Parsing to 1-Endpoint-Crossing, Pagenumber-2 Graphs

Junjie Cao*, **Sheng Huang***, **Weiwei Sun** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{junjie.cao,huangsheng,ws,wanxiaojun}@pku.edu.cn

Abstract

We study the Maximum Subgraph problem in deep dependency parsing. We consider two restrictions to deep dependency graphs: (a) 1-endpoint-crossing and (b) pagenumber-2. Our main contribution is an exact algorithm that obtains maximum subgraphs satisfying both restrictions simultaneously in time $O(n^5)$. Moreover, ignoring one linguistically-rare structure descreases the complexity to $O(n^4)$. We also extend our quartic-time algorithm into a practical parser with a discriminative disambiguation model and evaluate its performance on four linguistic data sets used in semantic dependency parsing.

1 Introduction

Dependency parsing has long been studied as a central issue in developing syntactic or semantic analysis. Recently, some linguistic projects grounded on deep grammar formalisms, including CCG, LFG, and HPSG, draw attentions to rich syntactic and semantic dependency annotations that are not limited to trees (Hockenmaier and Steedman, 2007; Sun et al., 2014; Ivanova et al., 2012). Parsing for these *deep* dependency representations can be viewed as the search for Maximum Subgraphs (Kuhlmann and Jonsson, 2015). This is a natural extension of the Maximum Spanning Tree (MST) perspective (McDonald et al., 2005) for dependency tree parisng.

One main challenge of the Maximum Subgraph perspective is to design tracTable algorithms for certain graph classes that have good empirical coverage for linguistic annotations. Unfortunately, no previously defined class simultaneously has high coverage and low-degree polynomial parsing algorithms. For example, noncrossing dependency graphs can be found in time $O(n^3)$, but cover only 48.23% of sentences in CCGBank (Kuhlmann and Jonsson, 2015).

We study two well-motivated restrictions to deep dependency graphs: (a) 1-endpoint-crossing (1EC hereafter; Pitler et al., 2013) and (b) pagenumber is less than or equal to 2 (P2 hereafter; Kuhlmann and Jonsson, 2015). We will show that if the output dependency graphs are restricted to satisfy both restrictions, the Maximum Subgraph problem can be solved using dynamic programming in time $O(n^5)$. Moreover, if we ignore one linguistically-rare sub-problem, we can reduce the time complexity to $O(n^4)$. Though this new algorithm is a degenerated one, it has the same empirical coverage for various deep dependency annotations. We evaluate the coverage of our algorithms on four linguistic data sets: CCGBank, DeepBank, Enju HPSGBank and Prague Dependency TreeBank. They cover 95.68%, 97.67%, 97.28% and 97.53% of dependency graphs in the four corpora. The relatively satisfactory coverage makes it possible to parse with high accuracy.

Based on the quartic-time algorithm, we implement a parser with a discriminative disambiguation model. Our new parser can be taken as a graph-based parser which is complementary to transition-based (Henderson et al., 2013; Zhang et al., 2016) and factorization-based (Martins and Almeida, 2014; Du et al., 2015a) systems. We evaluate our parser on four data sets: those used in SemEval 2014 Task 8 (Oepen et al., 2014), and the dependency graphs extracted from CCGbank (Hockenmaier and Steedman, 2007). Evaluations indicate that our parser produces very accurate deep dependency analysis. It reaches state-of-the-art results on average produced by a transition-based system of Zhang et al.

The first two authors contribute equally.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2110–2120
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1193

(2016) and factorization-based systems (Martins and Almeida, 2014; Du et al., 2015a).

The implementation of our parser is available at http://www.icst.pku.edu.cn/lcwm/grass.

2 Background

Dependency parsing is the task of mapping a natural language sentence into a dependency graph. Previous work on dependency parsing mainly focused on tree-shaped representations. Recently, it is shown that data-driven parsing techniques are also applicable to generate more flexible deep dependency graphs (Du et al., 2014; Martins and Almeida, 2014; Du et al., 2015b,a; Zhang et al., 2016; Sun et al., 2017). Parsing for deep dependency representations can be viewed as the search for Maximum Subgraphs for a certain graph class $\mathcal{G}$ (Kuhlmann and Jonsson, 2015), a generalization of the MST perspective for tree parsing. In particular, we have the following optimization problem:

Given an arc-weighted graph $G = (V, A)$, find a subgraph $G' = (V, A' \subseteq A)$ with maximum total weight such that G' belongs to $\mathcal{G}$.

The choice of $\mathcal{G}$ determines the computational complexity of dependency parsing. For example, if $\mathcal{G}$ is the set of projective trees, the problem can be solved in time $O(|V|^3)$, and if $\mathcal{G}$ is the set of noncrossing dependency graphs, the complexity is $O(|V|^3)$. Unfortunately, no previously defined class simultaneously has high coverage on deep dependency annotations and low-degree polynomial decoding algorithms for practical parsing. In this paper, we study well-motivated restrictions: 1EC (Pitler et al., 2013) and P2 (Kuhlmann and Jonsson, 2015). We will show that relatively satisfactory coverage and parsing complexity can be obtained for graphs that satisfy both restrictions.

