Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence

Caio Corro  Joseph Le Roux  Mathieu Lacroix
Laboratoire d’Informatique de Paris Nord, Université Paris 13 – SPC, CNRS UMR 7030, F-93430, Villetaneuse, France
{corro,leroux,lacroix}@lipn.fr

Abstract

We present a new method for the joint task of tagging and non-projective dependency parsing. We demonstrate its usefulness with an application to discontinuous phrase-structure parsing where decoding lexicalized spines and syntactic derivations is performed jointly. The main contributions of this paper are (1) a reduction from joint tagging and non-projective dependency parsing to the Generalized Maximum Spanning Arborescence problem, and (2) a novel decoding algorithm for this problem through Lagrangian relaxation. We evaluate this model and obtain state-of-the-art results despite strong independence assumptions.

1 Introduction

Discontinuous phrase-structure parsing relies either on formal grammars such as LCFRS, which suffer from a high complexity, or on reductions to non-projective dependency parsing with complex labels to encode phrase combinations. We propose an alternative approach based on a variant of spinal TAGs, which allows parses with discontinuity while grounding this work on a lexicalized phrase-structure grammar. Contrarily to previous approaches, (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015), we do not model supertagging nor spine interactions with a complex label scheme. We follow Carreras et al. (2008) but drop projectivity.

We first show that our discontinuous variant of spinal TAG reduces to the Generalized Maximum Spanning Arborescence (GMSA) problem (Myung et al., 1995). In a graph where vertices are partitioned into clusters, GMSA consists in finding the arborescence of maximum weight incident to exactly one vertex per cluster. This problem is NP-complete even for arc-factored models. In order to bypass complexity, we resort to Lagrangian relaxation and propose an efficient resolution based on dual decomposition which combines a simple non-projective dependency parser on a contracted graph and a local search on each cluster to find a global consensus.

We evaluated our model on the discontinuous PTB (Evang and Kallmeyer, 2011) and the Tiger (Brants et al., 2004) corpora. Moreover, we show that our algorithm is able to quickly parse the whole test sets.

Section 2 presents the parsing problem. Section 3 introduces GMSA from which we derive an effective resolution method in Section 4. In Section 5 we define a parameterization of the parser which uses neural networks to model local probabilities and present experimental results in Section 6. We discuss related work in Section 7.

2 Joint Supertagging and Spine Parsing

In this section we introduce our problem and set notation. The goal of phrase-structure parsing is to produce a derived tree by means of a sequence of operations called a derivation. For instance in context-free grammars the derived tree is built from a sequence of substitutions of a non-terminal symbol with a string of symbols, whereas in tree adjoining grammars (TAGs) a derivation is a sequence of substitutions and adjunctions over elementary trees. We are especially interested in building discontinuous phrase-structure trees which may contain constituents with gaps.1

We follow Shen (2006) and build derived trees from adjunctions performed on spines. Spines are lexicalized unary trees where each level represents

1 Although we will borrow concepts from TAGs, we do not require derivations to be TAG compatible (i.e. well-nested dependencies with a bounded number of gaps).
a lexical projection of the anchor. Carreras et al. (2008) showed how spine-based parsing could be reduced to dependency parsing: since spines are attached to words, equivalent derivations can be represented as a dependency tree where arcs are labeled by spine operations, an adjunction together with information about the adjunction site. However, we depart from previous approaches (Shen and Joshi, 2008; Carreras et al., 2008) by relaxing the projectivity constraint to represent all discontinuous phrase-structure trees (see Figure 1).

We assume a finite set of spines $S$. A spine $s$ can be defined as a sequence of grammatical categories, beginning at root. For a sentence $w = (w_0, w_1, \ldots, w_n)$ where $w_k$ is the word at position $k$ and $w_0$ is a dummy root symbol, a derivation is a triplet $(d, s, l)$ defined as follows. Adjunctions are described by a dependency tree rooted at 0 written as a sequence of arcs $d$. If $(h, m) \in d$ with $h \in \{0, \ldots, n\}$ and $m \in \{1, \ldots, n\}$, then the derivation contains an adjunction of the root of the spine at position $m$ to a node from the spine at position $h$. Supertagging, the assignment of a spine to each word, is represented by a sequence where $s$ is a triplet $s = (s_0, s_1, \ldots, s_n)$ of $n+1$ spines, each spine $s_k$ being assigned to word $w_k$. Finally, labeling $l = (l_1, \ldots, l_n)$ is a sequence where $l_k$ is the label of the $k^{th}$ arc $(h, m)$ of $d$. The label consists of a couple $(op, i)$ where $op$ is the type of adjunction, here sister or regular, and $i$ is the index of the adjunction node in $s_h$.

