Fast CFG Parsing Requires Fast Boolean Matrix Multiplication

Lillian Lee
Cornell University

ACL/EACL ’97, Madrid

CFG Parsing

- CKY and Earley parsers: $O(n^3)$.
- Graham et al. [1980] (variant of Earley): $O(n^3/\log n)$.
- Valiant [1975] (variant of CKY) : same running time as Boolean matrix multiplication (BMM).

The only practical algorithms are basically cubic and do not rely on BMM.

Boolean Matrix Multiplication

Let A and B be $m \times m$ boolean matrices, and let C be their Boolean product.

$$c_{ij} = \bigvee_{k=1}^{m} (a_{ik} \wedge b_{kj})$$

$c_{ij} = 1$ if and only if there exists a k such that $a_{ik} = b_{kj} = 1$.

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Standard algorithm: $O(m^3)$.

Strassen [1969] is $O(m^{2.81})$, Coppersmith and Winograd [1990] is $O(m^{2.376})$; neither is practical.

A practical sub-cubic BMM algorithm is not likely to exist.

Main Result

Since BMM is slow, can we speed up CFG parsing by avoiding the use of BMM?

No: Any practical parser running in time $O(n^{3-\epsilon})$ can be converted into a $O(m^{3-\epsilon/3})$ BMM algorithm with very little computational overhead.

This implies that practical parsers running in significantly lower than cubic time are unlikely to exist.

Practical Parsers

In order to talk about practical parsers, we need to define them.

A *c-derivation* is a substring derivation that is *consistent* with a sentential derivation. ($X \Rightarrow^* s_i \cdots s_j$ and $S \Rightarrow^* s_1 \cdots s_{i-1} X s_{j+1} \cdots s_n$).

A practical parser should

- Keep track of c-derivations (at least)
- Produce output from which parse info can be efficiently retrieved

C-Parsers

We formalize (strengthen) Lang's [1994] informal defintion of parsing.

Given G and s, a *c-parser* creates a parse oracle $\mathcal{F}_{G,s}$.
Given a nonterminal X and two numbers i and j,

- If X **c-derives** $s_i \cdots s_j$, then $\mathcal{F}_{G,s}$ says "yes".
- If X **does not derive** $s_i \cdots s_j$, then $\mathcal{F}_{G,s}$ says "no".
- $\mathcal{F}_{G,s}$ answers queries in constant time.

"If X derives ..." excludes Earley parsers.

"If X does not c-derive ..." excludes CKY-type parsers.

Claim: Practical parsers $\equiv$ c-parsers.

Sketch of the Reduction

We adapt the technique of Satta [1994]. Let $C = A \times B$ (we do not know C!)

Key point: $c_{ij} = 1$ if and only if there is a k such that

$$a_{ik} = b_{kj} = 1$$

We are checking for a match of "inner indices".

Given A and B, we will produce a grammar G and a string w that encode this inner index checking.

We will find C by querying the parse output $\mathcal{F}_{G,w}$ created by running a c-parser on G and w.

Construction Preliminaries

Let A and B be two $m \times m$ Boolean matrices, and let $C = A \times B$.

We create a dummy string $w = w_1 w_2 w_3 \cdots$, $|w| = O(m^{1/3})$.

Assume we can encode matrix indices as pairs of numbers:

$$i \longleftrightarrow (i_1, i_2)$$

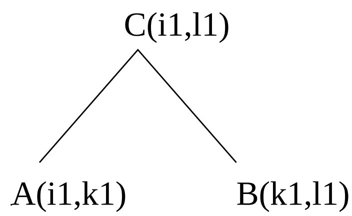
(the trick is to do this in such a way as to ensure time bounds).

Desired property: Matrix entry $x_{ij} = 1$ if and only if nonterminal $X_{i_1, j_1} \Rightarrow^* w_{i_2} \cdots w_{j_2}$.

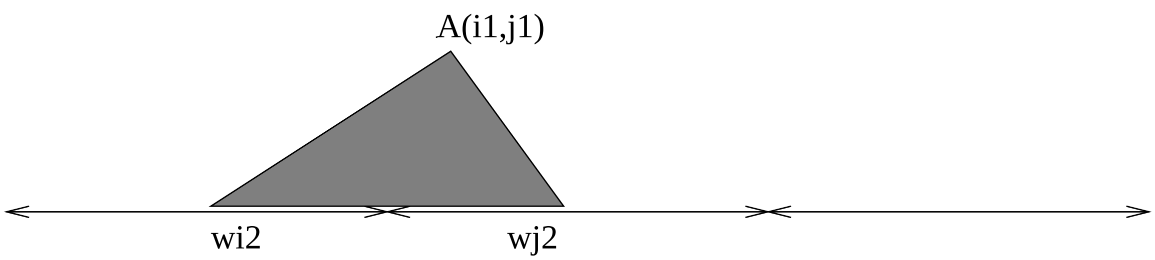
(The hard part is guaranteeing this for c_{ij}.)

