On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing

Alexander M. Rush, David Sontag,
Michael Collins, and Tommi Jaakkola

Dynamic Programming

Dynamic programming is a dominant technique in NLP.

- Fast
- Exact
- Easy to implement

Examples:

- Viterbi algorithm for hidden Markov models
- CKY algorithm for weighted context-free grammars

$$y^* = \arg\max_y f(y) \leftarrow \text{Decoding}$$

Model Complexity

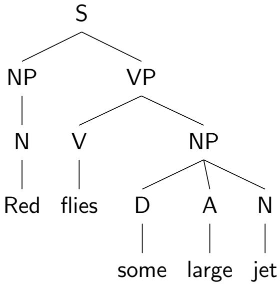
Unfortunately, dynamic programming algorithms do not scale well with model complexity.

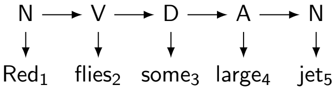
As our models become complex, these algorithms can explode in terms of computational or implementational complexity.

Integration:

- $f \leftarrow$ Easy
- $g \leftarrow$ Easy
- $f + g \leftarrow$ Hard

Integration (1)

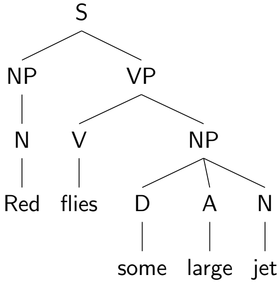


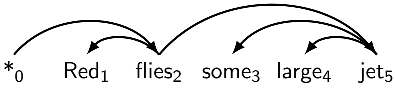


$$f(y) \quad + \quad g(z)$$

- Classical problem in NLP.
- The dynamic programming intersection is prohibitively slow and complicated to implement.

Integration (2)





$$f(y) \qquad + \qquad g(z)$$

- Important for improving parsing accuracy.
- The dynamic programming intersection is slow and complicated to implement.

Dual Decomposition

A general technique for constructing decoding algorithms

Solve complicated models

$$y^* = \arg\max_y f(y)$$

by decomposing into smaller problems.

Upshot: Can utilize a toolbox of combinatorial algorithms.

- Dynamic programming
- Minimum spanning tree
- Shortest path
- Min-Cut
- ...

Dual Decomposition Algorithms

Simple - Uses basic dynamic programming algorithms

Efficient - Faster than full dynamic programming intersections

Strong Guarantees - Gives a certificate of optimality when exact

In experiments, we find the global optimum on 99% of examples.

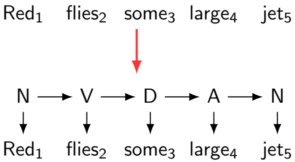
Widely Applicable - Similar techniques extend to other problems

Algorithm

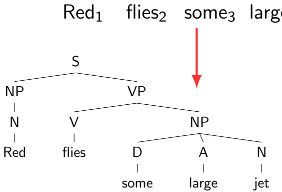
Experiments

LP Relaxations

Integrated Parsing and Tagging

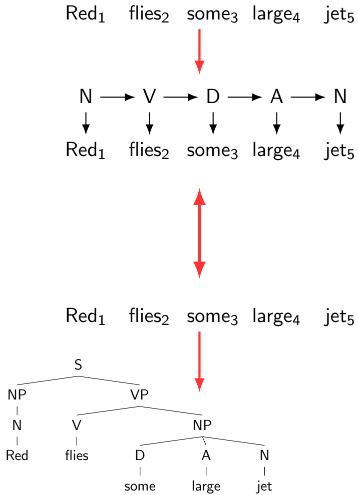


HMM



CFG

Integrated Parsing and Tagging



HMM

Dual Decomposition

CFG

HMM for Tagging

$$\begin{tikzcd}
\mathsf{N} \arrow[r] \arrow[d] & \mathsf{V} \arrow[r] \arrow[d] & \mathsf{D} \arrow[r] \arrow[d] & \mathsf{A} \arrow[r] \arrow[d] & \mathsf{N} \arrow[d] \\
\text{Red}_1 & \text{flies}_2 & \text{some}_3 & \text{large}_4 & \text{jet}_5
\end{tikzcd}$$

- Let $\mathcal{Z}$ be the set of all valid taggings of a sentence and $g(z)$ be a scoring function.

e.g. $g(z) = \log p(\mathrm{Red}_1|\mathrm{N}) + \log p(\mathrm{V}|\mathrm{N}) + \ldots$

HMM for Tagging

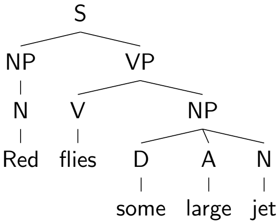
$$
\begin{array}{ccccccccc}
\text{N} & \longrightarrow & \text{V} & \longrightarrow & \text{D} & \longrightarrow & \text{A} & \longrightarrow & \text{N} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
\text{Red}_1 & & \text{flies}_2 & & \text{some}_3 & & \text{large}_4 & & \text{jet}_5
\end{array}
$$

