\alpha} \in \{0,1\}$ for every $\boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha$. For each edge $(i,\alpha) \in E$ and each $y_i \in \mathcal{Y}_i$, link variables $\{U_{\alpha,\boldsymbol{y}_\alpha} \mid \boldsymbol{y}_\alpha \sim y_i\}$ and $\neg U_{i,y_i}$ to a XOR factor; this imposes the constraint $p_i(y_i) = \sum_{\boldsymbol{y}_\alpha \sim y_i} q_\alpha(\boldsymbol{y}_\alpha)$.

The resulting binary graph is one for which we already presented the machinery needed for solving efficiently the corresponding AD^3 subproblems. As an example, for Potts models (graphs with only pairwise factors and variables that have more than two states), the computational cost per AD^3 iteration on the binarized graph is asymptotically the same as that of the PSDD and other message-passing algorithms; for details, see Martins (2012).

6. An Active Set Method For Solving the AD^3 Subproblems

In this section, we complement the results of Section 5 with a general *active-set procedure* for solving the AD^3 subproblems for *arbitrary* factors, the only requirement being a black-box MAP solver—the same as the PSDD algorithm. This makes AD^3 applicable to a wide range of problems. In particular, it makes possible to handle *structured factors*, by invoking specialized MAP decoders (functions COMPUTEMAP in Algorithm 1). In practice, as we will see in Section 7, the active set method we next present largely outperforms the graph binarization strategy outlined in Section 5.3.

Our active set method is based on Nocedal and Wright (1999, Section 16.4); it is an iterative algorithm that addresses the AD^3 subproblems (32) by solving a sequence of linear problems. The next crucial proposition (proved in Appendix C) states that the problem in Eq. 32 always admits a *sparse solution*.

Proposition 11 *Problem* (32) *admits a solution* $\boldsymbol{q}^*_\alpha \in \mathbb{R}^{|\mathcal{Y}_\alpha|}$ *with at most* $\sum_{i \in \partial(\alpha)} |\mathcal{Y}_i| - |\partial(\alpha)| + 1$ *non-zero components.*

The fact that the solution lies in a low dimensional subspace makes active set methods appealing, since they only keep track of an *active set* of variables, that is, the non-zero components of $\boldsymbol{q}_\alpha$. Proposition 11 tells us that such an algorithm only needs to maintain at most $O(\sum_i |\mathcal{Y}_i|)$ elements in the active set—note the *additive*, rather than multiplicative, dependency on the number of values of the variables. Our active set method seeks to identify the low-dimensional support of the solution $\boldsymbol{q}^*_\alpha$, by generating sparse iterates $\boldsymbol{q}^{(1)}_\alpha, \boldsymbol{q}^{(2)}_\alpha, \ldots$, while it maintains a working set $W \subseteq \mathcal{Y}_\alpha$ with the inequality constraints of Eq. 32 that are *inactive* along the way (*i.e.*, those $\boldsymbol{y}_\alpha$ for which $q_\alpha(\boldsymbol{y}_\alpha) > 0$ holds strictly). Each iteration adds or removes elements from the working set while it monotonically decreases the objective of Eq. 32.[5]

5. Our description differs from Nocedal and Wright (1999) in which their working set contains *active* constraints rather than the inactive ones. In our case, most constraints are active for the optimal $\boldsymbol{q}^*_\alpha$, therefore it is appealing to store the ones that are not.

Lagrangian and KKT conditions. Let τ and $\boldsymbol{\mu}$ be dual variables associated with the equality and inequality constraints of Eq. 32, respectively. The Lagrangian function is

$$L(\boldsymbol{q}_\alpha, \tau, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{M}\boldsymbol{q}_\alpha - \boldsymbol{a}\|^2 - \boldsymbol{b}^\top \boldsymbol{q}_\alpha - \tau(1 - \mathbf{1}^\top \boldsymbol{q}_\alpha) - \boldsymbol{\mu}^\top \boldsymbol{q}_\alpha. \tag{55}$$

This gives rise to the following Karush-Kuhn-Tucker (KKT) conditions:

$$\mathbf{M}^\top(\boldsymbol{a} - \mathbf{M}\boldsymbol{q}_\alpha) + \boldsymbol{b} = \tau\mathbf{1} - \boldsymbol{\mu} \quad (\nabla_{\boldsymbol{q}_\alpha} L = \mathbf{0}) \tag{56}$$

$$\mathbf{1}^\top \boldsymbol{q}_\alpha = 1,\ \boldsymbol{q}_\alpha \geq \mathbf{0},\ \boldsymbol{\mu} \geq \mathbf{0} \quad \text{(Primal/dual feasibility)} \tag{57}$$

$$\boldsymbol{\mu}^\top \boldsymbol{q}_\alpha = \mathbf{0} \quad \text{(Complementary slackness).} \tag{58}$$

The method works at follows. At each iteration s, it first checks if the current iterate $\boldsymbol{q}_\alpha^{(s)}$ is a *subspace minimizer*, *i.e.*, if it optimizes the objective of Eq. 32 in the sparse subspace defined by the working set W, $\{\boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|} \mid q_\alpha(\boldsymbol{y}_\alpha) = 0, \forall \boldsymbol{y}_\alpha \notin W\}$. This check can be made by first solving a relaxation where the inequality constraints are ignored. Since in this subspace the components of $\boldsymbol{q}_\alpha$ not in W will be zeros, one can simply delete those entries from $\boldsymbol{q}_\alpha$ and $\boldsymbol{b}$ and the corresponding columns in $\mathbf{M}$; we use a horizontal bar to denote these truncated $\mathbb{R}^{|W|}$-vectors. The problem can be written as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\bar{\mathbf{M}}\bar{\boldsymbol{q}}_\alpha - \boldsymbol{a}\|^2 - \bar{\boldsymbol{b}}^\top \bar{\boldsymbol{q}}_\alpha \\ \text{with respect to} \quad & \bar{\boldsymbol{q}}_\alpha \in \mathbb{R}^{|W|} \\ \text{subject to} \quad & \mathbf{1}^\top \bar{\boldsymbol{q}}_\alpha = 1. \end{aligned} \tag{59}$$

The solution of this equality-constrained QP can be found by solving a system of KKT equations:[6]

$$\begin{bmatrix} \bar{\mathbf{M}}^\top \bar{\mathbf{M}} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{q}}_\alpha \\ \tau \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{M}}^\top \boldsymbol{a} + \bar{\boldsymbol{b}} \\ 1 \end{bmatrix}. \tag{60}$$

The solution of Eq. 60 will give $(\widehat{\boldsymbol{q}}_\alpha, \widehat{\tau})$, where $\widehat{\boldsymbol{q}}_\alpha \in \mathbb{R}^{|\mathcal{Y}_\alpha|}$ is padded back with zeros. If it happens that $\widehat{\boldsymbol{q}}_\alpha = \boldsymbol{q}_\alpha^{(s)}$, then this means that the current iterate $\boldsymbol{q}_\alpha^{(s)}$ is a subspace minimizer; otherwise a new iterate $\boldsymbol{q}_\alpha^{(s+1)}$ will be computed. We next discuss these two events.

Case 1: $\boldsymbol{q}_\alpha^{(s)}$ is a subspace minimizer. If this happens, then it may be the case that $\boldsymbol{q}_\alpha^{(s)}$ is the optimal solution of Eq. 32. By looking at the KKT conditions (56)–(58), we have that this will happen iff $\mathbf{M}^\top(\boldsymbol{a} - \mathbf{M}\boldsymbol{q}_\alpha^{(s)}) + \boldsymbol{b} \leq \tau^{(s)}\mathbf{1}$. Define $\boldsymbol{w} := \boldsymbol{a} - \mathbf{M}\boldsymbol{q}_\alpha$. The condition above is equivalent to

$$\max_{\boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha} \left(b(\boldsymbol{y}_\alpha) + \sum_{i \in \partial(\alpha)} w_i(y_i) \right) \leq \tau^{(s)}. \tag{61}$$

6. Note that this is a low-dimensional problem, since we are working in a sparse working set. By caching the inverse of the matrix in the left-hand side, this system can be solved in time $O(|W|^2)$ at each iteration. Note also that adding a new configuration $\boldsymbol{y}_\alpha$ to the active set, corresponds to inserting a new column in $\bar{\mathbf{M}}$, thus the matrix inversion requires updating $\bar{\mathbf{M}}^\top \bar{\mathbf{M}}$. From the definition of $\mathbf{M}$ and simple algebra, the $(\boldsymbol{y}_\alpha, \boldsymbol{y}'_\alpha)$ entry in $\mathbf{M}^\top \mathbf{M}$ is simply the *number of common values* between the configurations $\boldsymbol{y}_\alpha$ and $\boldsymbol{y}'_\alpha$. Hence, when a new configuration $\boldsymbol{y}_\alpha$ is added to the active set W, that configuration needs to be compared with all the others already in W.

It turns out that this maximization is precisely a *local MAP inference problem*, given a vector of unary potentials $\boldsymbol{w}$ and factor potentials $\boldsymbol{b}$. Thus, the maximizer $\widehat{\boldsymbol{y}}_\alpha$ can be computed via the COMPUTEMAP procedure, which we assume available. If $b(\widehat{\boldsymbol{y}}_\alpha) + \sum_{i \in \partial(\alpha)} w_i(\widehat{y}_i) \leq \tau^{(s)}$, then the KKT conditions are satisfied and we are done. Otherwise, $\widehat{\boldsymbol{y}}_\alpha$ indicates the most violated condition; we will add it to the active set W, and proceed.

Case 2: $\boldsymbol{q}_\alpha^{(s)}$ is not a subspace minimizer. If this happens, then we compute a new iterate $\boldsymbol{q}_\alpha^{(s+1)}$ by keeping searching in the same subspace. We have already solved a relaxation in Eq. 59. If we have $\widehat{q}_\alpha(\boldsymbol{y}_\alpha) \geq 0$ for all $\boldsymbol{y}_\alpha \in W$, then the relaxation is tight, so we just set $\boldsymbol{q}_\alpha^{(s+1)} := \widehat{\boldsymbol{q}}_\alpha$ and proceed. Otherwise, we move as much as possible in the direction of $\widehat{\boldsymbol{q}}_\alpha$ while keeping feasibility, by defining $\boldsymbol{q}_\alpha^{(s+1)} := (1-\beta)\boldsymbol{q}_\alpha^{(s)} + \beta\widehat{\boldsymbol{q}}_\alpha$—as described in Nocedal and Wright (1999), the value of $\beta \in [0, 1]$ can be computed in closed form:

$$\beta = \min \left\{ 1, \min_{\boldsymbol{y}_\alpha \in W \,:\, q_\alpha^{(s)}(\boldsymbol{y}_\alpha) > \widehat{q}_\alpha(\boldsymbol{y}_\alpha)} \frac{q_\alpha^{(s)}(\boldsymbol{y}_\alpha)}{q_\alpha^{(s)}(\boldsymbol{y}_\alpha) - \widehat{q}_\alpha(\boldsymbol{y}_\alpha)} \right\}. \tag{62}$$

If $\beta < 1$, this update will have the effect of making one of the constraints active, by zeroing out $q_\alpha^{(s+1)}(\boldsymbol{y}_\alpha)$ for the minimizing $\boldsymbol{y}_\alpha$ above. This so-called "blocking constraint" is thus be removed from the working set W.

Algorithm 3 describes the complete procedure. The active set W is initialized arbitrarily: a strategy that works well in practice is, in the first AD^3 iteration, initialize $W := \{\widehat{\boldsymbol{y}}_\alpha\}$, where $\widehat{\boldsymbol{y}}_\alpha$ is the MAP configuration given log-potentials $\boldsymbol{a}$ and $\boldsymbol{b}$; and in subsequent AD^3 iterations, warm-start W with the support of the solution obtained in the previous iteration.

Each iteration of Algorithm 3 improves the objective of Eq. 32, and, with a suitable strategy to prevent cycles and stalling, the algorithm is guaranteed to stop after a finite number of steps (Nocedal and Wright, 1999, Theorem 16.5). In practice, since it is run as a subroutine of AD^3, Algorithm 3 does not need to be run to optimality, which is convenient in early iterations of AD^3 (this is supported by Proposition 9). The ability to warm-start with the solution from the previous round is very useful in practice: we have observed that, thanks to this warm-starting strategy, very few inner iterations are typically necessary for the correct active set to be identified. We will see some empirical evidence in Section 7.5.

7. Experiments

7.1 Baselines and Problems

In this section, we provide an empirical comparison between AD^3 (Algorithm 2) and four other algorithms: generalized MPLP (Globerson and Jaakkola, 2008); norm-product BP (Hazan and Shashua, 2010);[7] the PSDD algorithm of Komodakis et al. (2007) (Algorithm 1) and its accelerated version introduced by Jojic et al. (2010). All these algorithms address the LP-MAP problem; the first are message-passing methods performing block coordinate descent in the dual, whereas the last two are based on dual decomposition. The norm-product BP and accelerated dual decomposition algorithms introduce a temperature parameter to smooth their dual objectives. All the baselines have the same

7. For norm-product BP, we adapted the code provided by the authors, using the "trivial" counting numbers $c_\alpha = 1$, $c_{i\alpha} = 0$, and $c_i = 0, \forall (i, \alpha) \in E$, which leads to a concave entropy approximation.