Each derivation is assigned an arc-factored score $\sigma$ which is given by:

$$
\sigma(d, s, l; w) = \sum_{(h, m) \in d} \Omega(h, m, s_h, s_m, l_{hm}; w)
$$

For instance, following score functions developed in (Carreras et al., 2008), this function could read $s_h[i], s_h[i+1]$ and $s_m[0]$, where $s[i]$ denotes the $i^{th}$ grammatical category of the spine $s$. The score of the derivation in Figure 1 could then reflect that the spine $WHNP-WP$ associated with $What$ is adjoined on the spine $SBARQ-SQ-VP-VB$ associated with $do$ on a site with the grammatical triple $[VP WHNP VB]$. We assume that $\Omega$ accounts for the contribution of arcs, spines and labels to the score. The details of the contribution depend on the model. We choose the following:

$$
\sigma(d, s, l; w) = \sum_{(h, m) \in d} (\alpha(h, m; w) + \nu(s_m; h, m, w) + \gamma(l_{hm}; h, m, s_h, w))
$$

where $\alpha$ is the score related to the dependency tree, $\nu$ is the supertagging score and $\gamma$ the labeling score. Note that functions $\alpha$, $\nu$ and $\gamma$ have access to the entire input string $w$. Score function $\sigma$ can be parameterized in many ways and we discuss our implementation in Section 5. In this setting, parsing a sentence $w$ amounts to finding the highest-scoring derivation $(d^*, s^*, l^*) = \arg \max_{(d, s, l; w)} \sigma(d, s, l; w)$.

Recovering the derived tree from a derivation is performed by recursively mapping each spine and its dependencies to a possibly gappy constituent. Given a spine $s_h$ and site index $i$, we look for the leftmost $s_l$ and rightmost $s_r$ dependents attached with regular adjunction. If any, we insert a new node between $s_l[i]$ and $s_r[i+1]$ with the same grammatical category as the first one. This new node fills the role of the foot node in TAGs. Every dependent of $s_h[i]$ with anchor in interval $[l+1, r-1]$ is moved to the newly created node. Remaining sister and regular adjunctions are simply attached to $s_h[i]$.

The complexity of the parsing problem depends on the type of dependency trees. In the case of projective trees, it has been shown (Eisner, 2000; Carreras et al., 2008; Li et al., 2011) that this could be performed in cubic worst-case time complexity with dynamic programming, whether supertags are fixed beforehand or not. However, the modification of the original Eisner algorithm requires that chart cells must be indexed not only by spans, or pairs of positions, but also by pairs of supertags. In practice the problem is intractable unless heavy
pruning is performed first in order to select a subset of spines at each position.

In the case of non-projective dependency trees, the problem has quadratic worst-case time complexity when supertags are fixed, since the problem then amounts to non-projective parsing and reduces to the Maximum Spanning Arborescence problem (MSA) as in (McDonald et al., 2005). Unfortunately, the efficient algorithm for MSA is greedy and does not store potential substructure candidates. Hence, when supertags are not fixed beforehand, a new arborescence must be recomputed for each choice of supertags. This problem can be seen as instance of the Generalized Maximum Spanning Arborescence problem, an NP-complete problem, which we review in the next section. Note that arc labels do not impact the asymmetric complexity of an arc-factored model. Indeed, only the labeled arc with maximum weight between two vertices is considered when parsing.

3 The Generalized Maximum Spanning Arborescence

In this section, we first define GMSA introduced by Myung et al. (1995). We formulate this problem as an integer linear program. We then explain the reduction from the joint supertagging and spine parsing task to this problem.\(^3\)

3.1 Problem definition

Let \( D = (V, A) \) be a directed graph. Given a subset \( T \subseteq A \) of arcs, \( V[T] \) denotes the set of vertices of \( V \) which are the tail or the head of at least one arc of \( T \). These vertices are said to be covered by \( T \). A subset \( T \subseteq A \) of arcs is called an arborescence if the graph \( (V[T], T) \) is connected, acyclic, and each vertex has at most one entering arc. The vertex with no entering arc is called the root of \( T \). An arborescence covering all vertices is called a spanning arborescence.

Let \( \pi = \{V_0, \ldots, V_n\}, n \in \mathbb{N} \) be a partition of \( V \). Each element of \( \pi \) is called a cluster. An arborescence \( T \) of \( D \) covering exactly one vertex per cluster of \( \pi \) is called a generalized spanning arborescence (GSA). Figure 2 gives an example of a GSA. The partition of \( V \) is composed of a cluster having one vertex and six clusters having four vertices. Each cluster is depicted by a hatched area. The GSA is depicted by the dashed arcs.

Let \( W \) be a vertex subset of \( V \). We denote \( \delta^-(W) \) (resp. \( \delta^+(W) \)) the set of arcs entering (resp. leaving) \( W \) and \( \delta(W) = \delta^-(W) \cup \delta^+(W) \).\(^4\)

Contracting \( W \) consists in replacing in \( D \) all vertices in \( W \) by a new vertex \( w \), replacing each arc \( uv \in \delta^-(W) \) by the arc \( uw \) and each arc \( vu \in \delta^+(W) \) by \( wu \). Let \( D^\pi \) be the graph obtained by contracting each cluster of \( \pi \) in \( D \). Note that a GSA of \( D \) and \( \pi \) induces a spanning arborescence of \( D^\pi \).\(^5\) For instance, contracting each cluster in the graph given by Figure 2 leads to a graph \( D^\pi \) having 7 vertices and the set of dashed arcs corresponds to a spanning arborescence of \( D^\pi \).

