Turbo Parsers: Dependency Parsing by Approximate Variational Inference

André F. T. Martins[*†] **Noah A. Smith**[*] **Eric P. Xing**[*]
[*]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{afm,nasmith,epxing}@cs.cmu.edu

Pedro M. Q. Aguiar[‡]
[‡]Instituto de Sistemas e Robótica
Instituto Superior Técnico
Lisboa, Portugal
aguiar@isr.ist.utl.pt

Mário A. T. Figueiredo[†]
[†]Instituto de Telecomunicações
Instituto Superior Técnico
Lisboa, Portugal
mtf@lx.it.pt

Abstract

We present a unified view of two state-of-the-art non-projective dependency parsers, both approximate: the loopy belief propagation parser of Smith and Eisner (2008) and the relaxed linear program of Martins et al. (2009). By representing the model assumptions with a factor graph, we shed light on the optimization problems tackled in each method. We also propose a new aggressive online algorithm to learn the model parameters, which makes use of the underlying variational representation. The algorithm does not require a learning rate parameter and provides a single framework for a wide family of convex loss functions, including CRFs and structured SVMs. Experiments show state-of-the-art performance for 14 languages.

1 Introduction

Feature-rich discriminative models that break locality/independence assumptions can boost a parser's performance (McDonald et al., 2006; Huang, 2008; Finkel et al., 2008; Smith and Eisner, 2008; Martins et al., 2009; Koo and Collins, 2010). Often, inference with such models becomes computationally intractable, causing a demand for understanding and improving approximate parsing algorithms.

In this paper, we show a formal connection between two recently-proposed approximate inference techniques for non-projective dependency parsing: loopy belief propagation (Smith and Eisner, 2008) and linear programming relaxation (Martins et al., 2009). While those two parsers are differently motivated, we show that both correspond to inference in a factor graph, and both optimize objective functions over local approximations of the marginal polytope. The connection is made clear by writing the explicit declarative optimization problem underlying Smith and Eisner (2008) and by showing the factor graph underlying Martins et al. (2009). The success of both approaches parallels similar approximations in other fields, such as statistical image processing and error-correcting coding. Throughtout, we call these *turbo parsers*.[1]

Our contributions are not limited to dependency parsing: we present a general method for inference in factor graphs with hard constraints (§2), which extends some combinatorial factors considered by Smith and Eisner (2008). After presenting a geometric view of the variational approximations underlying message-passing algorithms (§3), and closing the gap between the two aforementioned parsers (§4), we consider the problem of learning the model parameters (§5). To this end, we propose an aggressive online algorithm that generalizes MIRA (Crammer et al., 2006) to arbitrary loss functions. We adopt a family of losses subsuming CRFs (Lafferty et al., 2001) and structured SVMs (Taskar et al., 2003; Tsochantaridis et al., 2004). Finally, we present a technique for including features not attested in the training data, allowing for richer models without substantial runtime costs. Our experiments (§6) show state-of-the-art performance on dependency parsing benchmarks.

[1]The name stems from "turbo codes," a class of high-performance error-correcting codes introduced by Berrou et al. (1993) for which decoding algorithms are equivalent to running belief propagation in a graph with loops (McEliece et al., 1998).

2 Structured Inference and Factor Graphs

Denote by $\mathcal{X}$ a set of input objects from which we want to infer some hidden structure conveyed in an output set $\mathcal{Y}$. Each input $x \in \mathcal{X}$ (e.g., a sentence) is associated with a set of candidate outputs $\mathcal{Y}(x) \subseteq \mathcal{Y}$ (e.g., parse trees); we are interested in the case where $\mathcal{Y}(x)$ is a large structured set.

Choices about the *representation* of elements of $\mathcal{Y}(x)$ play a major role in algorithm design. In many problems, the elements of $\mathcal{Y}(x)$ can be represented as discrete-valued vectors of the form $\mathbf{y} = \langle y_1, \ldots, y_I \rangle$, each y_i taking values in a label set $\mathcal{Y}_i$. For example, in unlabeled dependency parsing, I is the number of candidate dependency arcs (quadratic in the sentence length), and each $\mathcal{Y}_i = \{0, 1\}$. Of course, the y_i are highly interdependent.

Factor Graphs. Probabilistic models like CRFs (Lafferty et al., 2001) assume a factorization of the conditional distribution of Y,

$$\Pr(Y = \mathbf{y} \mid X = x) \propto \prod_{C \in \mathcal{C}} \Psi_C(x, \mathbf{y}_C), \quad (1)$$

where each $C \subseteq \{1, \ldots, I\}$ is a *factor*, $\mathcal{C}$ is the set of factors, each $\mathbf{y}_C \triangleq \langle y_i \rangle_{i \in C}$ denotes a partial output assignment, and each Ψ_C is a nonnegative *potential function* that depends on the output only via its restriction to C. A *factor graph* (Kschischang et al., 2001) is a convenient representation for the factorization in Eq. 1: it is a bipartite graph $\mathcal{G}_x$ comprised of variable nodes $\{1, \ldots, I\}$ and factor nodes $C \in \mathcal{C}$, with an edge connecting the ith variable node and a factor node C iff $i \in C$. Hence, the factor graph $\mathcal{G}_x$ makes explicit the direct dependencies among the variables $\{y_1, \ldots, y_I\}$.

Factor graphs have been used for several NLP tasks, such as dependency parsing, segmentation, and co-reference resolution (Sutton et al., 2007; Smith and Eisner, 2008; McCallum et al., 2009).

Hard and Soft Constraint Factors. It may be the case that valid outputs are a *proper* subset of $\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_I$—for example, in dependency parsing, the entries of the output vector $\mathbf{y}$ must jointly define a spanning tree. This requires *hard constraint factors* that rule out forbidden partial assignments by mapping them to zero potential values. See Table 1 for an inventory of hard constraint factors used in this paper. Factors that are not of this special kind are called *soft factors*, and have strictly positive potentials. We thus have a partition $\mathcal{C} = \mathcal{C}_{\text{hard}} \cup \mathcal{C}_{\text{soft}}$.

We let the soft factor potentials take the form $\Psi_C(x, \mathbf{y}_C) \triangleq \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}_C(x, \mathbf{y}_C))$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of parameters (shared across factors) and $\boldsymbol{\phi}_C(x, \mathbf{y}_C)$ is a local feature vector. The conditional distribution of Y (Eq. 1) thus becomes log-linear:

$$\Pr_{\boldsymbol{\theta}}(\mathbf{y}|x) = Z_x(\boldsymbol{\theta})^{-1} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(x, \mathbf{y})), \quad (2)$$

where $Z_x(\boldsymbol{\theta}) \triangleq \sum_{\mathbf{y}' \in \mathcal{Y}(x)} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(x, \mathbf{y}'))$ is the *partition function*, and the features decompose as:

$$\boldsymbol{\phi}(x, \mathbf{y}) \triangleq \sum_{C \in \mathcal{C}_{\text{soft}}} \boldsymbol{\phi}_C(x, \mathbf{y}_C). \quad (3)$$

Dependency Parsing. Smith and Eisner (2008) proposed a factor graph representation for dependency parsing (Fig. 1). The graph has $O(n^2)$ variable nodes (n is the sentence length), one per candidate arc $a \triangleq \langle h, m \rangle$ linking a head h and modifier m. Outputs are binary, with $y_a = 1$ iff arc a belongs to the dependency tree. There is a hard factor TREE connected to all variables, that constrains the overall arc configurations to form a spanning tree. There is a unary soft factor per arc, whose log-potential reflects the score of that arc. There are also $O(n^3)$ pairwise factors; their log-potentials reflect the scores of *sibling* and *grandparent* arcs. These factors create loops, thus calling for approximate inference. Without them, the model is arc-factored, and exact inference in it is well studied: finding the most probable parse tree takes $O(n^3)$ time with the Chu-Liu-Edmonds algorithm (McDonald et al., 2005),[2] and computing posterior marginals for all arcs takes $O(n^3)$ time via the matrix-tree theorem (Smith and Smith, 2007; Koo et al., 2007).

Message-passing algorithms. In general factor graphs, both inference problems—obtaining the most probable output (the MAP) $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$, and computing the marginals $\Pr_{\boldsymbol{\theta}}(Y_i = y_i|x)$—can be addressed with the belief propagation (BP) algorithm (Pearl, 1988), which iteratively passes messages between variables and factors reflecting their local "beliefs."

[2] There is a faster but more involved $O(n^2)$ algorithm due to Tarjan (1977).

