A Little Bit of History...

- Who invented Dynamic Programming? and when was it invented?
  - R. Bellman (1940s-50s)
  - A. Viterbi (1967)
  - E. Dijkstra (1959)
  - Hart, Nilsson, and Raphael (1968)  
    - Dijkstra => A* Algorithm
  - D. Knuth (1977)
  - Dijkstra on Grammar (Hypergraph)
Dynamic Programming

- Dynamic Programming is everywhere in NLP
  - Viterbi Algorithm for Hidden Markov Models
  - CKY Algorithm for Parsing and Machine Translation
  - Forward-Backward and Inside-Outside Algorithms
- Also everywhere in AI/ML
  - Reinforcement Learning, Planning (POMDP)
  - AI Search: Uniform-cost, A*, etc.
- This tutorial: a **unified** theoretical view of DP
  - Focusing on *Optimization Problems*

Review: DP Basics

- DP = Divide-and-Conquer + Two Principles:
  - **[required]** Optimal Subproblem Property
  - **[recommended]** Sharing of Common Subproblems
- Structure of the Search Space
  - Incremental
  - Graph
  - Knapsack, Edit Dist., Sequence Alignment
- Branching
  - Hypergraph
  - Matrix-Chain, Polygon Triangulation, Optimal BST
### Two Dimensional Survey

<table>
<thead>
<tr>
<th>Topological (acyclic)</th>
<th>Best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>graphs with semirings</strong></td>
<td>Viterbi</td>
</tr>
<tr>
<td><strong>hypergraphs with weight functions</strong></td>
<td>Generalized Viterbi</td>
</tr>
</tbody>
</table>

- **search space**

---

### Graphs in NLP

#### part-of-speech tagging

![Lattice in speech diagram](image)

- **lattice in speech**

---

Liang Huang (Penn)  Dynamic Programming
in a weighted graph, we need two operators:

- extension (multiplicative) and summary (additive)
- the weight of a path is the product of edge weights
- the weight of a vertex is the summary of path weights

\[ d(\pi_1) = \bigotimes_{e_i \in \pi_1} w(e_i) = w(e_1) \otimes w(e_2) \otimes w(e_3) \]

\[ d(t) = \bigoplus_{\pi_i} w(\pi_i) = w(p_1) \oplus w(p_2) \oplus \cdots \]

A **monoid** is a triple \((A, \otimes, 1)\) where

1. \(\otimes\) is a closed associative binary operator on the set \(A\),
2. \(1\) is the identity element for \(\otimes\), i.e., for all \(a \in A\), \(a \otimes 1 = 1 \otimes a = a\).

A monoid is **commutative** if \(\otimes\) is commutative.

A **semiring** is a 5-tuple \(R = (A, \oplus, \otimes, 0, 1)\) such that

1. \((A, \oplus, 0)\) is a commutative monoid.
2. \((A, \otimes, 1)\) is a monoid.
3. \(\otimes\) distributes over \(\oplus\): for all \(a, b, c\) in \(A\),

\[ (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c), \]

\[ c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b). \]

4. \(0\) is an **annihilator** for \(\otimes\): for all \(a\) in \(A\), \(0 \otimes a = a \otimes 0 = 0\).

\[(0, 1], +, \otimes, 0, 1) \quad \square\]
\((0, 1], \max, \times, 0, 1) \quad \times\]
Examples

<table>
<thead>
<tr>
<th>Semiring</th>
<th>Set</th>
<th>$\oplus$</th>
<th>$\otimes$</th>
<th>$\bar{0}$</th>
<th>$\bar{1}$</th>
<th>intuition/application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>${0, 1}$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>$0$</td>
<td>$1$</td>
<td>logical deduction, recognition</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$[0, 1]$</td>
<td>$\max$</td>
<td>$\times$</td>
<td>$0$</td>
<td>$1$</td>
<td>prob. of the best derivation</td>
</tr>
<tr>
<td>Inside</td>
<td>$\mathbb{R}^+ \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
<td>shortest-distance</td>
</tr>
<tr>
<td>Real</td>
<td>$\mathbb{R} \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
<td>with non-negative weights</td>
</tr>
<tr>
<td>Tropical</td>
<td>$\mathbb{R}^+ \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
<td>number of paths</td>
</tr>
</tbody>
</table>

Ordering

- **idempotent**
  A semiring $(A, \oplus, \otimes, \bar{0}, \bar{1})$ is idempotent if for all $a$ in $A$, $a \oplus a = a$.