Algorithm 3 Active Set Algorithm for Solving a General AD^3 Subproblem

input: Parameters $\boldsymbol{a}, \boldsymbol{b}, \mathbf{M}$, starting point $\boldsymbol{q}_\alpha^{(0)}$
initialize $W^{(0)}$ as the support of $\boldsymbol{q}_\alpha^{(0)}$
for $s = 0, 1, 2, \ldots$ **do**
  solve the KKT system and obtain $\widehat{\boldsymbol{q}}_\alpha$ and $\widehat{\tau}$ (Eq. 60)
  if $\widehat{\boldsymbol{q}}_\alpha = \boldsymbol{q}_\alpha^{(s)}$ **then**
    compute $\boldsymbol{w} := \boldsymbol{a} - \mathbf{M}\widehat{\boldsymbol{q}}_\alpha$
    obtain the tighter constraint $\widehat{\boldsymbol{y}}_\alpha$ via $\boldsymbol{e}_{\widehat{\boldsymbol{y}}_\alpha} = \textsc{ComputeMAP}(\boldsymbol{b} + \mathbf{M}^\top \boldsymbol{w})$
    if $b(\widehat{\boldsymbol{y}}_\alpha) + \sum_{i \in \partial(\alpha)} w_i(\widehat{y}_i) \leq \widehat{\tau}$ **then**
      return solution $\widehat{\boldsymbol{q}}_\alpha$
    else
      add the most violated constraint to the active set: $W^{(s+1)} := W^{(s)} \cup \{\widehat{\boldsymbol{y}}_\alpha\}$
    end if
  else
    compute the interpolation constant β as in Eq. 62
    set $\boldsymbol{q}_\alpha^{(s+1)} := (1 - \beta)\boldsymbol{q}_\alpha^{(s)} + \beta\widehat{\boldsymbol{q}}_\alpha$
    if if $\beta < 1$ **then**
      pick the blocking constraint $\widehat{\boldsymbol{y}}_\alpha$ in Eq. 62
      remove $\widehat{\boldsymbol{y}}_\alpha$ from the active set: $W^{(s+1)} := W^{(s)} \setminus \{\widehat{\boldsymbol{y}}_\alpha\}$
    end if
  end if
end for
output: $\widehat{\boldsymbol{q}}_\alpha$

algorithmic complexity per iteration, which is asymptotically the same as that of the AD^3 applied to a binarized graph, but different from that of AD^3 with the active set method.

We compare the performance of the algorithms above in several datasets, including synthetic Ising and Potts models, protein design problems, and two problems in natural language processing: frame-semantic parsing and non-projective dependency parsing. The graphical models associated with these problems are quite diverse, containing pairwise binary factors (AD^3 subproblems solved as described in Section 5.1), first-order logic factors (addressed using the tools of Section 5.2), dense factors, and structured factors (tackled with the active set method of Section 6).

7.2 Synthetic Ising and Potts Models

Ising models. Figure 4 reports experiments with random Ising models, with single-node log-potentials chosen as $\theta_i(1) - \theta_i(0) \sim \mathcal{U}[-1, 1]$ and random edge couplings in $\mathcal{U}[-\rho, \rho]$, where $\rho \in \{0.1, 0.2, 0.5, 1.0\}$. Decompositions are edge-based for all methods. For MPLP and norm-product BP, primal feasible solutions $(\widehat{y}_i)_{i \in V}$ are obtained by decoding the single node messages (Globerson and Jaakkola, 2008); for the dual decomposition methods, $\widehat{y}_i = \operatorname{argmax}_{y_i} p_i(y_i)$.

We observe that PSDD is the slowest algorithm, taking a long time to find a "good" primal feasible solution, arguably due to the large number of components. The accelerated dual decomposition method (Jojic et al., 2010) is also not competitive in this setting, as it takes many iterations to reach a near-optimal region. MPLP, norm-product, and AD^3 all perform very similarly regarding conver-

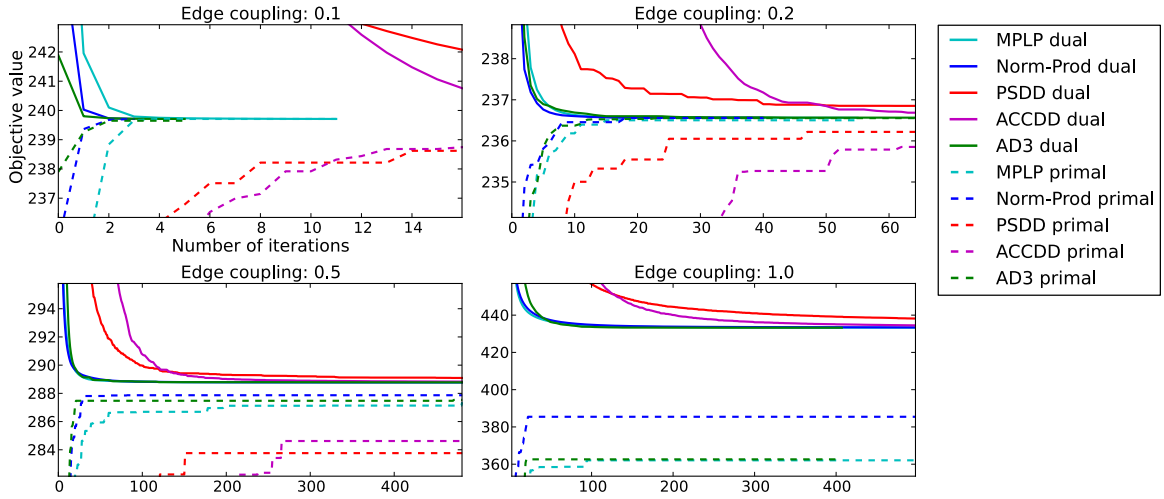

Figure 4: Evolution of the dual objective and the best primal feasible one in the experiments with 30×30 random Ising models, generated as described in the main text. For the subgradient method, the step sizes are $\eta_t = \eta_0 / k(t)$, where $k(t)$ is the number of times the dual decreased up to the tth iteration, and η_0 was chosen with hindsight in $\{0.001, 0.01, 0.1, 10\}$ to yield the best dual objective. For accelerated dual decomposition, the most favorable parameter $\epsilon \in \{0.1, 1, 10, 100\}$ was chosen. For norm-product BP, the temperature was set as $\tau = 0.001$, and the dual objective is computed with zero temperature (which led to better upper bounds). AD^3 uses $\eta = 0.1$ for all runs.

gence to the dual objective, with a slight advantage of the latter two. Regarding their ability to find a "good" feasible primal solution, AD^3 and norm-product BP seem to outperform their competitors. In a batch of 100 experiments using a coupling $\rho = 0.5$, AD^3 found a best dual than MPLP in 18 runs and it lost 11 times (the remaining 71 runs were ties); it won over norm-product BP 73 times and never lost. In terms of primal solutions, AD^3 won over MPLP in 47 runs and it lost 12 times (41 ties); and it won over norm-product in 49 runs and it lost 33 times (in all cases, relative differences lower than 1×10^{-6} were considered as ties).

Potts models. The effectiveness of AD^3 in the non-binary case is assessed using random Potts models, with single-node log-potentials chosen as $\theta_i(y_i) \sim \mathcal{U}[-1, 1]$ and pairwise log-potentials as $\theta_{ij}(y_i, y_j) \sim \mathcal{U}[-10, 10]$ if $y_i = y_j$ and 0 otherwise. All the baselines use the same edge decomposition as before, since they handle multi-valued variables; for AD^3, we tried two variants: one where the graph is binarized (see Section 5.3); and one which works in the original graph through the active set method, as described in Section 6.

As shown in Figure 5, MPLP and norm-product BP decrease the objective very rapidly in the beginning and then slow down considerably; the accelerated dual decomposition algorithm, although slower in early iterations, eventually surpasses them. Both variants of AD^3 converge as fast as the accelerated dual decomposition algorithm in later iterations, and are almost as fast as MPLP and norm-product in early iterations, getting the best of both worlds. Comparing the two variants of AD^3, we observe that the active set variant clearly outperforms the binarization variant. Notice that since AD^3 with the active set method involves more computation per iteration, we plot the objec-

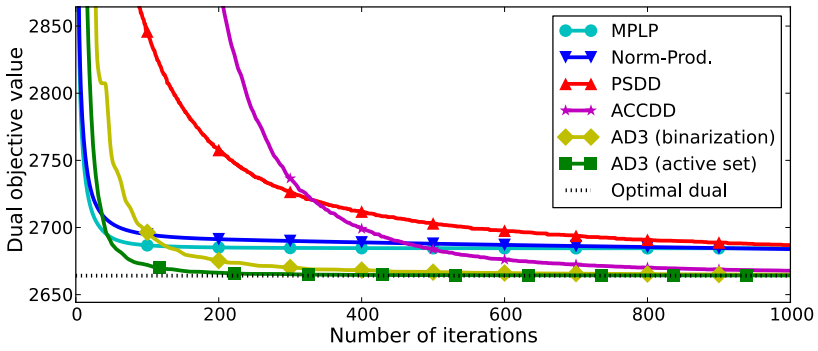

Figure 5: Evolution of the dual objective in the experiments with random 20×20 Potts models with 8-valued nodes, generated as described in the main text. For PSDD and the accelerated dual decomposition algorithm, we chose η_0 and ϵ as before. For AD^3, we set $\eta = 1.0$ in both settings (active set and binarization). In the active set method, no caching was used and the plotted number of iterations is corrected to make it comparable with the remaining algorithms, since each outer iteration of AD^3 requires several calls to a MAP oracle (we plot the normalized number of oracle calls instead). Yet, due to warm-starting, the average number of inner iterations is only 1.04, making the active set method extremely efficient. For all methods, the markers represent every 100 iterations.

tive values with respect to the normalized number of oracle calls (which matches the number of iterations for the other methods).

7.3 Protein Design

We compare AD^3 with the MPLP implementation[8] of Sontag et al. (2008) in the benchmark protein design problems[9] of Yanover et al. (2006). In these problems, the input is a three-dimensional shape, and the goal is to find the most stable sequence of amino acids in that shape. The problems can be represented as pairwise factor graphs, whose variables correspond to the identity of amino acids and rotamer configurations, thus having hundreds of possible states. Figure 6 plots the evolution of the dual objective over runtime, for two of the largest problem instances, *i.e.*, those with 3167 (1fbo) and 1163 (1kw4) factors. These plots are representative of the typical performance obtained in other instances. In both cases, MPLP steeply decreases the objective at early iterations, but then reaches a plateau with no further significant improvement. AD^3 rapidly surpasses MPLP in obtaining a better dual objective. Finally, observe that although earlier iterations of AD^3 take longer than those of MPLP, this cost is amortized in later iterations, by warm-starting the active set method.

7.4 Frame-Semantic Parsing

We now report experiments on a natural language processing task involving logic constraints: *frame-semantic parsing*, using the FrameNet lexicon (Fillmore, 1976). The goal is to predict the set of

8. Available at `http://cs.nyu.edu/~dsontag/code`; that code includes a "tightening" procedure for retrieving the exact MAP, which we don't use, since we are interested in the LP-MAP relaxation (which is what AD^3 addresses).
9. Available at `http://www.jmlr.org/papers/volume7/yanover06a/`.

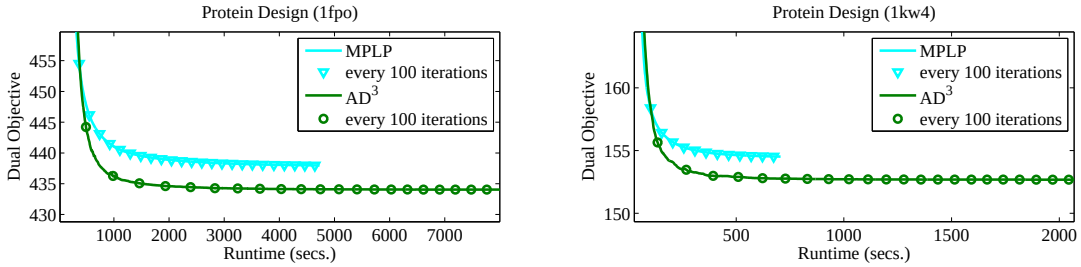

Figure 6: Protein design experiments (see main text for details). In AD^3, η is adjusted as proposed by Boyd et al. (2011, §3.4.1), initialized at $\eta = 1.0$ and the subproblems are solved by the proposed active set method. Although the plots are with respect to runtime, they also indicate iteration counts.