A general binary factor: $\Psi_C(v_1,\ldots,v_n) = \begin{cases} 1 & v_1,\ldots,v_n \in \mathcal{S}_C \\ 0 & \text{otherwise,} \end{cases}$ where $\mathcal{S}_C \subseteq \{0,1\}^n$.
• Message-induced distribution: $\boldsymbol{\omega} \triangleq \langle m_{j\to C}\rangle_{j=1,\ldots,n}$ • Partition function: $Z_C(\boldsymbol{\omega}) \triangleq \sum_{\langle v_1,\ldots,v_n\rangle\in\mathcal{S}_C} \prod_{i=1}^n m_{i\to C}^{v_i}$
• Marginals: $\textsc{marg}_i(\boldsymbol{\omega}) \triangleq \Pr_{\boldsymbol{\omega}}\{V_i = 1 \mid \langle V_1,\ldots,V_n\rangle \in \mathcal{S}_C\}$ • Max-marginals: $\textsc{max-marg}_{i,b}(\boldsymbol{\omega}) \triangleq \max_{\mathbf{v}\in\mathcal{S}_C} \Pr_{\boldsymbol{\omega}}(\mathbf{v} \mid v_i = b)$
• Sum-prod.: $m_{C\to i} = m_{i\to C}^{-1} \cdot \textsc{marg}_i(\boldsymbol{\omega})/(1-\textsc{marg}_i(\boldsymbol{\omega}))$ • Max-prod.: $m_{C\to i} = m_{i\to C}^{-1} \cdot \textsc{max-marg}_{i,1}(\boldsymbol{\omega})/\textsc{max-marg}_{i,0}(\boldsymbol{\omega})$
• Local agreem. constr.: $\mathbf{z} \in \operatorname{conv} \mathcal{S}_C$, where $\mathbf{z} = \langle \tau_i(1)\rangle_{i=1}^n$ • Entropy: $H_C = \log Z_C(\boldsymbol{\omega}) - \sum_{i=1}^n \textsc{marg}_i(\boldsymbol{\omega}) \log m_{i\to C}$
TREE $\Psi_{\textsc{tree}}(\langle y_a\rangle_{a\in A}) = \begin{cases} 1 & \mathbf{y} \in \mathcal{Y}_{\text{tree}} \text{ (i.e., } \{a \in A \mid y_a = 1\} \text{ is a directed spanning tree)} \\ 0 & \text{otherwise,} \end{cases}$ where A is the set of candidate arcs.
• Partition function $Z_{\text{tree}}(\boldsymbol{\omega})$ and marginals $\langle \textsc{marg}_a(\boldsymbol{\omega})\rangle_{a\in A}$ computed via the matrix-tree theorem, with $\boldsymbol{\omega} \triangleq \langle m_{a\to\textsc{tree}}\rangle_{a\in A}$
• Sum-prod.: $m_{\textsc{tree}\to a} = m_{a\to\textsc{tree}}^{-1} \cdot \textsc{marg}_a(\boldsymbol{\omega})/(1-\textsc{marg}_a(\boldsymbol{\omega}))$
• Max-prod.: $m_{\textsc{tree}\to a} = m_{a\to\textsc{tree}}^{-1} \cdot \textsc{max-marg}_{a,1}(\boldsymbol{\omega})/\textsc{max-marg}_{a,0}(\boldsymbol{\omega})$, where $\textsc{max-marg}_{a,b}(\boldsymbol{\omega}) \triangleq \max_{\mathbf{y}\in\mathcal{Y}_{\text{tree}}} \Pr_{\boldsymbol{\omega}}(\mathbf{y} \mid y_a = b)$
• Local agreem. constr.: $\mathbf{z} \in \mathcal{Z}_{\text{tree}}$, where $\mathcal{Z}_{\text{tree}} \triangleq \operatorname{conv} \mathcal{Y}_{\text{tree}}$ is the arborescence polytope
• Entropy: $H_{\text{tree}} = \log Z_{\text{tree}}(\boldsymbol{\omega}) - \sum_{a\in A} \textsc{marg}_a(\boldsymbol{\omega}) \log m_{a\to\textsc{tree}}$
XOR ("one-hot") $\Psi_{\textsc{xor}}(v_1,\ldots,v_n) = \begin{cases} 1 & \sum_{i=1}^n v_i = 1 \\ 0 & \text{otherwise.} \end{cases}$
• Sum-prod.: $m_{\textsc{xor}\to i} = \left(\sum_{j\neq i} m_{j\to\textsc{xor}}\right)^{-1}$ • Max-prod.: $m_{\textsc{xor}\to i} = \left(\max_{j\neq i} m_{j\to\textsc{xor}}\right)^{-1}$
• Local agreem. constr.: $\sum_i z_i = 1$, $z_i \in [0,1], \forall i$ • $H_{\textsc{xor}} = -\sum_i (m_{i\to\textsc{xor}}/\sum_j m_{j\to\textsc{xor}}) \log(m_{i\to\textsc{xor}}/\sum_j m_{j\to\textsc{xor}})$
OR $\Psi_{\textsc{or}}(v_1,\ldots,v_n) = \begin{cases} 1 & \sum_{i=1}^n v_i \geq 1 \\ 0 & \text{otherwise.} \end{cases}$
• Sum-prod.: $m_{\textsc{or}\to i} = \left(1 - \prod_{j\neq i}(1+m_{j\to\textsc{or}})^{-1}\right)^{-1}$ • Max-prod.: $m_{\textsc{or}\to i} = \max\{1, \min_{j\neq i} m_{j\to\textsc{or}}^{-1}\}$
• Local agreem. constr.: $\sum_i z_i \geq 1$, $z_i \in [0,1], \forall i$
OR-WITH-OUTPUT $\Psi_{\textsc{or-out}}(v_1,\ldots,v_n) = \begin{cases} 1 & v_n = \bigvee_{i=1}^{n-1} v_i \\ 0 & \text{otherwise.} \end{cases}$
• Sum-prod.: $m_{\textsc{or-out}\to i} = \begin{cases} \left(1 - (1 - m_{n\to\textsc{or-out}}^{-1}) \prod_{j\neq i,n}(1+m_{j\to\textsc{or-out}})^{-1}\right)^{-1} & i < n \\ \prod_{j\neq n}(1+m_{j\to\textsc{or-out}}) - 1 & i = n. \end{cases}$
• Max-prod.: $m_{\textsc{or-out}\to i} = \begin{cases} \min\left\{m_{n\to\textsc{or-out}} \prod_{j\neq i,n} \max\{1, m_{j\to\textsc{or-out}}\}, \max\{1, \min_{j\neq i,n} m_{j\to\textsc{or-out}}^{-1}\}\right\} & i < n \\ \prod_{j\neq n} \max\{1, m_{j\to\textsc{or-out}}\} \min\{1, \max_{j\neq n} m_{j\to\textsc{or-out}}\} & i = n. \end{cases}$

Table 1: Hard constraint factors, their potentials, messages, and entropies. The top row shows expressions for a general binary factor: each outgoing message is computed from incoming *marginals* (in the sum-product case), or *max-marginals* (in the max-product case); the entropy of the factor (see §3) is computed from these marginals and the partition function; the local agreement constraints (§4) involve the *convex hull* of the set $\mathcal{S}_C$ of allowed configurations (see footnote 5). The TREE, XOR, OR and OR-WITH-OUTPUT factors allow tractable computation of all these quantities (rows 2–5). Two of these factors (TREE and XOR) had been proposed by Smith and Eisner (2008); we provide further information (max-product messages, entropies, and local agreement constraints). Factors OR and OR-WITH-OUTPUT are novel to the best of our knowledge. This inventory covers many cases, since the above formulae can be extended to the case where some inputs are *negated*: just replace the corresponding messages by their reciprocal, v_i by $1 - v_i$, etc. This allows building factors NAND (an OR factor with negated inputs), IMPLY (a 2-input OR with the first input negated), and XOR-WITH-OUTPUT (an XOR factor with the last input negated).

In sum-product BP, the messages take the form:[3]

$$M_{i\to C}(y_i) \propto \textstyle\prod_{D\neq C} M_{D\to i}(y_i) \quad (4)$$

$$M_{C\to i}(y_i) \propto \textstyle\sum_{\mathbf{y}_C\sim y_i} \Psi_C(\mathbf{y}_C) \prod_{j\neq i} M_{j\to C}(y_j). \quad (5)$$

In max-product BP, the summation in Eq. 5 is replaced by a maximization. Upon convergence, variable and factor beliefs are computed as:

$$\tau_i(y_i) \propto \textstyle\prod_C M_{C\to i}(y_i) \quad (6)$$

$$\tau_C(\mathbf{y}_C) \propto \Psi_C(\mathbf{y}_C) \textstyle\prod_i M_{i\to C}(y_i). \quad (7)$$

BP is exact when the factor graph is a tree: in the sum-product case, the beliefs in Eqs. 6–7 correspond to the true marginals, and in the max-product case, maximizing each $\tau_i(y_i)$ yields the MAP output. In graphs with loops, BP is an approximate method, not guaranteed to converge, nicknamed *loopy* BP. We highlight a variational perspective of loopy BP in §3; for now we consider algorithmic issues. Note that computing the factor-to-variable messages for each factor C (Eq. 5) requires a summation/maximization over exponentially many configurations. Fortunately, for all the hard constraint factors in rows 3–5 of Table 1, this computation can be done in linear time (and polynomial for the TREE factor)—this extends results presented in Smith and Eisner (2008).[4]

[3] We employ the standard $\sim$ notation, where a summation/maximization indexed by $\mathbf{y}_C \sim y_i$ means that it is over all $\mathbf{y}_C$ with the i-th component held fixed and set to y_i.