3 The 1EC, P2 Graphs

3.1 The 1EC Restriction

Pitler et al. (2013) introduced a very nice property for modelling non-projective dependency trees, i.e. 1EC. This property not only covers a large amount of tree annotations in natural language treebanks, but also allows the corresponding MST problem to bo solved in time of $O(n^4)$. The formal description of the 1EC property is adopted from (Pitler et al., 2013).

Definition 1. *Edges e_1 and e_2 cross if e_1 and e_2 have distinct endpoints and exactly one of the endpoints of e_1 lies between the endpoints of e_2.*

Definition 2. *A dependency graph is 1-Endpoint-Crossing if for any edge e, all edges that cross e share an endpoint p.*

Given a sentence $s = w_0 w_1 \cdots w_{n-1}$ of length n, the vertices, i.e. words, are indexed with integers, an arc from w_i to w_j as $a_{(i,j)}$, and the common endpoint, namely pencil point, of all edges crossed with $a_{(i,j)}$ or $a_{(j,i)}$ as $pt(i, j)$. We denote an edge as $e_{(i,j)}$, if we do not consider its direction.

3.2 The P2 Restriction

The term *pagenumber* is referred to as *planar* by some other authors, e.g. (Titov et al., 2009; Gómez-Rodríguez and Nivre, 2010; Pitler et al., 2013). We give the definition of related concepts as follows.

Definition 3. *A book is a particular kind of topological space that consists of a single line called the spine, together with a collection of one or more half-planes, called the pages, each having the spine as its boundary.*

Definition 4. *A book embedding of a finite graph G onto a book B satisfies three conditions: (1) every vertex of G is drawn as a point on the spine of B; (2) every edge of G is drawn as a curve that lies within a single page of B; (3) every page of B does not have any edge crossings.*

Empirically, a deep dependency graph is not very dense and can typically be embedded onto a very *thin* book. To measure the thickness of a graph, we can use its *pagenumber*.

Definition 5. *The book pagenumber of G is the minimum number of pages required for a book embedding of G.*

For sake of concision, we say a graph is "pagenumber-k", meaning that the pagenumber is at most k.

Theorem 1. *The pagenumber of* 1EC *graph may be greater than 2.*

Proof. The graph in Figure 1 gives an instance which is 1EC but the pagenumber of which is 3. There is a cycle, namely $a \to c \to e \to b \to d \to a$, consisting of odd number of edges. □

Pitler et al. (2013) proved that 1EC trees are a subclass of graphs whose pagenumber is at most 2. This property provides the foundation to the

PN≤ 2	1EC	EnjuBank	DeepBank	PCEDT	CCGBank
Yes	Both	32236 (99.53%)	32287 (99.69%)	31866 (98.39%)	38848 (98.09%)
Both	Yes	31507 (97.28%)	31634 (97.67%)	31589 (97.53%)	37913 (95.73%)
Yes	Yes	31507 (97.28%)	31634 (97.67%)	31589 (97.53%)	37894 (95.68%)
No	Yes	0 (0.0%)	0 (0.0%)	0 (0.0%)	19 (0.05%)
Yes	No	729 (2.25%)	653 (2.02%)	277 (0.86%)	954 (2.41%)
Sentences		32389	32389	32389	39604

Table 1: Coverage in terms of complete graphs under various structural restrictions. Column "PN≤ 2" indicates whether the restriction "P2" is satisfied; Column "1EC" indicates whether the restriction "1EC" is satisfied.

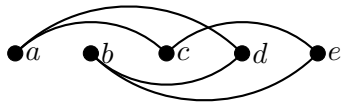


Figure 1: A 1EC graph whose pagenumber is 3.

success in designing dynamic programming algorithms for trees. Theorem 1 indicates that when we consider more general graph, the case is more complicated. In this paper, we study graphs that are constrained to be both 1EC and P2. We call them 1EC/P2 graphs.

3.3 Coverage on Linguistic Data

To show that the two restrictions above are well-motivated for describing linguistic data, we evaluate their empirical coverage on four deep dependency corpora (as defined in Section 5.2). These corpora are also used for training and evaluating our data-driven parsers. The coverage is evaluated using sentences in the training sets.

Table 1 shows the results. We can see that 1EC is also an empirical well-motivated restriction when it comes to deep dependency structures. The P2 property has an even better coverage. Unfortunately, it is a NP-hard problem to find optimal P2 graphs (Kuhlmann and Jonsson, 2015). Though theoretically a 1EC graph is not necessarily P2, the empirical evaluation demonstrates the high overlap of them on linguistic annotations. In particular, almost all 1EC deep dependency graphs are P2. The percentages of graphs satisfying both restrictions vary between 95.68% for CCGBank and 97.67% for DeepBank. The relatively satisfactory coverage enables accurate practical parsing.

4 The Algorithm

This section contains the main contribution of this paper: a polynomial time exact algorithm for solving the Maximum Subgraph problem for the class of 1EC/P2 graphs.

Theorem 2. *Take* 1EC/P2 *graphs as target subgraphs, the maximum subgraph problem can be solved in time* $O(|V|^5)$.

For sake of formal concision, we introduce the algorithm of which the goal is to calculate the maximum score of a subgraph. Extracting corresponding optimal graphs can be done in a number of ways. For example, we can maintain an auxiliary arc table which is populated parallel to the procedure of obtaining maximum scores.

