8.0 Search Algorithms for Speech Recognition

References: 1. 12.1-12.5 of Huang, or

2. 7.2-7.6 of Becchetti, or
3. 5.1-5.7, 6.1-6.5 of Jelinek
4. " Progress in Dynamic Programming Search for LVCSR (Large Vocabulary Continuous Speech Recognition)", Proceedings of the IEEE, Aug 2000
5. 4.7 up to 4.7.3 of Rabiner and Juang

DTW and Dynamic Programming

- **Dynamic Time Warping (DTW)**
  - well accepted pre-HMM approach
  - find an optimal path for matching two templates with different length
  - good for small-vocabulary isolated-word recognition even today
- **Test Template [y_j, j=1,2,...N] and Reference Template [x_i, i=1,2,..M]**
  - warping both templates to a common length L
    warping functions: $f_x(i)= m$, $f_y(j)= n$, m, n=1,2,...L
  - endpoint constraints: $f_x(1)= f_y(1)= 1$, $f_x(M)= f_y(N)= L$
    monotonic constraints: $f_x(i+1) \geq f_x(i)$, $f_y(j+1) \geq f_y(j)$
  - recursive relationship:

$$D(m,n)=\min_{(m',n')}\{D(m',n')+\bar{d}[(m',n');(m,n)]\} \quad \text{, D(m,n) : accumulated distance up to (m,n)}$$

$$\bar{d}[(m',n');(m,n)] = \sum_{(m',n')\rightarrow(m,n)} d(f_x^{-1}(k), f_y^{-1}(l))\, w(\Delta i, \Delta j)$$

$\mathrm{d}\,(\mathrm{i},\mathrm{j}) = $ distance measure for x_i and y_j

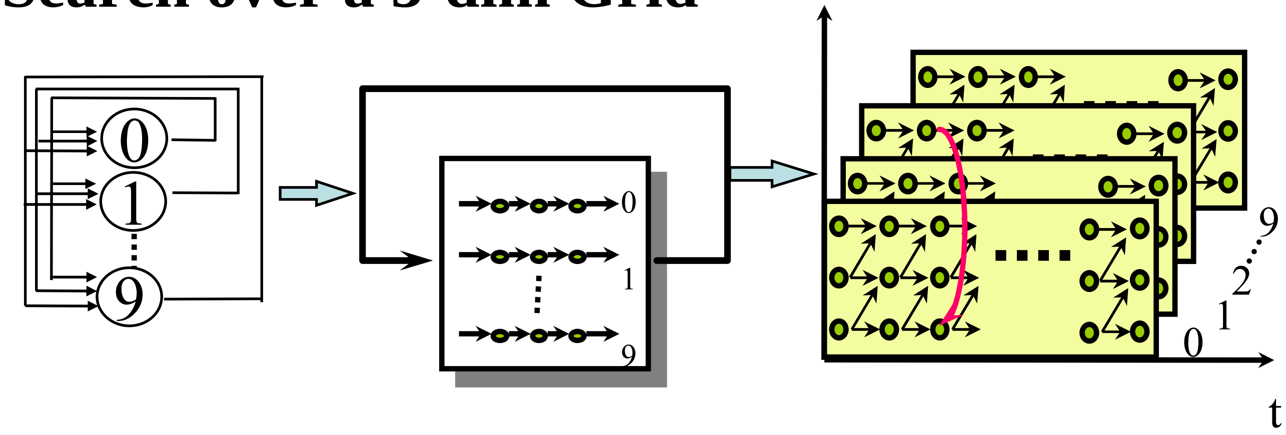
summation over all moves from (m', n') to (m, n)

$\mathrm{w}\,(\Delta\mathrm{i}, \Delta\mathrm{j})$: weights for different types of moves