- Let $\mathcal{Z}$ be the set of all valid taggings of a sentence and $g(z)$ be a scoring function.

e.g. $g(z) = \log p(\mathrm{Red}_1|\mathrm{N}) + \log p(\mathrm{V}|\mathrm{N}) + \ldots$

$$z^* = \arg\max_{z \in \mathcal{Z}} g(z) \leftarrow \text{Viterbi decoding}$$

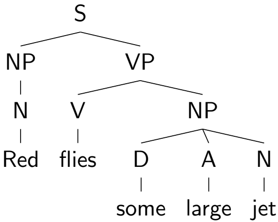
CFG for Parsing



- Let $\mathcal{Y}$ be the set of all valid parse trees for a sentence and $f(y)$ be a scoring function.

e.g. $f(y) = \log p(\mathrm{S} \ \rightarrow \mathrm{NP\ VP}|\mathrm{S}) + \log p(\mathrm{NP} \ \rightarrow \mathrm{N}|\mathrm{NP}) + \ldots$

CFG for Parsing

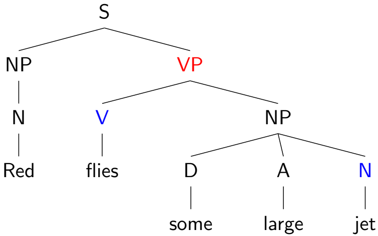


- Let $\mathcal{Y}$ be the set of all valid parse trees for a sentence and $f(y)$ be a scoring function.

e.g. $f(y) = \log p(\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}|\mathrm{S}) + \log p(\mathrm{NP} \rightarrow \mathrm{N}|\mathrm{NP}) + \ldots$

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y) \leftarrow \text{CKY Algorithm}$$

Problem Definition



- Find parse tree that optimizes

$$\textit{score}(\mathrm{S} \;\rightarrow \mathrm{NP}\ \mathrm{VP}) + \textit{score}(\mathrm{VP} \;\rightarrow \mathrm{V}\ \mathrm{NP}) +$$

$$\ldots + \textit{score}(\mathrm{Red}_1, \mathrm{N}) + \textit{score}(\mathrm{V}, \mathrm{N}) + \ldots$$

- Conventional Approach (Bar Hillel et al., 1961)
  - Replace rules like $S \;\rightarrow\; NP\;\; VP$

    with rules like $S_{N,N} \rightarrow NP_{N,V}\;\; VP_{V,N}$

    Painful. $O(t^6)$ increase in complexity for trigram tagging.

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \operatorname*{argmax}_{y \in \mathcal{Y},\, z \in \mathcal{Z}} \quad f(y) + g(z)$$

$$\text{such that for all } i, t, \quad y(i,t) = z(i,t)$$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \underset{y \in \mathcal{Y},\ z \in \mathcal{Z}}{\text{argmax}} \quad f(y) + g(z)$$

Trees

such that for all i, t, $y(i,t) = z(i,t)$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \operatorname*{argmax}_{\underbrace{y \in \mathcal{Y}}_{\textbf{Trees}},\ \underbrace{z \in \mathcal{Z}}_{\textbf{Taggings}}} \quad f(y) + g(z)$$

such that for all i, t, $\; y(i,t) = z(i,t)$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \underset{\underbrace{y \in \mathcal{Y}}_{\textbf{Trees}},\ \underbrace{z \in \mathcal{Z}}_{\textbf{Taggings}}}{\operatorname{argmax}} \quad \overbrace{f(y)}^{\textbf{CFG}} + g(z)$$

such that for all i, t, $y(i,t) = z(i,t)$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \underset{\underset{\textbf{Trees}}{y\in \mathcal{Y}},\ \underset{\textbf{Taggings}}{z\in \mathcal{Z}}}{\text{argmax}} \quad \overset{\textbf{CFG}}{\overset{\downarrow}{f(y)}} + \overset{\textbf{HMM}}{\overset{\downarrow}{g(z)}}$$

$$\text{such that for all } i, t, \quad y(i,t) = z(i,t)$$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

The Integrated Parsing and Tagging Problem

$$\text{Find} \quad \operatorname*{argmax}_{y \in \underset{\textbf{Trees}}{\mathcal{Y}},\; z \in \underset{\textbf{Taggings}}{\mathcal{Z}}} \quad \overset{\textbf{CFG}}{f(y)} + \overset{\textbf{HMM}}{g(z)}$$

$$\text{such that for all } i, t, \quad y(i,t) \underset{\textbf{Constraints}}{=} z(i,t)$$

Where $y(i,t) = 1$ if parse includes tag t at position i

$z(i,t) = 1$ if tagging includes tag t at position i

Algorithm Sketch

Set penalty weights equal to 0 for the tag at each position.

For $k = 1$ **to** K

Algorithm Sketch

Set penalty weights equal to 0 for the tag at each position.

For $k = 1$ **to** K

$y^{(k)} \leftarrow$ Decode $(f(y) + \text{penalty})$ by CKY Algorithm

Algorithm Sketch

Set penalty weights equal to 0 for the tag at each position.