Our algorithm is highly related to the following property: Every subgraph of a 1EC/P2 graph is also a 1EC/P2 graph. We therefore focus on maximal 1EC/P2 graphs, a particular type of 1EC/P2 graphs defined as follows.

Definition 6. *A maximal* 1EC/P2 *graph is a* 1EC/P2 *graph that cannot be extended by including one more edge.*

Our algorithm is a bottom-up dynamic programming algorithm. It defines different structures corresponding to different sub-problems, and visits all structures from bottom to top, finding the best combination of smaller structures to form a new structure. The key design is to make sure that it can produce all maximal 1EC/P2 graphs. During the search for maximal 1EC/P2 graphs, we can freely delete bad edges whose scores are negative. In particular, we figure out some edges, in each construction step, which can be created without violating either 1EC or P2 restriction. Assume the arc weight associated with $a_{(i,j)}$ is $w[i,j]$. Then we define a function SELECT(i,j) according to the comparison of 0 and $w[i,j]$ as well as $w[j,i]$. If $w[i,j] \geq 0$ (or $w[j,i] \geq 0$), we then *select* $a_{(i,j)}$ (or $a_{(j,i)}$) and add it to currently the best solution of a sub-problem. SELECT(i,j) returns $\max(\max(0, w[i,j]) + \max(0, w[j,i]))$. If we allow at most one arc between two nodes, SELECT(i,j) returns $\max(0, w[i,j], w[j,i])$.

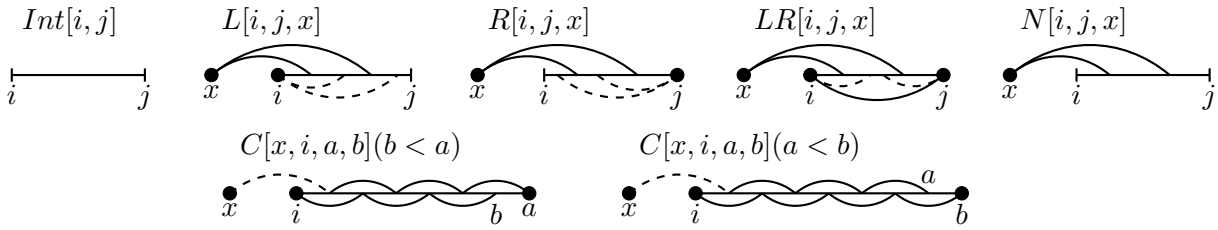


Figure 2: Graphic representations of sub-problems.

The graphical illustration of our algorithm uses undirected graphs[1]. In other words, we use $e_{(i,j)}$ to include the discussion about both $a_{(i,j)}$ and $a_{(j,i)}$.

4.1 Sub-problems

We consider six sub-problems when we construct a maximum dependency graph on a given (closed) interval $[i, k] \subseteq V$ of vertices. When we focus on the nodes strictly inside this interval, and we use an open interval (i, k) to exclude i and j. See Figure 2 for graphical visualization. The first five are adapted in concord with Pitler et al. (2013)'s solution for trees, and we introduce a new sub-problem, namely **C**. Because graphs allow for loops as well as disconnectedness, the sub-problems are simplified to some extent, while a special case of **LR** is now prominent. **C** is thus introduced to represent the special case. The sub-problems are explained as follows.

Int $Int[i, j]$ represents a partial analysis associated with an interval from i to j inclusively. $Int[i, j]$ may or may not contain edge $e_{(i,j)}$. To parse a given sentence is equivalent to solve the problem $Int[0, n-1]$.

L $L[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = i$. $L[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

R $R[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = j$. $R[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

LR $LR[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = i$ or j. $LR[i, j, x]$ must allow $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

N $N[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively and an external vertex x. $\forall p \in (i, j), pt(x, p) \notin [i, j]$. $N[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

C $C[x, i, a, b](a \neq b, a > i, b > i)$ represents a partial analysis associated with an interval from i to $max\{a, b\}$ inclusively and an external vertex x. Intuitively, **C** depicts a class of graphs constructed by upper- and lower-plane edges arranged in a staggered pattern. a stands for the last endpoint in the upper plane, and b the last endpoint in the lower plane.

We give a definition of **C**. There exists in $C[x, i, a, b]$ a series $\{s_1, \cdots, s_m\}$ that fulfills the following constraints:

1. $s_1 = i < s_2 < ... < s_m = \max\{a, b\}$.
2. $\exists e_{(x,s_2)}$.
3. $\forall k \in [1, m-2], \exists e_{(s_k, s_{k+2})}$.
4. $\forall k \in [1, m-2], \nexists e_{(l,r)} (s_k, s_{k+2}) \subset (l, r) \subset (s_1, s_m)$[2].
5. $\forall k \in [2, m-3]$, $e_{(s_k, s_{k+2})}$ crosses only with $e_{(s_{k-1}, s_{k+1})}$ and $e_{(s_{k+1}, s_{k+3})}$; $e_{(s_1, s_3)}$ crosses only with $e_{(s_2, s_4)}$ and $e_{(x,s_2)}$; $e_{(s_{m-2}, s_m)}$ crosses only with $e_{(s_{m-3}, s_{m-1})}$.
6. $e_{(x,s_{m-1})}, e_{(s_1,s_m)}, e_{(x,s_1)}, e_{(x,s_m)}$ are disallowed.
7. While $a < b$, the series can be written as $\{s_1 = i, \cdots, s_{m-1} = a, s_m = b\} (m \geq 5)$. While $b < a$, the series is $\{s_1, \cdots, s_{m-1} =$

[1] The *single-head* property does not hold. We currently do not consider other constraints of directions. So prediction of the direction of one edge does not affect prediction of other edges as well as their directions. The directions can be assigned *locally*, and our parser builds directed rather than undirected graphs in this way. Undirected graphs are only used to conveniently illustrate our algorithms. All experimental results in Section 5.2 consider directed dependencies in a standard way. We use the official evaluation tool provided by SDP2014 shared task. The numberic results reported in this paper are directly comparable to results in other papers.