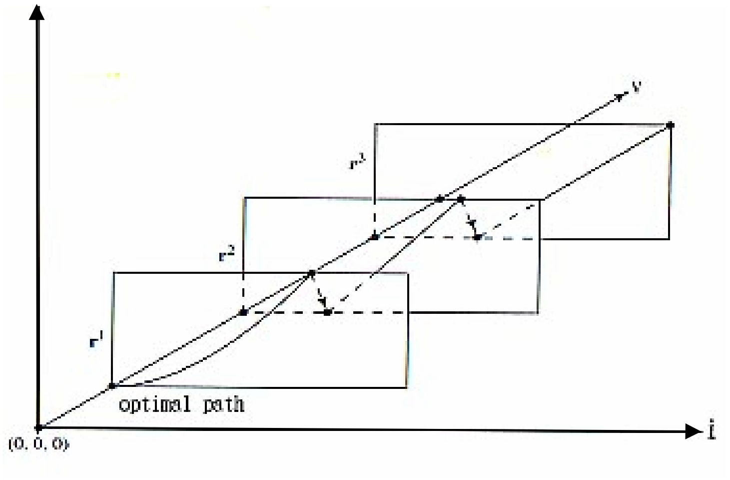
  - global constraints/local constraints
  - lack of a good approach to train a good reference pattern
- **Dynamic Programming Algorithm**
- **Search Algorithm within a Given HMM—Viterbi Algorithm in Basic Problem 2**
  - application example: isolated word recognition

Continuous Speech Recognition Example: Digit String Recognition— One-stage Search

- Unknown Number of Digits
- No Lexicon/Language Model Constraints
- Search over a 3-dim Grid

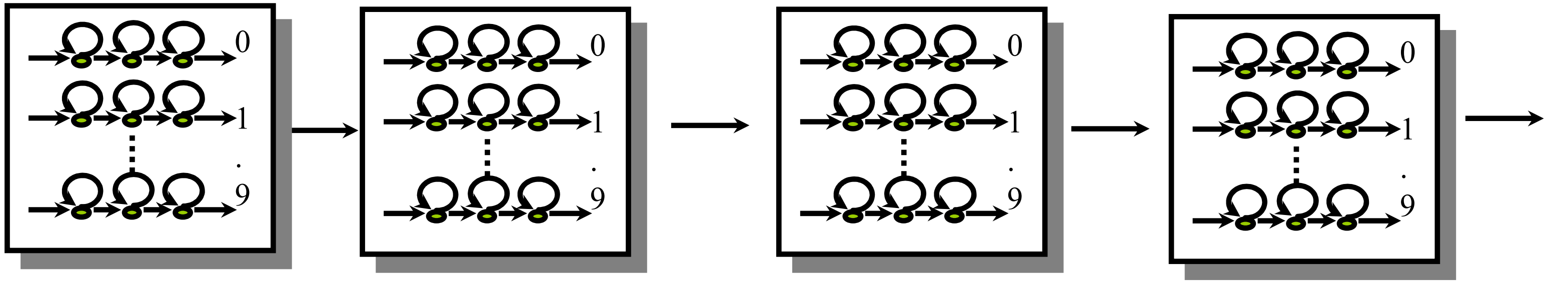


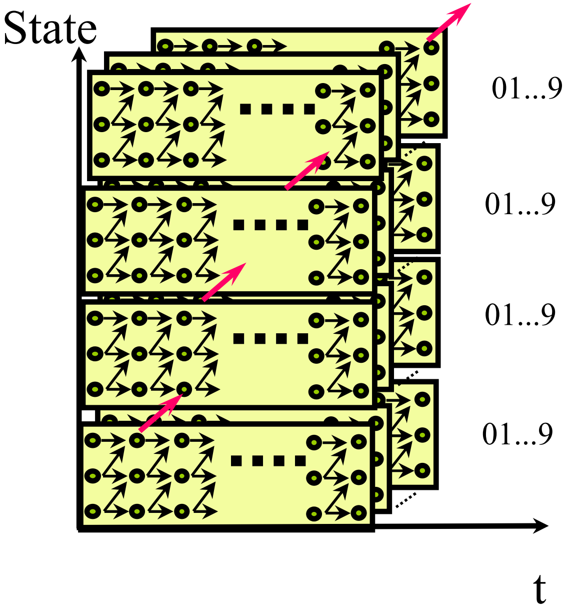
- Switched to the First State of the Next Model at the End of the Previous Model
- May Result with Substitution, Deletion and Insertion