[4] The insight behind these speed-ups is that messages on binary-valued potentials can be expressed as $M_{C\to i}(y_i) \propto$

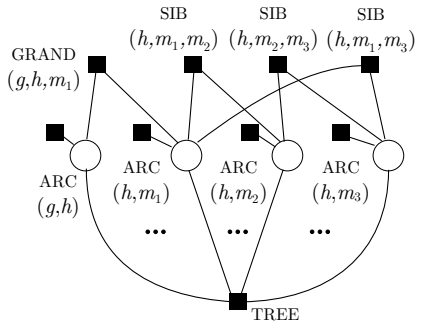

Figure 1: Factor graph corresponding to the dependency parsing model of Smith and Eisner (2008) with sibling and grandparent features. Circles denote variable nodes, and squares denote factor nodes. Note the loops created by the inclusion of pairwise factors (GRAND and SIB).

In Table 1 we present closed-form expressions for the factor-to-variable message ratios $m_{C\to i} \triangleq M_{C\to i}(1)/M_{C\to i}(0)$ in terms of their variable-to-factor counterparts $m_{i\to C} \triangleq M_{i\to C}(1)/M_{i\to C}(0)$; these ratios are all that is necessary when the variables are binary. Detailed derivations are presented in Appendix A.

3 Variational Representations

Let $\mathcal{P}_x \triangleq \{\Pr_{\boldsymbol{\theta}}(.|x) \mid \boldsymbol{\theta} \in \mathbb{R}^d\}$ be the family of all distributions of the form in Eq. 2. We next present an alternative parametrization for the distributions in $\mathcal{P}_x$ in terms of factor marginals. We will see that each distribution can be seen as a point in the so-called *marginal polytope* (Wainwright and Jordan, 2008); this will pave the way for the variational representations to be derived next.

Parts and Output Indicators. A *part* is a pair $\langle C, \mathbf{y}_C\rangle$, where C is a soft factor and $\mathbf{y}_C$ a partial output assignment. We let $\mathcal{R} = \{\langle C, \mathbf{y}_C\rangle \mid C \in \mathcal{C}_{\text{soft}}, \mathbf{y}_C \in \prod_{i\in C} \mathcal{Y}_i\}$ be the set of all parts. Given an output $\mathbf{y}' \in \mathcal{Y}(x)$, a part $\langle C, \mathbf{y}_C\rangle$ is said to be *active* if it locally matches the output, i.e., if $\mathbf{y}_C = \mathbf{y}'_C$. Any output $\mathbf{y}' \in \mathcal{Y}(x)$ can be mapped to a $|\mathcal{R}|$-dimensional binary vector $\boldsymbol{\chi}(\mathbf{y}')$ indicating which parts are active, i.e., $[\boldsymbol{\chi}(\mathbf{y}')]_{\langle C,\mathbf{y}_C\rangle} = 1$ if $\mathbf{y}_C = \mathbf{y}'_C$ and 0 otherwise; $\boldsymbol{\chi}(\mathbf{y}')$ is called the *output indicator vector*. This mapping allows decoupling the feature vector in Eq. 3 as the product of an input matrix and an output vector:

$$\boldsymbol{\phi}(x, \mathbf{y}) = \sum_{C\in\mathcal{C}_{\text{soft}}} \boldsymbol{\phi}_C(x, \mathbf{y}_C) = \mathbf{F}(x)\boldsymbol{\chi}(\mathbf{y}), \quad (8)$$

where $\mathbf{F}(x)$ is a d-by-$|\mathcal{R}|$ matrix whose columns contain the part-local feature vectors $\boldsymbol{\phi}_C(x, \mathbf{y}_C)$. Observe, however, that not every vector in $\{0,1\}^{|\mathcal{R}|}$ corresponds necessarily to a valid output in $\mathcal{Y}(x)$.

Marginal Polytope. Moving to vector representations of outputs leads naturally to a geometric view of the problem. The *marginal polytope* is the convex hull[5] of all the "valid" output indicator vectors:

$$\mathcal{M}(\mathcal{G}_x) \triangleq \text{conv}\{\boldsymbol{\chi}(\mathbf{y}) \mid \mathbf{y} \in \mathcal{Y}(x)\}.$$

Note that $\mathcal{M}(\mathcal{G}_x)$ only depends on the factor graph $\mathcal{G}_x$ and the hard constraints (i.e., it is independent of the parameters $\boldsymbol{\theta}$). The importance of the marginal polytope stems from two facts: (i) each vertex of $\mathcal{M}(\mathcal{G}_x)$ corresponds to an output in $\mathcal{Y}(x)$; (ii) each point in $\mathcal{M}(\mathcal{G}_x)$ corresponds to a vector of marginal probabilities that is realizable by some distribution (not necessarily in $\mathcal{P}_x$) that factors according to $\mathcal{G}_x$.

Variational Representations. We now describe formally how the points in $\mathcal{M}(\mathcal{G}_x)$ are linked to the distributions in $\mathcal{P}_x$. We extend the "canonical overcomplete parametrization" case, studied by Wainwright and Jordan (2008), to our scenario (common in NLP), where arbitrary features are allowed and the parameters are *tied* (shared by all factors). Let $H(\Pr_{\boldsymbol{\theta}}(.|x)) \triangleq -\sum_{\mathbf{y}\in\mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x) \log \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$ denote the *entropy* of $\Pr_{\boldsymbol{\theta}}(.|x)$, and $E_{\boldsymbol{\theta}}[.]$ the expectation under $\Pr_{\boldsymbol{\theta}}(.|x)$. The component of $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$ indexed by part $\langle C, \mathbf{y}_C\rangle$ is denoted $\mu_C(\mathbf{y}_C)$.

Proposition 1. *There is a map coupling each distribution* $\Pr_{\boldsymbol{\theta}}(.|x) \in \mathcal{P}_x$ *to a unique* $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$ *such that* $E_{\boldsymbol{\theta}}[\boldsymbol{\chi}(Y)] = \boldsymbol{\mu}$. *Define* $H(\boldsymbol{\mu}) \triangleq H(\Pr_{\boldsymbol{\theta}}(.|x))$ *if some* $\Pr_{\boldsymbol{\theta}}(.|x)$ *is coupled to* $\boldsymbol{\mu}$, *and* $H(\boldsymbol{\mu}) = -\infty$ *if no such* $\Pr_{\boldsymbol{\theta}}(.|x)$ *exists. Then:*

1. *The following variational representation for the log-partition function (mentioned in Eq. 2) holds:*

$$\log Z_x(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu}\in\mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)\boldsymbol{\mu} + H(\boldsymbol{\mu}). \quad (9)$$

$\Pr\{\Psi_C(Y_C) = 1|Y_i = y_i\}$ and $M_{C\to i}(y_i) \propto \max_{\Psi_C(\mathbf{y}_C)=1} \Pr\{Y_C = \mathbf{y}_C|Y_i = y_i\}$, respectively for the sum-product and max-product cases; these probabilities are induced by the messages in Eq. 4: for an event $A \subseteq \prod_{i\in C} \mathcal{Y}_i$, $\Pr\{Y_C \in A\} \triangleq \sum_{\mathbf{y}_C} \mathbb{I}(\mathbf{y}_C \in A) \prod_{i\in C} M_{i\to C}(y_i)$.

[5] The convex hull of $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ is the set of points that can be written as $\sum_{i=1}^k \lambda_i \mathbf{z}_i$, where $\sum_{i=1}^k \lambda_i = 1$ and each $\lambda_i \geq 0$.

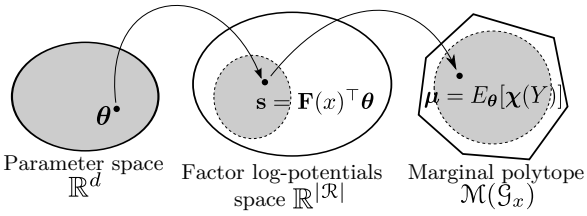

Figure 2: Dual parametrization of the distributions in $\mathcal{P}_x$. Our parameter space (left) is first linearly mapped to the space of factor log-potentials (middle). The latter is mapped to the marginal polytope $\mathcal{M}(\mathcal{G}_x)$ (right). In general only a subset of $\mathcal{M}(\mathcal{G}_x)$ is reachable from our parameter space. Any distribution in $\mathcal{P}_x$ can be parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ or by a point $\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)$.

2. *The problem in Eq. 9 is convex and its solution is attained at the factor marginals, i.e., there is a maximizer $\bar{\boldsymbol{\mu}}$ s.t. $\bar{\mu}_C(\mathbf{y}_C) = \Pr_{\boldsymbol{\theta}}(Y_C = \mathbf{y}_C|x)$ for each $C \in \mathcal{C}$. The gradient of the log-partition function is $\nabla \log Z_x(\boldsymbol{\theta}) = \mathbf{F}(x)\bar{\boldsymbol{\mu}}$.*
3. *The MAP $\hat{\mathbf{y}} \triangleq \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(x)} \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$ can be obtained by solving the linear program*

$$\hat{\boldsymbol{\mu}} \triangleq \boldsymbol{\chi}(\hat{\mathbf{y}}) = \operatorname*{argmax}_{\boldsymbol{\mu} \in \mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)\boldsymbol{\mu}. \quad (10)$$

A proof of this proposition can be found in Martins et al. (2010). Fig. 2 provides an illustration of the dual parametrization implied by Prop. 1.

4 Approximate Inference & Turbo Parsing

We now show how the variational machinery just described relates to message-passing algorithms and provides a common framework for analyzing two recent dependency parsers. Later (§5), Prop. 1 is used constructively for learning the model parameters.