[2] By "$(x, y) \subset (z, w)$," we mean $x \geq z, y < w$ or $x > z, y \leq w$.

$b, s_m = a\}(m \geq 4)$. We denote the two cases using the signs **C1** and **C2** respectively.

The distinction between **C1** and **C2** is whether there is one more edge below than above.

4.2 Decomposing an Int Sub-problem

Consider an $Int[i,j]$ sub-problem. Assume that $k(k \in (i,j))$ is the farthest vertex that is linked with i, and $l = pt(i,k)$. When $j - i > 1$, there must be such a k given that we consider maximal 1EC/P2 graphs. There are three cases.

Case 1: $l = j$**.** Vertex k divides the interval $[i,j]$ into two parts: $[i,k]$ and $[k,j]$. First notice that the edges linking (i,k) and j can only cross with $e_{(i,k)}$. Thus i or k can be the pencil points of those edges, which entails that interval $[i,k]$ is an **LR** in respect to external vertex j. Because there exist no edge from i to any node in (k,j), interval $[k,j]$ is an **Int**. The problem is eventually decomposed to: $LR[i,k,j] + Int[k,j] + \text{SELECT}[i,j]$.

Case 2: $l \in (k,j)$**.** In this case, we can freely add $e_{(i,l)}$ without violating either 1EC or P2 conditions. Therefore Case 2 does not lead to any maximal 1EC/P2 graph. Our algorithm does not need to explicitly handle this case, given that they can be derived from solutions to other cases.

Case 3: $l \in (i,k)$**.** Now assume that there is an edge from i to a vertex in (l,k). Consider the farthest vertex that is linked with l, say $p(p \in (k,j)$. We can freely add $e_{(i,p)}$ without violating the 1EC and P2 restrictions. Similar to Case 2, we do not explicitly deal with this case.

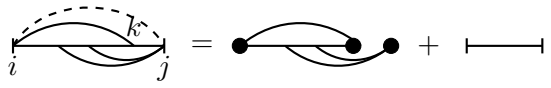
If there is no edge from i to any vertex in (l,k), then $[i,l]$, $[l,k]$, $[k,j]$ are **R**, **Int**, **L** respectively. Three external edges are $e_{(i,k)}$, $e_{(l,j)}$, and $e_{(i,j)}$. The decomposition is: $R[i,l,k]+Int[l,k]+L[k,j,l] + \text{SELECT}[l,j] + \text{SELECT}[i,j]$.

4.3 Decomposing an L Sub-problem

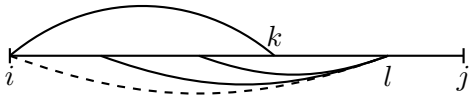
If there is no edge from x to any node in (i,j), the graph is reduced to $Int[i,j]$. If there is one, let k be the vertex farthest from i and adjacent to x. There are two different cases, as shown in Figure 4.

1. If there exists an edge from x to some node in (i,k), intervals $[i,k]$, $[k,j]$ are classified as **L**, **N** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $L[i,k,x]+N[k,j,i]+\text{SELECT}[x,k]+\text{SELECT}[i,j]$.

Case 1: $l = j$



Case 2: $l \in (k,j)$



Case 3: $l \in (i,k)$

Does such a dashed edge exist?

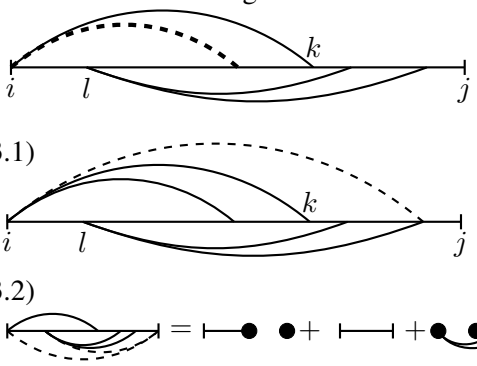


Figure 3: Decomposition for $Int[i,j]$, with $pt(i,k) = l$.

Does such a dashed edge exist?

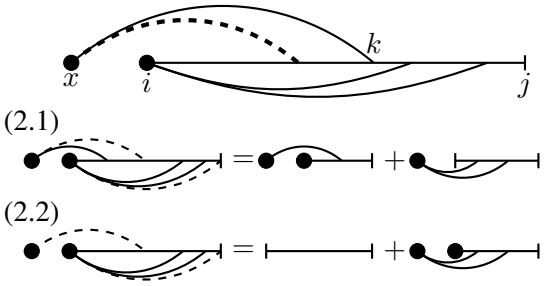


Figure 4: Decomposition for $L[i,j,x]$.