Continuous Speech Recognition Example: Digit String Recognition — Level-Building

- **Known Number of Digits**
- **No Lexicon/Language Model Constraints**
- **Higher Computation Complexity, No Deletion/Insertion**





- number of levels = number of digits in an utterance
- automatic transition from the last state of the previous model to the first state of the next model

Time (Frame)- Synchronous Viterbi Search for Large-Vocabulary Continuous Speech Recognition

- **MAP Principle**

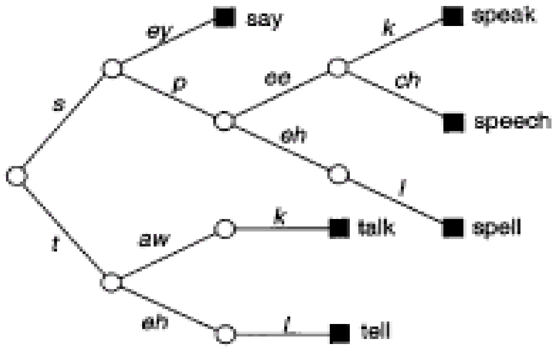
$$W^* = \arg\max_W [p(W|X)] = \arg\max_W [\frac{p(X|W)p(W)}{p(X)}] = \arg\max_W [\underset{\substack{\uparrow \\ \text{from HMM}}}{p(X|W)}\,\underset{\substack{\uparrow \\ \text{from Language Model}}}{p(W)}]$$

$$p(X|W) = \sum_{all\ \bar{q}} p(X, \bar{q}|W), \bar{q} : a\ state\ \ sequence$$

- **An Approximation**

$$W^* = \arg\max_W [p(W) \sum_{all\ \bar{q}} p(X, \bar{q}|W)] \cong \arg\max_W [p(W) \cdot \max_{\bar{q}} p(X, \bar{q}|W)]$$

  - the most likely word sequence is approximated by the most likely state sequence
  - Viterbi search, a sub-optimal approach
- **Viterbi Search—Dynamic Programming**
  - replacing the problem by a smaller sub-problem and formulating an iterative procedure
  - time (frame)- synchronous: the best score at time t is updated from all states at time t-1
- **Tree Lexicon as the Basic Working Structure**



  - each arc is an HMM
  - each leaf node is a word
  - search processes for a segment of utterance through some common units for different words can be shared
  - the same tree copy reproduced at each leaf node in principle

Time (Frame)- Synchronous Viterbi Search for Large –Vocabulary Continuous Speech Recognition

- **Define Key Parameters**

  D (t, q_t, w) : objective function for the best partial path ending at time t in state q_t for the word w

  h (t, q_t, w) : backtrack pointer for the previous state at the pervious time when the best partial path ends at time t in state q_t for the word w

- **Intra-word Transition—HMM only, no Language Model**

$$D(t,q_t,w)=\max_{q_{t-1}}[d(o_t,q_t|q_{t-1},w)+D(t-1,q_{t-1},w)]$$

$$d(o_t,q_t|q_{t-1},w)=\log p(o_t|q_t,w)+\log p(q_t|q_{t-1},w)$$

$$\overline{q}(t,q_t,w)=\arg\max_{\mathrm{q}_{\mathrm{t-1}}}\left[d(o_t,q_t|q_{t-1},w)+D(t-1,q_{t-1},w)\right]$$

$$h(t,q_t,w)=\overline{q}(t,q_t,w)$$

- **Inter-word Transition—Language Model only, no HMM (bi-gram as an example)**

$$D(t,Q,w)=\max_{v}[\log p(w|v)+D(t,q_f(v),v)]$$

Q : a pseudo initial state for the word w

$q_f(v)$: the final state for the word v

$\overline{v}$: $\arg\max_{v}[\log p(w|v)+D(t,q_f(v),v)]$

$$\mathrm{h}(t,Q,w)=q_f(\overline{v})$$

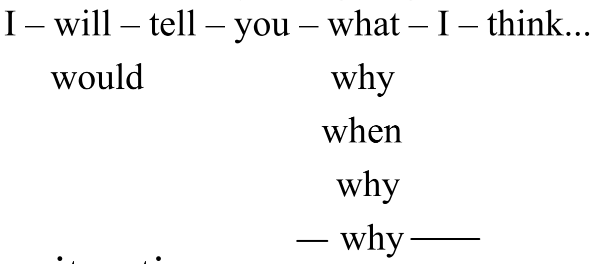
Time (Frame)- Synchronous Viterbi Search for Large-Vocabulary Continuous Speech Recognition