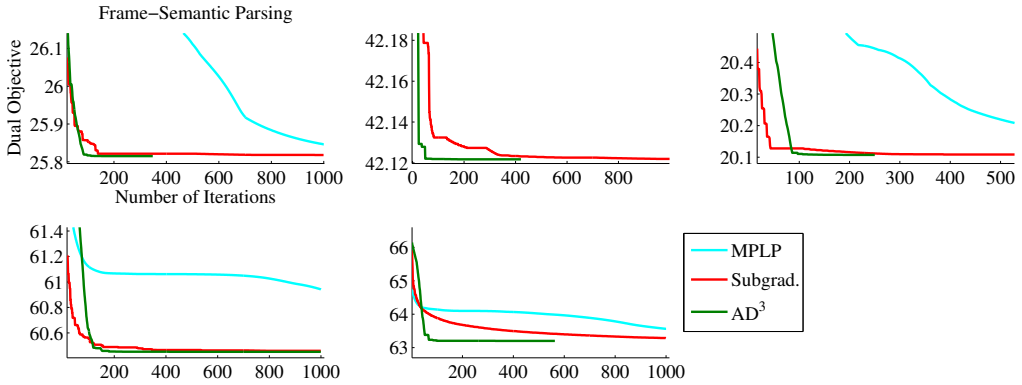

Figure 7: Experiments in five frame-semantic parsing problems (Das, 2012, Section 5.5). The projected subgradient uses $\eta_t = \eta_0/t$, with $\eta_0 = 1.0$ (found to be the best choice for all examples). In AD^3, η is adjusted as proposed by Boyd et al. (2011), initialized at $\eta = 1.0$.

arguments and roles for a predicate word in a sentence, while respecting several constraints about the frames that can be evoked. The resulting graphical models are binary constrained factor graphs with FOL constraints (see Das et al. 2012 for details about this task). Figure 7 shows the results of AD^3, MPLP, and PSDD on the five most difficult problems (which have between 321 and 884 variables, and between 32 and 59 factors), the ones in which the LP relaxation is not tight. Unlike MPLP and PSDD, which did not converge after 1000 iterations, AD^3 achieves convergence in a few hundreds of iterations for all but one example. Since these examples have a fractional LP-MAP solution, we applied the branch-and-bound procedure described in Section 4.5 to obtain the exact MAP for these examples. The whole dataset contains 4,462 instances, which were parsed by this exact variant of the AD^3 algorithm in only 4.78 seconds, against 43.12 seconds of CPLEX, a state-of-the-art commercial ILP solver.

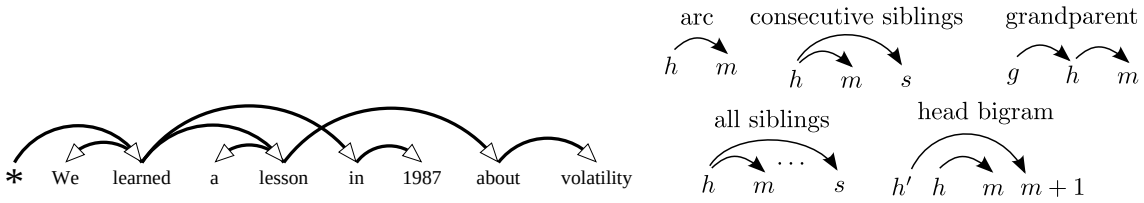

Figure 8: Left: example of a sentence (input) and its dependency parse tree (output to be predicted); this is a directed spanning tree where each arc (h, m) represent a syntactic relationships between a *head* word h and the a *modifier* word m. Right: the parts used in our models. *Arcs* are the basic parts: any dependency tree can be "read out" from its arcs. *Consecutive siblings* and *grandparent* parts introduce horizontal and vertical Markovization. We break the horizontal Markovianity via *all siblings* parts (which look at arbitrary pairs of siblings, not necessarily consecutive). Inspired by transition-based parsers, we also adopt *head bigram* parts, which look at the heads attached to consecutive words.

7.5 Dependency Parsing

The final set of experiments assesses the ability of AD^3 to handle problems with structured factors. The task is *dependency parsing* (illustrated in the left part of Figure 8), an important problem in natural language processing (Eisner, 1996; McDonald et al., 2005), to which dual decomposition has been recently applied (Koo et al., 2010). We use an English dataset derived from the Penn Treebank (PTB)(Marcus et al., 1993), converted to dependencies by applying the head rules of Yamada and Matsumoto (2003); we follow the common procedure of training in sections §02–21 (39,832 sentences), using §22 as validation data (1,700 sentences), and testing on §23 (2,416 sentences). We ran a part-of-speech tagger on the validation and test splits, and devised a linear model using various features depending on words, part-of-speech tags, and arc direction and length. Our features decompose over the parts illustrated in the right part of Figure 8. We consider two different models in our experiments: a *second order model* with scores for arcs, consecutive siblings, and grandparents; a *full model*, which also has scores for arbitrary siblings and head bigrams.

If only scores for arcs were used, the problem of obtaining a parse tree with maximal score could be solved efficiently with a maximum directed spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005); the addition of any of the other scores makes the problem NP-hard (McDonald and Satta, 2007). A factor graph representing the second order model, proposed by Smith and Eisner (2008) and Koo et al. (2010), contains binary variables representing the candidate arcs, a hard-constraint factor imposing the tree constraint, and head automata factors modeling the sequences of consecutive siblings and grandparents. The full model has additional binary pairwise factors for each possible pair of siblings (significantly increasing the number of factors), and a sequential factor modeling the sequence of heads.[10] We compare the PSDD and AD^3 algorithms for this task, using the decompositions above, which are the same for both methods. These decompositions select the largest factors for which efficient MAP oracles exist, based on the

10. In previous work (Martins et al., 2011b), we implemented a similar model with a more complex factor graph based on a multi-commodity flow formulation, requiring only the FOL factors described in Section 5.2. In the current paper, we consider a smaller graph with structured factors, which leads to significantly faster runtimes. More involved models, including third-order features, were recently considered in Martins et al. (2013).

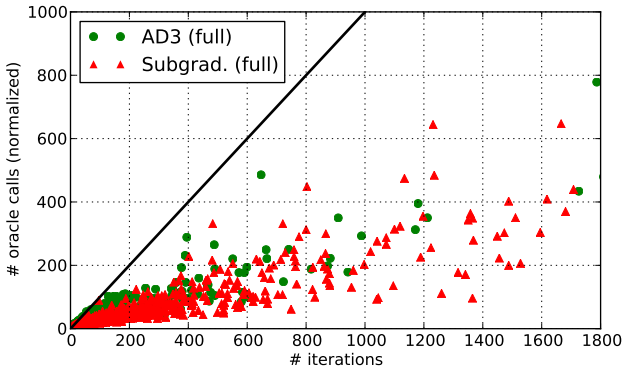

Figure 9: Number of calls to COMPUTEMAP for AD^3 and PSDD, as a function of the number of iterations. The number of calls is normalized by dividing by the number of factors: in PSDD, this number would equal the number of iterations if there was no caching (black line); each iteration of AD^3 runs 10 iterations of the active set method, thus without caching or warm-starting the normalized number of calls would be ten times the number of AD^3 iterations. Yet, it is clear that both algorithms make significantly fewer calls. Remarkably, after just a few iterations, the number of calls made by the AD^3 and PSDD algorithms are comparable, which means that the number of active set iterations is quickly amortized during the execution of AD^3.

Chu-Liu-Edmonds algorithm and on dynamic programming. The active set method enables AD^3 to depend only on these MAP oracles.

Figure 9 illustrates the remarkable speed-ups that the caching and warm-starting procedures bring to both the AD^3 and PSDD algorithms. A similar conclusion was obtained by Koo et al. (2010) for PSDD and by Martins et al. (2011b) for AD^3 in a different factor graph. Figure 10 shows average runtimes for both algorithms, as a function of the sentence length, and plots the percentage of instances for which the exact solution was obtained, along with a certificate of optimality. For the second-order model, AD^3 was able to solve all the instances to optimality, and in 98.2% of the cases, the LP-MAP was exact. For the full model, AD^3 solved 99.8% of the instances to optimality, being exact in 96.5% of the cases. For the second order model, we obtained in the test set (PTB §23) a parsing speed of 1200 tokens per second and an unlabeled attachment score of 92.48% (fraction of correct dependency attachments excluding punctuation). For the full model, we obtained a speed of 900 tokens per second and a score of 92.62%. All speeds were measured in a desktop PC with Intel Core i7 CPU 3.4 GHz and 8GB RAM. The parser is publicly available as an open-source project in `http://www.ark.cs.cmu.edu/TurboParser`.

8. Discussion and Related Work

We next discuss some of the strengths and weaknesses of AD^3 over other recently proposed LP-MAP inference algorithms. As mentioned in the beginning of Section 4, one of the main sources of difficulty is the *non-smoothness* of the dual objective function (Eq. 11). This affects both block

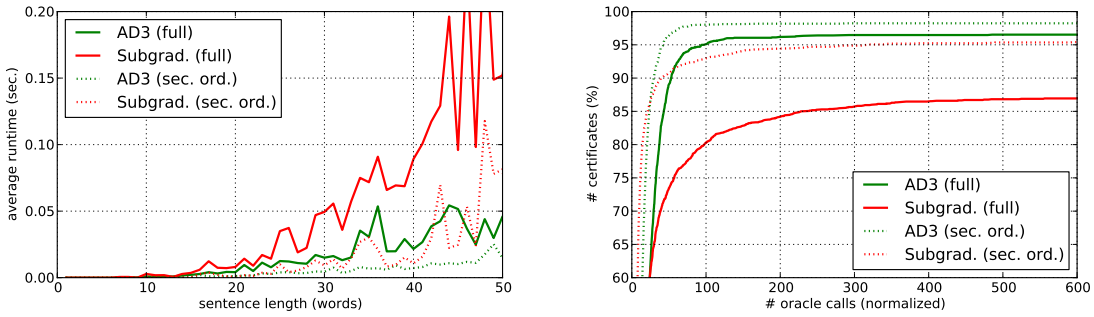

Figure 10: Left: average runtime in PTB §22, as a function of sentence length. Right: percentage of instances, as a function of the normalized number of COMPUTEMAP calls (see the caption of Figure 9), for which the exact solution was obtained along with a certificate of optimality. The maximum number of iterations is 2000 for both methods.

coordinate descent methods (such as MPLP), which can get stuck at suboptimal stationary points, and the PSDD algorithm, which is tied to the slow $O(1/\epsilon^2)$ convergence of subgradient methods.

Several "smoothing methods" have been proposed in the literature to obviate these drawbacks. Johnson et al. (2007) added an entropic regularization term to the dual objective (Eq. 11), opening the door for gradient methods; and Jojic et al. (2010) applied an accelerated gradient method to the smoothed problem (Nesterov, 1983), yielding a $O(1/\epsilon)$ iteration bound (the same asymptotic bound as AD^3, as established in Proposition 15). This method has been recently improved by (Savchynskyy et al., 2011), through adaptive smoothing and a dynamic estimation of the Lipschitz constant. In a related line of research, Hazan and Shashua (2010) proposed a class of norm-product message-passing algorithms that can be used for both marginal and MAP inference. Norm-product BP implements a primal-dual ascent scheme for optimizing a fractional entropy approximation, constructed as a linear combination of variable and factor entropic terms. For a proper choice of counting numbers, the resulting objective function is convex and smooth, and the amount of smoothness can be controlled by a temperature parameter τ. With $\tau = 0$, norm-product is similar to MPLP and can get stuck at a suboptimal solution; but with a positive τ, the norm-product algorithm is globally convergent to a solution which is $O(\tau)$-close to the LP-MAP optimal value.

Compared with AD^3, the smoothing-based methods mentioned above have the advantage that their local subproblems can typically be transformed into marginal inference problems, which in many cases can be solved with brute-force counting or dynamic programming. However, they also have important drawbacks. First, their precision depends critically on the temperature parameter; *e.g.*, the $O(1/\epsilon)$ iteration bound of Jojic et al. (2010) requires setting the temperature to $O(\epsilon)$, which scales the potentials by $O(1/\epsilon)$ and may lead to numerical instabilities. Second, the solution of the local subproblems are always *dense*; although some marginal values may be low, they are never exactly zero. This contrasts with the projected subgradient and the AD^3 algorithms, for which the solutions of the local subproblems are spanned by one or a small number of MAP configurations. As shown in the experimental section (Figure 9), caching these configurations across iterations may lead to great speedups.

While smoothing-based methods that use quadratic regularizers (as opposed to entropic ones) have also been proposed—most notably the proximal point method of Ravikumar et al. (2010)—these methods also have disadvantages over AD^3. The proximal point method of Ravikumar et al. (2010) for pairwise MRFs is a double-loop algorithm, where a penalty term with varying magnitude is added to the primal objective, and a globally smooth problem is solved iteratively in the inner loop, using cyclic Bregman projections. Applied to a general factor graph and using a quadratic penalty, the problems solved in the inner loop resemble the AD^3 subproblems, with an important difference: there is an extra Euclidean penalty of the form $\|\boldsymbol{q}_\alpha - \boldsymbol{q}_\alpha^{(t)}\|^2$. While this term makes the subproblems strongly convex, it also destroys the sparsity property mentioned in Proposition 11, which results in substantially more messages needing to be passed around (in particular, messages with size $|\mathcal{Y}_\alpha|$, which can be prohibitive for factors with large degree). A different strategy has been proposed by Pletscher and Wulff (2012), who combined the LP-MAP relaxation described here with a non-convex QP relaxation, which unlike other smoothing methods increases the effect of the penalty term through the progression of the algorithm.

Finally, it should be noted that other strategies have been recently proposed to overcome the weaknesses of coordinate descent algorithms and PSDD, which are not based on smoothing the dual objective. The fact that the PSDD algorithm has "no memory" across iterations (pointed out in the beginning of Section 4) has been addressed by Kappes et al. (2012) in their bundle method, which remembers past updates, at the cost of extra memory storage and more involved local subproblems. The fact that coordinate descent methods can get stuck in suboptimal solutions has been addressed by Schwing et al. (2012), who proposed a ϵ-descent strategy as a way to move away from corners, mixing coordinate and steepest descent steps; the latter, however, require solving QPs as an intermediate step.