Given arc weights \( \phi \in \mathbb{R}^A \), the weight of an arborescence \( T \) is \( \sum_{v \in T} \phi_v \). Given \( (D, \pi, \phi) \), the Generalized Maximum Spanning Arborescence problem (GMSA) consists in finding a GSA of \( D \) and \( \pi \) of maximum weight whose root is in \( V_0 \).

3.2 Integer linear program

Given a set \( S, z \in \mathbb{R}^S \) is a vector indexed by elements in \( S \). For \( S' \subseteq S, z(S') = \sum_{s \in S'} z_s \).

A GSA \( T \subseteq A \) is represented by variables \( x \in \{0, 1\}^V \) and \( y \in \{0, 1\}^A \) such that \( x_v \) (resp. \( y_a \)) is equal to 1 iff \( v \in V[T] \) (resp. \( a \in T \)).

Since a GSA of \( D \) and \( \pi \) induces a spanning arborescence of \( D^\pi \), the arc-incident vector \( y \in \{0, 1\}^A \) of a GSA with root in \( V_0 \) satisfies the following, adapted from MSA (Schrijver, 2003):

\[
\begin{align*}
\mathcal{Y} &= \{y \in \{0, 1\}^A | y \text{ satisfies (1)-(3)}\}. \\
\text{GMSA can be formulated with the following integer linear program:} & \\
\max_{x,y} & \phi \cdot y \\
\text{s.t.} & \\
y & \in \mathcal{Y} \\
x_v & \geq y_a \quad \forall v \in V, a \in \delta(v), \\
x_v(V_k) & = 1 \quad \forall 0 \leq k \leq n, \\
x_v & \in \{0, 1\} \quad \forall v \in V.
\end{align*}
\]

\(^4\)By an abuse of notation, we identify any singleton \( \{v\} \) with its element \( v \).

\(^5\)The converse does not hold: an arc subset of \( A \) corresponding to a spanning arborescence of \( D^\pi \) may not be a GSA of \( D \) and \( \pi \) since it may not induce a connected graph.
Let \( W \) and \( T \) be the vertex and arc sets given by \( x_v = 1 \) and \( y_a = 1 \) respectively. Since \( T \) is a spanning arborescence of \( D^\pi \) by \((5)\), \( V[T], T \) is an acyclic directed graph with \( n \) arcs such that \( V_0 \) has no entering arc and \( V_i, i \in \{1, \ldots, n\}, \) has one entering arc. By constraints \((7)\), \( W \) contains one vertex per cluster of \( \pi \). Moreover, by inequalities \((6)\), \( V[T] \subseteq W \). Since \( |W| = n + 1 \) and \( |T| = n \), \( W = V[T] \) and \((V[T], T) \) is connected, so it is a GSA. Because its root is in \( V_0 \) by \((5)\), it is an optimal solution for GMSA by \((4)\).

### 3.3 Reduction from joint parsing to GMSA

Given an instance of the joint parsing problem, we construct an instance of GMSA as follows. With every spine \( s \) of every word \( w_k \) different from \( w_0 \), we associate a vertex \( v \). For \( k = 1, \ldots, n \), we denote by \( V_k \) the set of vertices associated with the spines of \( w_k \). We associate with \( w_0 \) a set \( V_0 \) containing only one vertex and \( V_0 \) will now refer both the cluster and the vertex it contains depending on the context. Let \( \pi = \{V_0, \ldots, V_n\} \) and \( V = \bigcup_{k=0}^n V_k \). For every couple \( u, v \) of vertices such that \( u \in V_h \) and \( v \in V_m \), \( h \neq m \) and \( m \neq 0 \), we associate an arc \( uv \) corresponding to the best adjunction of the root of spine \( s_m \) associated with \( v \) of \( V_m \) to spine \( s_h \) associated with vertex \( u \) of \( V_h \). The weight of this arc is given by

\[
\phi_{uv} = \alpha(h, m; w) + v(s_m; h, m, w) + \max_{l, m} \gamma(l, m; h, m, s_h, w)
\]

which is the score of the best adjunction of \( s_m \) to \( s_h \). This ends the construction of \( (D, \pi, \phi) \).

There is a 1-to-1 correspondence between the solutions to GMSA and those to the joint supertagging and spine parsing task in which each adjunction is performed with the label maximizing the score of the adjunction. Indeed, the vertices covered by a GSA \( T \) with root \( V_0 \) correspond to the spines on which the derivation is performed. By definition of GSAs, one spine per word is chosen. Each arc of \( T \) corresponds to an adjunction. The score of the arborescence is the sum of the scores of the selected spines plus the sum of the scores of the best adjunctions with respect to \( T \). Hence, one can solve GMSA to perform joint parsing.