For $k = 1$ **to** K

$y^{(k)} \leftarrow$ Decode $(f(y) + \text{penalty})$ by CKY Algorithm

$z^{(k)} \leftarrow$ Decode $(g(z) - \text{penalty})$ by Viterbi Decoding

Algorithm Sketch

Set penalty weights equal to 0 for the tag at each position.

For $k = 1$ **to** K

$y^{(k)} \leftarrow$ Decode $(f(y) + \text{penalty})$ by CKY Algorithm

$z^{(k)} \leftarrow$ Decode $(g(z) - \text{penalty})$ by Viterbi Decoding

If $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all i, t **Return** $(y^{(k)}, z^{(k)})$

Algorithm Sketch

Set penalty weights equal to 0 for the tag at each position.

For $k = 1$ **to** K

$y^{(k)} \leftarrow$ Decode $(f(y) + \text{penalty})$ by CKY Algorithm

$z^{(k)} \leftarrow$ Decode $(g(z) - \text{penalty})$ by Viterbi Decoding

If $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all i, t **Return** $(y^{(k)}, z^{(k)})$

Else Update penalty weights based on $y^{(k)}(i,t) - z^{(k)}(i,t)$

CKY Parsing

Penalties

$u(i, t) = 0$ for all i,t

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t) y(i, t))$$

Viterbi Decoding

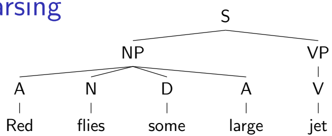
Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t) z(i, t))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i, t) = 1$	if	y contains tag t at position i			

CKY Parsing



Penalties

$u(i,t) = 0$ for all i,t

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

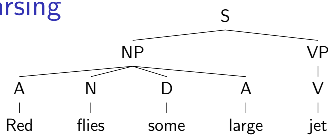
Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Key

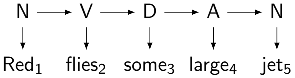
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} \left(f(y) + \sum_{i,t} u(i,t) y(i,t) \right)$$

Viterbi Decoding



$$z^* = \arg\max_{z \in \mathcal{Z}} \left(g(z) - \sum_{i,t} u(i,t) z(i,t) \right)$$

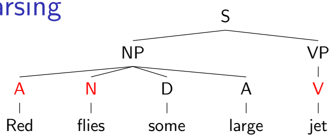
Penalties

$u(i,t) = 0$ for all i,t

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing

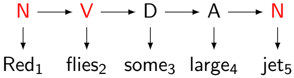


Penalties

$u(i,t) = 0$ for all i,t

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

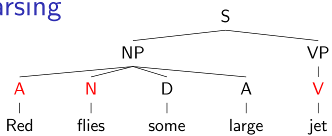


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Key

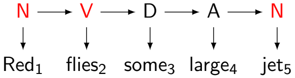
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing

Penalties

$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1,A)$	-1
$u(1,N)$	1
$u(2,N)$	-1
$u(2,V)$	1
$u(5,V)$	-1
$u(5,N)$	1

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

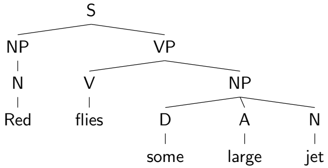
$\text{Red}_1 \quad \text{flies}_2 \quad \text{some}_3 \quad \text{large}_4 \quad \text{jet}_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

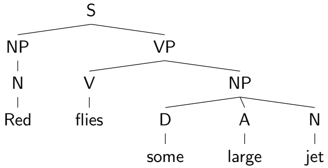
$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Key

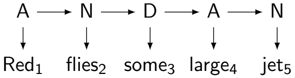
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

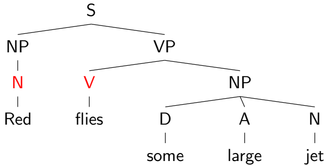
$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Key

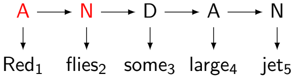
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Iteration 2	
$u(5, V)$	-1
$u(5, N)$	1

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

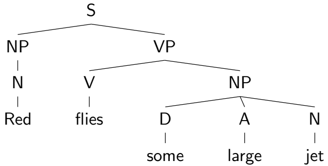
Iteration 1	
$u(1,A)$	-1
$u(1,N)$	1
$u(2,N)$	-1
$u(2,V)$	1
$u(5,V)$	-1
$u(5,N)$	1

Iteration 2	
$u(5,V)$	-1
$u(5,N)$	1

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding

Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

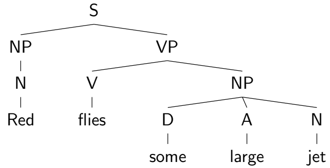
Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Iteration 2	
$u(5, V)$	-1
$u(5, N)$	1