During the preparation of this paper, and following our earlier work (Martins et al., 2010, 2011a), AD^3 has been successfully applied to several NLP problems (Martins et al., 2011b, 2013; Das et al., 2012; Almeida and Martins, 2013), and a few related methods have appeared. Meshi and Globerson (2011) also applied ADMM to MAP inference in graphical models, although addressing the dual problem (the one underlying the MPLP algorithm) rather than the primal. Yedidia et al. (2011) proposed the "divide-and-concur" algorithm for LDPC (low-density parity check) decoding, which shares aspects of AD^3, and can be seen as an instance of non-convex ADMM. Barman et al. (2011) proposed an algorithm analogous to AD^3 for the same LDPC decoding problem; their subproblems correspond to projections onto the parity polytope, for which they have derived an efficient algorithm. More recently, Fu et al. (2013) proposed a Bethe-ADMM procedure resembling AD^3, but with an inexact variant of ADMM that makes the subproblems become marginal computations. Recent work also addressed budget and knapsack constraints, important for dealing with cardinality-based potentials and to promote diversity (Tarlow et al., 2010; Almeida and Martins, 2013).

9. Conclusions

We introduced AD^3, a new LP-MAP inference algorithm based on the alternating directions method of multipliers (ADMM) (Glowinski and Marroco, 1975; Gabay and Mercier, 1976).

AD^3 enjoys the modularity of dual decomposition methods, but achieves faster consensus, by penalizing, for each subproblem, deviations from the current global solution. Using recent results, we showed that AD^3 converges to an ϵ-accurate solution with an iteration bound of $O(1/\epsilon)$. AD^3

can handle factor graphs with hard constraints in first-order logic, using efficient procedures for projecting onto the marginal polytopes of the corresponding factors. This opens the door for using AD^3 in problems with declarative constraints (Roth and Yih, 2004; Richardson and Domingos, 2006). A closed-form solution of the AD^3 subproblem was also derived for pairwise binary factors.

We introduced a new active set method for solving the AD^3 subproblems for arbitrary factors. This method requires only a local MAP oracle, as the PSDD algorithm. The active set method is particularly suitable for these problems, since it can take advantage of warm starting and it deals well with sparse solutions—which are guaranteed by Proposition 11. We also show how AD^3 can be wrapped in a branch-and-bound procedure to retrieve the exact MAP.

Experiments with synthetic and real-world datasets have shown that AD^3 is able to solve the LP-MAP problem more efficiently than other methods for a variety of problems, including MAP inference in Ising and Potts models, protein design, frame-semantic parsing, and dependency parsing.

Our contributions open several directions for future research. One possible extension is to replace the Euclidean penalty of ADMM by a general Mahalanobis distance. The convergence proofs can be trivially extended to Mahalanobis distances, since they correspond to an affine transformation of the subspace defined by the equality constraints of Eq. 15. Simple operations, such as scaling these constraints, do not affect the algorithms that are used to solve the subproblems, thus AD^3 can be generalized by including scaling parameters.

Since the AD^3 subproblems can be solved in parallel, significant speed-ups may be obtained in multi-core architectures or using GPU programming. This has been shown to be very useful for large-scale message-passing inference in graphical models (Felzenszwalb and Huttenlocher, 2006; Low et al., 2010; Schwing et al., 2011).

The branch-and-bound algorithm for obtaining the exact MAP deserves further experimental study, as similar approaches have been proven useful in MAP inference problems (Sun et al., 2012). An advantage of AD^3 is its ability to quickly produce sharp upper bounds. For many problems, there are effective rounding procedures that can also produce lower bounds, which can be exploited for guiding the search. There are also alternatives to branch-and-bound, such as tightening procedures (Sontag et al., 2008; Batra et al., 2011), which progressively add larger factors to decrease the duality gap. The variant of AD^3 with the active set method can be used to handle these larger factors.

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.

Appendix A. Proof of Convergence Rate of AD^3

In this appendix, we show the $O(1/\epsilon)$ convergence bound of the ADMM algorithm. We use a recent result established by Wang and Banerjee (2012) regarding convergence in a variational setting, from which we derive the convergence of ADMM in the dual objective. We then consider the special case of AD^3, interpreting the constants in the bound in terms of properties of the graphical model.

We start with the following proposition, which states the variational inequality associated with the Lagrangian saddle point problem associated with (14),

$$\min_{\boldsymbol{\lambda}\in\Lambda}\max_{\boldsymbol{q}\in\mathcal{Q},\boldsymbol{p}\in\mathcal{P}} L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}), \tag{63}$$

where $L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}) := f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) + \boldsymbol{\lambda}^\top(\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c})$ is the standard Lagrangian, and $\Lambda := \{\boldsymbol{\lambda} \mid \max_{\boldsymbol{q}\in\mathcal{Q},\boldsymbol{p}\in\mathcal{P}} L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}) < \infty\}$.

Proposition 12 (Variational inequality) *Let $\mathcal{W} := \mathcal{Q}\times\mathcal{P}\times\Lambda$. Given $\boldsymbol{w} = (\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}) \in \mathcal{W}$, define $h(\boldsymbol{w}) := f_1(\boldsymbol{q}) + f_2(\boldsymbol{p})$ and $F(\boldsymbol{w}) := (\mathbf{A}^\top\boldsymbol{\lambda}, \mathbf{B}^\top\boldsymbol{\lambda}, -(\mathbf{A}\boldsymbol{q}+\mathbf{B}\boldsymbol{p}-\boldsymbol{c}))$. Then, $\boldsymbol{w}^* := (\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}^*) \in \mathcal{W}$ is a primal-dual solution of Eq. 63 if and only if:*

$$\forall \boldsymbol{w}\in\mathcal{W},\quad h(\boldsymbol{w}) - h(\boldsymbol{w}^*) + (\boldsymbol{w}-\boldsymbol{w}^*)^\top F(\boldsymbol{w}^*) \le 0. \tag{64}$$

Proof: Assume $\boldsymbol{w}^*$ is a primal-dual solution of Eq. 63. Then, the saddle point conditions imply $L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}^*) \le L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}^*) \le L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda})$ for every $\boldsymbol{w} := (\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}) \in \mathcal{W}$. Hence:

$$\begin{aligned} 0 &\ge L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}^*) - L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}) \\ &= f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) + \boldsymbol{\lambda}^{*\top}(\mathbf{A}\boldsymbol{q}+\mathbf{B}\boldsymbol{p}-\boldsymbol{c}) - f_1(\boldsymbol{q}^*) - f_2(\boldsymbol{p}^*) - \boldsymbol{\lambda}^\top(\mathbf{A}\boldsymbol{q}^*+\mathbf{B}\boldsymbol{p}^*-\boldsymbol{c}) \\ &= h(\boldsymbol{w}) - h(\boldsymbol{w}^*) + (\boldsymbol{w}-\boldsymbol{w}^*)^\top F(\boldsymbol{w}^*). \end{aligned} \tag{65}$$

Conversely, let $\boldsymbol{w}^*$ satisfy Eq. 64. Taking $\boldsymbol{w} = (\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda})$, we obtain $L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}^*) \le L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda})$. Taking $\boldsymbol{w} = (\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}^*)$, we obtain $L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda}^*) \le L(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}^*)$. Hence $(\boldsymbol{q}^*,\boldsymbol{p}^*,\boldsymbol{\lambda}^*)$ is a saddle point, and therefore a primal-dual solution. ■

The next result, due to Wang and Banerjee (2012) and related to previous work by He and Yuan (2011), concerns the convergence rate of ADMM in terms of the variational inequality stated above.

Proposition 13 (Variational convergence rate) *Assume the conditions in Proposition 7. Let $\bar{\boldsymbol{w}}_T = \frac{1}{T}\sum_{t=1}^T \boldsymbol{w}^t$, where $\boldsymbol{w}^t := (\boldsymbol{q}^t,\boldsymbol{p}^t,\boldsymbol{\lambda}^t)$ are the ADMM iterates with $\boldsymbol{\lambda}^0 = \mathbf{0}$. Then, after T iterations:*

$$\forall \boldsymbol{w}\in\mathcal{W},\quad h(\boldsymbol{w}) - h(\bar{\boldsymbol{w}}_T) + (\boldsymbol{w}-\bar{\boldsymbol{w}}_T)^\top F(\bar{\boldsymbol{w}}_T) \le C/T, \tag{66}$$

where $C = \frac{\eta}{2}\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2 + \frac{1}{2\eta}\|\boldsymbol{\lambda}\|^2$ is independent of T.

Proof: From the variational inequality associated with the $\boldsymbol{q}$-update (19) we have for every $\boldsymbol{q}\in\mathcal{Q}$[11]

$$\begin{aligned} 0 &\ge \nabla_{\boldsymbol{q}} L_\eta(\boldsymbol{q}^{t+1},\boldsymbol{p}^t,\boldsymbol{\lambda}^t)^\top(\boldsymbol{q}-\boldsymbol{q}^{t+1}) \\ &= \nabla f_1(\boldsymbol{q}^{t+1})^\top(\boldsymbol{q}-\boldsymbol{q}^{t+1}) + (\boldsymbol{q}-\boldsymbol{q}^{t+1})^\top\mathbf{A}^\top(\boldsymbol{\lambda}^t - \eta(\mathbf{A}\boldsymbol{q}^{t+1}+\mathbf{B}\boldsymbol{p}^t-\boldsymbol{c})) \\ &\ge^{(i)} f_1(\boldsymbol{q}) - f_1(\boldsymbol{q}^{t+1}) + (\boldsymbol{q}-\boldsymbol{q}^{t+1})^\top\mathbf{A}^\top(\boldsymbol{\lambda}^t - \eta(\mathbf{A}\boldsymbol{q}^{t+1}+\mathbf{B}\boldsymbol{p}^t-\boldsymbol{c})) \\ &=^{(ii)} f_1(\boldsymbol{q}) - f_1(\boldsymbol{q}^{t+1}) + (\boldsymbol{q}-\boldsymbol{q}^{t+1})^\top\mathbf{A}^\top\boldsymbol{\lambda}^{t+1} - \eta(\mathbf{A}(\boldsymbol{q}-\boldsymbol{q}^{t+1}))^\top\mathbf{B}(\boldsymbol{p}^t-\boldsymbol{p}^{t+1}), \end{aligned} \tag{67}$$

11. Given a problem $\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$, where f is concave and differentiable and $\mathcal{X}$ is convex, a point $\boldsymbol{x}^* \in \mathcal{X}$ is a maximizer iff it satisfies the variational inequality $\nabla f(\boldsymbol{x}^*)^\top(\boldsymbol{x}-\boldsymbol{x}^*) \le 0$ for all $\boldsymbol{x}\in\mathcal{X}$ (Facchinei and Pang, 2003).

where in (i) we have used the concavity of f_1, and in (ii) we used Eq. 18 for the $\boldsymbol{\lambda}$-updates. Similarly, the variational inequality associated with the $\boldsymbol{p}$-updates (20) yields, for every $\boldsymbol{p} \in \mathcal{P}$:

$$\begin{aligned} 0 &\geq \nabla_{\boldsymbol{p}} L_\eta(\boldsymbol{q}^{t+1}, \boldsymbol{p}^{t+1}, \boldsymbol{\lambda}^t)^\top (\boldsymbol{p} - \boldsymbol{p}^{t+1}) \\ &= \nabla f_2(\boldsymbol{p}^{t+1})^\top (\boldsymbol{p} - \boldsymbol{p}^{t+1}) + (\boldsymbol{p} - \boldsymbol{p}^{t+1})^\top \mathbf{B}^\top (\boldsymbol{\lambda}^t - \eta(\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c})) \\ &\geq^{(i)} f_2(\boldsymbol{p}) - f_2(\boldsymbol{p}^{t+1}) + (\boldsymbol{p} - \boldsymbol{p}^{t+1})^\top \mathbf{B}^\top \boldsymbol{\lambda}^{t+1}, \end{aligned} \tag{68}$$

where in (i) we have used the concavity of f_2. Summing (67) and (68), and noting again that $\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t - \eta(\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c})$, we obtain, for every $\boldsymbol{w} \in \mathcal{W}$,

$$\begin{aligned} & h(\boldsymbol{w}^{t+1}) - h(\boldsymbol{w}) + (\boldsymbol{w}^{t+1} - \boldsymbol{w})^\top F(\boldsymbol{w}^{t+1}) \\ & \quad \geq -\eta \mathbf{A}(\boldsymbol{q} - \boldsymbol{q}^{t+1})^\top \mathbf{B}(\boldsymbol{p}^t - \boldsymbol{p}^{t+1}) - \eta^{-1}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{t+1})^\top (\boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^t). \end{aligned} \tag{69}$$