4.1 Loopy BP as a Variational Approximation

For general factor graphs with loops, the marginal polytope $\mathcal{M}(\mathcal{G}_x)$ cannot be compactly specified and the entropy term $H(\boldsymbol{\mu})$ lacks a closed form, rendering exact optimizations in Eqs. 9–10 intractable. A popular *approximate* algorithm for marginal inference is sum-product loopy BP, which passes messages as described in §2 and, upon convergence, computes beliefs via Eqs. 6–7. Were loopy BP exact, these beliefs would be the true marginals and hence a point in the marginal polytope $\mathcal{M}(\mathcal{G}_x)$. However, this need not be the case, as elucidated by Yedidia et al. (2001) and others, who first analyzed loopy BP from a variational perspective. The following two approximations underlie loopy BP:

- The marginal polytope $\mathcal{M}(\mathcal{G}_x)$ is approximated by the *local polytope* $\mathcal{L}(\mathcal{G}_x)$. This is an *outer* bound; its name derives from the fact that it only imposes *local agreement constraints* $\forall i, y_i \in \mathcal{Y}_i, C \in \mathcal{C}$:

$$\textstyle\sum_{y_i} \tau_i(y_i) = 1,\ \sum_{\mathbf{y}_C \sim y_i} \tau_C(\mathbf{y}_C) = \tau_i(y_i). \quad (11)$$

  Namely, it is characterized by $\mathcal{L}(\mathcal{G}_x) \triangleq \{\boldsymbol{\tau} \in \mathbb{R}_+^{|\mathcal{R}|} \mid \text{Eq. 11 holds } \forall i, y_i \in \mathcal{Y}_i, C \in \mathcal{C}\}$. The elements of $\mathcal{L}(\mathcal{G}_x)$ are called *pseudo-marginals*. Clearly, the true marginals satisfy Eq. 11, and therefore $\mathcal{M}(\mathcal{G}_x) \subseteq \mathcal{L}(\mathcal{G}_x)$.
- The entropy H is replaced by its *Bethe approximation* $H_{\text{Bethe}}(\boldsymbol{\tau}) \triangleq \sum_{i=1}^{I}(1 - d_i)H(\boldsymbol{\tau}_i) + \sum_{C \in \mathcal{C}} H(\boldsymbol{\tau}_C)$, where $d_i = |\{C \mid i \in C\}|$ is the number of factors connected to the ith variable, $H(\boldsymbol{\tau}_i) \triangleq -\sum_{y_i} \tau_i(y_i) \log \tau_i(y_i)$ and $H(\boldsymbol{\tau}_C) \triangleq -\sum_{\mathbf{y}_C} \tau_C(\mathbf{y}_C) \log \tau_C(\mathbf{y}_C)$.

Any stationary point of sum-product BP is a local optimum of the variational problem in Eq. 9 with $\mathcal{M}(\mathcal{G}_x)$ replaced by $\mathcal{L}(\mathcal{G}_x)$ and H replaced by H_{Bethe} (Yedidia et al., 2001). Note however that multiple optima may exist, since H_{Bethe} is not necessarily concave, and that BP may not converge.

Table 1 shows closed form expressions for the local agreement constraints and entropies of some hard-constraint factors, obtained by invoking Eq. 7 and observing that $\tau_C(\mathbf{y}_C)$ must be zero if configuration $\mathbf{y}_C$ is forbidden. See Appendix A.

4.2 Two Dependency Turbo Parsers

We next present our main contribution: a formal connection between two recent approximate dependency parsers, which at first sight appear unrelated. Recall that (i) Smith and Eisner (2008) proposed a factor graph (Fig. 1) in which they run loopy BP, and that (ii) Martins et al. (2009) approximate parsing as the solution of a linear program. Here, we fill the blanks in the two approaches: we derive explicitly the variational problem addressed in (i) and we provide the underlying factor graph in (ii). This puts the two approaches side-by-side as approximate methods for marginal and MAP inference. Since both rely on "local" approximations (in the sense

of Eq. 11) that ignore the loops in their graphical models, we dub them *turbo parsers* by analogy with error-correcting turbo decoders (see footnote 1).

Turbo Parser #1: Sum-Product Loopy BP. The factor graph depicted in Fig. 1—call it $\mathcal{G}_x$—includes pairwise soft factors connecting sibling and grandparent arcs.[6] We next characterize the local polytope $\mathcal{L}(\mathcal{G}_x)$ and the Bethe approximation H_{Bethe} inherent in Smith and Eisner's loopy BP algorithm.

Let A be the set of candidate arcs, and $P \subseteq A^2$ the set of pairs of arcs that have factors. Let $\boldsymbol{\tau} = \langle \boldsymbol{\tau}_A, \boldsymbol{\tau}_P \rangle$ with $\boldsymbol{\tau}_A = \langle \tau_a \rangle_{a \in A}$ and $\boldsymbol{\tau}_P = \langle \tau_{ab} \rangle_{\langle a,b \rangle \in P}$. Since all variables are binary, we may write, for each $a \in A$, $\tau_a(1) = z_a$ and $\tau_a(0) = 1 - z_a$, where z_a is a variable constrained to $[0, 1]$. Let $\mathbf{z}_A \triangleq \langle z_a \rangle_{a \in A}$; the local agreement constraints at the TREE factor (see Table 1) are written as $\mathbf{z}_A \in \mathcal{Z}_{\text{tree}}(x)$, where $\mathcal{Z}_{\text{tree}}(x)$ is the *arborescence polytope*, i.e., the convex hull of all incidence vectors of dependency trees (Martins et al., 2009). It is straightforward to write a contingency table and obtain the following local agreement constraints at the pairwise factors:

$$\begin{aligned} &\tau_{ab}(1,1) = z_{ab}, \qquad &&\tau_{ab}(0,0) = 1 - z_a - z_b + z_{ab} \\ &\tau_{ab}(1,0) = z_a - z_{ab}, &&\tau_{ab}(0,1) = z_b - z_{ab}. \end{aligned}$$

Noting that all these pseudo-marginals are constrained to the unit interval, one can get rid of all variables τ_{ab} and write everything as

$$\begin{aligned} &z_a \in [0,1], \quad z_b \in [0,1], \quad z_{ab} \in [0,1], \\ &z_{ab} \leq z_a, \quad z_{ab} \leq z_b, \quad z_{ab} \geq z_a + z_b - 1, \end{aligned} \tag{12}$$

inequalities which, along with $\mathbf{z}_A \in \mathcal{Z}_{\text{tree}}(x)$, define the local polytope $\mathcal{L}(\mathcal{G}_x)$. As for the factor entropies, start by noting that the TREE-factor entropy H_{tree} can be obtained in closed form by computing the marginals $\bar{\mathbf{z}}_A$ and the partition function $Z_x(\boldsymbol{\theta})$ (via the matrix-tree theorem) and recalling the variational representation in Eq. 9, yielding $H_{\text{tree}} = \log Z_x(\boldsymbol{\theta}) - \boldsymbol{\theta}^\top \mathbf{F}(x) \bar{\mathbf{z}}_A$. Some algebra allows writing the overall Bethe entropy approximation as:

$$H_{\text{Bethe}}(\boldsymbol{\tau}) = H_{\text{tree}}(\mathbf{z}_A) - \sum_{\langle a,b \rangle \in P} I_{a;b}(z_a, z_b, z_{ab}), \tag{13}$$

where we introduced the mutual information associated with each pairwise factor, $I_{a;b}(z_a, z_b, z_{ab}) = \sum_{y_a, y_b} \tau_{ab}(y_a, y_b) \log \frac{\tau_{ab}(y_a, y_b)}{\tau_a(y_a)\tau_b(y_b)}$. The approximate variational expression becomes $\log Z_x(\boldsymbol{\theta}) \approx$

$$\begin{aligned} \max_{\mathbf{z}} \quad & \boldsymbol{\theta}^\top \mathbf{F}(x) \mathbf{z} + H_{\text{tree}}(\mathbf{z}_A) - \sum_{\langle a,b \rangle \in P} I_{a;b}(z_a, z_b, z_{ab}) \\ \text{s.t.} \quad & z_{ab} \leq z_a, \quad z_{ab} \leq z_b, \\ & z_{ab} \geq z_a + z_b - 1, \quad \forall \langle a, b \rangle \in P, \\ & \mathbf{z}_A \in \mathcal{Z}_{\text{tree}}, \end{aligned} \tag{14}$$

whose maximizer corresponds to the beliefs returned by the Smith and Eisner's loopy BP algorithm (if it converges).

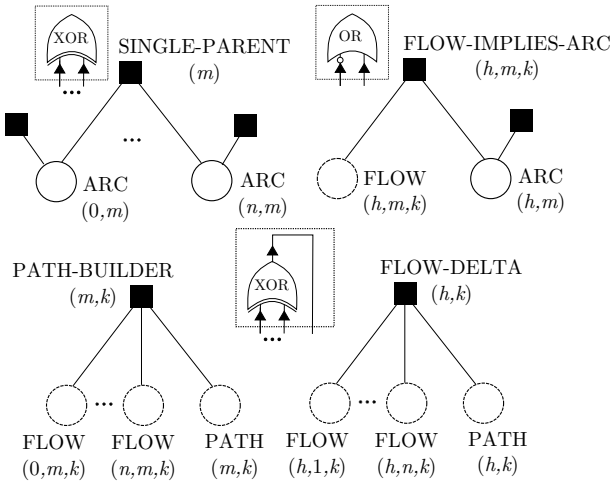

Figure 3: Details of the factor graph underlying the parser of Martins et al. (2009). Dashed circles represent auxiliary variables. See text and Table 1.

