

# Basic Search Algorithms for Speech Recognition

Professor Marie Roch  
San Diego State University

Huang et. al. 12.1-12.1.2,12.2-12.4

## Problem

- Want to maximize

$$\hat{W} = \arg \max_w \Pr(X | W) \Pr(W)$$

- How can we find  $W$  in a reasonable manner?

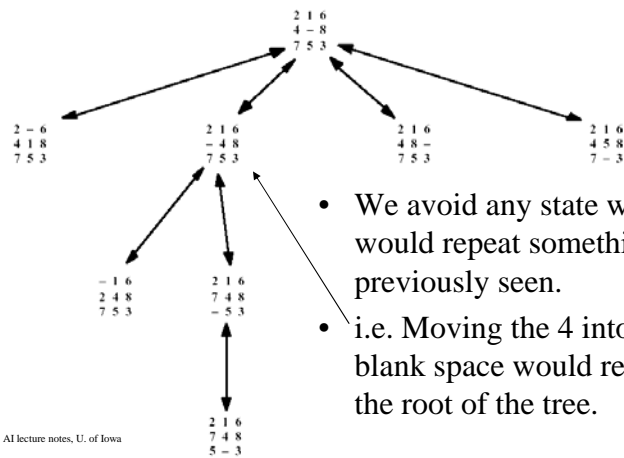
# Search

- A search is a systematic exploration of a state space.
- Example: The eight puzzle.

2	1	6
4		8
7	5	3

Oden, AI lecture notes, U. of Iowa

# Exploration by depth

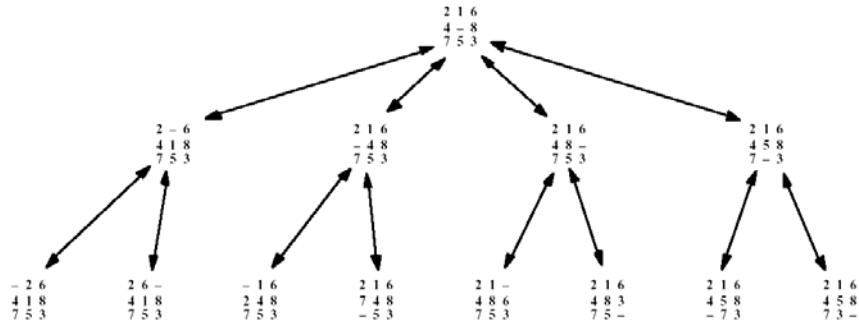


- We avoid any state which would repeat something previously seen.
- i.e. Moving the 4 into the blank space would repeat the root of the tree.

Oden, AI lecture notes, U. of Iowa

## Exploration by breadth

- Similar restrictions to avoid repeated states.

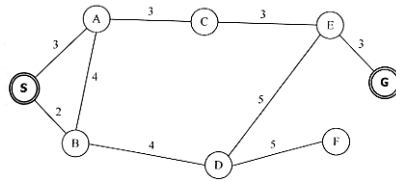


## Exploring search spaces

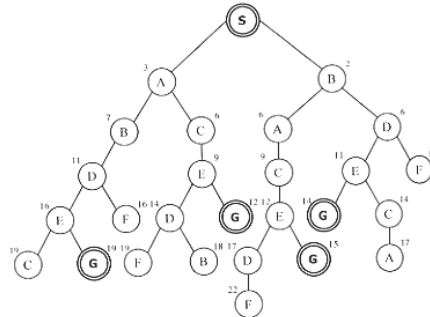
- When deciding how to search, we have several choices to make:
  - breadth vs. depth search
  - blind vs. heuristic search
  - optimal path vs. first/any path
- ASR algorithms typically use breadth or heuristic search, we'll cover breadth.

## Example: Traveling from city A → G

- Graph with costs
- Search space



Huang et al. p. 593



Huang et al. p. 594



7

## Breadth-first search

```

Queue = { start-node }

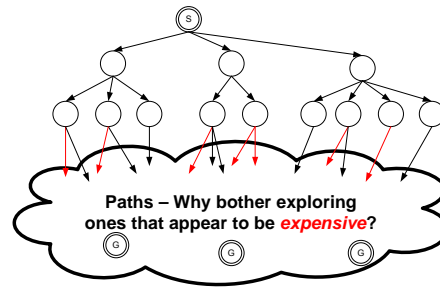
while Queue is not empty {
  node = Remove first node from Queue.
  If node is a goal node, exit loop
  Determine successors of node excluding its ancestors
  For each successor
    find cost of to each successor:
      cost to(node) + cost edge(node, successor)
  Merge successor nodes into Queue sorted by cost
}
if node is a goal node, the search was successful.
  
```



8

# Pruning

- Some partial paths not worth pursuing.
- Elimination of these paths called pruning.
- Likely to result in an inadmissible search (doesn't find optimal path).