2. Otherwise, Intervals $[i,k]$, $[k,j]$ are classified as **Int**, **L** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $Int[i,k] + L[k,j,i] + \text{SELECT}[x,k] + \text{SELECT}[i,j]$.

4.4 Decomposing an R Sub-problem

If there is no edge from x to (i,j), then the graph is reduced to $Int[i,j]$. If there is one, let k be the farthest vertex from j and adjacent to x. There are two different cases:

1. If there exist an edge from x to (k,j), Intervals $[i,k]$, $[k,j]$ are classified as **N**, **R** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition

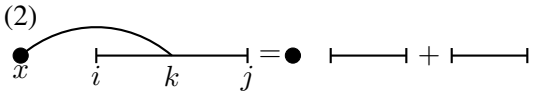


Figure 5: Decomposition for $N[i, j, x]$.

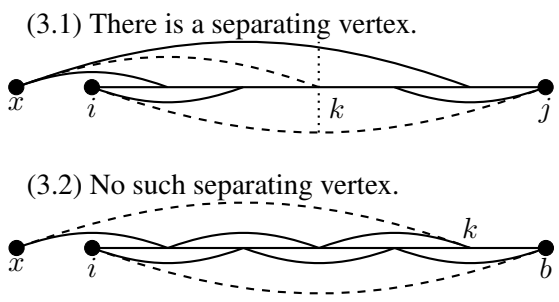


Figure 6: Decomposition for $LR[i, j, x]$.

is $N[i, k, j] + R[k, j, x] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

2. Otherwise, Intervals $[i, k]$, $[k, j]$ are classified as **R**, **Int** respectively. Two edges external to the interval are $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $R[i, k, j] + Int[k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

The decomposition is similar to **L**, we thus do not give a graphical representation to save space.

4.5 Decomposing an N Sub-problem

If there is no edge from x to (i, j), then the graph is reduced to $Int[i, j]$. If there is one, let k be the farthest vertex from i and adjacent to x. By definition, $N[i, j, x]$ does not allow for $e_{(x,i)}$ or $e_{(x,j)}$. Thus $k \neq i$ or j. Intervals $[i, k]$, $[k, j]$ are classified as **N**, **Int** respectively. Two edges external to the interval are $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $N[i, k, x] + Int[k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

4.6 Decomposing an LR Sub-problem

If the pencil point of all edges from x to (i, j) is i, then the model is the same as $L[i, j, x]$. Similary, if the pencil point is j, then the model is the same as $R[i, j, x]$.

If some of the edges from x to (i, j) share a pencil point i, and the others share j, there are two different cases.

1. If there is a k which satisfies that within $[i, j]$, only $e_{(i,j)}$ crosses over k (i.e., $[i, j]$ can be divided along dashed line k into two), then, k divides $[i, j]$ into $[i, k]$ and $[k, j]$. Because k is not allowed to be pencil point, the two subintervals must be an **L** and an **R** in terms of external x, respectively. In addition, there are two edges, namely $e_{(x,k)}$ and $e_{(i,j)}$ not included by the subintervals. The problem is thus decomposed as $L[i, k, x] + R[k, j, x] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

2. If there is no such k in concord with the condition in (1), it comes a much more difficult case for which we introduce sub-problem **C**. Here we put forward the conclusion:

Lemma 1. *Assume that $k(k \in (i, j))$ is the vertex that is adjacent to x and farthest from i. The decomposition for the second case is $C[x, i, k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.*

Proof. The distinction between Case 1 and 2 implies the following property, which is essential, $\forall t \in (i, j), \exists e_{(pl,pr)}$ such that $t \in (pl, pr) \subset [i, j]$.

We can recursively generate a series of length n—$\{e_{(sl_k,sr_k)}\}$—in $LR[i, j, x]$ as follows.

$k = 1$ Let $sl_k = i, sr_k = \max\{p | p \in (i + 1, j) \text{ and } \exists e_{(i,p)}\}$;

$k > 1$ For sr_{k-1}, we denote all edges that cover it as $e_{(pl_1,pr_1)}, \cdots, e_{(pl_s,pr_s)}$. Note that there is at least one such edge. For any two edges in them, viz $e_{(pl_u,pr_u)}$ and $e_{(pl_v,pr_v)}$, $(pl_u, pr_u) \subset (pl_v, pr_v)$ or $(pl_v, pr_v) \subset (pl_u, pr_u)$. Otherwise, the P2 property no longer holds due to the interaction among $e_{(sl_{k-1},sr_{k-1})}$, $e_{(pl_u,pr_u)}$ and $e_{(pl_v,pr_v)}$. Assume (pl_w, pr_w) is the largest one, then we let $sl_k = pl_w, sr_k = pr_w$. When $sr_k = j$, recursion ends.

We are going to prove that if we delete two edges $e_{(x,sr_{n-1})}$ and $e_{(i,j)}$ from $LR[i, j, x]$, the series $\{sl_1, sl_2, sl_3, ..., sl_{n-2}, sl_{n-1}, sl_n, sr_{n-1}, sr_n\}$ satisfies each and all the conditions of **C1**.

Condition 1. Because $e_{(sl_n,sr_n)}$ covers sr_{n-1}, Condition 1 holds for $k = m-3, m-2$. Consider $k \leq m-4 = n-2$. Assume that $s_{k+1} < s_k$, then we have $e_{(s_{k+1},sr_{k+1})}$ is larger than $e_{(s_k,sr_{k+1})}$. This is impossible because we select the largest edge in every step.