Turbo Parser #2: LP-Relaxed MAP. We now turn to the concise integer LP formulation of Martins et al. (2009). The formulation is exact but NP-hard, and so an LP relaxation is made there by dropping the integer constraints. We next construct a factor graph $\mathcal{G}'_x$ and show that the LP relaxation corresponds to an optimization of the form in Eq. 10, with the marginal polytope $\mathcal{M}(\mathcal{G}'_x)$ replaced by $\mathcal{L}(\mathcal{G}'_x)$.

$\mathcal{G}'_x$ includes the following auxiliary variable nodes: *path variables* $\langle p_{ij} \rangle_{i=0,\ldots,n, j=1,\ldots,n}$, which indicate whether word j descends from i in the dependency tree, and *flow variables* $\langle f_a^k \rangle_{a \in A, k=1,\ldots,n}$, which evaluate to 1 iff arc a "carries flow" to k, i.e., iff there is a path from the root to k that passes through a. We need to seed these variables imposing

$$p_{0k} = p_{kk} = 1, \forall k, \quad f^h_{\langle h,m \rangle} = 0, \forall h, m; \tag{15}$$

i.e., any word descends from the root and from itself, and arcs leaving a word carry no flow to that

[6] Smith and Eisner (2008) also proposed other variants with more factors, which we omit for brevity.

word. This can be done with unary hard constraint factors. We then replace the TREE factor in Fig. 1 by the factors shown in Fig. 3:

- $O(n)$ XOR factors, each connecting all arc variables of the form $\{\langle h, m\rangle\}_{h=0,\ldots,n}$. These ensure that each word has exactly one parent. Each factor yields a local agreement constraint (see Table 1):

$$\sum_{h=0}^{n} z_{\langle h,m\rangle} = 1, \quad m \in \{1, \ldots, n\} \tag{16}$$

- $O(n^3)$ IMPLY factors, each expressing that if an arc carries flow, then that arc *must* be active. Such factors are OR factors with the first input negated, hence, the local agreement constraints are:

$$f_a^k \leq z_a, \quad a \in A, k \in \{1, \ldots, n\}. \tag{17}$$

- $O(n^2)$ XOR-WITH-OUTPUT factors, which impose the constraint that each path variable p_{mk} is active if and only if exactly one incoming arc in $\{\langle h, m\rangle\}_{h=0,\ldots,n}$ carries flow to k. Such factors are XOR factors with the last input negated, and hence their local constraints are:

$$p_{mk} = \sum_{h=0}^{n} f_{\langle h,m\rangle}^k, \quad m, k \in \{1, \ldots, n\} \tag{18}$$

- $O(n^2)$ XOR-WITH-OUTPUT factors to impose the constraint that words don't consume other words' commodities; i.e., if $h \neq k$ and $k \neq 0$, then there is a path from h to k iff exactly one outgoing arc in $\{\langle h, m\rangle\}_{m=1,\ldots,n}$ carries flow to k:

$$p_{hk} = \sum_{m=1}^{n} f_{\langle h,m\rangle}^k, \ h, k \in \{0, \ldots, n\}, \ k \notin \{0, h\}. \tag{19}$$

$\mathcal{L}(\mathcal{G}'_x)$ is thus defined by the constraints in Eq. 12 and 15–19. The approximate MAP problem, that replaces $\mathcal{M}(\mathcal{G}'_x)$ by $\mathcal{L}(\mathcal{G}'_x)$ in Eq. 10, thus becomes:

$$\begin{array}{ll} \max_{\mathbf{z},\mathbf{f},\mathbf{p}} & \boldsymbol{\theta}^\top \mathbf{F}(x)\mathbf{z} \\ \text{s.t.} & \text{Eqs. 12 and 15–19 are satisfied.} \end{array} \tag{20}$$

This is exactly the LP relaxation considered by Martins et al. (2009) in their multi-commodity flow model, for the configuration with siblings and grandparent features.[7] They also considered a configuration with non-projectivity features—which fire if an arc is non-projective.[8] That configuration can also be obtained here if variables $\{n_{\langle h,m\rangle}\}$ are added to indicate non-projective arcs and OR-WITH-OUTPUT hard constraint factors are inserted to enforce $n_{\langle h,m\rangle} = z_{\langle h,m\rangle} \wedge \bigvee_{\min(h,m)<j<\min(h,m)} \neg p_{hj}$. Details are omitted for space.

In sum, although the approaches of Smith and Eisner (2008) and Martins et al. (2009) look very different, in reality both are variational approximations emanating from Prop. 1, respectively for marginal and MAP inference. However, they operate on distinct factor graphs, respectively Figs. 1 and 3.[9]

5 Online Learning

Our learning algorithm is presented in Alg. 1. It is a generalized online learner that tackles ℓ_2-regularized empirical risk minimization of the form

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \tfrac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + \tfrac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{\theta}; x_i, \mathbf{y}_i), \tag{21}$$

where each $\langle x_i, \mathbf{y}_i\rangle$ is a training example, $\lambda \geq 0$ is the regularization constant, and $L(\boldsymbol{\theta}; x, \mathbf{y})$ is a non-negative convex loss. Examples include the logistic loss used in CRFs ($-\log \Pr_{\boldsymbol{\theta}}(\mathbf{y}|x)$) and the hinge loss of structured SVMs ($\max_{\mathbf{y}'\in\mathcal{Y}(x)} \boldsymbol{\theta}^\top(\boldsymbol{\phi}(x, \mathbf{y}') - \boldsymbol{\phi}(x, \mathbf{y})) + \ell(\mathbf{y}', \mathbf{y})$ for some cost function ℓ). These are both special cases of the family defined in Fig. 4, which also includes the structured perceptron's loss ($\beta \to \infty$, $\gamma = 0$) and the softmax-margin loss of Gimpel and Smith (2010; $\beta = \gamma = 1$).

Alg. 1 is closely related to stochastic or online gradient descent methods, but with the key advantage of not needing a learning rate hyperparameter. We sketch the derivation of Alg. 1; full details can be found in Martins et al. (2010). On the tth round, one example $\langle x_t, \mathbf{y}_t\rangle$ is considered. We seek to solve

$$\begin{array}{ll} \min_{\boldsymbol{\theta},\xi} & \frac{\lambda m}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 + \xi \\ \text{s.t.} & L(\boldsymbol{\theta}; x_t, \mathbf{y}_t) \leq \xi, \quad \xi \geq 0, \end{array} \tag{23}$$

[7] To be precise, the constraints of Martins et al. (2009) are recovered after eliminating the path variables, via Eqs. 18–19.

[8] An arc $\langle h, m\rangle$ is non-projective if there is some word in its span not descending from h (Kahane et al., 1998).

[9] Given what was just exposed, it seems appealing to try max-product loopy BP on the factor graph of Fig. 1, or sum-product loopy BP on the one in Fig. 3. Both attempts present serious challenges: the former requires computing messages sent by the tree factor, which requires $O(n^2)$ calls to the Chu-Liu-Edmonds algorithm and hence $O(n^5)$ time. No obvious strategy seems to exist for simultaneous computation of all messages, unlike in the sum-product case. The latter is even more challenging, as standard sum-product loopy BP has serious issues in the factor graph of Fig. 3; we construct in Appendix B a simple example with a very poor Bethe approximation. This might be fixed by using other variants of sum-product BP, e.g., ones in which the entropy approximation is concave.

$$L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) \triangleq \frac{1}{\beta} \log \sum_{\mathbf{y}' \in \mathcal{Y}(x)} \exp \left[\beta \Big(\boldsymbol{\theta}^\top \big(\boldsymbol{\phi}(x, \mathbf{y}') - \boldsymbol{\phi}(x, \mathbf{y}) \big) + \gamma \ell(\mathbf{y}', \mathbf{y}) \Big) \right] \tag{22}$$

Figure 4: A family of loss functions including as particular cases the ones used in CRFs, structured SVMs, and the structured perceptron. The hyperparameter β is the analogue of the inverse temperature in a Gibbs distribution, while γ scales the cost. For any choice of $\beta > 0$ and $\gamma \geq 0$, the resulting loss function is convex in $\boldsymbol{\theta}$, since, up to a scale factor, it is the composition of the (convex) log-sum-exp function with an affine map.