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t)y(i,t))$$

Viterbi Decoding

$$\begin{tikzcd}
N \arrow[r] \arrow[d] & V \arrow[r] \arrow[d] & D \arrow[r] \arrow[d] & A \arrow[r] \arrow[d] & N \arrow[d] \\
\text{Red}_1 & \text{flies}_2 & \text{some}_3 & \text{large}_4 & \text{jet}_5
\end{tikzcd}$$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t)z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

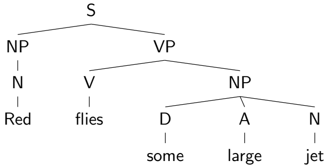
Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Iteration 2	
$u(5, V)$	-1
$u(5, N)$	1

Key

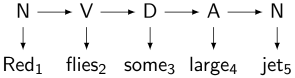
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Viterbi Decoding



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Penalties

$u(i,t) = 0$ for all i,t

Iteration 1	
$u(1,A)$	-1
$u(1,N)$	1
$u(2,N)$	-1
$u(2,V)$	1
$u(5,V)$	-1
$u(5,N)$	1

Iteration 2	
$u(5,V)$	-1
$u(5,N)$	1

Converged

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y) + g(y)$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

Guarantees

Theorem
If at any iteration $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all i, t, then $(y^{(k)}, z^{(k)})$ is the global optimum.

In experiments, we find the global optimum on 99% of examples.

Guarantees

Theorem
If at any iteration $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all i, t, then $(y^{(k)}, z^{(k)})$ is the global optimum.

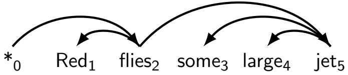
In experiments, we find the global optimum on 99% of examples.

If we do not converge to a match, we can still get a result (more in paper).

Integrated CFG and Dependency Parsing

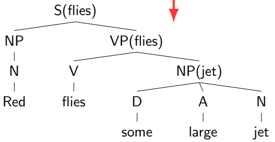
$\text{Red}_1 \quad \text{flies}_2 \quad \text{some}_3 \quad \text{large}_4 \quad \text{jet}_5$

Dependency Model

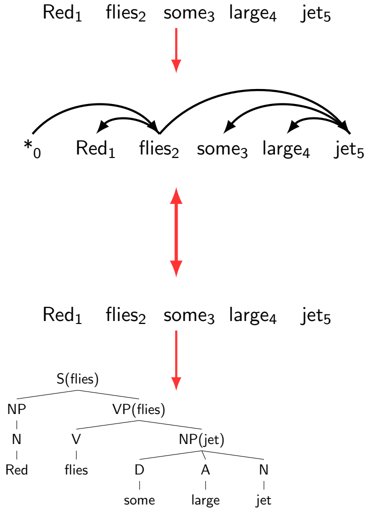


$\text{Red}_1 \quad \text{flies}_2 \quad \text{some}_3 \quad \text{large}_4 \quad \text{jet}_5$

Lexicalized CFG



Integrated CFG and Dependency Parsing

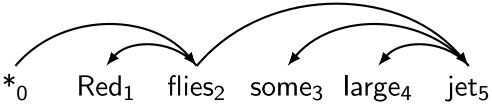


Dependency Model

Dual Decomposition

Lexicalized CFG

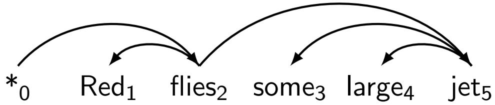
Dependency Parsing



- Let $\mathcal{Z}$ be the set of all valid dependency parses of a sentence and $g(z)$ be a scoring function.

  e.g. $g(z) = \log p(\text{some}_3|\text{jet}_5, \text{large}_4) + ...$

Dependency Parsing

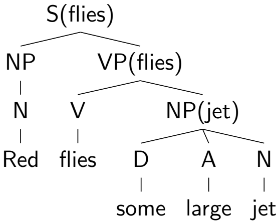


- Let $\mathcal{Z}$ be the set of all valid dependency parses of a sentence and $g(z)$ be a scoring function.

e.g. $g(z) = \log p(\text{some}_3|\text{jet}_5, \text{large}_4) + \ldots$

$$z^* = \arg\max_{z \in \mathcal{Z}} g(z) \leftarrow \text{Eisner (2000) algorithm}$$

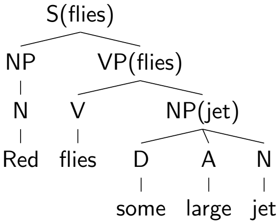
Lexicalized PCFG



- Let $\mathcal{Y}$ be the set of all valid dependency parses of a sentence and $f(y)$ be a scoring function.

e.g. $f(y) = \log p(\mathrm{S(flies)} \ \to \mathrm{NP(Red)\ VP(flies)}|\mathrm{S(flies)}) + ...$

Lexicalized PCFG



- Let $\mathcal{Y}$ be the set of all valid dependency parses of a sentence and $f(y)$ be a scoring function.

e.g. $f(y) = \log p(\text{S(flies)} \rightarrow \text{NP(Red) VP(flies)}|\text{S(flies)}) + \ldots$