As an illustration, the GSA depicted in Figure 2 represents the derivation tree of Figure 1: the vertices of \( V \setminus V_0 \) covered by the GSA are those associated with the spines of Figure 1 and the arcs represent the different adjunctions. For instance the arc from \( V_3 \) to \( V_2 \) represents the adjunction of spine NP–PRP to spine S–VP–VB at index 0.

### 4 Efficient Decoding

Lagrangian relaxation has been successfully applied to various NLP tasks (Koo et al., 2010; Le Roux et al., 2013; Almeida and Martins, 2013; Das et al., 2012; Corro et al., 2016). Intuitively, given an integer linear program, it consists in relaxing some linear constraints which make the program difficult to solve and penalizing their violation in the objective function.

We propose a new decoding method for GMSA based on dual decomposition, a special flavor of Lagrangian relaxation where the problem is decomposed in several independent subproblems.

#### 4.1 Dual decomposition

To perform the dual decomposition, we first reformulate the integer linear program \((4)-(8)\) before relaxing linear constraints. For this purpose, we replace the variables \( y \) by three copies \( \{y^i\} = \{y^0, y^1, y^2\}, y^i \in \{0, 1\}^A \). We also consider variables \( z \in \mathbb{R}^A \). Let \( \phi^0, \phi^1 \) and \( \phi^2 \) be arc weight vectors such that \( \sum_i \phi^i = \phi \). GMSA can then be reformulated as:

\[
\max_{x, (y^i), z} \sum_i \phi_i \cdot y_i \quad (9)
\]

s.t.

\[
y^0 \in \mathcal{Y} \quad (10)
\]

\[
x_v \geq y_0^a \quad \forall v \in V, a \in \delta^- (v), \quad (11)
\]

\[
x_v \geq y_2^a \quad \forall v \in V, a \in \delta^+ (v), \quad (12)
\]

\[
x_v (V_k) = 1 \quad \forall 0 \leq k \leq n, \quad (13)
\]

\[
x_v \in \{0, 1\} \quad \forall v \in V, \quad (14)
\]

\[
z = y^i \quad \forall i. \quad (15)
\]

---

*In our implementation, we choose \( \phi^0 = \phi^1 = \phi^2 = \frac{1}{3} \phi \).*

---
Note that variables \( z \) only appear in equations (15). Their goal is to ensure equality between copies \( y^0, y^1 \) and \( y^2 \). Variables \( z \) are usually called witness variables (Komodakis et al., 2007). Equality between \( y^0, y^1 \) and \( y^2 \) implies that (10)-(12) are equivalent to (5) and (6).

We now relax constraints (15) and build the dual objective (Lemaréchal, 2001) \( L^*(\{\lambda^i\}) \):

\[
\max_{x,\{y^i\},z} \sum_i \phi^i \cdot y^i + \sum_{i \in \{0,1,2\}} \lambda^i \cdot (z - y^i) \\
\text{s.t.} \quad (10) - (14)
\]

where \( \{\lambda^i\} = \{\lambda^0, \lambda^1, \lambda^2\}, \lambda^i \in \mathbb{R}^A \) for \( i = 0, 1, 2 \), is the set of Lagrangian multipliers. The dual problem is then:

\[
\min_{\{\lambda^i\}} L^*(\{\lambda^i\})
\]

Note that, as there is no constraint on \( z \), if \( \sum_i \lambda^i \neq 0 \) then \( L^*(\{\lambda^i\}) = +\infty \). Therefore, we can restrict the domain of \( \{\lambda^i\} \) in the dual problem to the set \( \Lambda = \{\{\lambda^i\} | \sum_i \lambda^i = 0\} \). This implies that \( z \) may be removed in the dual objective. This latter can be rewritten as:

\[
L^*(\{\lambda^i\}) = \max_{x,\{y^i\}} \sum_i \phi^i \cdot y^i \\
\text{s.t.} \quad (10) - (14)
\]

where \( \tilde{\phi}^i = \phi^i - \lambda^i \) for \( i = 0, 1, 2 \).

### 4.2 Computing the dual objective

Given \( \{\lambda^i\} \in \Lambda \), computing the dual objective \( L^*(\{\lambda^i\}) \) can be done by solving the two following distinct subproblems:

\[
P_1(\phi^0) = \max_{y^0} \phi^0 \cdot y^0 \\
\text{s.t.} \quad y^0 \in \mathcal{Y} \\
P_2(\phi^1, \phi^2) = \max_{x,y^1,y^2} \phi^1 \cdot y^1 + \phi^2 \cdot y^2 \\
\text{s.t.} \quad (11) - (14) \\
\quad y^a \in \{0,1\} \quad \forall a \in A, i = 1, 2.
\]

Subproblem \( P_1 \) can be solved by simply running the MSA algorithm on the contracted graph \( D^\pi \).