Algorithm 1 Aggressive Online Learning

1: **Input:** $\{\langle x_i, \mathbf{y}_i \rangle\}_{i=1}^m$, λ, number of epochs K
2: Initialize $\boldsymbol{\theta}_1 \leftarrow \mathbf{0}$; set $T = mK$
3: **for** $t = 1$ **to** T **do**
4: Receive instance $\langle x_t, \mathbf{y}_t \rangle$ and set $\boldsymbol{\mu}_t = \boldsymbol{\chi}(\mathbf{y}_t)$
5: Solve Eq. 24 to obtain $\bar{\boldsymbol{\mu}}_t$ and $L_{\beta,\gamma}(\boldsymbol{\theta}_t, x_t, \mathbf{y}_t)$
6: Compute $\nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t, x_t, \mathbf{y}_t) = \mathbf{F}(x_t)(\bar{\boldsymbol{\mu}}_t - \boldsymbol{\mu}_t)$
7: Compute $\eta_t = \min \left\{ \frac{1}{\lambda m}, \frac{L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)}{\|\nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)\|^2} \right\}$
8: Return $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla L_{\beta,\gamma}(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)$
9: **end for**
10: Return the averaged model $\bar{\boldsymbol{\theta}} \leftarrow \frac{1}{T} \sum_{t=1}^T \boldsymbol{\theta}_t$.

which trades off conservativeness (stay close to the most recent solution $\boldsymbol{\theta}_t$) and correctness (keep the loss small). Alg. 1's lines 7–8 are the result of taking the first-order Taylor approximation of L around $\boldsymbol{\theta}_t$, which yields the lower bound $L(\boldsymbol{\theta}; x_t, \mathbf{y}_t) \geq L(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t) + (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \nabla L(\boldsymbol{\theta}_t; x_t, \mathbf{y}_t)$, and plugging that linear approximation into the constraint of Eq. 23, which gives a simple Euclidean projection problem (with slack) with a closed-form solution.

The online updating requires evaluating the loss and computing its gradient. Both quantities can be computed using the variational expression in Prop. 1, for any loss $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y})$ in Fig. 4.[10] Our only assumption is that the cost function $\ell(\mathbf{y}', \mathbf{y})$ can be written as a sum over factor-local costs; letting $\boldsymbol{\mu} = \boldsymbol{\chi}(\mathbf{y})$ and $\boldsymbol{\mu}' = \boldsymbol{\chi}(\mathbf{y}')$, this implies $\ell(\mathbf{y}', \mathbf{y}) = \mathbf{p}^\top \boldsymbol{\mu}' + q$ for some $\mathbf{p}$ and q which are constant with respect to $\boldsymbol{\mu}'$.[11] Under this assumption, $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y})$ becomes expressible in terms of the log-partition function of a distribution whose log-potentials are set to $\beta(\mathbf{F}(x)^\top \boldsymbol{\theta} + \gamma \mathbf{p})$. From Eq. 9 and after some algebra, we finally obtain $L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) =$

$$\max_{\boldsymbol{\mu}' \in \mathcal{M}(\mathcal{G}_x)} \boldsymbol{\theta}^\top \mathbf{F}(x)(\boldsymbol{\mu}' - \boldsymbol{\mu}) + \frac{1}{\beta} H(\boldsymbol{\mu}') + \gamma(\mathbf{p}^\top \boldsymbol{\mu}' + q). \tag{24}$$

Let $\bar{\boldsymbol{\mu}}$ be a maximizer in Eq. 24; from the second statement of Prop. 1 we obtain $\nabla L_{\beta,\gamma}(\boldsymbol{\theta}; x, \mathbf{y}) = \mathbf{F}(x)(\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})$. When the inference problem in Eq. 24 is intractable, approximate message-passing algorithms like loopy BP still allow us to obtain approximations of the loss $L_{\beta,\gamma}$ and its gradient.

For the hinge loss, we arrive precisely at the max-loss variant of 1-best MIRA (Crammer et al., 2006). For the logistic loss, we arrive at a new online learning algorithm for CRFs that resembles stochastic gradient descent but with an automatic step size that follows from our variational representation.

Unsupported Features. As datasets grow, so do the sets of features, creating further computational challenges. Often only "supported" features—those observed in the training data—are included, and even those are commonly eliminated when their frequencies fall below a threshold. Important information may be lost as a result of these expedient choices. Formally, the *supported feature set* is $\mathcal{F}_{\text{supp}} \triangleq \bigcup_{i=1}^m \operatorname{supp} \boldsymbol{\phi}(x_i, \mathbf{y}_i)$, where $\operatorname{supp} \mathbf{u} \triangleq \{j \mid u_j \neq 0\}$ denotes the *support* of vector $\mathbf{u}$. $\mathcal{F}_{\text{supp}}$ is a subset of the *complete* feature set, comprised of those features that occur in some candidate output, $\mathcal{F}_{\text{comp}} \triangleq \bigcup_{i=1}^m \bigcup_{\mathbf{y}_i' \in \mathcal{Y}(x_i)} \operatorname{supp} \boldsymbol{\phi}(x_i, \mathbf{y}_i')$. Features in $\mathcal{F}_{\text{comp}} \setminus \mathcal{F}_{\text{supp}}$ are called *unsupported*.

Sha and Pereira (2003) have shown that training a CRF-based shallow parser with the complete feature set may improve performance (over the supported one), at the cost of 4.6 times more features. Dependency parsing has a much higher ratio (around 20 for bilexical word-word features, as estimated in the Penn Treebank), due to the quadratic or faster growth of the number of parts, of which only a few are active in a legal output. We propose a simple strategy for handling $\mathcal{F}_{\text{comp}}$ efficiently, which can be applied for those losses in Fig. 4 where $\beta = \infty$. (e.g., the structured SVM and perceptron). Our procedure is the following: keep an active set $\mathcal{F}$ contain-

[10] Our description also applies to the (non-differentiable) hinge loss case, when $\beta \to \infty$, if we replace all instances of "the gradient" in the text by "a subgradient."

[11] For the Hamming cost, this holds with $\mathbf{p} = 1 - 2\boldsymbol{\mu}$ and $q = \mathbf{1}^\top \boldsymbol{\mu}$. See Taskar et al. (2006) for other examples.

	CRF (TURBO PARS. #1)		SVM (TURBO PARS. #2)		SVM (TURBO #2)		
	ARC-FACT.	SEC. ORD.	ARC-FACT.	SEC. ORD.	$\lvert\mathcal{F}\rvert$	$\frac{\lvert\mathcal{F}\rvert}{\lvert\mathcal{F}_{\text{supp}}\rvert}$	+NONPROJ., COMPL.
ARABIC	78.28	79.12	79.04	79.42	6,643,191	2.8	80.02 (-0.14)
BULGARIAN	91.02	91.78	90.84	92.30	13,018,431	2.1	92.88 (+0.34) (†)
CHINESE	90.58	90.87	91.09	91.77	28,271,086	2.1	91.89 (+0.26)
CZECH	86.18	87.72	86.78	88.52	83,264,645	2.3	88.78 (+0.44) (†)
DANISH	89.58	90.08	89.78	90.78	7,900,061	2.3	91.50 (+0.68)
DUTCH	82.91	84.31	82.73	84.17	15,652,800	2.1	84.91 (-0.08)
GERMAN	89.34	90.58	89.04	91.19	49,934,403	2.5	91.49 (+0.32) (†)
JAPANESE	92.90	93.22	93.18	93.38	4,256,857	2.2	93.42 (+0.32)
PORTUGUESE	90.64	91.00	90.56	91.50	16,067,150	2.1	91.87 (-0.04)
SLOVENE	83.03	83.17	83.49	84.35	4,603,295	2.7	85.53 (+0.80)
SPANISH	83.83	85.07	84.19	85.95	11,629,964	2.6	87.04 (+0.50) (†)
SWEDISH	87.81	89.01	88.55	88.99	18,374,160	2.8	89.80 (+0.42)
TURKISH	76.86	76.28	74.79	76.10	6,688,373	2.2	76.62 (+0.62)
ENGLISH NON-PROJ.	90.15	91.08	90.66	91.79	57,615,709	2.5	92.13 (+0.12)
ENGLISH PROJ.	91.23	91.94	91.65	92.91	55,247,093	2.4	93.26 (+0.41) (†)

Table 2: Unlabeled attachment scores, ignoring punctuation. The leftmost columns show the performance of arc-factored and second-order models for the CRF and SVM losses, after 10 epochs with $1/(\lambda m) = 0.001$ (tuned on the English Non-Proj. dev.-set). The rightmost columns refer to a model to which non-projectivity features were added, trained under the SVM loss, that handles the *complete* feature set. Shown is the total number of features instantiated, the multiplicative factor w.r.t. the number of supported features, and the accuracies (in parenthesis, we display the difference w.r.t. a model trained with the supported features only). Entries marked with † are the highest reported in the literature, to the best of our knowledge, beating (sometimes slightly) McDonald et al. (2006), Martins et al. (2008), Martins et al. (2009), and, in the case of English Proj., also the third-order parser of Koo and Collins (2010), which achieves 93.04% on that dataset (their experiments in Czech are not comparable, since the datasets are different).

ing all features that have been instantiated in Alg. 1. At each round, run lines 4–5 as usual, using only features in $\mathcal{F}$. Since the other features have not been used before, they have a zero weight, hence can be ignored. When $\beta = \infty$, the variational problem in Eq. 24 consists of a MAP computation and the solution corresponds to one output $\hat{\mathbf{y}}_t \in \mathcal{Y}(x_t)$. Only the parts that are active in $\hat{\mathbf{y}}_t$ but not in $\mathbf{y}_t$, or vice-versa, will have features that might receive a nonzero update. Those parts are reexamined for new features and the active set $\mathcal{F}$ is updated accordingly.