- **comparison**
  $\ (a \leq b) \iff (a \oplus b = a)$ defines a partial ordering.

- **examples: boolean, viterbi, tropical, real, ...**
  
  
  $(\{0, 1\}, \lor, \land, 0, 1) \quad (\mathbb{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0)$

  
  
  $([0, 1], \max, \otimes, 0, 1) \quad (\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)$

- **total-order for optimization problems**
  A semiring is totally-ordered if defines a total ordering.

- **examples: all of the above**
Monotonicity

- **monotonicity**
  Let $K = (A, \oplus, \otimes, \overline{0}, \overline{1})$ be a semiring, and $\leq$ a partial ordering over $A$. We say $K$ is **monotonic** if for all $a, b, c \in A$

  $$(a \leq b) \Rightarrow (a \otimes c \leq b \otimes c) \quad (a \leq b) \Rightarrow (c \otimes a \leq c \otimes b)$$

- **optimal substructure** in dynamic programming

- idempotent $\Rightarrow$ monotone (from distributivity)
  - $(a+b)\otimes c = (a\otimes c)+(b\otimes c)$; if $a \leq b$, $(a\otimes c) = (a\otimes c)+(b\otimes c)$
  - by def. of comparison, $a\otimes c \leq b\otimes c$

---

DP on Graphs

- optimization problems on graphs
  => generic shortest-path problem

- weighted directed graph $G=(V, E)$ with a function $w$ that assigns each edge a weight from a semiring

- compute the best weight of the target vertex $t$

- generic update along edge $(u, v)$
  $$d(v) \oplus = d(u) \otimes w(u, v)$$

- how to avoid cyclic updates?
  - only update when $d(u)$ is fixed
# Two Dimensional Survey

<table>
<thead>
<tr>
<th>search space</th>
<th>traversing order</th>
<th>best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td>topological (acyclic)</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>

Viterbi Algorithm for DAGs

1. **topological sort**
2. visit each vertex \( v \) in sorted order and do updates
   - for each **incoming** edge \((u, v)\) in \( E \)
   - use \( d(u) \) to update \( d(v) \): \( d(v) \oplus = d(u) \otimes w(u, v) \)
   - key observation: \( d(u) \) is fixed to optimal at this time

\[
\begin{align*}
\text{for each incoming edge } & (u, v) \text{ in } E \\
\text{use } d(u) \text{ to update } & d(v): d(v) \oplus = d(u) \otimes w(u, v) \\
\text{key observation: } & d(u) \text{ is fixed to optimal at this time}
\end{align*}
\]

- time complexity: \( O(V + E) \)
Variant 1: forward-update

1. topological sort

2. visit each vertex $v$ in sorted order and do updates
   - for each outgoing edge $(v, u)$ in $E$
   - use $d(v)$ to update $d(u)$: $d(u) \oplus = d(v) \otimes w(v, u)$
   - key observation: $d(v)$ is fixed to optimal at this time

\[ d(u) \oplus = d(v) \otimes w(v, u) \]

- time complexity: $O(V + E)$

Examples

- [Number of Paths in a DAG]
  - just use the counting semiring $(\mathbb{N}, +, \times, 0, 1)$
  - note: this is not an optimization problem!