Condition 2. The **LR** sub-problem we discussed now cannot be reduced to **L** nor **R**, so there must be two edges from x that respectively cross edges linked to i and j. We are going to prove that

the two edges must be $e_{(x,s_2)}$ and $e_{(x,sr_{n-1})}$. Assume that there is $e_{(x,p)}$, where $p \in (i,j)$, $p \neq s_2$ and $p \neq sr_{n-1}$. If $p \in (i,s_2)$, then $e_{(s_1,s_3)}$ crosses with $e_{(x,p)}$ and $e_{(s_2,s_4)}$ simultaneously. 1EC is violated. If $p \in (s_2, sr_{n-1})$, $e_{(x,p)}$ necessarily crosses with some edge $e_{(s_k,s_{k+2})}$. Furthermore, $i < s_k < s_{k+2} < j$. Thus 1EC is violated. If $p \in (sr_{n-1}, j)$, the situation is similar to $p \in (i, s_2)$.

Condition 3. $\forall k \in [1, n-2]$, $e_{(sl_k,sr_k)}$ and $e_{(sl_{k+1},sr_{k+1})}$ cross, $e_{(sl_{k+1},sr_{k+1})}$ and $e_{(sl_{k+2},sr_{k+2})}$ cross, so $sr_k \leq sl_{k+2}$. Otherwise the interaction of the three edges results in the violation of P2. If $sr_k < sl_{k+2}$, $e_{(sl_k,sr_k)}$ and $e_{(sl_{k+2},sr_{k+2})}$ share no common endpoint, violating 1EC. Therefore, $sr_k = sl_{k+2} = s_{k+2}$, and Condition 3 is satisfied.

We also reach proposition that $pt(s_k, s_{k+2}) = s_{k+1}$.

Condition 4. This condition is easy to verify because (s_k, s_{k+2}) is the largest with respective to sr_k.

Condition 5. Assume, that there is $e_{(pl,pr)}$ which intersects with $e_{(s_k,s_{k+2})}$, and at the same time satisfy the conditions: $e_{(pl,pr)} \notin \{e_{(s_t,s_{t+2})} | t \in [1, m-2]\} \cup \{e_{(x,s_2)}, e_{(x,sr_{n-1})}\}$. Since $pt(s_k, s_{k+2}) = s_{k+1}$, $pl = s_{k+1}$ or $pr = s_{k+1}$.

If $pl = s_{k+1}$, then $pl < sl_{k+2} < pr$, and in turn $k < m-2$. In addition, according to Condition 4, $(pl, pr) \subset (s_{k+1}, s_{k+3})$. So $pr < s_{k+3}$. If $k = m-3$ then $e_{(x,s_{n-1})}$ crosses with $e_{(pl,pr)}$ and $e_{(i,j)}$ simultaneously. 1EC is violated. If $k < m-3$ then $e_{(s_{k+2},s_{k+4})}$ cross with $e_{(pl,pr)}$, and $pr < s_{k+3} = pt(e_{(s_{k+2},s_{k+4})})$. Again 1EC is violated. If $pr = s_{k+1}$ The symmetry of our proof entails the violation of 1ec.

All in all, the assumption does not hold and thus satisfies Condition 5.

Condition 6. $e_{(x,s_1)}, e_{(x,s_m)}$ are disallowed due to definition of an **LR** problem. $e_{(x,s_{m-1})}, e_{(s_1,s_m)}$ are disallowed due to the decomposition.

Condition 7. Due to the existence of $e_{(x,s_2)}$ and $e_{(x,sr_{n-1})}$, there must be two edges: $e_{(x,p_1)}$ and $e_{(x,p_2)}$ that cross $e_{(i,s_2)}$ and $e_{(sr_{n-1},j)}$ respectively. There must be an odd number of edges in the series $\{e_{(sl_k,sr_k)}\}$, otherwise P2 is violated as the case shown in Figure 1. In summary, the last condition is satisfied and we have a **C1** structure in this **LR** sub-problem.

□

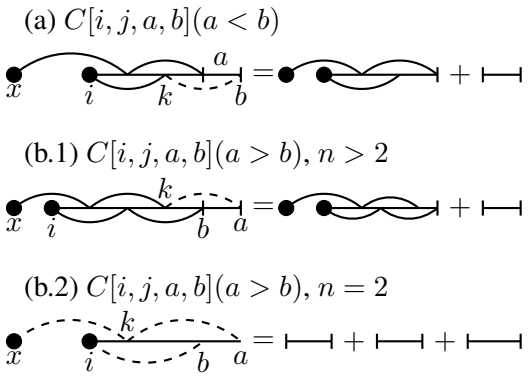


Figure 7: Decomposition for $C[x,i,a,b]$.

4.7 Decomposing a C Sub-problem

We illustrate the decomposition using the graphical representations shown in Figure 7. When $a < b$, since a is the *upper-plane* endpoint farthest to the right, and b is the *lower-plane* counterpart, in this case a precedes b (i.e., a is to the left of b).

Let $C[x,i,a,k]$ be a **C** in which the lower-plane endpoint k precedes a. Add $e_{(k,b)}$ gives a new **C** sub-problem with lower-plane endpoint preceded by the upper-plane one. The decomposition is then $C[x,i,a,k] + Int[a,b] + \text{SELECT}[k,b]$.