6 Experiments

We trained non-projective dependency parsers for 14 languages, using datasets from the CoNLL-X shared task (Buchholz and Marsi, 2006) and two datasets for English: one from the CoNLL-2008 shared task (Surdeanu et al., 2008), which contains non-projective arcs, and another derived from the Penn Treebank applying the standard head rules of Yamada and Matsumoto (2003), in which all parse trees are projective.[12] We implemented Alg. 1, which handles any loss function $L_{\beta,\gamma}$.[13] When $\beta < \infty$, Turbo Parser #1 and the loopy BP algorithm of Smith and Eisner (2008) is used; otherwise, Turbo Parser #2 is used and the LP relaxation is solved with CPLEX. In both cases, we employed the same pruning strategy as Martins et al. (2009).

Two different feature configurations were first tried: an arc-factored model and a model with second-order features (siblings and grandparents). We used the same arc-factored features as McDonald et al. (2005) and second-order features that conjoin words and lemmas (at most two), parts-of-speech tags, and (if available) morphological information; this was the same set of features as in Martins et al. (2009). Table 2 shows the results obtained in both configurations, for CRF and SVM loss functions. While in the arc-factored case performance is similar, in second-order models there seems to be a consistent gain when the SVM loss is used. There are two possible reasons: first, SVMs take the cost function into consideration; second, Turbo Parser #2 is less approximate than Turbo Parser #1, since only the marginal polytope is approximated (the entropy function is not involved).

[12] We used the provided train/test splits for all datasets. For English, we used the standard test partitions (section 23 of the Wall Street Journal). We did not exploit the fact that some datasets only contain projective trees and have unique roots.

[13] The code is available at `http://www.ark.cs.cmu.edu/TurboParser`.

β	1	1	1	1	3	5	∞
γ	0 (CRF)	1	3	5	1	1	1 (SVM)
ARC-F.	90.15	90.41	90.38	90.53	90.80	90.83	90.66
2 ORD.	91.08	91.85	91.89	91.51	92.04	91.98	91.79

Table 3: Varying β and γ: neither the CRF nor the SVM is optimal. Results are UAS on the English Non-Projective dataset, with λ tuned with dev.-set validation.

The loopy BP algorithm managed to converge for nearly all sentences (with message damping). The last three columns show the beneficial effect of unsupported features for the SVM case (with a more powerful model with non-projectivity features). For most languages, unsupported features convey helpful information, which can be used with little extra cost (on average, 2.5 times more features are instantiated). A combination of the techniques discussed here yields parsers that are in line with very strong competitors—for example, the parser of Koo and Collins (2010), which is exact, third-order, and constrains the outputs to be projective, does not outperform ours on the projective English dataset.[14]

Finally, Table 3 shows results obtained for different settings of β and γ. Interestingly, we observe that higher scores are obtained for loss functions that are "between" SVMs and CRFs.

7 Related Work

There has been recent work studying efficient computation of messages in combinatorial factors: bipartite matchings (Duchi et al., 2007), projective and non-projective arborescences (Smith and Eisner, 2008), as well as high order factors with count-based potentials (Tarlow et al., 2010), among others. Some of our combinatorial factors (OR, OR-WITH-OUTPUT) and the analogous entropy computations were never considered, to the best of our knowledge.

Prop. 1 appears in Wainwright and Jordan (2008) for canonical overcomplete models; we adapt it here for models with shared features. We rely on the variational interpretation of loopy BP, due to Yedidia et al. (2001), to derive the objective being optimized by Smith and Eisner's loopy BP parser.

Independently of our work, Koo et al. (2010) recently proposed an efficient dual decomposition method to solve an LP problem similar (but not equal) to the one in Eq. 20,[15] with excellent parsing performance. Their parser is also an instance of a turbo parser since it relies on a local approximation of a marginal polytope. While one can also use dual decomposition to address our MAP problem, the fact that our model does not decompose as nicely as the one in Koo et al. (2010) would likely result in slower convergence.

8 Conclusion

We presented a unified view of two recent approximate dependency parsers, by stating their underlying factor graphs and by deriving the variational problems that they address. We introduced new hard constraint factors, along with formulae for their messages, local belief constraints, and entropies. We provided an aggressive online algorithm for training the models with a broad family of losses.

There are several possible directions for future work. Recent progress in message-passing algorithms yield "convexified" Bethe approximations that can be used for marginal inference (Wainwright et al., 2005), and provably convergent max-product variants that solve the relaxed LP (Globerson and Jaakkola, 2008). Other parsing formalisms can be handled with the inventory of factors shown here—among them, phrase-structure parsing.

A Proofs

We derive the expressions in the first row of Table 1; the remaining rows are instantiations of the general case. For the sum-product case, we have

$$\begin{aligned} m_{C\to i} &= \frac{M_{C\to i}(1)}{M_{C\to i}(0)} = \frac{\Pr\{\Psi_C(Y_C)=1|Y_i=1\}}{\Pr\{\Psi_C(Y_C)=1|Y_i=0\}} \\ &= \frac{\Pr\{Y_i=1|\Psi_C(Y_C)=1\}\Pr\{Y_i=0\}}{\Pr\{Y_i=0|\Psi_C(Y_C)=1\}\Pr\{Y_i=1\}} \\ &= m_{i\to C}^{-1}\cdot \textsc{marg}_i(\boldsymbol{\omega})/(1-\textsc{marg}_i(\boldsymbol{\omega})); \end{aligned}$$

the proof for the max-product case is similar. To derive the local agreement constraints and the factor

[14] This might be due to the fact that Koo and Collins (2010) trained with the perceptron algorithm and did not use unsupported features. Experiments plugging the perceptron loss ($\beta\to\infty, \gamma\to 0$) into Alg. 1 yielded worse performance than with the hinge loss.

[15] The difference is that the model of Koo et al. (2010) includes features that depend on *consecutive* siblings—making it decompose into subproblems amenable to dynamic programming—while we have factors for *all pairs* of siblings.

entropies, note first that $\tau_C(\mathbf{y}_C) = 0$ if $\mathbf{y}_C \notin \mathcal{S}_C$ (this comes from the fact that forbidden configurations' log-potentials are $-\infty$). Since the allowed configurations have zero log-potentials, the variables $\tau_C(\mathbf{y}_C)$ need not be explicit; we will also see that the factor entropy (involved in Bethe's approximation) can be computed from the variable-to-factor messages, once a fixed point has been reached. Replacing $z_i \triangleq \tau_i(1)$ and $1-z_i \triangleq \tau_i(0)$, we can rewrite the local agreement constraints in Eq. 11 as $\mathbf{z} \in \mathcal{Z}$, where

$$\begin{aligned}\mathcal{Z} \;\triangleq\; & \{\mathbf{z} \geq 0 \mid \exists \tau_C(\mathbf{y}_C) \geq 0 \text{ s.t. } \forall i \\ & \quad z_i = \sum_{\substack{\mathbf{y}_C \in \mathcal{S}_C \\ [\mathbf{y}_C]_i = 1}} \tau_C(\mathbf{y}_C) = 1 - \sum_{\substack{\mathbf{y}_C \in \mathcal{S}_C \\ [\mathbf{y}_C]_i = 0}} \tau_C(\mathbf{y}_C) \Bigg\} \\ \triangleq\; & \Bigg\{\mathbf{z} \mid \exists \tau_C(\mathbf{y}_C) \geq 0, \sum_{\mathbf{y}_C \in \mathcal{S}_C} \tau_C(\mathbf{y}_C) = 1 \text{ s.t.} \\ & \quad \mathbf{z} = \sum_{\mathbf{y}_C \in \mathcal{S}_C} \tau_C(\mathbf{y}_C)\mathbf{y}_C \Bigg\} = \operatorname{conv} \mathcal{S}_C. \end{aligned} \tag{25}$$

To derive a formula for the entropy $H_C = -\sum_{\mathbf{y}_C} \tau_C(\mathbf{y}_C) \log \tau_C(\mathbf{y}_C)$, consider Eq. 7 and set the proportionality constant of the variable-to-factor messages so that $M_{i \to C}(1) = m_{i \to C}$ and $M_{i \to C}(0) = 1$. The partition function becomes

$$Z_C = \sum_{\mathbf{y}_C} \Psi_C(\mathbf{y}_C) \prod_{i=1}^{n} M_{i \to C}(y_i) = \sum_{\mathbf{y}_C \in \mathcal{S}_C} \prod_{i=1}^{n} m_{i \to C}^{y_i}$$

and one can now invoke the relation in Eq. 9 to obtain $H_C = \log Z_C - \sum_{i=1}^{n} \textsc{marg}_i(\boldsymbol{\omega}) \log m_{i \to C}$.