- [Longest Path in a DAG]
  - just use the semiring $(\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$

- [Part-of-Speech Tagging with a Hidden Markov Model]
Example: Speech Alignment

- Time complexity: \(O(n^2)\)
- Also used in: edit distance, biological sequence alignment

Example: Word Alignment

- Key difference
- Reorderings in translation!
- Sequence/speech alignment is always monotonic
- Complexity under HMM
- Word alignment is \(O(n^3)\)
  - For every \((i, j)\)
  - Enumerate all \((i-1, k)\)
- Sequence alignment \(O(n^2)\)
Chinese Word Segmentation

Liang Huang (Penn)

Dynamic Programming

Phrase-based Decoding

Huang and Chiang

Forest Rescoring
Huang and Chiang

Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon

with Sharon held a talk

yu Shalong juxing le huitan

source-side: coverage vector
target-side: grow hypotheses strictly left-to-right

space: $O(2^n)$, time: $O(2^n n^2)$ -- cf. traveling salesman problem
Traveling Salesman Problem & MT

- a classical NP-hard problem
- goal: visit each city once and only once
- exponential-time dynamic programming
- state: cities visited so far (bit-vector)
- search in this $O(2^n)$ transformed graph
- MT: each city is a source-language word
- restrictions in reordering can reduce complexity => distortion limit
- $=>$ syntax-based MT

Adding a Bigram Model

- “refined” graph: annotated with language model words
- still dynamic programming, just larger search space

space: $O(2^n)$,  
time: $O(2^n n^2)$
$=>$ space: $O(2^n V^{m-1})$,  
time: $O(2^n V^{m-1} n^2)$

for $m$-gram language models
Two Dimensional Survey

<table>
<thead>
<tr>
<th>search space</th>
<th>topological (acyclic)</th>
<th>best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>

**Dijkstra Algorithm**

- Dijkstra does not require acyclicity
  - instead of topological order, we use **best-first** order
- but this requires **superiority** of the semiring

Let $K = (A, \oplus, \otimes, 0, 1)$ be a semiring, and $\leq$ a partial ordering over $A$. We say $K$ is **superior** if for all $a, b \in A$

$$a \leq a \otimes b, \quad b \leq a \otimes b.$$  

- intuition: combination always gets worse
- contrast: monotonicity: combination preserves order

$$\begin{align*}
(a \leq b) & \implies (a \otimes c \leq b \otimes c) \\
& \quad \text{(e.g., } \{0, 1\}, \vee, \wedge, 0, 1) \checkmark \\
& \quad \text{(e.g., } [0, 1], \min, +, +\infty, 0) \checkmark \\
& \quad \text{(e.g., } R^+ \cup \{+\infty\}, \min, +, +\infty, 0) \checkmark \\
& \quad \text{(e.g., } R \cup \{+\infty\}, \min, +, +\infty, 0) \times
\end{align*}$$

\[ \text{d}(u) \quad \overrightarrow{\text{w}(e)} \quad \text{d}(u) \otimes \text{w}(e) \]
Dijkstra Algorithm

- keep a cut $(S : V - S)$ where $S$ vertices are fixed
- maintain a priority queue $Q$ of $V - S$ vertices
- each iteration choose the best vertex $v$ from $Q$
- move $v$ to $S$, and use $d(v)$ to forward-update others

\[ d(u) \oplus = d(v) \odot w(v, u) \]

**time complexity:**
- $O((V+E) \lg V)$ (binary heap)
- $O(V \lg V + E)$ (fib. heap)

Viterbi vs. Dijkstra

- structural vs. algebraic constraints
- Dijkstra only applicable to optimization problems

forward-backward (Inside semiring)

acyclic: Viterbi

many NLP problems

superior: Dijkstra

cyclic FSMs/grammars

non-probabilistic models
What if both fail?

monotonic optimization problems

acyclic: Viterbi

many NLP problems

superior: Dijkstra

generalized Bellman-Ford
(CLR, 1990; Mohri, 2002)

or, first do strongly-connected components (SCC)
which gives a DAG; use Viterbi globally on this SCC-DAG;
use Bellman-Ford locally within each SCC