We next rewrite the two terms in the right hand side:

$$\begin{aligned} \eta \mathbf{A}(\boldsymbol{q} - \boldsymbol{q}^{t+1})^\top \mathbf{B}(\boldsymbol{p}^t - \boldsymbol{p}^{t+1}) &= \frac{\eta}{2} \left(\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^t - \boldsymbol{c}\|^2 - \|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c}\|^2 \right. \\ &+ \left. \|\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c}\|^2 - \|\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^t - \boldsymbol{c}\|^2 \right); \end{aligned} \tag{70}$$

$$\eta^{-1}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{t+1})^\top (\boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^t) = \frac{1}{2\eta} \left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^t\|^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{t+1}\|^2 - \|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t+1}\|^2 \right). \tag{71}$$

Summing (69) over t and noting that $\eta^{-1}\|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t+1}\|^2 = \eta\|\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^{t+1} - \boldsymbol{c}\|^2$:

$$\begin{aligned} & \sum_{t=0}^{T-1} \left(h(\boldsymbol{w}^{t+1}) - h(\boldsymbol{w}) + (\boldsymbol{w}^{t+1} - \boldsymbol{w})^\top F(\boldsymbol{w}^{t+1}) \right) \\ & \quad \geq -\frac{\eta}{2} \left(\|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2 - \|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^T - \boldsymbol{c}\|^2 - \sum_{t=0}^{T-1} \|\mathbf{A}\boldsymbol{q}^{t+1} + \mathbf{B}\boldsymbol{p}^t - \boldsymbol{c}\|^2 \right) \\ & \qquad -\frac{1}{2\eta} \left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^0\|^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\|^2 \right) \\ & \quad \geq -\frac{\eta}{2} \|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2 - \frac{1}{2\eta} \|\boldsymbol{\lambda}\|^2. \end{aligned} \tag{72}$$

From the concavity of h, we have that $h(\bar{\boldsymbol{w}}_T) \geq \frac{1}{T} \sum_{t=0}^{T-1} h(\boldsymbol{w}^{t+1})$. Note also that, for every $\tilde{\boldsymbol{w}}$, the function $\boldsymbol{w} \mapsto (\boldsymbol{w} - \tilde{\boldsymbol{w}})^\top F(\boldsymbol{w})$ is affine:

$$\begin{aligned} (\boldsymbol{w} - \tilde{\boldsymbol{w}})^\top F(\boldsymbol{w}) &= (\boldsymbol{q} - \tilde{\boldsymbol{q}})^\top \mathbf{A}^\top \boldsymbol{\lambda} + (\boldsymbol{p} - \tilde{\boldsymbol{p}})^\top \mathbf{B}^\top \boldsymbol{\lambda} - (\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}})^\top (\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}) \\ &= -(\mathbf{A}\tilde{\boldsymbol{q}} + \mathbf{B}\tilde{\boldsymbol{p}} - \boldsymbol{c})^\top \boldsymbol{\lambda} + \tilde{\boldsymbol{\lambda}}^\top (\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}) \\ &= F(\tilde{\boldsymbol{w}})^\top \boldsymbol{w} - \boldsymbol{c}^\top \tilde{\boldsymbol{\lambda}}. \end{aligned} \tag{73}$$

As a consequence, $\frac{1}{T} \sum_{t=0}^{T-1} \left(h(\boldsymbol{w}^{t+1}) + (\boldsymbol{w}^{t+1} - \boldsymbol{w})^\top F(\boldsymbol{w}^{t+1}) \right) \leq h(\bar{\boldsymbol{w}}_T) + (\bar{\boldsymbol{w}}_T - \boldsymbol{w})^\top F(\bar{\boldsymbol{w}}_T)$, and from (72), we have that $h(\boldsymbol{w}) - h(\bar{\boldsymbol{w}}_T) + (\boldsymbol{w} - \bar{\boldsymbol{w}}_T)^\top F(\bar{\boldsymbol{w}}_T) \leq C/T$, with C as in Eq. 66. Note also that, since Λ is convex, we must have $\bar{\boldsymbol{\lambda}}_T \in \Lambda$. ■

Next, we use the bound in Proposition 13 to derive a convergence rate for the dual problem.

Proposition 14 (Dual convergence rate) *Assume the conditions stated in Proposition 13, with $\bar{\boldsymbol{w}}_T$ defined analogously. Let $g : \Lambda \to \mathbb{R}$ be the dual objective function, $g(\boldsymbol{\lambda}) := \max_{\boldsymbol{q}\in\mathcal{Q},\boldsymbol{p}\in\mathcal{P}} L(\boldsymbol{q},\boldsymbol{p},\boldsymbol{\lambda})$, and let $\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda}\in\Lambda} g(\boldsymbol{\lambda})$ be a dual solution. Then, after T iterations, ADMM achieves an $O(\frac{1}{T})$-accurate solution $\bar{\boldsymbol{\lambda}}_T$:*

$$g(\boldsymbol{\lambda}^*) \leq g(\bar{\boldsymbol{\lambda}}_T) \leq g(\boldsymbol{\lambda}^*) + \frac{C}{T}, \tag{74}$$

where the constant C is given by

$$C \;=\; \frac{5\eta}{2}\left(\max_{\boldsymbol{q}\in\mathcal{Q}} \|\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2\right) + \frac{5}{2\eta}\|\boldsymbol{\lambda}^*\|^2. \tag{75}$$

Proof: By applying Proposition 13 to $\boldsymbol{w} = (\bar{\boldsymbol{q}}_T, \bar{\boldsymbol{p}}_T, \boldsymbol{\lambda})$ we obtain for arbitrary $\boldsymbol{\lambda} \in \Lambda$:

$$-(\boldsymbol{\lambda} - \bar{\boldsymbol{\lambda}}_T)^\top(\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\bar{\boldsymbol{p}}_T - \boldsymbol{c}) \leq O(1/T). \tag{76}$$

By applying Proposition 13 to $\boldsymbol{w} = (\boldsymbol{q}, \boldsymbol{p}, \bar{\boldsymbol{\lambda}}_T)$ we obtain for arbitrary $\boldsymbol{q} \in \mathcal{Q}$ and $\boldsymbol{p} \in \mathcal{P}$:

$$\begin{aligned} & f_1(\bar{\boldsymbol{q}}_T) + f_2(\bar{\boldsymbol{p}}_T) + (\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\bar{\boldsymbol{p}}_T - \boldsymbol{c})^\top\bar{\boldsymbol{\lambda}}_T \\ \geq\;& f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) + (\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c})^\top\bar{\boldsymbol{\lambda}}_T - O(1/T). \end{aligned} \tag{77}$$

In particular, let $g(\bar{\boldsymbol{\lambda}}_T) = \max_{\boldsymbol{q}\in\mathcal{Q},\boldsymbol{p}\in\mathcal{P}} L(\boldsymbol{q},\boldsymbol{p},\bar{\boldsymbol{\lambda}}_T) = L(\widehat{\boldsymbol{q}}_T, \widehat{\boldsymbol{p}}_T, \bar{\boldsymbol{\lambda}}_T)$ be the value of the dual objective at $\bar{\boldsymbol{\lambda}}_T$, where $(\widehat{\boldsymbol{q}}_T, \widehat{\boldsymbol{p}}_T)$ are the corresponding maximizers. We then have:

$$f_1(\bar{\boldsymbol{q}}_T) + f_2(\bar{\boldsymbol{p}}_T) + (\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\bar{\boldsymbol{p}}_T - \boldsymbol{c})^\top\bar{\boldsymbol{\lambda}}_T \geq g(\bar{\boldsymbol{\lambda}}_T) - O(1/T). \tag{78}$$

Finally we have (letting $\boldsymbol{w}^* = (\boldsymbol{q}^*, \boldsymbol{p}^*, \boldsymbol{\lambda}^*)$ be the optimal primal-dual solution):

$$\begin{aligned} g(\boldsymbol{\lambda}^*) \;&=\; \max_{\boldsymbol{q}\in\mathcal{Q},\boldsymbol{p}\in\mathcal{P}} f_1(\boldsymbol{q}) + f_2(\boldsymbol{p}) + \boldsymbol{\lambda}^{*\top}(\mathbf{A}\boldsymbol{q} + \mathbf{B}\boldsymbol{p} - \boldsymbol{c}) \\ &\geq\; f_1(\bar{\boldsymbol{q}}_T) + f_2(\bar{\boldsymbol{p}}_T) + \boldsymbol{\lambda}^{*\top}(\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\bar{\boldsymbol{p}}_T - \boldsymbol{c}) \\ &\geq^{(i)}\; f_1(\bar{\boldsymbol{q}}_T) + f_2(\bar{\boldsymbol{p}}_T) + \bar{\boldsymbol{\lambda}}_T^\top(\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\bar{\boldsymbol{p}}_T - \boldsymbol{c}) - O(1/T) \\ &\geq^{(ii)}\; g(\bar{\boldsymbol{\lambda}}_T) - O(1/T), \end{aligned} \tag{79}$$

where in (i) we used Eq. 76 and in (ii) we used Eq. 78. By definition of $\boldsymbol{\lambda}^*$, we also have $g(\bar{\boldsymbol{\lambda}}_T) \geq g(\boldsymbol{\lambda}^*)$. Since we applied Proposition 13 twice, the constant inside the O-notation becomes

$$C = \frac{\eta}{2}\left(\|\mathbf{A}\bar{\boldsymbol{q}}_T + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2 + \|\mathbf{A}\widehat{\boldsymbol{q}}_T + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2\right) + \frac{1}{2\eta}\left(\|\boldsymbol{\lambda}^*\|^2 + \|\bar{\boldsymbol{\lambda}}_T\|^2\right). \tag{80}$$

Even though C depends on $\bar{\boldsymbol{q}}_T$, $\widehat{\boldsymbol{q}}_T$, and $\bar{\boldsymbol{\lambda}}_T$, it is easy to obtain an upper bound on C when $\mathcal{Q}$ is a bounded set, using the fact that the sequence $(\boldsymbol{\lambda}^t)_{t\in\mathbb{N}}$ is bounded by a constant, which implies that the average $\bar{\boldsymbol{\lambda}}_T$ is also bounded. Indeed, from Boyd et al. (2011, p.107), we have that

$$V^t := \eta^{-1}\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^t\|^2 + \eta\|\mathbf{B}(\boldsymbol{p}^* - \boldsymbol{p}^t)\|^2 \tag{81}$$

is a Lyapunov function, *i.e.*, $0 \leq V^{t+1} \leq V^t$ for every $t \in \mathbb{N}$. This implies that $V^t \leq V^0 = \eta^{-1}\|\boldsymbol{\lambda}^*\|^2 + \eta\|\mathbf{B}(\boldsymbol{p}^* - \boldsymbol{p}^0)\|^2$; since $V^t \geq \eta^{-1}\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^t\|^2$, we can replace above and write:

$$\begin{aligned} 0 \;&\geq\; \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^t\|^2 - \|\boldsymbol{\lambda}^*\|^2 - \eta^2\|\mathbf{B}(\boldsymbol{p}^* - \boldsymbol{p}^0)\|^2 = \|\boldsymbol{\lambda}^t\|^2 - 2\boldsymbol{\lambda}^{*\top}\boldsymbol{\lambda}^t - \eta^2\|\mathbf{B}(\boldsymbol{p}^* - \boldsymbol{p}^0)\|^2 \\ &\geq\; \|\boldsymbol{\lambda}^t\|^2 - 2\|\boldsymbol{\lambda}^*\|\|\boldsymbol{\lambda}^t\| - \eta^2\|\mathbf{B}(\boldsymbol{p}^* - \boldsymbol{p}^0)\|^2, \end{aligned} \tag{82}$$

where in the last line we invoked the Cauchy-Schwarz inequality. Solving the quadratic equation, we obtain $\|\boldsymbol{\lambda}^t\| \leq \|\boldsymbol{\lambda}^*\| + \sqrt{\|\boldsymbol{\lambda}^*\|^2 + \eta^2\|\mathbf{B}(\boldsymbol{p}^0 - \boldsymbol{p}^*)\|^2}$, which in turn implies

$$\begin{aligned}\|\boldsymbol{\lambda}^t\|^2 &\leq 2\|\boldsymbol{\lambda}^*\|^2 + \eta^2\|\mathbf{B}(\boldsymbol{p}^0 - \boldsymbol{p}^*)\|^2 + 2\|\boldsymbol{\lambda}^*\|\sqrt{\|\boldsymbol{\lambda}^*\|^2 + \eta^2\|\mathbf{B}(\boldsymbol{p}^0 - \boldsymbol{p}^*)\|^2} \\ &\leq 2\|\boldsymbol{\lambda}^*\|^2 + \eta^2\|\mathbf{B}(\boldsymbol{p}^0 - \boldsymbol{p}^*)\|^2 + 2(\|\boldsymbol{\lambda}^*\|^2 + \eta^2\|\mathbf{B}(\boldsymbol{p}^0 - \boldsymbol{p}^*)\|^2) \\ &= 4\|\boldsymbol{\lambda}^*\|^2 + 3\eta^2\|\mathbf{A}\boldsymbol{q}^* + \mathbf{B}\boldsymbol{p}^0 - \boldsymbol{c}\|^2, \end{aligned} \tag{83}$$

the last line following from $\mathbf{A}\boldsymbol{q}^* + \mathbf{B}\boldsymbol{p}^* = \boldsymbol{c}$. Replacing (83) in (80) yields the result. ■

Finally, we will see how the bounds above apply to the AD^3 algorithm.