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y) \leftarrow \text{Modified CKY algorithm}$$

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \operatorname*{argmax}_{y\in \mathcal{Y},\ z\in \mathcal{Z}} \quad f(y) + g(z)$$

$$\text{such that for all } i, j, \ \ y(i,j) = z(i,j)$$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \underset{y \in \mathcal{Y},\ z \in \mathcal{Z}}{\text{argmax}} \quad f(y) + g(z)$$

Trees

such that for all i, j, $y(i,j) = z(i,j)$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \underset{\underbrace{y \in \mathcal{Y}}_{\textbf{Trees}},\ \underbrace{z \in \mathcal{Z}}_{\textbf{Dependency Trees}}}{\operatorname{argmax}} \quad f(y) + g(z)$$

such that for all i, j, $y(i,j) = z(i,j)$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \underset{\underbrace{y\in \mathcal{Y}}_{\textbf{Trees}},\ \underbrace{z\in \mathcal{Z}}_{\textbf{Dependency Trees}}}{\operatorname{argmax}} \quad \overbrace{f(y)}^{\textbf{CFG}} + g(z)$$

$$\text{such that for all } i,j, \;\; y(i,j) = z(i,j)$$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \underset{\underbrace{y \in \mathcal{Y}}_{\text{Trees}},\ \underbrace{z \in \mathcal{Z}}_{\text{Dependency Trees}}}{\text{argmax}} \quad \underbrace{f(y)}_{\text{CFG}} + \underbrace{g(z)}_{\text{Dependency}}$$

$$\text{such that for all } i, j, \ \ y(i,j) = z(i,j)$$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

The Integrated Constituency and Dependency Parsing Problem

$$\text{Find} \quad \underset{\substack{y \in \mathcal{Y},\ z \in \mathcal{Z} \\ \textbf{Trees} \qquad \textbf{Dependency Trees}}}{\text{argmax}} \quad \overbrace{f(y)}^{\textbf{CFG}} + \overbrace{g(z)}^{\textbf{Dependency}}$$

$$\text{such that for all } i, j, \quad \underbrace{y(i,j) = z(i,j)}_{\textbf{Constraints}}$$

Where $y(i,j) = 1$ if parse includes dependency from word i to j

$z(i,j) = 1$ if parse includes dependency from word i to j

CKY Parsing

Penalties

$u(i,j) = 0$ for all i,j

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Dependency Parsing

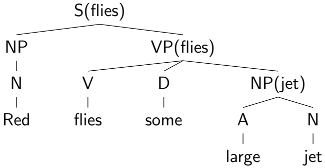
$*_0$ Red$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

$u(i,j) = 0$ for all i,j

Dependency Parsing

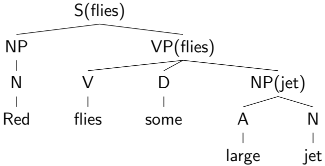
$*_0$ $\quad$ Red$_1$ $\quad$ flies$_2$ $\quad$ some$_3$ $\quad$ large$_4$ $\quad$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

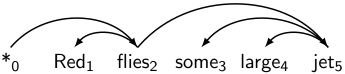


$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

$u(i,j) = 0$ for all i,j

Dependency Parsing

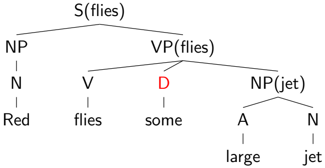


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

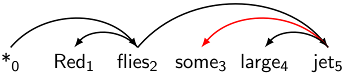


$$y^* = \arg\max_{y \in \mathcal{Y}} \Big(f(y) + \sum_{i,j} u(i,j) y(i,j)\Big)$$

Penalties

$u(i,j) = 0$ for all i,j

Dependency Parsing

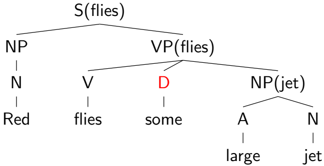


$$z^* = \arg\max_{z \in \mathcal{Z}} \Big(g(z) - \sum_{i,j} u(i,j) z(i,j)\Big)$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



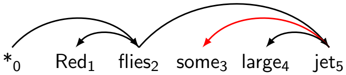
$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

$u(i,j) = 0$ for all i,j

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

Penalties

$u(i,j) = 0$ for all i,j

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Dependency Parsing

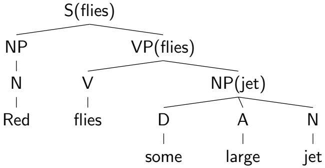
$*_0$ Red_1 flies_2 some_3 large_4 jet_5

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

$u(i,j) = 0$ for all i,j

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing

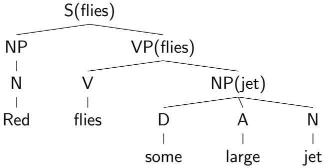
$*_0$ $\quad$ Red$_1$ $\quad$ flies$_2$ $\quad$ some$_3$ $\quad$ large$_4$ $\quad$ jet$_5$