What if both work?

monotonic optimization problems

acyclic: Viterbi

many NLP problems

superior: Dijkstra

full Dijkstra is slower than Viterbi
\[ O((V + E) \lg V) \quad \text{vs.} \quad O(V + E) \]

but it can finish as early as the target vertex is popped
\[ a \ (V + E) \lg V \quad \text{vs.} \quad V + E \]

\[ Q: \text{how to (magically) reduce } a? \]
A* Search: Intuition

- Dijkstra is “blind” about how far the target is
- may get “trapped” by obstacles
- can we be more intelligent about the future?
- idea: prioritize by \( s-v \) distance + \( v-t \) estimate

A* Heuristic

- \( h(v) \): the distance from \( v \) to target \( t \)
- \( \hat{h}(v) \) must be an optimistic estimate of \( h(v) \): \( \hat{h}(v) \leq h(v) \)
- Dijkstra is a special case where \( \hat{h}(v) = 0 \) for dist.
- now, prioritize the queue by \( d(v) \times \hat{h}(v) \)
- can stop when target gets popped -- why?
- optimal subpaths should pop earlier than non-optimal
  - \( d(v) \times \hat{h}(v) \leq d(v) \times h(v) \leq d(t) \leq \) non-optimal paths of \( t \)
How to design a heuristic?

• more of an art than science
• basic idea: projection into coarser space
• cluster: \[ w'(U, V) = \min \{ w(u, v) \mid u \in U, v \in V \} \]
• exact cost in coarser graph is estimate of finer graph

Viterbi or A*?

• A* intuition: \( d(t) \otimes \hat{h}(t) \) ranks higher among \( d(v) \otimes \hat{h}(v) \)
• can finish early if lucky
• actually, \( d(t) \otimes \hat{h}(t) = d(t) \otimes h(t) = d(t) \otimes \bar{I} = d(t) \)
• with the price of maintaining priority queue - \( O(\log V) \)
• Q: how early? worth the price?
• if the rank is \( r \), then A* is better when \( \frac{r}{V \log V} < 1 \)
Two Dimensional Survey

<table>
<thead>
<tr>
<th>Search Space</th>
<th>Topological (acyclic)</th>
<th>Best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphs with semirings (e.g., FSMs)</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>Hypergraphs with weight functions (e.g., CFGs)</td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>

Background: CFG and Parsing

- For each diff \( (\leq n) \)
  - For each rule \( X \rightarrow Y Z \)
  - For each split point \( k \)
    \[
    \text{score}[X][i][j] = \max \text{score}[X][i][k], \text{score}(X\rightarrow YZ) * \text{score}[Y][i][k] * \text{score}[Z][k][j]
    \]
Background: CFG and Parsing

- For each diff (<= n)
  - For each i (<= n)
    - For each rule X \rightarrow Y Z
      - For each split point k
        \[ \text{score}[X][i][j] = \max \text{ score}[X][i][j], \]
        \[ \text{score}(X \rightarrow Y Z) * \]
        \[ \text{score}[Y][i][k] * \]
        \[ \text{score}[Z][k][j] \]

\[ (S, 0, n) \]

(Directed) Hypergraphs

- a generalization of graphs
- edge \rightarrow hyperedge: several vertices to one vertex
- \( e = (T(e), h(e), f_e) \). arity \( |e| = |T(e)| \)
- a totally-ordered weight set \( R \)
  - we borrow the \( \oplus \) operator to be the comparison
  - weight function \( f_e : R^{|e|} \rightarrow R \)
- generalizes the \( \otimes \) operator in semirings

\[ \text{simple case: } f_e(a, b) = a \otimes b \otimes w(e) \]

\[ d(v) \oplus = f_e(d(u_1), d(u_2)) \]
Hypergraphs and Deduction

\begin{align*}
(B, i, k) & \quad (C, k, j) \\
\begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b \\
\mathbf{v} : a \times b \times \Pr(A \rightarrow B C)
\end{array} \\
\mathbf{f}_e
\end{array} & \quad \begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b
\end{array} & \quad \mathbf{v} : \mathbf{f}_e(a, b)
\end{array}
\end{align*}