Proposition 15 (Dual convergence rate of AD^3) *After T iterations of AD^3, we achieve an $O(\frac{1}{T})$-accurate solution $\bar{\boldsymbol{\lambda}}_T := \sum_{t=0}^{T-1} \boldsymbol{\lambda}^{(t)}$:*

$$g(\boldsymbol{\lambda}^*) \leq g(\bar{\boldsymbol{\lambda}}_T) \leq g(\boldsymbol{\lambda}^*) + \frac{C}{T}, \tag{84}$$

where $C = \frac{5\eta}{2}\sum_i |\partial(i)|(1 - |\mathcal{Y}_i|^{-1}) + \frac{5}{2\eta}\|\boldsymbol{\lambda}^*\|^2$ is a constant independent of T.

Proof: With the uniform initialization of the $\boldsymbol{p}$-variables in AD^3, the first term of Eq. 75 is maximized by a choice of $\boldsymbol{q}_\alpha$-variables that puts all mass in a single configuration. That is:

$$\max_{\boldsymbol{q}_{i\alpha}} \|\boldsymbol{q}_{i\alpha} - |\mathcal{Y}_i|^{-1}\mathbf{1}\|^2 = \big((1 - |\mathcal{Y}_i|^{-1})^2 + (|\mathcal{Y}_i| - 1)|\mathcal{Y}_i|^{-2}\big) = 1 - |\mathcal{Y}_i|^{-1}. \tag{85}$$

This leads to the desired bound. ■

Appendix B. Derivation of Solutions for AD^3 Subproblems

B.1 Binary Pairwise Factors

In this section, we prove Proposition 10. Let us first assume that $c_{12} \geq 0$. In this case, the constraints $z_{12} \geq z_1 + z_2 - 1$ and $z_{12} \geq 0$ in (33) are always inactive and the problem can be simplified to:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(z_1 - c_1)^2 + \frac{1}{2}(z_2 - c_2)^2 - c_{12}z_{12} \\ \text{with respect to} \quad & z_1, z_2, z_{12} \\ \text{subject to} \quad & z_{12} \leq z_1, \quad z_{12} \leq z_2, \quad z_1 \in [0,1], \quad z_2 \in [0,1]. \end{aligned} \tag{86}$$

If $c_{12} = 0$, the problem becomes separable, and a solution is

$$z_1^* = [c_1]_{\mathbb{U}}, \quad z_2^* = [c_2]_{\mathbb{U}}, \quad z_{12}^* = \min\{z_1^*, z_2^*\}, \tag{87}$$

which complies with Eq. 37. We next analyze the case where $c_{12} > 0$. The Lagrangian of (86) is:

$$\begin{aligned} L(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\nu}) &= \frac{1}{2}(z_1 - c_1)^2 + \frac{1}{2}(z_2 - c_2)^2 - c_{12}z_{12} + \mu_1(z_{12} - z_1) + \mu_2(z_{12} - z_2) \\ &\quad -\lambda_1 z_1 - \lambda_2 z_2 + \nu_1(z_1 - 1) + \nu_2(z_2 - 1). \end{aligned} \tag{88}$$

At optimality, the following Karush-Kuhn-Tucker (KKT) conditions need to be satisfied:

$$\nabla_{z_1} L(\boldsymbol{z}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = 0 \quad \Rightarrow \quad z_1^* = c_1 + \mu_1^* + \lambda_1^* - \nu_1^* \tag{89}$$

$$\nabla_{z_2} L(\boldsymbol{z}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = 0 \quad \Rightarrow \quad z_2^* = c_2 + \mu_2^* + \lambda_2^* - \nu_2^* \tag{90}$$

$$\nabla_{z_{12}} L(\boldsymbol{z}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = 0 \quad \Rightarrow \quad c_{12} = \mu_1^* + \mu_2^* \tag{91}$$

$$\lambda_1^* z_1^* = 0, \quad \lambda_2^* z_2^* = 0 \tag{92}$$

$$\mu_1^*(z_{12}^* - z_1^*) = 0, \quad \mu_2^*(z_{12}^* - z_2^*) = 0 \tag{93}$$

$$\nu_1^*(z_1^* - 1) = 0, \quad \nu_2^*(z_2^* - 1) = 0 \tag{94}$$

$$\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^* \geq 0 \tag{95}$$

$$z_{12}^* \leq z_1^*, \quad z_{12}^* \leq z_2^*, \quad z_1^* \in [0,1], \quad z_2^* \in [0,1] \tag{96}$$

We are going to consider three cases separately:

1. $\boxed{z_1^* > z_2^*}$ From the primal feasibility conditions (96), this implies $z_1^* > 0$, $z_2^* < 1$, and $z_{12}^* < z_1^*$. Complementary slackness (92,93,94) implies in turn $\lambda_1^* = 0$, $\nu_2^* = 0$, and $\mu_1^* = 0$. From (91) we have $\mu_2^* = c_{12}$. Since we are assuming $c_{12} > 0$, we then have $\mu_2^* > 0$, and complementary slackness (93) implies $z_{12}^* = z_2^*$. Plugging this into (89)–(90) we obtain

$$z_1^* = c_1 - \nu_1^* \leq c_1, \qquad z_2^* = c_2 + \lambda_2^* + c_{12} \geq c_2 + c_{12}. \tag{97}$$

   Now we have the following:

   - Either $z_1^* = 1$ or $z_1^* < 1$. In the latter case, $\nu_1^* = 0$ due to (94), hence $z_1^* = c_1$. Since in any case we must have $z_1^* \leq c_1$, we conclude that $z_1^* = \min\{c_1, 1\}$.
   - Either $z_2^* = 0$ or $z_2^* > 0$. In the latter case, $\lambda_2^* = 0$ due to (92), hence $z_2^* = c_2 + c_{12}$. Since in any case we must have $z_2^* \geq \lambda_2$, we conclude that $z_2^* = \max\{0, c_2 + c_{12}\}$.

   In sum:

$$z_1^* = \min\{c_1, 1\}, \quad z_{12}^* = z_2^* = \max\{0, c_2 + c_{12}\}, \tag{98}$$

   and our assumption $z_1^* > z_2^*$ can only be valid if $c_1 > c_2 + c_{12}$.

2. $\boxed{z_1^* < z_2^*}$ By symmetry, we have

$$z_2^* = \min\{c_2, 1\}, \quad z_{12}^* = z_1^* = \max\{0, c_1 + c_{12}\}, \tag{99}$$

   and our assumption $z_1^* < z_2^*$ can only be valid if $c_2 > c_1 + c_{12}$.

3. $\boxed{z_1^* = z_2^*}$ In this case, it is easy to verify that we must have $z_{12}^* = z_1^* = z_2^*$, and we can rewrite our optimization problem in terms of one variable only (call it z). The problem becomes that of minimizing $\frac{1}{2}(z - c_1)^2 + \frac{1}{2}(z - c_2)^2 - c_{12} z$, which equals a constant plus $(z - \frac{c_1 + c_2 + c_{12}}{2})^2$, subject to $z \in \mathbb{U} \triangleq [0, 1]$. Hence:

$$z_{12}^* = z_1^* = z_2^* = [(c_1 + c_2 + c_{12})/2]_{\mathbb{U}}\,. \tag{100}$$

Putting all the pieces together, we obtain the solution displayed in Eq. 37.

It remains to address the case where $c_{12} < 0$. By redefining $c_1' = c_1 + c_{12}$, $c_2' = 1 - c_2$, $c_{12}' = -c_{12}$, $z_2' = 1 - z_2$, and $z_{12}' = z_1 - z_{12}$, we can reduce (33) to the form in (86). Substituting back in Eq. 37, we obtain the solution displayed in Eq. 38.

B.2 Marginal Polytope of Hard Constraint Factors

The following proposition establishes that the marginal polytope of a hard constraint factor is the convex hull of its acceptance set.

Proposition 16 *Let α be a binary hard constraint factor with degree K, and consider the set of all possible distributions $\mathbb{P}(\boldsymbol{Y}_\alpha)$ which satisfy $\mathbb{P}(\boldsymbol{Y}_\alpha = \boldsymbol{y}_\alpha) = 0$ for every $\boldsymbol{y}_\alpha \notin \mathcal{S}_\alpha$. Then, the set of possible marginals realizable for some distribution in that set is given by*

$$\begin{aligned} \mathcal{Z}_\alpha &:= \left\{ (q_{1\alpha}(1), \ldots, q_{K\alpha}(1)) \;\middle|\; \boldsymbol{q}_{i\alpha} = \mathbf{M}_{i\alpha}\boldsymbol{q}_\alpha, \text{ for some } \boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|} \text{ s.t. } q_\alpha(\boldsymbol{y}_\alpha) = 0, \forall \boldsymbol{y}_\alpha \notin \mathcal{S}_\alpha \right\} \\ &= \operatorname{conv} \mathcal{S}_\alpha. \end{aligned} \tag{101}$$

Proof: From the fact that we are constraining $q_\alpha(\boldsymbol{y}_\alpha) = 0, \forall \boldsymbol{y}_\alpha \notin \mathcal{S}_\alpha$, it follows:

$$\begin{aligned} \mathcal{Z}_\alpha &= \left\{ \boldsymbol{z} \geq 0 \;\middle|\; \exists \boldsymbol{q}_\alpha \geq 0 \text{ s.t. } \forall i \in \partial(\alpha), z_i = \sum_{\substack{\boldsymbol{y}_\alpha \in \mathcal{S}_\alpha \\ \boldsymbol{y}_i = 1}} q_\alpha(\boldsymbol{y}_\alpha) = 1 - \sum_{\substack{\boldsymbol{y}_\alpha \in \mathcal{S}_\alpha \\ \boldsymbol{y}_i = 0}} q_\alpha(\boldsymbol{y}_\alpha) \right\} \\ &= \left\{ \boldsymbol{z} \geq 0 \;\middle|\; \exists \boldsymbol{q}_\alpha \geq 0, \sum_{\boldsymbol{y}_\alpha \in \mathcal{S}_\alpha} q_\alpha(\boldsymbol{y}_\alpha) = 1 \text{ s.t. } \boldsymbol{z} = \sum_{\boldsymbol{y}_\alpha \in \mathcal{S}_\alpha} q_\alpha(\boldsymbol{y}_\alpha)\boldsymbol{y}_\alpha \right\} \\ &= \operatorname{conv} \mathcal{S}_\alpha. \end{aligned} \tag{102}$$

■

For hard constraint factors, the AD^3 subproblems take the following form (cf. Eq. 32):

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2} \sum_{i \in \partial(\alpha)} \|\boldsymbol{q}_{i\alpha} - \boldsymbol{a}_i\|^2 \quad \text{with respect to} \quad \boldsymbol{q}_\alpha \in \Delta^{|\mathcal{Y}_\alpha|}, \; \boldsymbol{q}_{i\alpha} \in \mathbb{R}^{|\mathcal{Y}_i|}, \; \forall i \in \partial(\alpha) \\ &\text{subject to} \quad \boldsymbol{q}_{i\alpha} = \mathbf{M}_{i\alpha}\boldsymbol{q}_\alpha, \quad q_\alpha(\boldsymbol{y}_\alpha) = 0, \; \forall \boldsymbol{y}_\alpha \neq \mathcal{S}_\alpha. \end{aligned} \tag{103}$$

From Proposition 16, and making use of a reduced parametrization, noting that $\|\boldsymbol{q}_{i\alpha} - \boldsymbol{a}_i\|^2 = (q_{i\alpha}(1) - a_i(1))^2 + (1 - q_{i\alpha}(1) - a_i(0))^2$, which equals a constant plus $2(q_{i\alpha}(1) - (a_i(1) + 1 - a_i(0))/2)^2$, we have that this problem is equivalent to:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{z}_0\|^2 \quad \text{with respect to} \quad \boldsymbol{z} \in \mathcal{Z}_\alpha, \tag{104}$$

where $z_{0i} := (a_i(1) + 1 - a_i(0))/2$, for each $i \in \partial(\alpha)$.

B.3 XOR Factor

For the XOR factor, the quadratic problem in Eq. 32 reduces to that of *projecting onto the simplex*. That problem is well-known in the optimization community (see, *e.g.*, Brucker 1984; Michelot 1986); by writing the KKT conditions, it is simple to show that the solution $\boldsymbol{z}^*$ is a soft-thresholding of $\boldsymbol{z}_0$, and therefore the problem can be reduced to that of finding the right threshold. Algorithm 4 provides an efficient procedure; it requires a sort operation, which renders its cost $O(K \log K)$. A proof of correctness appears in Duchi et al. (2008).[12]

12. This cost can be reduced to $O(K)$ using linear-time selection algorithms (Pardalos and Kovoor, 1990).

Algorithm 4 Projection onto simplex (Duchi et al., 2008)

Input: $\boldsymbol{z}_0$
Sort $\boldsymbol{z}_0$ into $\boldsymbol{y}_0$: $y_1 \geq \ldots \geq y_K$
Find $\rho = \max\left\{j \in [K] \mid y_{0j} - \frac{1}{j}\left(\left(\sum_{r=1}^{j} y_{0r}\right) - 1\right) > 0\right\}$
Define $\tau = \frac{1}{\rho}\left(\sum_{r=1}^{\rho} y_{0r} - 1\right)$
Output: $\boldsymbol{z}$ defined as $z_i = \max\{z_{0i} - \tau, 0\}$.