- **Beam Search**
  - at each time t only a subset of promising paths are kept
  - example 1: define a beam width L (i.e. keeping only L paths at each time)
    example 2: define a threshold Th (i.e. all paths with $D < D_{max,t}$-Th are deleted)
  - very helpful in reducing the search space
- **Other Pruning Approaches**
  - by acoustic scores, language model scores, etc.
- **N-best List and Word Graph (Lattice)**
  - decouple the complicated search process into simpler processes
  - the first primarily by acoustic scores (HMMs), the second by language models for example

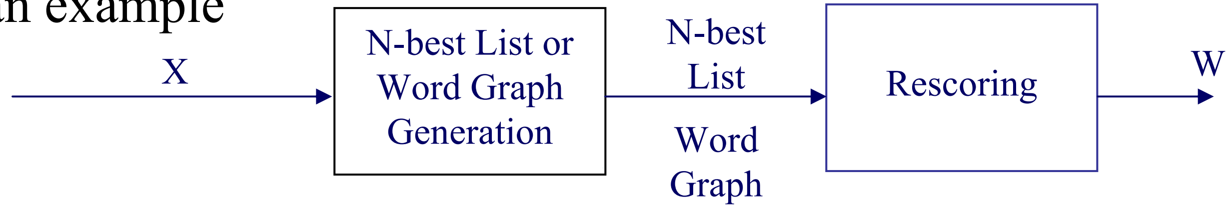
I will tell you what I think....

I will tell you why I think....

I will tell you when I think....


I – will – tell – you – what – I – think...
would why
when
why
— why ——

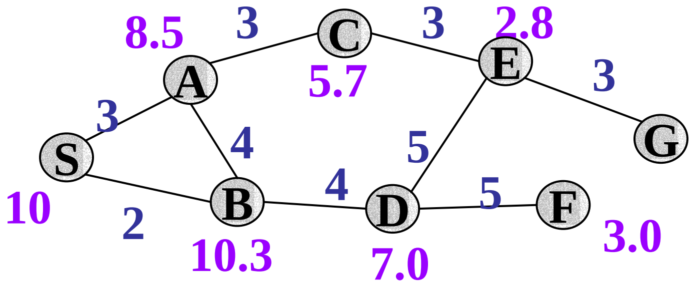
  - similarly constructed with dynamic programming iterations
- **Multi-pass Search**
  - an example


X
N-best List or Word Graph Generation
N-best List
Word Graph
Rescoring
W

  - use less knowledge or less constraints in the first stage, etc.

Some Search Algorithm Fundamentals

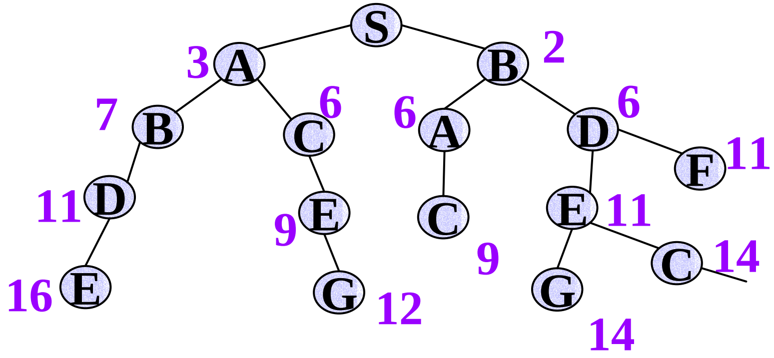
- **An Example – a city traveling problem**