(A, i, j)

\begin{align*}
(A, i, j) & \quad (C, k, j) \\
\begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b
\end{array} & \quad \mathbf{v} : a \times b \times \Pr(A \rightarrow B C)
\end{array} & \quad \begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b
\end{array} & \quad \mathbf{v} : \mathbf{f}_e(a, b)
\end{array}
\end{align*}

\begin{align*}
(B, i, k) & \quad (C, k, j) \\
\begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b
\end{array} & \quad \mathbf{v} : a \times b \times \Pr(A \rightarrow B C)
\end{array} & \quad \begin{array}{c}
\begin{array}{c}
\mathbf{u}_1 : a \\
\mathbf{u}_2 : b
\end{array} & \quad \mathbf{v} : \mathbf{f}_e(a, b)
\end{array}
\end{align*}

\begin{align*}
(A, i, j) & \quad (C, k, j)
\end{align*}

\begin{align*}
(A, i, j) & \quad (C, k, j)
\end{align*}

(Nederhof, 2003)

Related Formalisms

<table>
<thead>
<tr>
<th>hypergraph</th>
<th>AND/OR graph</th>
<th>context-free grammar</th>
<th>deductive system</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertex</td>
<td>OR-node</td>
<td>symbol</td>
<td>item</td>
</tr>
<tr>
<td>source-vertex</td>
<td>leaf OR-node</td>
<td>terminal</td>
<td>axiom</td>
</tr>
<tr>
<td>target-vertex</td>
<td>root OR-node</td>
<td>start symbol</td>
<td>goal item</td>
</tr>
<tr>
<td>hyperedge</td>
<td>AND-node</td>
<td>production</td>
<td>instantiated deduction</td>
</tr>
<tr>
<td>{(u_1, u_2), v, f}</td>
<td>OR-nodes</td>
<td></td>
<td>\frac{u_1 : a \quad u_2 : b}{v : f(a, b)}</td>
</tr>
</tbody>
</table>

Liang Huang (Penn) 39 Dynamic Programming

Liang Huang (Penn) 40 Dynamic Programming
Packed Forests

- a compact representation of many parses
- by sharing common sub-derivations
- polynomial-space encoding of exponentially large set

(Klein and Manning, 2001; Huang and Chiang, 2005)

Weight Functions and Semirings

\[
d(u) \xrightarrow{w(e)} d(u) \otimes w(e)
\]

\[
d(u) \xrightarrow{f_e} f_e(d(u))
\]

\[
f_e(a_1, \ldots, a_k) = a_1 \otimes \ldots \otimes a_k \otimes w(e)
\]

can also extend monotoncity and superiority to general weight functions
Generalizing Semiring Properties

- monotonicity
  - semiring: \( a \leq b \Rightarrow a \times c \leq b \times c \)
  - for all weight function \( f \), for all \( a_1 \ldots a_k \), for all \( i \), if \( a_i' \leq a_i \) then \( f(a_1 \ldots a_i' \ldots a_k) \leq f(a_1 \ldots a_i \ldots a_k) \)

- superiority
  - semiring: \( a \leq a \times b, \quad b \leq a \times b \)
  - for all \( f \), for all \( a_1 \ldots a_k \), for all \( i \), \( a_i \leq f(a_1, \ldots, a_k) \)

- acyclicity
  - degenerate a hypergraph back into a graph

Two Dimensional Survey

<table>
<thead>
<tr>
<th>search space</th>
<th>topological (acyclic)</th>
<th>best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>
Viterbi Algorithm for DAGs

