Greed is Good if Randomized: New Inference for Dependency Parsing

Yuan Zhang
CSAIL, MIT

Joint work with Tao Lei, Regina Barzilay, and Tommi Jaakkola
Inference vs. Scoring

Inference

Approximate

Exact

Scoring Function

Limited

Expressive
Inference vs. Scoring

- Inference
  - Exact
  - Approximate

- Scoring
  - Limited
  - Expressive

Minimum Spanning Tree
Inference vs. Scoring

- **Inference**: Approximate
- **Scoring**: Limited

- **Exact**: Minimum Spanning Tree
- **Expressive**: Reranking

- Reranking: incorporate arbitrary features
Inference vs. Scoring

- **Reranking**: incorporate arbitrary features
- **Dual Decomposition**: search in full space
Parsing Complexity

• High-order parsing is NP-hard (McDonald et al., 2006)
• Hypothesis: parsing is easy on average
• Many NP-hard problems are easy on average
  – MAX-SAT (Resende et al., 1997)
  – Set cover (Hochbaum, 1982)
Parsing Complexity

- High-order parsing is NP-hard (McDonald et al., 2006)
- Hypothesis: parsing is easy on average
- Many NP-hard problems are easy on average
  - MAX-SAT (Resende et al., 1997)
  - Set cover (Hochbaum, 1982)

We show
- Analysis on average parsing complexity
- A simple inference algorithm based on the analysis
Our Approach

- **Reranking**: incorporate arbitrary features
- **Dual Decomposition**: search in full space
Core Idea

• Climb to the optimal tree in a few small greedy steps

**Randomized Hill-climbing**

For $k = 1$ to $K$

1) Randomly sample a dependency tree
2) Greedily improve the tree one edge at a time
3) Repeat (2) until converge

Select the tree with the highest score
Core Idea

- Climb to the optimal tree in a few small greedy steps

**Randomized Hill-climbing**

For $k = 1$ to $K$

1) Randomly sample a dependency tree
2) Greedily improve the tree one edge at a time
3) Repeat (2) until converge

Select the tree with the highest score

That’s it!
It Works!

Parsing Performance on CoNLL Dataset

- Dual Decomposition: 88.73%
- Our Full: 89.44%
Example

“$I$ ate an apple today$”
I ate an apple today

"I ate an apple today"
 Example

**Initial tree**

```
ROOT
   apple
   
   ate
   
   I
   
   an
   
   today
```

**Target tree**

```
ROOT
   ate
   
   I
   
   apple
   
   an
   
   today
```

"I ate an apple today"
Example

```
I ate an apple today
```

Initial tree

```
ROOT
  ↓
  apple
    ↓
    ate
      ↓
      I
      ↓
      an
    ↓
    today
```

Target tree

```
ROOT
  ↓
  ate
    ↓
    I
      ↓
      an
    ↓
    apple
    ↓
    today
```
Example

“I ate an apple today”
Example

"I ate an apple today"

Target tree
Example

“"I ate an apple today"”

Target tree
Example

"I ate an apple today"

Target tree
Example

“I ate an apple today”

Target tree
Example

```
I ate an apple today
```

Target tree
Example

“I ate an apple today”
Example

"I ate an apple today"

Target tree
Example

“I ate an apple today”

Target tree
Example

"I ate an apple today"

Target tree
Why Greedy Has a Chance to Work

Reachability: transforming any tree to any other tree

- maintaining the structure a valid tree at any point
- using as few as $d$ steps ($d$: head differences/hamming distance)
Greedy Hill-climbing

\[ y^{(0)} \xrightarrow{\text{increase}} S(x, y^{(t)}) \xrightarrow{\text{}} y^{(T)} \]
Greedy Hill-climbing

\[ y^{(0)} \xrightarrow{\text{increase}} S(x, y^{(t)}) \xrightarrow{\text{increase}} y^{(T)} \]

Arbitrary features in the scoring function
Challenge: Local Optimum

\[ y^{(0)} \xrightarrow{\text{increase}} S(x, y^{(t)}) \xrightarrow{\text{---}} y^{(T)} \]

score \( S \)

global optimum

local optimum
Hill-climbing with Restarts

Overcome local optima via **restarts**
Hill-climbing with Restarts

Random initialization (e.g. uniform)

Overcome local optima via **restarts**
Learning Algorithm

• Follow common max-margin framework

\[ \forall y \in T(x) \quad S(x, \hat{y}) \geq S(x, y) + |\hat{y} - y| - \xi \]

- \( \hat{y} \) is the gold tree
Learning Algorithm

• Follow common max-margin framework

\[ \forall y \in T(x) \quad S(x, \hat{y}) \geq S(x, y) + |\hat{y} - y| - \xi \]

\[ \hat{y} \] is the gold tree

• Adopt passive-aggressive online learning framework (Crammer et al. 2006)

• Decode with our randomized greedy algorithm
Analysis
Analysis

First-order

<table>
<thead>
<tr>
<th>Theoretical</th>
<th>Empirical</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Analysis