Subproblem \( P_2 \) can be solved in a combinatorial way. Indeed, observe that each value of \( y^1 \) and \( y^2 \) is only constrained by a single value of \( x \). The problem amounts to selecting for each cluster a vertex as well as all the arcs with positive weight covering it. More precisely, for each vertex \( v \in \mathcal{V} \), compute the local weight \( c_v \) defined by:

\[
\sum_{a \in \delta^-(v)} \max\{0, \phi^1\} + \sum_{a \in \delta^+(v)} \max\{0, \phi^2\}.
\]

Let \( V^{\max} \) be the set of vertices defined as follows. For \( k = 0, \ldots, n \), add in \( V^{\max} \) the vertex \( v \in V_k \) with the maximum weight \( c_v \). Let \( A^1 \) and \( A^2 \) be the sets of arcs such that \( A^1 \) (resp. \( A^2 \)) contains all the arcs with positive weights entering (resp. leaving) a vertex of \( V^{\max} \). The vectors \( x, y^1 \) and \( y^2 \) corresponding respectively to the incidence vectors of \( V^{\max} \), \( A^1 \) and \( A^2 \) form an optimal solution to \( P_2 \).

Hence, both subproblems can be be solved with a \( O(|\pi|^2) \) time complexity, that is quadratic w.r.t. the length of the input sentence.\(^7\)

### 4.3 Decoding algorithm

Our algorithm seeks for a solution to GMSA by solving the dual problem since its solution is optimal to GMSA whenever it is a GSA. If not, a solution is constructed by returning the highest GSA on the spines computed during the resolution of the dual problem.

We solve the dual problem using a projected subgradient descent which consists in iteratively updating \( \{\lambda^i\} \) in order to reduce the distance to the optimal assignment. Let \( \{\lambda^{t+1}\} \) denote the value of \( \{\lambda^i\} \) at iteration \( t \). \( \{\lambda^{0}\} \) is initially set to 0. At each iteration, the value of \( \{\lambda^{t+1}\} \) is computed from the value of \( \{\lambda^t\} \) thanks to a subgradient of the dual objective. More precisely, we have

\[
\{\lambda^{t+1}\} = \{\lambda^{t}\} - \eta^t \times \nabla L^*(\{\lambda^t\})
\]

where \( \nabla L^*(\{\lambda^t\}) \) is a subgradient of \( L^*(\{\lambda^t\}) \) and \( \eta^t \in \mathbb{R} \) is the stepsize at iteration \( t \). We use the projected subgradient from Komodakis et al. (2007). Hence, at iteration \( t \), we must solve reparameterized subproblems \( P_1 \) and \( P_2 \) to obtain the current solution \( (\bar{x}^t, \bar{y}^0,t, \bar{y}^1,t, \bar{y}^2,t) \) of the dual objective. Then each multiplier is updated following

\[
\lambda^{t+1} = \lambda^t - \eta^t \times \left( \bar{y}^t - \sum_{j=0}^{3} \bar{y}^{j,t} \right) / 3.
\]

Note that for any value of \( \{\lambda^i\} \), \( L^*(\{\lambda^i\}) \) gives an upper bound for GMSA. So, whenever the

\(^7\)In the general case, the time complexity is \( O(|V|^2) \). But in our problem, the number of vertices per cluster is bounded by the grammar size: \( O(|V|^2) = O(|S\pi|^2) = O(|\pi|^2) \).
optimal solution \( \bar{x}^t, \{\bar{y}^{k,t}\} \) to the dual objective \( \mathcal{L}^\ast(\{\lambda^{k,t}\}) \) at iteration \( t \) is a primal feasible solution, that is \( \bar{y}^{k,t} = \bar{y}^{1,t} = \bar{y}^{2,t} \), it is an optimal solution to GMSA and the algorithm ends. Otherwise, we construct a pipeline solution by performing a MSA on the vertices given by \( \bar{x}^t \).

If after a fixed number of iterations we have not found an optimal solution to GMSA, we return the pipeline solution with maximum weight.

### 4.4 Lagrangian enhancement

The previously defined Lagrangian dual is valid but may lead to slow convergence. Thus, we propose three additional techniques which empirically improve the decoding time and the convergence rate: constraint tightening, arc reweighing and problem reduction.

**Constraint tightening:** In subproblem \( P_2 \), we consider a vertex and all of its adjacent arcs of positive weight. However, we know that our optimal solution must satisfy tree-shape constraints (5). Thus, every cluster except the root must have exactly one incoming arc and there is at most one arc between two clusters. Both constraints are added to \( P_2 \) without hurting its time complexity.

**Reweighing:** By modifying weights such that less incoming arcs have a positive weight, the solution of \( P_2 \) tends to be an arborescence. For each cluster \( V_k \in \pi \setminus V_0 \), let \( A_k \) be the set of incoming arcs with the highest weight \( \phi_k \). Then, let \( \gamma_k \) be a value such that \( \phi_a - \gamma_k \) is positive only for arcs in \( A_k \). Subtracting \( \gamma_k \) from the weight \( \phi_a \) of each arc of \( \delta^-(V_k) \) and adding \( \gamma_k \) to the objective score does not modify the weight of the solution because only one entering arc per cluster is selected.