Construction

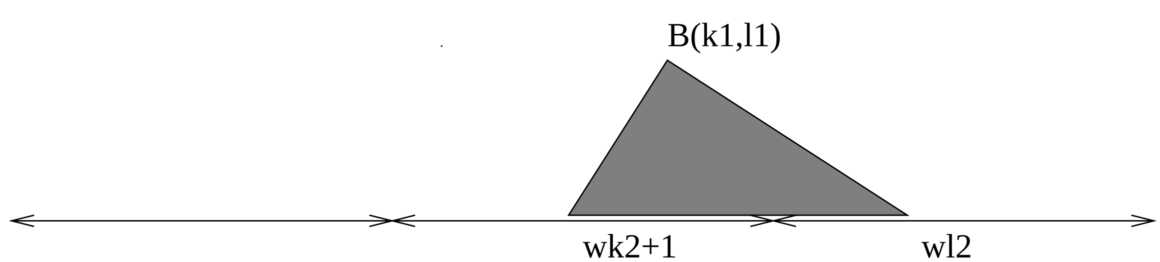
For all i_1, k_1, l_1, $C_{i_1,l_1} \rightarrow A_{i_1,k_1} B_{k_1,l_1}$.



If matrix entry $a_{ij} = 1$, then $A_{i_1, j_i} \Rightarrow^* w_{i_2} \cdots w_{j_2}$.



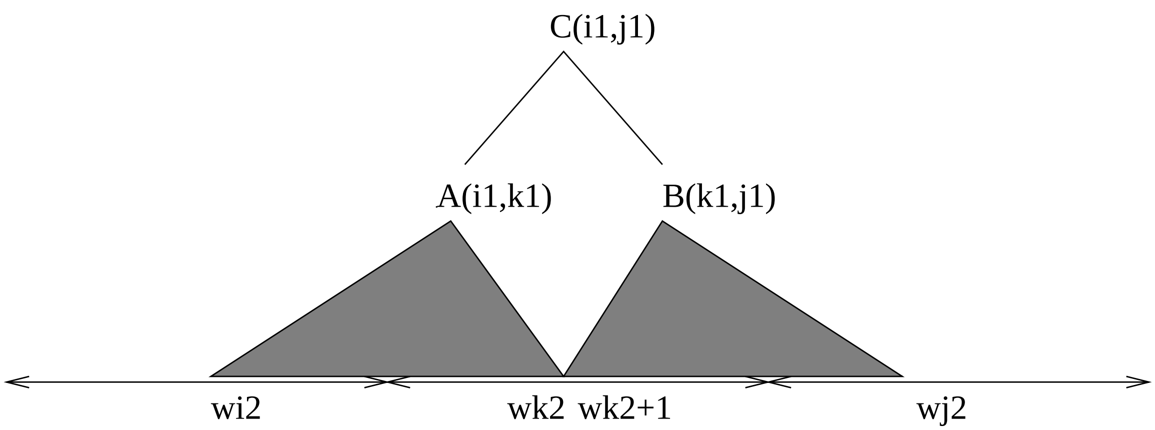
If matrix entry $b_{kl} = 1$, then $B_{k_1,l_i} \Rightarrow^* w_{k_2+1} \cdots w_{l_2}$



When $k = j$, the substrings line up!

Construction Results

If $a_{ik} = b_{kj} = 1$, then $C_{i_1,j_1} \Rightarrow^* w_{i_2} \cdots w_{j_2}$.



In fact, C_{i_1,j_1} c-derives $w_{i_2} \cdots w_{j_2}$.

Theorem: The matrix entry $c_{ij} = 1$ if and only if C_{i_1,j_1} c-derives $w_{i_2}^{j_2}$.

Note: our index encoding allows us to write G in Chomsky Normal Form such that $|G| = O(m^2)$ (which is the size of the input matrices).

Resulting BMM Algorithm

Let P be any c-parser.

1. Given A and B, construct G and w.
2. Run P to yield output oracle $\mathcal{F}_{G,w}$.
3. For all i and j, ask $\mathcal{F}_{G,w}$ whether C_{i_1,j_1} c-derives $w_{i_2} \cdots w_{j_2}$.
   - Set $c_{ij} = 1$ if and only if $\mathcal{F}_{G,w}$ answers "yes".

Computational Consequences

Recall that $|G| = O(m^2)$ and $|w| = O(m^{1/3})$.

Theorem: If P takes time $O(gn^\alpha)$ to c-parse a string of length n wrt a grammar of size g, then BMM can be done in time $O(m^{2+\alpha/3})$.

CFGP		BMM
$O(n^3)$	$\Rightarrow$	$O(m^3)$
$O(n^{2.43})$	$\Rightarrow$	$O(m^{2.81})$ (Strassen)
$O(n^{1.12})$	$\Rightarrow$	$O(m^{2.376})$ (Coppersmith and Winograd)

Key point: Given that substantially sub-cubic practical BMM algorithms are unlikely, substantially sub-cubic c-parsers are also unlikely to exist.

Related Results

To our knowledge, there are no non-trivial hardness results for CFG parsing.

- Harrison and Havel [1974]: BMM checking reduces to CFG recognition
  - ▷ Different problem
  - ▷ $O(n^{1.5}) \Rightarrow O(m^3)$
- Seiferas [86], Gallaire [1969]: sub-quadratic lower bound on on-line CFL recognition
  - ▷ Different problem
  - ▷ Different computational model
- Valiant [1975]: CFG parsing reduces to BMM
  - ▷ Better time bound
  - ▷ Other direction of reduction