S: starting
G: goal

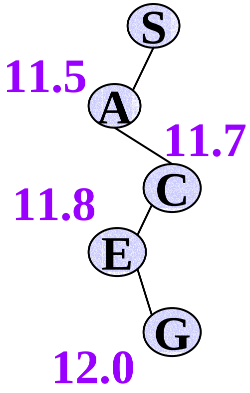
– to find the minimum distance path

- **Search Tree(Graph)**



- **Blind Search Algorithms**
  - Depth-first Search: pick up an arbitrary alternative and proceed
  - Breath-first Search: consider all nodes on the same level before going to the next level
  - no sense about where the goal is

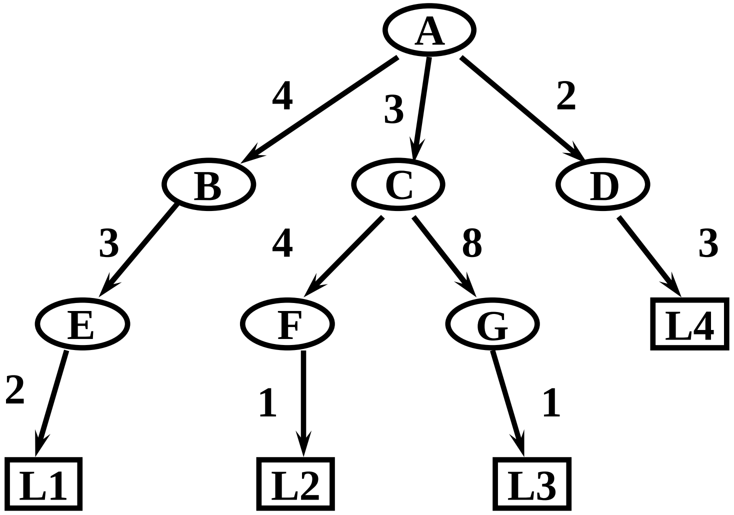
- **Heuristic Search**
  - Best-first Search
  - based on some knowledge, or “heuristic information”
    f(n) = g(n)+h*(n)
    g(n): distance up to node n
    h*(n): heuristic estimate for the remaining distance up to G
  - heuristic pruning



$h^*(n)$:
straight-line
distance

Heuristic Search: Another Example

- **Problem: Find a path with the highest score from root node “A” to some leaf node (one of “L1”,”L2”,”L3”,”L4”)**



$$f(n) = g(n) + h^*(n)$$

$g(n)$: score from root node to node n

$h(n)$: exact score from node n to a specific leaf node

$h^*(n)$: estimated value for h(n)

Node	**g(n)**	**h*(n)**	**f(n)**
A	0	15	15
B	4	9	13
C	3	12	15
D	2	5	7
E	7	4	11
F	7	2	9
G	11	3	14
L1	9	0	9
L2	8	0	8
L3	12	0	12
L4	5	0	5

List of Candidate Steps

Top	**Candidate List**
A(15)	A(15)
C(15)	C(15), B(13), D(7)
G(14)	G (14), B(13), F(9), D(7)
B(13)	B(13), L3(12), F(9), D(7)
L3(12)	L3 (12), E(11), F(9), D(7)

A* Search and Speech Recognition

- **Admissibility**
  - a search algorithm is admissible if it is guaranteed that the first solution found is optimal, if one exists (for example, beam search is NOT admissible)
- **It can be shown**
  - the heuristic search is admissible if

$$h^*(n) \geq h(n) \text{ for all n with a highest-score problem}$$

  - A^* search when the above is satisfied
- **Procedure**
  - Keep a list of next-step candidates, and proceed with the one with the highest f(n) (for a highest-score problem)
- **A^* search in Speech Recognition**
  - example 1: estimated average score per frame as the heuristic information

$$s_f = [\log P(\bar{o}_{i,j} \mid \bar{q}_{i,j})]/(j-i+1)$$

$$\bar{o}_{i,j} : \text{observations from frame i to j}, \ \bar{q}_{i,j} : \text{state sequences from frame i to j}$$

$$\text{estimated with many (i, j) pairs from training data}$$

$$\text{h}^*(n) \text{ obtained from Max}[s_f], \text{Ave}[s_f], \text{Min}[s_f] \text{ and (T - t)}$$

  - example 2: use of weak constraints in the first pass to generate heuristic estimates in multi-pass search