**Problem reduction:** We use the pipeline solutions computed at each iteration to set the value of some variables. Let \( \bar{x}, \{\bar{y}^j\} \) be the optimal solution of \( \mathcal{L}^\ast(\{\lambda^j\}) \) computed at any iteration of the subgradient algorithm. For \( k = 1, \ldots, n \), let \( \bar{v} \) be the vertex of \( V_k \) such that \( \bar{x}_\bar{v} = 1 \). Using the local weights (Section 4.2), for all \( v \in V_k \setminus \{\bar{v}\}, \mathcal{L}^\ast(\{\lambda^j\}) + c_v - c_\bar{v} \) is an upper bound on the weight of any solution \( (x, y) \) to GMSA with \( x_v = 1 \). Hence, if it is lower than the weight of the best pipeline solution found so far, we can guarantee that \( x_v = 0 \) in any optimal solution. We can check the whole graph in linear time if we keep local weights \( c \) in memory.

### 5 Neural Parameterization

We present a probabilistic model for our framework. We implement our probability distributions with neural networks, more specifically we build a neural architecture on top of bidirectional recurrent networks that compute context sensitive representations of words. At each step, the recurrent architecture is given as input a concatenation of word and part-of-speech embeddings. We refer the reader to (Kiperwasser and Goldberg, 2016; Dozat and Manning) for further explanations about bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). In the rest of this section, \( b_m \) denotes the context sensitive representation of word \( w_m \).

We now describe the neural network models used to learn and assign weight functions \( \alpha, \nu \) and \( \gamma \) under a probabilistic model. Given a sentence \( w \) of length \( n \), we assume a derivation \( (d, s, l) \) is generated by three distinct tasks. By chain rule, \( P(d, s, l|w) = P_a(d|w) \times P_v(s|d, w) \times P_\gamma(l|d, s, w) \). We follow a common approach in dependency parsing and assign labels \( l \) in a post-processing step, although our model is able to incorporate label scores directly. Thus, we are left with jointly decoding a dependency structure and assigning a sequence of spines. We note \( s_i \) the \( i \)th spine:\[P_a(d|w) \times P_v(s|d, w) = \prod_{(h,m) \in d} P_a(h|m, w) \times P_v(s_m|m, d, w) = \prod_{(h,m) \in d} P_a(h|m, w) \times P_v(s_m|m, h, w)\]

We suppose that adjunctions are generated by an arc-factored model, and that a spine prediction depends on both current position and head position. Then parsing amounts to finding the most probable derivation and can be realized in the log space, which gives following weight functions:

\[\alpha(h, m; w) = \log P_a(h|m, w)\]
\[\nu(s_m; h, m, w) = \log P_v(s_m|m, h, w)\]

where \( \alpha \) represents the arc contribution and \( \nu \) the spine contribution (cf. Section 2).

Word embeddings \( b_k \) are first passed through specific feed-forward networks depending on the

---

8We assume that the spine for the root \( w_0 \) is unique.
distribution and role. The result of the feed-forward transformation parameterized by set of parameters \( p \) of a word embedding \( b_s \) is a vector denoted \( b_s^{(p)} \). We first define a biaffine attention networks weighting dependency relations (Dozat and Manning):

\[
o^{(a)}_{h,m} = b^{(α)}_m W^{(α)} b^{(α^2)} + V^{(α)} b^{(α^2)}_h
\]

where \( W^{(α)} \) and \( V^{(α)} \) are trainable parameters. Moreover, we define a biaffine attention classifier networks for class \( c \) as:

\[
o^{(c)}_{s,h,m} = b^{(β)}_m W^{(β)} b^{(β^2)} + V^{(β)} b^{(β^2)}_h
\]

where \( β \) is the concatenation. \( W^{(β)}_c, V^{(β)}_c \) and \( u^{(β)} \) are trainable parameters. Then, we define the weight of assigning spine \( s \) to word at position \( m \) with head \( h \) as \( o^{(β)}_{s,h,m} \).

Distributions \( P_a \) and \( P_v \) are parameterized by these biaffine attention networks followed by a softmax layer:

\[
P_a(h|m,w) = \frac{\exp o^{(a)}_{h,m}}{\sum_{h'} \exp o^{(a)}_{h',m}}
\]

\[
P_v(s|h,m,w) = \frac{\exp o^{(v)}_{s,h,m}}{\sum_{s'} \exp o^{(v)}_{s',h,m}}
\]

Now we move on to the post-processing step predicting arc labels. For each adjunction of spine \( s \) at position \( m \) to spine \( t \) at position \( h \), instead of predicting a site index \( i \), we predict the non-terminal \( nt \) at \( t[i] \) with a biaffine attention classifier.\(^9\) The probability of the adjunction of spine \( s \) at position \( m \) to a site labeled with \( nt \) on spine \( t \) at position \( h \) with type \( a \) in \{regular, sister\} is:

\[
P_γ(nt,a|h,m,w) = P_γ(nt|h,m,w) \times P_v(a|h,m,w)
\]

\( P_γ \) and \( P_v \) are again defined as distributions from the exponential family using biaffine attention classifiers:

\[
P_γ(nt|h,m,t) = \frac{\exp o^{(γ)}_{nt,h,m}}{\sum_{nt'} \exp o^{(γ)}_{nt',h,m}}
\]

\[
P_v(a|h,m,t) = \frac{\exp o^{(γ)}_{a,h,m}}{\sum_{a'} \exp o^{(γ)}_{a',h,m}}
\]

\( \gamma \) is a hyperparameter controlling the strength of the attention.