1. topological sort
2. visit each vertex $v$ in sorted order and do updates
   - for each incoming edge $(u, v)$ in $E$
   - use $d(u)$ to update $d(v)$:
   - key observation: $d(u)$ is fixed to optimal at this time
     \[
     d(v) \oplus = d(u) \otimes w(u, v)
     \]
   - time complexity: $O(V + E)$

Viterbi Algorithm for DAHs

1. topological sort
2. visit each vertex $v$ in sorted order and do updates
   - for each incoming hyperedge $e = ((u_1, \ldots, u_{|e|}), v, f_e)$
   - use $d(u_i)$'s to update $d(v)$
   - key observation: $d(u_i)$'s are fixed to optimal at this time
     \[
     d(v) \oplus = f_e(d(u_1), \ldots, d(u_{|e|}))
     \]
   - time complexity: $O(V + E)$ (assuming constant arity)
Example: CKY Parsing

- parsing with CFGs in Chomsky Normal Form (CNF)
- typical instance of the generalized Viterbi for DAHs
- many variants of CKY ~ various topological ordering

- For each diff ($<= n$)
  - For each $i$ ($<= n$)
    - For each rule $X \rightarrow Y Z$
      - For each split point $k$
        \[
        \text{score}[X][i][j] = \max
        \]

$O(n^3 |P|)$

Example: CKY Parsing

- parsing with CFGs in Chomsky Normal Form (CNF)
- typical instance of the generalized Viterbi for DAHs
- many variants of CKY ~ various topological ordering

$O(n^3 |P|)$
Example: Syntax-based MT

- synchronous context-free grammars (SCFGs)
- context-free grammar in two dimensions
- generating pairs of strings/trees simultaneously
- co-indexed nonterminal further rewritten as a unit

\[
\begin{align*}
\text{VP} & \rightarrow \text{PP}^{(1)} \text{ VP}^{(2)}, \\
\text{VP} & \rightarrow \text{ju xing le huitan, } \text{held a meeting} \\
\text{PP} & \rightarrow \text{yu Shalong, } \text{with Sharon}
\end{align*}
\]

Translation as Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
- build the corresponding target sub-strings in parallel

\[
\begin{align*}
\text{VP} & \rightarrow \text{PP}^{(1)} \text{ VP}^{(2)}, \\
\text{VP} & \rightarrow \text{ju xing le huitan, } \text{held a talk with Sharon} \\
\text{PP} & \rightarrow \text{yu Shalong, }
\end{align*}
\]

complexity: same as CKY parsing -- $O(n^3)$
Adding a Bigram Model

\[
\text{held ... talk with ... Sharon}
\]

\[
\text{VP}_1, 6
\]

\[
\text{with ... Sharon along ... Shalong with ... Shalong}
\]

\[
\text{VP}_3, 6
\]

\[
\text{PP}_1, 3
\]

complexity: \(O(n^3 V^{4(m-1)})\)

Two Dimensional Survey

<table>
<thead>
<tr>
<th>search space</th>
<th>topological (acyclic)</th>
<th>best-first (superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td>Viterbi</td>
<td>Generalized Viterbi</td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td></td>
<td>Dijkstra</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knuth</td>
</tr>
</tbody>
</table>
Viterbi Algorithm for $\text{DAHs}$

1. topological sort
2. visit each vertex $v$ in sorted order and do updates
   - for each incoming *hyperedge* $e = ((u_1, ..., u_{|e|}), v, f_e)$
   - use $d(u_i)$’s to update $d(v)$
   - key observation: $d(u_i)$’s are fixed to optimal at this time
     \[ d(v) \oplus = f_e(d(u_1), \cdots, d(u_{|e|})) \]
   - time complexity: $O(V + E)$ (assuming constant arity)

---

**Forward Variant for $\text{DAHs}$**

1. topological sort
2. visit each vertex $v$ in sorted order and do updates
   - for each *outgoing* hyperedge $e = ((u_1, ..., u_{|e|}), h(e), f_e)$
   - if $d(u_i)$’s have all been fixed to optimal
     - use $d(u_i)$’s to update $d(h(e))$
   - time complexity: $O(V + E)$