<table>
<thead>
<tr>
<th></th>
<th>Theoretical</th>
<th>Empirical</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-order</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>High-order</td>
<td>?</td>
<td>☑</td>
</tr>
</tbody>
</table>
## Analysis

<table>
<thead>
<tr>
<th></th>
<th>Theoretical</th>
<th>Empirical</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-order</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High-order</td>
<td>?</td>
<td>✓</td>
</tr>
</tbody>
</table>
Search Space Complexity: First-order

10 words
Search Space Complexity: First-order

10 words

\[ \approx 2 \text{ billion trees} \]
Search Space Complexity: First-order

\[ \approx 2 \text{ billion trees} \]

10 words

\[ < 512 \text{ local optima} \]
Search Space Complexity: First-order

**Theorem**: For any first-order scoring function:

- there are at most $2^{n-1}$ locally optimal trees
- this upper bound is **tight**
Theorem: For any first-order scoring function:

- there are at most $2^{n-1}$ locally optimal trees
- this upper bound is **tight**

$2^{n-1}$ is still a lot, but it is the worst case.
Search Space Complexity: First-order

**Theorem**: For any first-order scoring function:

- there are at most $2^{n-1}$ locally optimal trees
- this upper bound is **tight**

$2^{n-1}$ is still a lot, but it is the worst case

*What about the average case?*
Algorithm for Counting Local Optima
Algorithm for Counting Local Optima

The method is based on Chu-Liu-Edmonds algorithm

- Select the best heads independently
Algorithm for Counting Local Optima

- Contract the cycle and recursively count the local optima
  - Any local optimum exactly reassigns one edge in the cycle
Empirical Results: First-order

How many local optima in real data?
Empirical Results: First-order

How many local optima in real data?

# Optima on English Dataset

% sentences 50%

21
Empirical Results: First-order

How many local optima in real data?

# Optima on English Dataset

% sentences  | 50%  | 70%  
---------|------|------
21       | 121  

Empirical Results: First-order

How many **local optima in real data**?

![Graph showing the number of optimas on an English dataset vs. percentage of sentences.](image-url)
Empirical Results: First-order

Does the hill-climbing find the argmax?

Finding Global Optimum on English
Empirical Results: First-order

Does the hill-climbing find the argmax?

Finding Global Optimum on English

<table>
<thead>
<tr>
<th>Len. ≤ 15</th>
<th>Len. &gt; 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>99.3%</td>
</tr>
</tbody>
</table>

Easy search space leads to successful decoding
Empirical Results: High-order

Does the hill-climbing find the argmax?

Comparison on English Given DD Cert.

<table>
<thead>
<tr>
<th></th>
<th>% Certificate</th>
<th>$S_{DD} = S_{HC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual decomposition</td>
<td>94.5%</td>
<td>99.8%</td>
</tr>
<tr>
<td>(Koo et al., 2010)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Given a certificate by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English
Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English

<table>
<thead>
<tr>
<th></th>
<th>$S_{DD}$</th>
<th>$S_{HC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>98.7%</td>
</tr>
</tbody>
</table>
Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English

\[ S_{DD} = S_{HC} \]

- 98.7%

\[ S_{DD} < S_{HC} \]

- 1.0%

\[ S_{DD} > S_{HC} \]

- 0.3%
Experimental Setup

Datasets
- 14 languages in CoNLL 2006 & 2008 shared tasks

Features
- Up to 3rd-order (three arcs) features used in MST/Turbo parsers
- Global features used in re-ranking

Implementation
- Adaptive restarting strategy with $K = 300$
Baselines and Evaluation Measure

Baseline:

- Turbo Parser: Dual Decomposition with 3rd-order features (Martins et al., 2013)
- Sampling-based Parser: MCMC sampling with global features (Zhang et al., 2014)

Evaluation Measure:

- Unlabeled Attachment Score (UAS), without punctuations
Comparing with Baselines

- Turbo (DD) - 88.73%
- Our 3rd (Full) - 88.66%
Comparing with Baselines

- **Turbo (DD)**: 88.73%
- **Our 3rd**: 88.66%
- **Sampling-based (MCMC)**: 89.23%
- **Our Full**: 89.24%
Comparing with Baselines

- **Turbo (DD)**: 88.73%
- **Our 3rd**: 88.66%
- **Sampling-based (MCMC)**: 89.23%
- **Our Full**: 89.24%
Impact of Initialization

Uniform: 88.0%
Rnd-1st: 88.1%
Impact of Restarts

No Restart: 85.4%
300 Restarts: 88.1%
Convergence Property

Convergence Analysis on English

- Score normalized by the highest score in 3000 restarts

- Score convergence analysis for English text, showing convergence within 500 restarts.
Trade-off between Speed and Performance

Decoding Speed on English

- 3rd-order Model
- Full Model

Fast → Slow
Conclusion

• **Analysis:** we investigate average case complexity of parsing

• **Algorithm:** we introduce a simple randomized greedy inference algorithm

Source code available at:  
https://github.com/taolei87/RBGParser
Thank You!