\( \gamma \) is fixed following the rule of Polyak (1987).

We use embeddings of size 100 for words and size 50 for parts-of-speech tags. We stack two bidirectional LSTMs with a hidden layer of size 300, resulting in a context sensitive embedding of size 600. Embeddings are shared across distributions. All feed-forward networks have a unique elu-activated hidden layer of size 100 (Clevert et al., 2016). We regularize parameters with a dropout ratio of 0.5 on LSTM input. We estimate parameters by maximizing the likelihood of the training data through stochastic subgradient descent using Adam (Kingma and Ba, 2015). Our implementation uses the DyNet library (Neubig et al., 2017) with default parameters.

6 Experiments

We ran a series of experiments on two corpora annotated with discontinuous constituents.

English We used an updated version of the Wall Street Journal part of the Penn Treebank corpus (Marcus et al., 1994) which introduces discontinuity (Evang and Kallmeyer, 2011). Sections 2-21 are used for training, 22 for development and 23 for testing. We used gold and predicted POS tags by the Stanford tagger,\(^10\) trained with 10-jackknifing. Dependencies are extracted following the head-percolation table of Collins (1997).

German We used the Tiger corpus (Brants et al., 2004) with the split defined for the SPMRL 2014 shared task (Maier, 2015; Seddah et al., 2013). Following Maier (2015) and Coavoux and Crabbé (2017), we removed sentences number 46234 and 50224 as they contain annotation errors. We only used the given gold POS tags. Dependencies are extracted following the head-percolation table distributed with Tulipa (Kallmeyer et al., 2008).

We emphasize that long sentences are not filtered out. Our derivation extraction algorithm is similar to the one proposed in Carreras et al. (2008). Regarding decoding, we use a beam of size 10 for spines w.r.t. \( P_v(s_m|m,w) = \sum_h P_v(s_m|h,m,w) \times P_a(h|m,w) \) but allow every possible adjunction. The maximum number of iterations of the subgradient descent is set to 500 and the stepsize \( η \) is fixed following the rule of Polyak (1987).

Parsing results and timing on short sentences only (≤ 40 words) and full test set using the de-

\(^9\)If a spine contains repeated non-terminal sequences, we select the lowest match.

\(^10\)http://nlp.stanford.edu/software/tagger.shtml
fault discodop\textsuperscript{11} eval script are reported on Table 1 and Table 2.\textsuperscript{12} We report labeled recall (LR), precision (LP), F-measure (LF) and time measured in minutes. We also report results published by van Cranenburgh et al. (2016) for the discontinuous PTB and Coavoux and Crabb\textsuperscript{e} (2017) for Tiger. Moreover, dependency unlabeled attachment scores (UAS) and tagging accuracies (Spine acc.) are given on Table 3. We achieve significantly better results on the discontinuous PTB, while being roughly 36 times faster together with a low memory footprint.\textsuperscript{13} On the Tiger corpus, we achieve on par results. Note however that Coavoux and Crabb\textsuperscript{e} (2017) rely on a greedy parser combined with beam search.

Fast and efficient parsing of discontinuous constituent is a challenging task. Our method can quickly parse the whole test set, without any parallelization or GPU, obtaining an optimality certificate for more than 90% of the sentences in less than 500 iterations of the subgradient descent. When using a non exact decoding algorithm, such as a greedy transition based method, we may not be able to deduce the best opportunity for improving scores on benchmarks, such as the parameterization method or the decoding algorithm. Here the behavior may be easier to interpret and directions for future improvement easier to see. We stress that our method is able to produce an optimality certificate on more than 99% of the test examples thanks to the enhancement presented in Section 4.4.

## 7 Related Work

Spine-based parsing has been investigated in (Shen and Joshi, 2005) for Lexicalized TAGs with a left-to-right shift-reduce parser which was subsequently extended to a bidirectional version in (Shen and Joshi, 2008). A graph-based algorithm was proposed in (Carreras et al., 2008) for second-order projective dependencies, and for a form of non-projectivity occurring in machine translation (i.e. projective parses of permuted input sentences) in (Carreras and Collins, 2009).