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



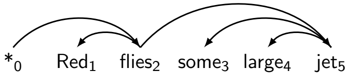
$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

$u(i,j) = 0$ for all i,j

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing

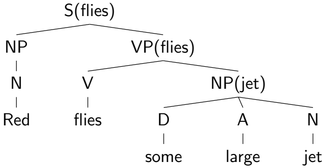


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

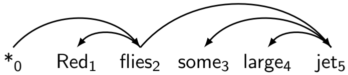
$f(y)$	$\Leftarrow$	CFG
$\mathcal{Y}$	$\Leftarrow$	Parse Trees
$y(i,j) = 1$	if	y contains dependency i,j
$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Z}$	$\Leftarrow$	Dependency Trees

CKY Parsing



$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Dependency Parsing



$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

Penalties

$u(i,j) = 0$ for all i,j

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Converged

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y) + g(y)$$

Roadmap

Algorithm

Experiments

LP Relaxations

Experiment

Properties:

- Exactness
- Parsing Accuracy

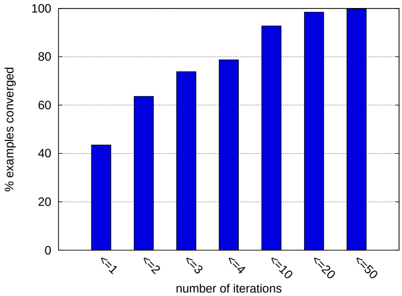
Experiments on:

- English Penn Treebank

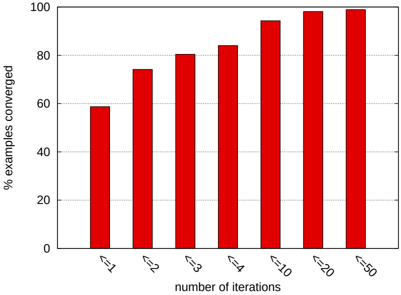
Models

- Collins (1997) Model 1
- Semi-Supervised Dependency Parser (Koo, 2008)
- Trigram Tagger (Toutanova, 2000)

How quickly do the models converge?

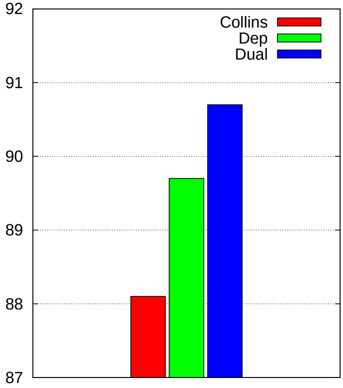


Integrated Dependency Parsing



Integrated POS Tagging

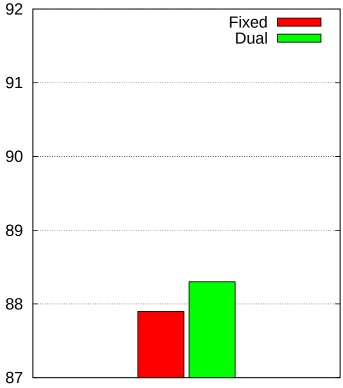
Integrated Constituency and Dependency Parsing: Accuracy



F_1 Score

- Collins (1997) Model 1
- Fixed, First-best Dependencies from Koo (2008)
- Dual Decomposition

Integrated Parsing and Tagging: Accuracy



F_1 Score

- Fixed, First-Best Tags From Toutanova (2000)
- Dual Decomposition

Roadmap

Algorithm

Experiments

LP Relaxations

Dual Decomposition and Linear Programming Relaxations

Theorem

- If the dual decomposition algorithm converges, then $(y^{(k)}, z^{(k)})$ is the global optimum.

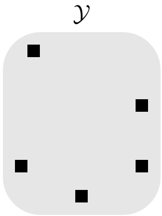
Questions

- What problem is dual decomposition solving?
- How come the algorithm doesn't always converge?

Dual decomposition searches over a linear programming relaxation of the original problem.

Convex Hulls for CKY

A parse tree can be represented as a binary vector $y \in \mathcal{Y}$.
$y(A \rightarrow B\ C, i, j, k) = 1$ if rule $A \rightarrow B\ C$ is used at span i, j, k.

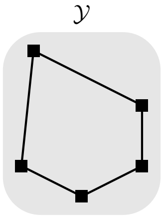


Parsing

- If f is linear, $\arg \max_{y \in conv(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.

Convex Hulls for CKY

A parse tree can be represented as a binary vector $y \in \mathcal{Y}$.
$y(A \rightarrow B\ C, i, j, k) = 1$ if rule $A \rightarrow B\ C$ is used at span i, j, k.

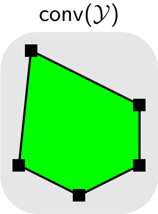


Parsing

- If f is linear, $\arg \max_{y \in conv(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.

Convex Hulls for CKY

A parse tree can be represented as a binary vector $y \in \mathcal{Y}$.
$y(A \rightarrow B\ C, i, j, k) = 1$ if rule $A \rightarrow B\ C$ is used at span i, j, k.