When $a > b$ and $n > 2$, the lower-plane endpoint b precedes a. In analogy, the case can be obtained by adding $e_{(k,a)}$ to $C[x,i,k,b]$. The decomposition: $C[x,i,k,b] + Int[b,a] + \text{SELECT}[k,a]$.

When $n = 2$, we reach the most fundamental case. Only 4 vertices are in the series, namely i,k,b,a. Moreover, there are three edges: $e_{(x,k)}$, $e_{(i,b)}$, $e_{(k,a)}$, and the interval [i,a] is divided by k,b into three parts. The decomposition is $Int[i,k] + Int[k,b] + Int[b,a] + \text{SELECT}[x,k] + \text{SELECT}[i,b] + \text{SELECT}[k,a]$.

4.8 Discussion

4.8.1 Soundness and Completeness

The algorithm is sound and complete with respective to 1EC/P2 graphs. We present our algorithms by detailing the decomposition rules. The completeness is obvious because we can decompose any 1EC/P2 graph from an **Int**, use our rules to reduce it into smaller sub-problems, and repeat this procedure. The decomposition rules are also construction rules. During constructing graphs by applying these rules, we never violate 1EC nor P2

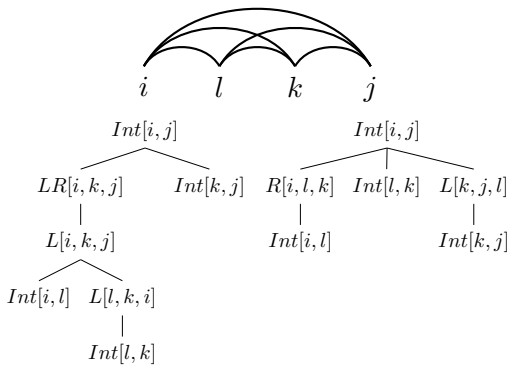


Figure 8: A maximal 1EC/P2 graph and its two derivations. For brevity, we elide the edges created in each derivation step.

restrictions. So our algorithm is sound.

4.8.2 Greedy Search during Construction

There is an important difference between our algorithm and Eisner-style MST algorithms (Eisner, 1996b; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010) for trees as well as Kuhlmann and Jonsson's Maximum Subgraph algorithm for noncrossing graphs. In each construction step, our algorithm allows multiple arcs to be constructed, but whether or not such arcs are added to the target graph depends on their arc-weights. In each step, we do greedy search and decide if adding an related arc according to local scores. If all arcs are assigned scores that are greater than 0, the output of our algorithm includes the most complicated 1EC/P2 graphs. That means adding one more arc voilates the 1EC or P2 restrictions. For all other aforementioned algorithms, in a single construction step, it is clear whether to add a new arc, and which one. There is no local search.

4.8.3 Spurious Ambiguity

To generate the same graph, even a maximal 1EC/P2 graph, we may have different derivations. Figure 8 is an example. This is similar to syntactic analysis licensed by Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000). To derive one surface string, there usually exists multiple CCG derivations. A practice of CCG parsing is defining one particular derivation as the *standard* one, namely normal form (Eisner, 1996a). The spurious ambiguity in our algorithm does not affect the correctness of first-order parsing, because scores are assigned to individual dependencies, rather than derivation processes. There is no need to distinguish one special derivation here.

4.8.4 Complexity

The sub-problem **Int** is of size $O(n^2)$, each graph of which takes a calculating time of order $O(n^2)$. For sub-problems **L**, **R**, **LR**, and **N**, each has $O(n^3)$ elements, with a unit calculating time $O(n)$. **C** has $O(n^4)$ elements, with a unit calculating time $O(n)$. Therefore the full version algorithm runs in time of $O(n^5)$ with a space requirement of $O(n^4)$.

4.9 A Degenerated Version

We find that graphical structures involved in the **C** sub-problem, namely coupled staggered pattern, is extremely rare in linguistic analysis. If we ignore this special case, we get a degenerated version of dynamic programming algorithm. This algorithm can find a strict subset of 1EC/P2 graphs. We can improve efficiency without sacrificing *expressiveness* in terms of linguistic data. This degenerated version algorithm requires $O(n^4)$ time and $O(n^3)$ space.

5 Practical Parsing

5.1 Disambiguation

We extend our quartic-time parsing algorithm into a practical parser. In the context of data-driven parsing, this requires an extra disambiguation model. As with many other parsers, we employ a global linear model. Following Zhang et al. (2016)'s experience, we define rich features extracted from word, POS-tags and pseudo trees. For details we refer to the source code. To estimate parameters, we utilize the averaged perceptron algorithm (Collins, 2002).

5.2 Data

We conduct experiments on unlabeled parsing using four corpora: CCGBank (Hockenmaier and Steedman, 2007), DeepBank (Flickinger et al., 2012), Enju HPSGBank (EnjuBank; Miyao et al., 2004) and Prague Dependency TreeBank (PCEDT; Hajic et al., 2012), We use "standard" training, validation, and test splits to facilitate comparisons. Following previous experimental setup for CCG parsing, we use section 02-21 as training data, section 00 as the development data, and section 23 for testing. The other three data sets are from SemEval 2014 Task 8 (Oepen et al.,

	DeepBank			EnjuBank			CCGBank			PCEDT		
	UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
P1	90.75	86.13	88.38	93.38	90.20	91.76	94.21	88.55	91.29	90.61	85.69	88.08
1ECP2^d	91.05	87.22	89.09	93.41	91.83	92.61	94.41	91.41	92.89	90.76	86.31	88.48

Table 2: Parsing accuracy evaluated on the development sets.