Discontinuous phrase-structure parsing through dependencies in contexts other that TAGs have been explored in (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015). The first two encode spine information as arc labels while the third one relaxes spine information by keeping only the root and height of the adjunction, thus avoiding combinatorial explosion. Labeling is performed as a post-processing step in these approaches, since the number of labels can be very high. Our model also performs labeling after structure construction, but it could be performed jointly without major issue. This is one way our model could be improved.

GMSA has been studied mostly as a way to solve the non directed version (i.e. with symmetric arc weights) (Myung et al., 1995), see (Pop, 2009; Feremans et al., 1999) for surveys on resolution methods. Myung et al. (1995) proposed an exact decoding algorithm through branch-and-bound using a dual ascent algorithm to compute bounds. Pop (2002) also used Lagrangian relaxation – in the non directed case – where a single subproblem is solved in polynomial time. However, the relaxed constraints are inequalities: if the dual objective returns a valid primal solution, it is not a sufficient condition in order to guarantee that

\[
\begin{array}{cccc}
\text{LR} & \text{LP} & \text{LF} & \text{Time} \\
\hline
\text{Short sentences only} \\
\text{This work} & 90.63 & 91.01 & 90.82 & \approx 4 \\
\text{This work} & 89.57 & 90.13 & 89.85 & \approx 4 \\
\text{VC2016} & 87.00 & 87.09 & 87.09 & \approx 180 \\
\hline
\text{Full test set} \\
\text{This work} & 89.89 & 90.29 & 90.09 & \approx 6.5 \\
\text{This work} & 88.90 & 89.45 & 89.17 & \approx 5.5 \\
\end{array}
\]

Table 1: Parsing results and processing time on the english discontinuous PTB corpus. Results marked with † use predicted part-of-speech tags. VC2016 indicates results of van Cranenburgh et al. (2016).

\[
\begin{array}{cccc}
\text{LR} & \text{LP} & \text{LF} & \text{Time} \\
\hline
\text{Short sentences only} \\
\text{This work} & 82.69 & 84.68 & 83.67 & \approx 7.5 \\
\text{C2017} & 81.60 & 81.63 & 81.63 & \approx 11 \\
\hline
\text{Full test set} \\
\text{This work} & 80.66 & 82.63 & 81.63 & \approx 11 \\
\text{C2017} & 81.60 & 81.60 & 81.60 & \approx 2.5 \\
\end{array}
\]

Table 2: Parsing results and processing time on the german Tiger corpus. C2017 indicates results of Coavoux and Crabb\textsuperscript{e} (2017).
Table 3: Dependency parsing and tagging results. Results marked with † use predicted part-of-speech tags.

<table>
<thead>
<tr>
<th>Language</th>
<th>UAS</th>
<th>Spine acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>93.70</td>
<td>97.32</td>
</tr>
<tr>
<td>English†</td>
<td>93.04</td>
<td>96.81</td>
</tr>
<tr>
<td>German</td>
<td>92.25</td>
<td>96.49</td>
</tr>
</tbody>
</table>

it is the optimal solution (Beasley, 1993), and thus the stopping criterion for the subgradient descent is usually slow to obtain. To our knowledge, our system is the first time that GMSA is used to solve a NLP problem.

Dual decomposition has been used to derive efficient practical resolution methods in NLP, mostly for machine translation and parsing, see (Rush et al., 2010) for an overview and (Koo et al., 2010) for an application to dependency parsing.

To accelerate the resolution, our method relies heavily on problem reduction (Beasley, 1993), which uses the primal/dual bounds to filter out suboptimal assignments. Exact pruning based on duality has already been studied in parsing, with branch and bound (Corro et al., 2016) or column generation (Riedel et al., 2012) and in machine translation with beam search (Rush et al., 2013).

8 Conclusion

We presented a novel framework for the joint task of supertagging and parsing by a reduction to GMSA. Within this framework we developed a model able to produce discontinuous constituents. The scoring model can be decomposed into tagging and dependency parsing and thus may rely on advances in those active fields.

This work could benefit from several extensions. Bigram scores on spines could be added at the expense of a third subproblem in the dual objective. High-order scores on arcs like grandparent or siblings can be handled in subproblem $P_2$ with the algorithms described in (Koo et al., 2010). In this work, the parameters are learned as separate models. Joint learning in the max-margin framework (Komodakis, 2011; Komodakis et al., 2015) may model interactions between vertex and arc weights better and lead to improved accuracy. Finally, we restricted our grammar to spinal trees but it could be possible to allow full lexicalized TAG-like trees, with substitution nodes and even obligatory adjunction sites. Derivations compatible with the TAG formalism (or more generally LCFRS) could be recovered by the use of a constrained version of MSA (Corro et al., 2016).

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We thank Laura Kallmeyer and Kilian Evang for providing us with the script for the discontinuous PTB. First author is supported by a public grant overseen by the French National Research Agency (ANR) as part of the Investissements d’Avenir program (ANR-10-LABX-0083). Second author, supported by a public grant overseen by the French ANR (ANR-16-CE33-0021), completed this work during a CNRS research leave at LIMSI, CNRS / Université Paris Saclay.

References