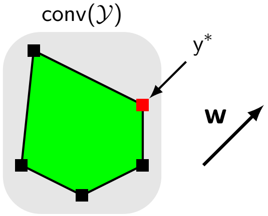


Parsing

- If f is linear, $\arg \max_{y \in conv(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.

Convex Hulls for CKY

A parse tree can be represented as a binary vector $y \in \mathcal{Y}$.
$y(A \rightarrow B\ C, i, j, k) = 1$ if rule $A \rightarrow B\ C$ is used at span i, j, k.

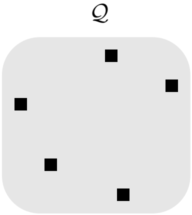


Parsing

- If f is linear, $\arg \max_{y \in conv(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.

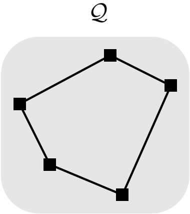
Combined Problem

$$\mathcal{Q} = \{(y,z)\colon y \in \mathcal{Y}, z \in \mathcal{Z},$$
$$y(i,t) = z(i,t) \text{ for all } (i,t)\}$$



Combined Problem

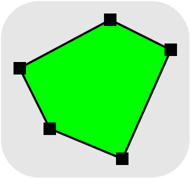
$$\mathcal{Q} = \{(y,z)\colon y \in \mathcal{Y}, z \in \mathcal{Z},$$
$$y(i,t) = z(i,t) \text{ for all } (i,t)\}$$



Combined Problem

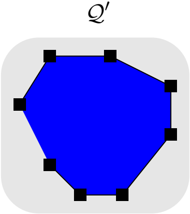
$$\mathcal{Q} = \{(y, z) \colon y \in \mathcal{Y}, z \in \mathcal{Z},$$
$$y(i, t) = z(i, t) \text{ for all } (i, t)\}$$

conv($\mathcal{Q}$)



Combined Problem

$$\mathcal{Q} = \{(y,z)\colon y \in \mathcal{Y}, z \in \mathcal{Z},$$
$$y(i,t) = z(i,t) \text{ for all } (i,t)\}$$

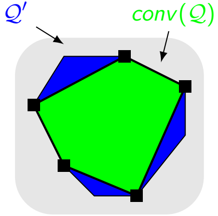


$$\mathcal{Q}' = \{(\mu,\nu)\colon \mu \in \mathrm{conv}(\mathcal{Y}), \nu \in \mathrm{conv}(\mathcal{Z}),$$
$$\mu(i,t) = \nu(i,t) \text{ for all } (i,t)\}$$

Dual decomposition searches over $\mathcal{Q}'$

Combined Problem

$$\mathcal{Q} = \{(y,z) \colon y \in \mathcal{Y}, z \in \mathcal{Z},$$
$$y(i,t) = z(i,t) \text{ for all } (i,t)\}$$

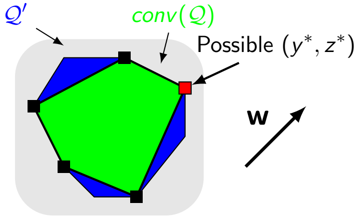


$$\mathcal{Q}' = \{(\mu,\nu) \colon \mu \in \mathrm{conv}(\mathcal{Y}), \nu \in \mathrm{conv}(\mathcal{Z}),$$
$$\mu(i,t) = \nu(i,t) \text{ for all } (i,t)\}$$

Dual decomposition searches over $\mathcal{Q}'$

Combined Problem

$$\mathcal{Q} = \{(y,z)\colon y \in \mathcal{Y}, z \in \mathcal{Z}, \\ y(i,t) = z(i,t) \text{ for all } (i,t)\}$$



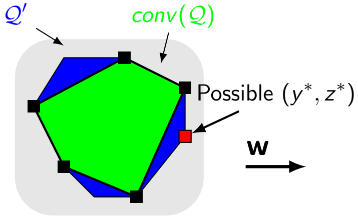
$$\mathcal{Q}' = \{(\mu,\nu)\colon \mu \in \mathrm{conv}(\mathcal{Y}), \nu \in \mathrm{conv}(\mathcal{Z}), \\ \mu(i,t) = \nu(i,t) \text{ for all } (i,t)\}$$

Dual decomposition searches over $\mathcal{Q}'$

Depending on the weight vector, $(y^*, z^*) \in \mathcal{Q}'$ could be in $\mathcal{Q}$ or in the strict outer bound.