	DeepBank			EnjuBank			CCGBank			PCEDT		
	UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
Ours	90.91	86.98	88.90	93.83	91.49	92.64	94.23	91.13	92.66	90.09	85.90	87.95
ZDSW	89.04	88.85	88.95	92.92	92.83	92.87	92.49	92.30	92.40	- -	- -	- -
MA	90.14	88.65	89.39	93.18	91.12	92.14	- -	- -	- -	90.21	85.51	87.80
DSW	- -	- -	- -	- -	- -	- -	93.03	92.03	92.53	- -	- -	- -

Table 3: Parsing accuracy evaluated on the test sets.

2014), and the data splitting policy follows the shared task. All the four data sets are publicly available from LDC (Oepen et al., 2016).

Experiments for CCG-grounded analysis were performed using automatically assigned POS-tags that are generated by a symbol-refined HMM tagger (Huang et al., 2010). Experiments for the other three data sets used POS-tags provided by the shared task. We also use features extracted from pseudo trees. We utilize the Mate parser (Bohnet, 2010) to generate pseudo trees. The pre-processing for CCGBank, DeepBank and EnjuBank are exactly the same as in experiments reported in (Zhang et al., 2016).

5.3 Accuracy

We evaluate two parsing algorithms, the algorithm for noncrossing dependency graphs (Kuhlmann and Jonsson, 2015), i.e. pagenumber-1 (denoted as *P1*) graphs, and our quartic-time algorithm (denoted as *1ECP2*^{d}). Table 2 summerizes the accuracy obtained our parser. Same feature templates are applied for disambiguation. We can see that our new algorithm yields significant improvements on all data sets, as expected. Especially, due to the improved coverage, the recall is improved more.

5.4 Comparison with Other Parsers

Our new parser can be taken as a graph-based parser which employ a different architecture from transition-based and factorization-based (Martins and Almeida, 2014; Du et al., 2015a) systems. We compare our parser with the best reported systems in the other two architectures. *ZDSW* (Zhang et al., 2016) is transition-based parser while *MA* (Martins and Almeida, 2014) and DSW (Du et al., 2015a) are two factorization-based systems. All of them achieves state-of-the-art performance. All results on the test set is shown in Table 3. We can see that our parser, as a graph-based parser, is comparable to state-of-the-art transition-based and factorization-based parsers.

6 Conclusion and Future Work

In this paper, we explore the strength of the graph-based approach. In particular, we enhance the Maximum Subgraph model with new parsing algorithms for 1EC/P2 graphs. Our work indicates the importance of finding appropriate graph classes that on the one hand are linguistically expressive and on the other hand allow efficient search. Within tree-structured dependency parsing, higher-order factorization that conditions on wider syntactic contexts than arc-factored relationships have been proved very useful. The arc-factored model proposed in this paper may be enhanced with higher-order features too. We leave this for future investigation.

Acknowledgments

This work was supported by 863 Program of China (2015AA015403), NSFC (61331011), and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

We thank the first anonymous reviewer whose valuable comments led to significant revisions. We thank Xingfeng Shi for his help in explicating the idea.

Weiwei Sun is the corresponding author.

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 89–97. http://www.aclweb.org/anthology/C10-1011.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *In Proc. EMNLP-CoNLL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015a. A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1545–1555. http://www.aclweb.org/anthology/P15-1149.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 459–464. http://www.aclweb.org/anthology/S14-2080.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015b. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 927–931. http://www.aclweb.org/anthology/S15-2154.

Jason Eisner. 1996a. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*. Santa Cruz, pages 79–86.

Jason M. Eisner. 1996b. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 340–345.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1492–1501. http://www.aclweb.org/anthology/P10-1151.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jirí Semecký, Jana Sindlerová, Jan Stepánek, Josef Toman, Zdenka Uresová, and Zdenek Zabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics* 39(4):949–998.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 12–22. http://www.aclweb.org/anthology/D10-1002.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1–11. http://www.aclweb.org/anthology/P10-1001.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 471–476. http://www.aclweb.org/anthology/S14-2082.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*. volume 6, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 523–530.

Yusuke Miyao, Takashi Ninomiya, and Jun ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *IJCNLP*. pages 684–693.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Semantic Dependency Parsing (SDP) graph banks release 1.0 LDC2016T10. Web Download.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *TACL* 1:13–24. http://www.transacl.org/wp-content/uploads/2013/03/paper13.pdf.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monographs. Mit Press. http://books.google.ca/books?id=Mh1vQgAACAAJ.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Weiwei Sun, Junjie Cao, and Xiaojun Wan. 2017. Semantic dependency parsing via book embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding, and Xiaojun Wan. 2014. Grammatical relations in Chinese: GB-ground extraction and data-driven parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 446–456. http://www.aclweb.org/anthology/P14-1042.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 1562–1567. http://dl.acm.org/citation.cfm?id=1661445.1661696.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.