B.4 OR Factor

The following procedure can be used for computing a projection onto $\mathcal{Z}_{\mathrm{OR}}$:

1. Set $\tilde{\boldsymbol{z}}$ as the projection of $\boldsymbol{z}_0$ onto the unit cube. This can be done by clipping each coordinate to the unit interval $\mathbb{U} = [0, 1]$, i.e., by setting $\tilde{z}_i = [z_{0i}]_{\mathbb{U}} = \min\{1, \max\{0, z_{0i}\}\}$. If $\sum_{i=1}^{K} \tilde{z}_i \geq 1$, then return $\tilde{\boldsymbol{z}}$. Else go to step 2.

2. Return the projection of $\boldsymbol{z}_0$ onto the simplex (use Algorithm 4).

The correctness of this procedure is justified by the following lemma:

Lemma 17 (Sifting Lemma.) *Consider a problem of the form*

$$P: \qquad \min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \quad \textit{subject to} \quad g(\boldsymbol{x}) \leq 0, \tag{105}$$

where $\mathcal{X}$ is nonempty convex subset of $\mathbb{R}^D$ and $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$ are convex functions. Suppose that the problem (105) *is feasible and bounded below, and let $\mathcal{A}$ be the set of solutions of the* relaxed problem $\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$*, i.e. $\mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}'),\ \forall \boldsymbol{x}' \in \mathcal{X}\}$. Then:*

1. *if for some $\tilde{\boldsymbol{x}} \in \mathcal{A}$ we have $g(\tilde{\boldsymbol{x}}) \leq 0$, then $\tilde{\boldsymbol{x}}$ is also a solution of the original problem P;*

2. *otherwise (if for all $\tilde{\boldsymbol{x}} \in \mathcal{A}$ we have $g(\tilde{\boldsymbol{x}}) > 0$), then the inequality constraint is necessarily active in P, i.e., problem P is equivalent to $\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ subject to $g(\boldsymbol{x}) = 0$.*

Proof: Let f^* be the optimal value of P. The first statement is obvious: since $\tilde{\boldsymbol{x}}$ is a solution of a relaxed problem we have $f(\tilde{\boldsymbol{x}}) \leq f^*$; hence if $\tilde{\boldsymbol{x}}$ is feasible this becomes an equality. For the second statement, assume that $\exists \boldsymbol{x} \in \mathcal{X}$ subject to $g(\boldsymbol{x}) < 0$ (otherwise, the statement holds trivially). The nonlinear Farkas' lemma (Bertsekas et al., 2003, Prop. 3.5.4, p. 204) implies that there exists some $\lambda^* \geq 0$ subject to $f(\boldsymbol{x}) - f^* + \lambda^* g(\boldsymbol{x}) \geq 0$ holds for all $\boldsymbol{x} \in \mathcal{X}$. In particular, this also holds for an optimal $\boldsymbol{x}^*$ (i.e., such that $f^* = f(\boldsymbol{x}^*)$), which implies that $\lambda^* g(\boldsymbol{x}^*) \geq 0$. However, since $\lambda^* \geq 0$ and $g(\boldsymbol{x}^*) \leq 0$ (since $\boldsymbol{x}^*$ has to be feasible), we also have $\lambda^* g(\boldsymbol{x}^*) \leq 0$, *i.e.*, $\lambda^* g(\boldsymbol{x}^*) = 0$. Now suppose that $\lambda^* = 0$. Then we have $f(\boldsymbol{x}) - f^* \geq 0$, $\forall \boldsymbol{x} \in \mathcal{X}$, which implies that $\boldsymbol{x}^* \in \mathcal{A}$ and contradicts the assumption that $g(\tilde{\boldsymbol{x}}) > 0, \forall \tilde{\boldsymbol{x}} \in \mathcal{A}$. Hence we must have $g(\boldsymbol{x}^*) = 0$. ■

Let us see how the Sifting Lemma applies to the problem of projecting onto $\mathcal{Z}_{\mathrm{OR}}$. If the relaxed problem in the first step does not return a feasible point then, from the Sifting Lemma, the constraint $\sum_{i=1}^{K} z_i \geq 1$ has to be active, *i.e.*, we must have $\sum_{i=1}^{K} z_i = 1$. This, in turn, implies that $\boldsymbol{z} \leq 1$, hence the problem becomes equivalent to the XOR case. In sum, the worst-case runtime is $O(K \log K)$, although it is $O(K)$ if the first step succeeds.

B.5 OR-with-output Factor

Solving the AD^3 subproblem for the OR-with-output factor is slightly more complicated than in the previous cases; however, we next see that it can also be addressed in $O(K \log K)$ time with a sort operation. The polytope $\mathcal{Z}_{\mathrm{OR-out}}$ can be expressed as the intersection of the following three sets:[13]

$$\mathbb{U}^{K+1} := [0,1]^{K+1} \tag{106}$$

$$\mathcal{A}_1 := \{\boldsymbol{z} \in \mathbb{R}^{K+1} \mid z_k \leq z_{K+1}, \forall k = 1, \ldots, K\} \tag{107}$$

$$\mathcal{A}_2 := \left\{\boldsymbol{z} \in [0,1]^{K+1} \;\middle|\; \textstyle\sum_{k=1}^{K} z_k \geq z_{K+1}\right\}. \tag{108}$$

We further define $\mathcal{A}_0 := [0,1]^{K+1} \cap \mathcal{A}_1$, and we denote by $\mathrm{proj}_{\mathcal{Z}}(\boldsymbol{z})$ the Euclidean projection of a point $\boldsymbol{z}$ onto a convex set $\mathcal{Z}$. From Lemma 17, we have that the following procedure is correct:

1. Set $\tilde{\boldsymbol{z}} := \mathrm{proj}_{\mathbb{U}^{K+1}}(\boldsymbol{z}_0)$. If $\tilde{\boldsymbol{z}} \in \mathcal{A}_1 \cap \mathcal{A}_2$, then we are done: just return $\tilde{\boldsymbol{z}}$. Else, if $\tilde{\boldsymbol{z}} \in \mathcal{A}_1$ but $\tilde{\boldsymbol{z}} \notin \mathcal{A}_2$, discard $\tilde{\boldsymbol{z}}$ and go to step 3. Otherwise, discard $\tilde{\boldsymbol{z}}$ and go to step 2.

2. Set $\tilde{\boldsymbol{z}} := \mathrm{proj}_{\mathcal{A}_0}(\boldsymbol{z}_0)$ (we will describe later how to compute this projection). If $\tilde{\boldsymbol{z}} \in \mathcal{A}_2$, return $\tilde{\boldsymbol{z}}$. Otherwise, discard $\tilde{\boldsymbol{z}}$ and go to step 3.

3. Set $\tilde{\boldsymbol{z}} := \mathrm{proj}_{\bar{\mathcal{A}}_2}(\boldsymbol{z}_0)$, where $\bar{\mathcal{A}}_2 := \{\boldsymbol{z} \in [0,1]^{K+1} \mid \sum_{k=1}^{K} z_k = z_{K+1}\}$ (this set is precisely the marginal polytope of a XOR factor with the last output negated, hence the projection corresponds to the local subproblem for that factor, for which we can employ Algorithm 4).

Note that the first step above can be omitted; however, it avoids performing step 2 (which requires a sort) unless it is really necessary. To completely specify the algorithm, we only need to explain how to compute the projection onto $\mathcal{A}_0$ (step 2). The next proposition states that this can be done by first projecting onto $\mathcal{A}_1$, and then projecting the result onto $[0,1]^{K+1}$.

We first start with a lemma establishing a sufficient condition for the composition of two individual projections be equivalent to projecting onto the intersection (which is not true in general).[14]

Lemma 18 *Let $X \subseteq \mathbb{R}^D$ and $Y \subseteq \mathbb{R}^D$ be convex sets, and suppose $\boldsymbol{z}^* = \mathrm{proj}_Y(\boldsymbol{z}_0 + \boldsymbol{z}^* - \boldsymbol{z}')$ holds for any $\boldsymbol{z}_0 \in \mathbb{R}^D$, where $\boldsymbol{z}' = \mathrm{proj}_Y(\boldsymbol{z}_0)$, and $\boldsymbol{z}^* = \mathrm{proj}_X(\boldsymbol{z}')$. Then, we have $\mathrm{proj}_{X \cap Y} = \mathrm{proj}_X \circ \mathrm{proj}_Y$.*

Proof: Assume $\boldsymbol{z}^* = \mathrm{proj}_Y(\boldsymbol{z}_0 + \boldsymbol{z}^* - \boldsymbol{z}')$. Then, we have $(\boldsymbol{z}_0 + \boldsymbol{z}^* - \boldsymbol{z}' - \boldsymbol{z}^*)^\top (\boldsymbol{z} - \boldsymbol{z}^*) \leq 0$ for all $\boldsymbol{z} \in Y$; in particular, $(\boldsymbol{z}_0 - \boldsymbol{z}')^\top (\boldsymbol{z} - \boldsymbol{z}^*) \leq 0$ for all $\boldsymbol{z} \in X \cap Y$. On the other hand, the definition of $\boldsymbol{z}^*$ implies $(\boldsymbol{z}' - \boldsymbol{z}^*)^\top (\boldsymbol{z} - \boldsymbol{z}^*) \leq 0$ for all $\boldsymbol{z} \in X$, and in particular for $\boldsymbol{z} \in X \cap Y$. Summing these two inequalities, we obtain $(\boldsymbol{z}_0 - \boldsymbol{z}^*)^\top (\boldsymbol{z} - \boldsymbol{z}^*) \leq 0$ for all $\boldsymbol{z} \in X \cap Y$, that is, $\boldsymbol{z}^* = \mathrm{proj}_{X \cap Y}(\boldsymbol{z}_0)$. ■

Proposition 19 *It holds $\mathrm{proj}_{\mathcal{A}_0} = \mathrm{proj}_{\mathbb{U}^{K+1}} \circ \mathrm{proj}_{\mathcal{A}_1}$. Furthermore, a projection onto $\mathcal{A}_1$ can be computed in $O(K \log K)$ time using Algorithm 5.*

13. Actually, the set $\mathbb{U}^{K+1}$ is redundant, since we have $\mathcal{A}_2 \subseteq \mathbb{U}^{K+1}$ and therefore $\mathcal{Z}_{\mathrm{OR-out}} = \mathcal{A}_1 \cap \mathcal{A}_2$. However it is computationally advantageous to consider this redundancy, as we shall see.

14. This is equivalent to Dykstra's projection algorithm (Boyle and Dykstra, 1986) converging in one iteration.

Proof: We first prove the second part. Note that a projection onto $\mathcal{A}_1$ can be written as the following problem:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{z}_0\|^2 \quad \text{subject to} \quad z_k \leq z_{K+1},\ \forall k=1,\ldots,K, \tag{109}$$

and we have successively:

$$\begin{aligned}
\min_{z_k\leq z_{K+1},\ \forall k} \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{z}_0\|^2 &= \min_{z_{K+1}} \frac{1}{2}(z_{K+1}-z_{0,K+1})^2 + \sum_{k=1}^{K} \min_{z_k\leq z_{K+1}} \frac{1}{2}(z_k-z_{0k})^2 \\
&= \min_{z_{K+1}} \frac{1}{2}(z_{K+1}-z_{0,K+1})^2 + \sum_{k=1}^{K} \frac{1}{2}(\min\{z_{K+1},z_{0k}\}-z_{0k})^2 \\
&= \min_{z_{K+1}} \frac{1}{2}(z_{K+1}-z_{0,K+1})^2 + \frac{1}{2}\sum_{k\in\mathcal{I}(z_{K+1})} (z_{K+1}-z_{0k})^2. \qquad (110)
\end{aligned}$$

where $\mathcal{I}(z_{K+1}) \triangleq \{k\in[K] : z_{0k}\geq z_{K+1}\}$. Assuming that the set $\mathcal{I}(z_{K+1})$ is given, the previous is a sum-of-squares problem whose solution is

$$z^*_{K+1} = \frac{z_{0,K+1}+\sum_{k\in\mathcal{I}(z_{K+1})} z_{0k}}{1+|\mathcal{I}(z_{K+1})|}. \tag{111}$$

The set $\mathcal{I}(z_{K+1})$ can be determined by inspection after sorting $z_{01},\ldots,z_{0K}$. The procedure is shown in Algorithm 5.