Combined Problem

$$\mathcal{Q} = \{(y, z) \colon y \in \mathcal{Y}, z \in \mathcal{Z}, \\ y(i, t) = z(i, t) \text{ for all } (i, t)\}$$

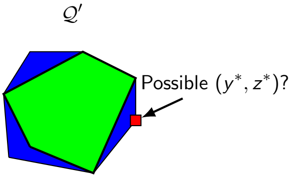


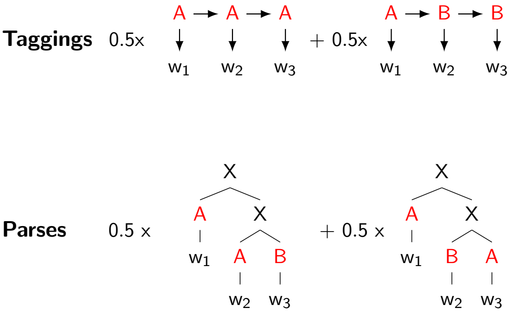
$$\mathcal{Q}' = \{(\mu, \nu) \colon \mu \in \mathrm{conv}(\mathcal{Y}), \nu \in \mathrm{conv}(\mathcal{Z}), \\ \mu(i, t) = \nu(i, t) \text{ for all } (i, t)\}$$

Dual decomposition searches over $\mathcal{Q}'$

Depending on the weight vector, $(y^*, z^*) \in \mathcal{Q}'$ could be in $\mathcal{Q}$ or in the strict outer bound.

Are there points strictly in the outer bound?





Best result can be a fractional solution.

Convex combination of these structures.

Summary

A Dual Decomposition algorithm for integrated decoding

Simple - Uses only simple, off-the-shelf dynamic programming algorithms to solve a harder problem.

Efficient - Faster than classical methods for dynamic programming intersection.

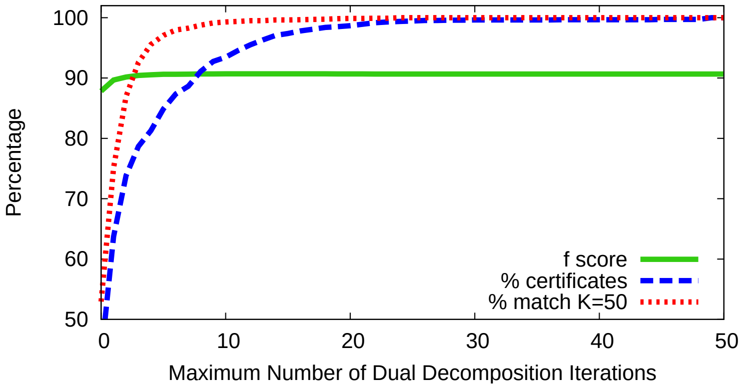
Strong Guarantees - Solves a linear programming relaxation which gives a certificate of optimality.

Finds the exact solution on 99% of the examples.

Widely Applicable - Similar techniques extend to other problems

Appendix

Iterative Progress



Deriving the Algorithm

Goal:

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y)$$

Rewrite:

$$\arg \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y)$$

$$\text{s.t. } z(i,j) = y(i,j) \text{ for all } i,j$$

Lagrangian: $L(u, y, z) = f(z) + g(y) + \sum_{i,j} u(i,j) \left(y(i,j) - z(i,j) \right)$

Deriving the Algorithm

Goal:

$$y^* = \arg\max_{y \in \mathcal{Y}} f(y)$$

Rewrite:

$$\arg\max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y)$$

$$\text{s.t. } z(i,j) = y(i,j) \text{ for all } i,j$$

Lagrangian: $L(u,y,z) = f(z) + g(y) + \sum_{i,j} u(i,j)\,(y(i,j) - z(i,j))$

The dual problem is to find $\min_u L(u)$ where

$$L(u) = \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u,y,z) \quad = \quad \max_{z \in \mathcal{Z}} \left(f(z) + \sum_{i,j} u(i,j) z(i,j) \right) + \max_{y \in \mathcal{Y}} \left(g(y) - \sum_{i,j} u(i,j) y(i,j) \right)$$

Dual is an upper bound: $L(u) \geq f(z^*) + g(y^*)$ for any u

A Subgradient Algorithm for Minimizing $L(u)$

$$L(u) = \max_{z \in \mathcal{Z}} \left(f(z) + \sum_{i,j} u(i,j) y(i,j) \right) + \max_{y \in \mathcal{Y}} \left(g(y) - \sum_{i,j} u(i,j) z(i,j) \right)$$

$L(u)$ is convex, but not differentiable. A *subgradient* of $L(u)$ at u is a vector g_u such that for all v,

$$L(v) \geq L(u) + g_u \cdot (v - u)$$

Subgradient methods use updates $u' = u - \alpha g_u$

In fact, for our $L(u)$, $g_u(i,j) = z^*(i,j) - y^*(i,j)$

Related Work

- Methods that use general purpose linear programming or integer linear programming solvers (Martins et al. 2009; Riedel and Clarke 2006; Roth and Yih 2005)
- Dual decomposition/Lagrangian relaxation in combinatorial optimization (Dantzig and Wolfe, 1960; Held and Karp, 1970; Fisher 1981)
- Dual decomposition for inference in MRFs (Komodakis et al., 2007; Wainwright et al., 2005)
- Methods that incorporate combinatorial solvers within loopy belief propagation (Duchi et al. 2007; Smith and Eisner 2008)