B Non-Convergence of Loopy Sum-Product BP for the Flow-Based Graph

We next show that there are very serious issues with the sum-product loopy BP algorithm when ran in the factor graph of Fig. 3. We provide a very simple example—a sentence with 2 words, using an arc-factored model with all log-potentials set to zero—where it will not converge to the desired solution. For such a sentence, only three possible dependency trees exist: one tree (T_1) where the root is the parent of both words, and two symmetric trees (T_2 and T_3) where one of the words is the parent of the other word. Since the log-potentials are zero, all the trees are equiprobable. Table 4 shows the values of all the variables associated with the factor graph in Fig. 3 for each of the trees, as well as their true marginals (which are their values averaged over the trees). The factor graph is comprised of the following factors (all of them pairwise):

- Two "single-parent" factors $\textsc{sp}(1)$ and $\textsc{sp}(2)$ implementing the XOR function. $\textsc{sp}(1)$ is connected to the variables $\textsc{arc}(0,1)$ and $\textsc{arc}(2,1)$, and $\textsc{sp}(2)$ is connected to $\textsc{arc}(0,2)$ and $\textsc{arc}(1,2)$.
- Two "flow-implies-arc" factors $\textsc{fia}(0,1,2)$ and $\textsc{fia}(0,2,1)$ implementing the OR function (with the first input negated). $\textsc{fia}(0,1,2)$ is connected to the variables $\textsc{flow}(0,1,2)$ and $\textsc{arc}(0,1)$, and $\textsc{fia}(0,2,1)$ is connected to $\textsc{flow}(0,2,1)$ and $\textsc{arc}(0,2)$.
- Two "path-builder" factors $\textsc{pb}(1,2)$ and $\textsc{pb}(2,1)$ implementing the XOR function (with the last input negated). $\textsc{pb}(1,2)$ is connected to the variables $\textsc{flow}(0,1,2)$ and $\textsc{path}(1,2)$, and $\textsc{pb}(2,1)$ is connected to $\textsc{flow}(0,2,1)$ and $\textsc{path}(2,1)$.[16]
- Four "flow-delta" factors $\textsc{fd}(0,1)$, $\textsc{fd}(0,2)$, $\textsc{fd}(1,2)$, $\textsc{fd}(2,1)$ implementing the XOR function. $\textsc{fd}(0,1)$ is connected to the variables $\textsc{arc}(0,1)$ (originally $\textsc{flow}(0,1,1)$) and $\textsc{flow}(0,2,1)$ ($\textsc{path}(0,1)$ is seeded to 1 and hence removed); $\textsc{fd}(0,2)$ is connected to $\textsc{arc}(0,2)$ and $\textsc{flow}(0,1,2)$ ($\textsc{path}(0,2)$ is seeded to 1 and hence removed); $\textsc{fd}(1,2)$ is connected to $\textsc{arc}(1,2)$ and $\textsc{path}(1,2)$ (which

[16] Note that the original, unsimplified model has two more path-builder factors $\textsc{pb}(1,1)$ and $\textsc{pb}(2,2)$. For example, the former is originally connected to $\textsc{flow}(0,1,1)$ (which is replaced by $\textsc{arc}(0,1)$), $\textsc{flow}(2,1,1)$ (which is replaced by $\textsc{arc}(2,1)$), and $\textsc{path}(1,1)$. Since the last variable is seeded to 1 and negated, it behaves as a neutral element and can be removed, turning $\textsc{pb}(1,1)$ into a factor exactly equal to $\textsc{sp}(1)$. The same happens for $\textsc{pb}(2,2)$, which becomes equal to $\textsc{sp}(2)$. Hence, the model can be simplified by omitting these two factors. Note that the Bethe approximation (and hence, the outcome of loopy BP) become different with and without the simplification. However, they both yield the same conclusion drawn in this section.

	T_1	T_2	T_3	MARGINALS	DEGREE
$z_{\langle 0,1\rangle}$	1	1	0	2/3	3
$z_{\langle 0,2\rangle}$	1	0	1	2/3	3
$z_{\langle 1,2\rangle}$	0	1	0	1/3	2
$z_{\langle 2,1\rangle}$	0	0	1	1/3	2
$f^2_{\langle 0,1\rangle}$	0	1	0	1/3	3
$f^1_{\langle 0,2\rangle}$	0	0	1	1/3	3
p_{12}	0	1	0	1/3	2
p_{21}	0	0	1	1/3	2

Table 4: Arc, flow, and path variables, along with their values for each tree, marginals, degrees and entropies. "Seed variables" (see Eq. 15) are not included; we also do not consider the "redundant" variables $f^m_{\langle h,m\rangle}$ which necessarily take the same value as $z_{\langle h,m\rangle}$.

is negated); $\text{FD}(2,1)$ is connected to $\text{ARC}(2,1)$ and $\text{PATH}(2,1)$ (which is negated).

This is schematized in Table 5. We now turn to deriving the entropies of the variables and factors. The former are obtained directly from the value of the marginals, yielding $\sum_i (1-d_i) H_i(\boldsymbol{\tau}_i) = (-2-2-1-1-2-2-1-1)H(\frac{1}{3},\frac{2}{3}) = -12H(\frac{1}{3},\frac{2}{3})$. For the factors, note that the entropy of a XOR factor whose incoming variables have marginals $z_1,\ldots,z_n$ equals the entropy of $H(z_1,\ldots,z_n)$; all eight XOR factors in Table 5 have thus entropy $H(\frac{1}{3},\frac{2}{3})$. It only remains to compute the entropy of the two "flow-implies-arc" OR factors. Enumerating the feasible three states of these factors, each with probability proportional to the product of input marginals, we have that each of these entropies is $H(\frac{1}{3},\frac{1}{3},\frac{1}{3}) = \log 3$. Therefore, the Bethe approximation is

$$\begin{aligned} H_{\text{Bethe}}(\boldsymbol{\tau}) &= -12H(\tfrac{1}{3},\tfrac{2}{3}) + 8H(\tfrac{1}{3},\tfrac{2}{3}) + 2\log 3 \\ &= -4H(\tfrac{1}{3},\tfrac{2}{3}) + 2\log 3 \\ &= -0.3488 < 0, \end{aligned} \quad (26)$$

which is negative! This is a very poor approximation of the true entropy, which equals $\log 3$ (since there are three equiprobable trees). Furthermore, any set of degenerate marginals (corresponding to one of the three trees) would yield a zero entropy and a zero Bethe entropy. Since loopy BP will attempt to maximize the variational problem described in §4.1 with zero log-potentials, even if it converges to the global optimum it will not retrieve the true marginals depicted in Table 4, since these yield the objective value 0, which is lower than the one corresponding to the degenerate marginals.

$\text{SP}(1)$	$\text{XOR}(\text{ARC}(0,1), \text{ARC}(2,1))$
$\text{SP}(2)$	$\text{XOR}(\text{ARC}(0,2), \text{ARC}(1,2))$
$\text{FIA}(0,1,2)$	$\text{OR}(\neg\text{FLOW}(0,1,2), \text{ARC}(0,1))$
$\text{FIA}(0,2,1)$	$\text{OR}(\neg\text{FLOW}(0,2,1), \text{ARC}(0,2))$
$\text{PB}(1,2)$	$\text{XOR}(\text{FLOW}(0,1,2), \neg\text{PATH}(1,2))$
$\text{PB}(2,1)$	$\text{XOR}(\text{FLOW}(0,2,1), \neg\text{PATH}(2,1))$
$\text{FD}(0,1)$	$\text{XOR}(\text{ARC}(0,1), \text{FLOW}(0,2,1))$
$\text{FD}(0,2)$	$\text{XOR}(\text{ARC}(0,2), \text{FLOW}(0,1,2))$
$\text{FD}(1,2)$	$\text{XOR}(\text{ARC}(1,2), \neg\text{PATH}(1,2))$
$\text{FD}(2,1)$	$\text{XOR}(\text{ARC}(2,1), \neg\text{PATH}(2,1))$

Table 5: Factors for the two-word sentence example.

Acknowledgments

The authors would like to thank the reviewers for their comments, and Kevin Gimpel, David Smith, David Sontag, and Terry Koo for helpful discussions. A. M. was supported by a grant from FCT/ICTI through the CMU-Portugal Program, and also by Priberam Informática. N. S. was supported in part by Qatar NRF NPRP-08-485-1-083. E. X. was supported by AFOSR FA9550010247, ONR N000140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship. M. F. and P. A. were supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250).

References

C. Berrou, A. Glavieux, and P. Thitimajshima. 1993. Near Shannon limit error-correcting coding and decoding. In *Proc. of ICC*, volume 93, pages 1064–1070.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. 2007. Using combinatorial optimization within max-product belief propagation. *NIPS*, 19.

J. R. Finkel, A. Kleeman, and C. D. Manning. 2008. Efficient, feature-based, conditional random field parsing. *Proc. of ACL*.

K. Gimpel and N. A. Smith. 2010. Softmax-margin crfs: Training log-linear models with loss functions. In *Proc. of NAACL*.

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

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *Proc. of COLING*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

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

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

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498–519.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *EMNLP*.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*.

A. F. T. Martins, K. Gimpel, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Learning structured classifiers with dual coordinate descent. Technical Report CMU-ML-10-109.

A. McCallum, K. Schultz, and S. Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *NIPS*.

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of CoNLL*.

R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. 1998. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2).

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

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

D. A. Smith and N. A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP*.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *CoNLL*.

C. Sutton, A. McCallum, and K. Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *JMLR*, 8:693–723.

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

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

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS*.

B. Taskar, S. Lacoste-Julien, and M. I. Jordan. 2006. Structured prediction, dual extragradient and Bregman projections. *JMLR*, 7:1627–1653.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

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

M. J. Wainwright, T.S. Jaakkola, and A.S. Willsky. 2005. A new class of upper bounds on the log partition function. *IEEE Trans. Inf. Theory*, 51(7):2313–2335.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. 2001. Generalized belief propagation. In *NIPS*.