To prove the first part, we invoke Lemma 18. It suffices to show that $\boldsymbol{z}^* = \text{proj}_{\mathcal{A}_1}(\boldsymbol{z}_0+\boldsymbol{z}^*-\boldsymbol{z}')$ holds for any $\boldsymbol{z}_0\in\mathbb{R}^D$, where $\boldsymbol{z}' = \text{proj}_{\mathcal{A}_1}(\boldsymbol{z}_0)$, and $\boldsymbol{z}^* = \text{proj}_{\mathbb{U}^{K+1}}(\boldsymbol{z}')$. Looking at Algorithm 5, we see that:

$$z'_k = \begin{cases} \tau, & \text{if } k=K+1 \text{ or } z_{0k}\geq\tau \\ z_{0k}, & \text{otherwise,} \end{cases} \qquad z^*_k = [z'_k]_{\mathbb{U}} = \begin{cases} [\tau]_{\mathbb{U}}, & \text{if } k=K+1 \text{ or } z_{0k}\geq\tau \\ [z_{0k}]_{\mathbb{U}}, & \text{otherwise.} \end{cases}$$

$$z_{0k}+z^*_k-z'_k = \begin{cases} [\tau]_{\mathbb{U}}-\tau+z_{0k}, & \text{if } k=K+1 \text{ or } z_{0k}\geq\tau \\ [z_{0k}]_{\mathbb{U}}, & \text{otherwise.} \end{cases} \tag{112}$$

Now two things should be noted about Algorithm 5:

- If a constant is added to all entries in $\boldsymbol{z}_0$, the set $\mathcal{I}(z_{K+1})$ remains the same, and τ and $\boldsymbol{z}$ are affected by the same constant;
- Let $\tilde{\boldsymbol{z}}_0$ be such that $\tilde{z}_{0k}=z_{0k}$ if $k=K+1$ or $z_{0k}\geq\tau$, and $\tilde{z}_{0k}\leq\tau$ otherwise. Let $\tilde{\boldsymbol{z}}$ be the projected point when such $\tilde{\boldsymbol{z}}_0$ is given as input. Then $\mathcal{I}(\tilde{z}_{K+1})=\mathcal{I}(z_{K+1})$, $\tilde{\tau}=\tau$, $\tilde{z}_k=z_k$ if $k=K+1$ or $z_{0k}\geq\tau$, and $\tilde{z}_k=\tilde{z}_{0k}$ otherwise.

The two facts above allow to relate the projection of $\boldsymbol{z}_0+\boldsymbol{z}^*-\boldsymbol{z}'$ with that of $\boldsymbol{z}_0$. Using $[\tau]_{\mathbb{U}}-\tau$ as the constant, and noting that, for $k\neq K+1$ and $z_{0k}<\tau$, we have $[z_{0k}]_{\mathbb{U}}-[\tau]_{\mathbb{U}}+\tau\geq\tau$ if $z_{0k}<\tau$, the two facts imply that:

$$\text{proj}_{\mathcal{A}_1}(\boldsymbol{z}_0+\boldsymbol{z}^*-\boldsymbol{z}') = \begin{cases} z'_k+[\tau]_{\mathbb{U}}-\tau=[\tau]_{\mathbb{U}}, & \text{if } k=K+1 \text{ or } z_{0k}\geq\tau \\ [z_{0k}]_{\mathbb{U}}, & \text{otherwise;} \end{cases} \tag{113}$$

hence $\boldsymbol{z}^* = \text{proj}_{\mathcal{A}_1}(\boldsymbol{z}_0+\boldsymbol{z}^*-\boldsymbol{z}')$, which concludes the proof. ■

Algorithm 5 Projection onto $\mathcal{A}_1$

Input: $\boldsymbol{z}_0$
Sort $z_{01}, \ldots, z_{0K}$ into $y_1 \geq \ldots \geq y_K$
Find $\rho = \min \left\{ j \in [K+1] \mid \frac{1}{j} \left(z_{0,K+1} + \sum_{r=1}^{j-1} y_r \right) > y_j \right\}$
Define $\tau = \frac{1}{\rho} \left(z_{0,K+1} + \sum_{r=1}^{\rho-1} y_r \right)$
Output: $\boldsymbol{z}$ defined as $z_{K+1} = \tau$ and $z_i = \min\{z_{0i}, \tau\}$, $i = 1, \ldots, K$.

Appendix C. Proof of Proposition 11

We first show that the rank of the matrix $\mathbf{M}$ is at most $\sum_{i \in \partial(\alpha)} |\mathcal{Y}_i| - \partial(\alpha) + 1$. For each $i \in \partial(\alpha)$, let us consider the $|\mathcal{Y}_i|$ rows of $\mathbf{M}$. By definition of $\mathbf{M}$, the set of entries on these rows which have the value 1 form a partition of $\mathcal{Y}_\alpha$, hence, summing these rows yields the all-ones row vector, and this happens for each $i \in \partial(\alpha)$. Hence we have at least $\partial(\alpha) - 1$ rows that are linearly dependent. This shows that the rank of $\mathbf{M}$ is at most $\sum_{i \in \partial(\alpha)} |\mathcal{Y}_i| - \partial(\alpha) + 1$. Let us now rewrite (32) as

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{a}\|^2 + g(\boldsymbol{u}) \quad \text{with respect to} \quad \boldsymbol{u} \in \mathbb{R}^{\sum_i |\mathcal{Y}_i|}, \tag{114}$$

where $g(\boldsymbol{u})$ is the solution value of the following linear problem:

$$\begin{aligned} &\text{minimize} \quad -\boldsymbol{b}^\top \boldsymbol{q}_\alpha \quad \text{with respect to} \quad \boldsymbol{q}_\alpha \in \mathbb{R}^{|\mathcal{Y}_\alpha|} \\ &\text{subject to} \quad \begin{cases} \mathbf{M}\boldsymbol{q}_{\boldsymbol{\alpha}} = \boldsymbol{u} \\ \mathbf{1}^\top \boldsymbol{q}_{\boldsymbol{\alpha}} = 1 \\ \boldsymbol{q}_{\boldsymbol{\alpha}} \geq 0. \end{cases} \end{aligned} \tag{115}$$

From the simplex constraints (last two lines), we have that problem (115) is bounded below (i.e., $g(\boldsymbol{u}) > -\infty$). Furthermore, problem (115) is feasible (i.e., $g(\boldsymbol{u}) < +\infty$) iff $\boldsymbol{u} \in \prod_{i \in \partial(\alpha)} \Delta^{|\mathcal{Y}_i|}$, which in turn implies $\mathbf{1}^\top \boldsymbol{q}_\alpha = 1$. Hence we can add these constraints to the problem in (114), discard the constraint $\mathbf{1}^\top \boldsymbol{q}_\alpha = 1$ in (115), and assume that the resulting problem (which we reproduce below) is feasible and bounded below:

$$\begin{aligned} &\text{minimize} \quad -\boldsymbol{b}^\top \boldsymbol{q}_\alpha \quad \text{with respect to} \quad \boldsymbol{q}_\alpha \in \mathbb{R}^{|\mathcal{Y}_\alpha|} \\ &\text{subject to} \quad \mathbf{M}\boldsymbol{q}_\alpha = \boldsymbol{u}, \quad \boldsymbol{q}_\alpha \geq 0. \end{aligned} \tag{116}$$

Problem (116) is a linear program in standard form. Since it is feasible and bounded, it admits a solution at a vertex of the constraint set (Rockafellar, 1970). We have that a feasible point $\widehat{\boldsymbol{q}}_\alpha$ is a vertex if and only if the columns of $\mathbf{M}$ indexed by $\{\boldsymbol{y}_\alpha \mid \widehat{\boldsymbol{q}}\alpha(\boldsymbol{y}_\alpha) \neq 0\}$ are linearly independent. We cannot have more than $\sum_{i \in \partial(\alpha)} |\mathcal{Y}_i| - \partial(\alpha) + 1$ of these columns, since this is the rank of $\mathbf{M}$. It follows that (116) (and hence (32)) has a solution $\boldsymbol{q}^*_\alpha$ with at most $\sum_{i \in \partial(\alpha)} |\mathcal{Y}_i| - \partial(\alpha) + 1$ nonzeros.

References

M. B. Almeida and A. F. T. Martins. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2013.

S. Barman, X. Liu, S. Draper, and B. Recht. Decomposition methods for large scale LP decoding. In *49th Annual Allerton Conference on Communication, Control, and Computing*, pages 253–260. IEEE, 2011.

Dhruv Batra, Sebastian Nowozin, and Pushmeet Kohli. Tighter relaxations for map-mrf inference: A local primal-dual gap based separation algorithm. In *International Conference on Artificial Intelligence and Statistics*, pages 146–154, 2011.

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

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

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

S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

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

P. Brucker. An $o(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3 (3):163–166, 1984.

M. Chang, L. Ratinov, and D. Roth. Constraints as prior knowledge. In *International Conference of Machine Learning: Workshop on Prior Knowledge for Text and Language Processing*, July 2008.

Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14: 1396–1400, 1965.

D. Das. *Semi-Supervised and Latent-Variable Models of Natural Language Semantics*. PhD thesis, Carnegie Mellon University, 2012.

D. Das, A.F.T. Martins, and N.A. Smith. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*, 2012.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. *Advances in Neural Information Processing Systems*, 19, 2007.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proc. of International Conference of Machine Learning*, 2008.

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

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B: 233–240, 1967.

J.M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of International Conference on Computational Linguistics*, pages 340–345, 1996.

F. Facchinei and J.S. Pang. *Finite-dimensional variational inequalities and complementarity problems*, volume 1. Springer Verlag, 2003.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

C.J. Fillmore. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.

Qiang Fu, Huahua Wang, and Arindam Banerjee. Bethe-ADMM for tree decomposition based parallel MAP inference. In *Proc. of Uncertainty in Artificial Intelligence*, 2013.

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

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

R. Glowinski and A. Marroco. 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.

T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

BS He and XM Yuan. On the $O(1/t)$ convergence rate of alternating direction method. *SIAM Journal of Numerical Analysis (to appear)*, 2011.

M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:302–320, 1969.

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

V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *International Conference of Machine Learning*, 2010.

Jörg Hendrik Kappes, Bogdan Savchynskyy, and C Schnorr. A bundle approach to efficient map-inference by lagrangian relaxation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

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

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

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of International Conference on Computer Vision*, 2007.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proc. of Empirical Methods for Natural Language Processing*, 2010.

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

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

A. Kulesza and F. Pereira. Structured Learning with Approximate Inference. *Neural Information Processing Systems*, 2007.

Steffen Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996. ISBN 0-19-852219-3.

Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J.M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *International Conference on Uncertainty in Artificial Intelligence*, 2010.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comp. Ling.*, 19(2):313–330, 1993.

A. F. T. Martins. *The Geometry of Constrained Structured Prediction: Applications to Inference and Learning of Natural Language Syntax*. PhD thesis, Carnegie Mellon University and Instituto Superior Técnico, 2012.

A. F. T. Martins, N. A. Smith, and E. P. Xing. Polyhedral Outer Approximations with Application to Natural Language Parsing. In *Proc. of International Conference of Machine Learning*, 2009.

A. F. T. Martins, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. Augmented Dual Decomposition for MAP Inference. In *Neural Information Processing Systems: Workshop in Optimization for Machine Learning*, 2010.

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. of International Conference on Machine Learning*, 2011a.

A. F. T. Martins, N. A. Smith, P. M. Q. Aguiar, and M. A. T. Figueiredo. Dual Decomposition with Many Overlapping Components. In *Proc. of Empirical Methods for Natural Language Processing*, 2011b.

A. F. T. Martins, M. B. Almeida, and N. A. Smith. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2013.

R. McDonald and G. Satta. On the complexity of non-projective data-driven dependency parsing. In *Proc. of International Conference on Parsing Technologies*, 2007.

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of Empirical Methods for Natural Language Processing*, 2005.

O. Meshi and A. Globerson. An Alternating Direction Method for Dual MAP LP Relaxation. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011.

C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *Journal of Optimization Theory and Applications*, 50(1):195–200, 1986.

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

J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

S. Nowozin and C.H. Lampert. Global connectivity potentials for random field models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 818–825. IEEE, 2009.

Panos M. Pardalos and Naina Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1):321–328, 1990.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Patrick Pletscher and Sharon Wulff. LPQP for MAP: Putting LP Solvers to Better Use. 2012.

H. Poon and P. Domingos. Unsupervised semantic parsing. In *Proc. of Empirical Methods in Natural Language Processing*, 2009.

M. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, 1969.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and Inference over Constrained Output. In *Proc. of International Joint Conference on Artificial Intelligence*, 2005.

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

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

T.J. Richardson and R.L. Urbanke. *Modern coding theory*. Cambridge University Press, 2008.

R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *International Conference on Natural Language Learning*, 2004.

A.M. Rush and M. Collins. A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing. *Journal of Artificial Intelligence Research*, 45:305–362, 2012.

Bogdan Savchynskyy, Stefan Schmidt, Jörg Kappes, and Christoph Schnörr. A study of nesterov's scheme for lagrangian decomposition and map labeling. In *IEEE Conference on Computer Vision*

and Pattern Recognition, 2011.

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

Alex Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Globally convergent dual map lp relaxation solvers using fenchel-young margins. In *Advances in Neural Information Processing Systems 25*, pages 2393–2401, 2012.

Alexander Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Distributed message passing for large scale graphical models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1833–1840, 2011.

D. Smith and J. Eisner. Dependency parsing by belief propagation. In *Proc. of Empirical Methods for Natural Language Processing*, 2008.

D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T Jaakkola. Tightening LP relaxations for MAP using message-passing. In *Proc. of Uncertainty in Artificial Intelligence*, 2008.

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

M. Sun, M. Telaprolu, H. Lee, and S. Savarese. An efficient branch-and-bound algorithm for optimal human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1616–1623, 2012.

R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

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

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

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

Huahua Wang and Arindam Banerjee. Online Alternating Direction Method. In *Proc. of International Conference on Machine Learning*, 2012.

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

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proc. of International Conference on Parsing Technologies*, 2003.

C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation–an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

J.S. Yedidia, Y. Wang, and S.C. Draper. Divide and concur and difference-map BP decoders for LDPC codes. *IEEE Transactions on Information Theory*, 57(2):786–802, 2011.