---

Q: *how to avoid repeated checking?*
maintain a counter $r[e]$ for each $e$: how many tails yet to be fixed?
fire this hyperedge only if $r[e]=0$
Dijkstra Algorithm

- keep a cut \((S : V - S)\) where \(S\) vertices are fixed
- maintain a priority queue \(Q\) of \(V - S\) vertices
- each iteration choose the best vertex \(v\) from \(Q\)
- move \(v\) to \(S\), and use \(d(v)\) to forward-update others

\[
d(u) \oplus = d(v) \otimes w(v, u)
\]

Time complexity:
- \(O((V+E) \lg V)\) (binary heap)
- \(O(V \lg V + E)\) (fib. heap)

Knuth (1977) Algorithm

- keep a cut \((S : V - S)\) where \(S\) vertices are fixed
- maintain a priority queue \(Q\) of \(V - S\) vertices
- each iteration choose the best vertex \(v\) from \(Q\)
- move \(v\) to \(S\), and use \(d(v)\) to forward-update others

Time complexity:
- \(O((V+E) \lg V)\) (binary heap)
- \(O(V \lg V + E)\) (fib. heap)
Example: Best-First/A* Parsing

- Knuth for parsing: best-first (Caraballo & Charniak, 1998)
- further speed-up: use A* heuristics
  - showed significant speed up with carefully designed heuristic functions (Klein and Manning, 2003)
- heuristic function: an estimate of outside cost

Outside Cost in Hypergraph

- outside cost: yet to pay to reach goal
- let’s only consider semiring-composed case
  - and only acyclic hypergraphs
- after computing $d(v)$ for all $v$ from bottom-up
- backwards Viterbi from top-down (outside-in)

$$h(S_{0,n}) = \bar{1}$$

$$h(v) \oplus = h(u) \otimes w(e) \otimes d(v')$$
Projection-based Heuristics

- how to guess? project onto a coarser-grained space
- and parse with the coarser grammar
- outside cost of of the coarser item as heuristics

(Klein and Manning, 2003)
Projection-based Heuristics

- how to guess? project onto a coarser-grained space
- and parse with the coarser grammar
- outside cost of of the coarser item as heuristics

\[ \hat{h} \left( VBD_{2,3} \right) = h' \left( V_{2,3} \right) \]

(Klein and Manning, 2003)

Analogy with Graphs
More on Coarse-to-Fine

- multilevel coarse-to-fine A*
- heuristic = exact outside cost in
- $\hat{h}_i(v) = h_{i-1}(\text{proj}_{i-1}(v))$
- $VBD>V>X$. $\hat{h}_i(VBD_{1,5}) = h_{i-1}(V_{1,5})$; $\hat{h}_{i-1}(V_{1,5}) = h_{i-2}(X_{1,5})$
- multilevel coarse-to-fine Viterbi w/ beam-search
- Viterbi + beam pruning in each stage
- prune according to merit: $d(v) \odot h(v) \odot d(\text{TOP})$
- hard to derive a provably correct threshold
- in practice: use a preset threshold (but works well!)

Same Picture Again

monotonic optimization problems

acyclic: Viterbi

Many NLP problems

Superior: Knuth

Inside-Outside Alg. (Inside semiring)

non-prob. (discriminative) parsing

PCFG parsing with CNF

cyclic grammars

generalized Bellman-Ford (open)
Take Home Message

- Dynamic Programming is cool, easy, and universal!
- two frameworks and two types of algorithms
  - monotonicity; acyclicity and/or superiority
- topological (Viterbi) vs. best-first style (Dijkstra/Knuth/A*)
  - when to choose which: A* can finish early if lucky
- graph (lattice) vs. hypergraph (forest)
  - incremental, finite-state vs. branching, context-free
- covered many typical NLP applications
- a better understanding of theory helps in practice

Liang Huang (Penn)

Thanks!

Questions? Comments?

final slides will be available on my website.