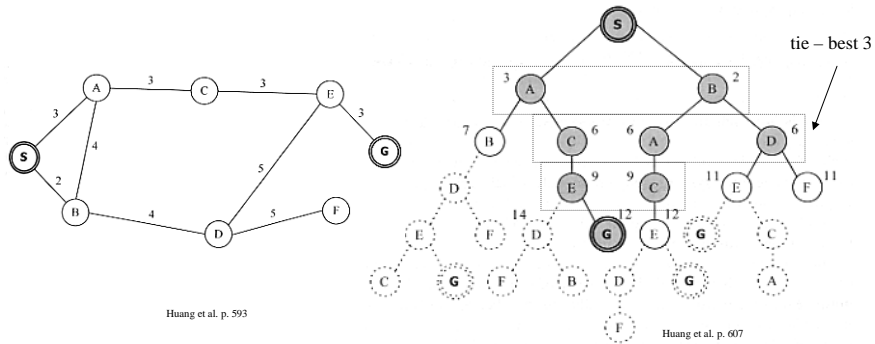


# Our searches

- Beam search
  - Breadth-first search with pruning.
  - Search may fail and is inadmissible.
- Questions about breadth-first search:
  - Admissible?
  - Can it fail?

## Beam search city travel – width 2

- Shaded nodes added to Queue
- Clear nodes and their dotted descendants have been pruned



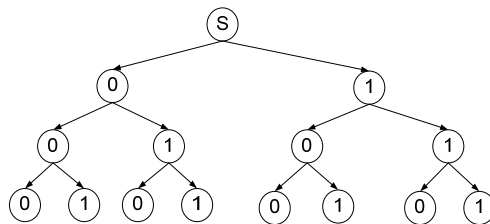
## Search & speech recognition

- Recall that our goal is to determine a sequence of words:

$$\begin{aligned}
 \hat{W} &= \arg \max_w \Pr(W | X) \\
 &= \arg \max_w \frac{\Pr(X | W) \Pr(W)}{\Pr(X)} \\
 &= \arg \max_w \Pr(X | W) \Pr(W)
 \end{aligned}$$

## Search & speech recognition

- We can think of this as a search of a state-space with a branch for at each word:



- Search space for spoken binary digits.

## Probabilities & cost

- Searches
  - Look for lowest cost path
  - Problem: Want highest probability path

- Solution:

$$C(Z) = \log \frac{1}{\Pr(Z)} = -\log \Pr(Z)$$

## Probabilities & cost

Thus

$$\begin{aligned}C(W | X) &= -\log(\Pr(W | X)) \\ &= -\log\left(\frac{\Pr(X | W)\Pr(W)}{\Pr(X)}\right) \\ &= -\log(\Pr(X | W)) - \log(\Pr(W)) + \log(\Pr(X)) \\ &= C(X | W) + C(W) - C(X)\end{aligned}$$

## Probabilities & cost

$C(X)$  is constant across all hypotheses and can be discarded:

$$C(W | X) \propto C(X | W) + C(W)$$

resulting in the minimization problem

$$\hat{W} = \arg \min_w C(W | X) = \arg \min_w [C(X | W) + C(W)]$$

## Acoustic & language models

- Problems with combination:
  - Continuous HMMs likelihood scores are not probabilities
  - Acoustic models tend to underestimate the probability (simplifying assumptions: Markov, independence)
  - Dynamic ranges very different

## Acoustic & language models

- As a consequence, a language model weight  $LW$  is introduced:

$$[\text{Pr}(W)]^{LW} \quad LW > 1$$

- Brings language model probabilities more in line with those produced by HMMs.

## Insertion penalties

- Language model scores reduced by language model weight. 👉
- Each new word doesn't cost as much any more... 👈 leads to tendency to add words (insertion errors)
- Address by using a fixed per word Pr

$$IP^{N(W)}, IP \in [0,1]$$

## Revised language model

- Thus, the total contribution of the language model becomes:

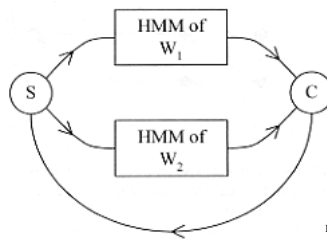
$$\Pr(W)^{LW} IP^{N(W)}$$

translating to a cost of:

$$\begin{aligned} C(W) &= -\log[\Pr(W)^{LW} IP^{N(W)}] \\ &= -LW \log \Pr(W) - N(W) \log IP \end{aligned}$$

# Continuous speech recognition

- Simple example:
  - Two words with uniform language model.
  - Use null transitions to add start & collector states to word models to create:



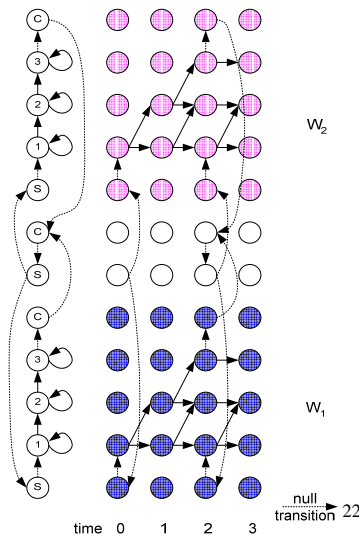
Huang et al. p. 611



21

# Decoding continuous speech

- Partial trellis to time 3.
- When a collector (final) state is reached, null transitions move to the next words.
- Some decoders are time-synchronous: finish time  $t$  before starting  $t+1$

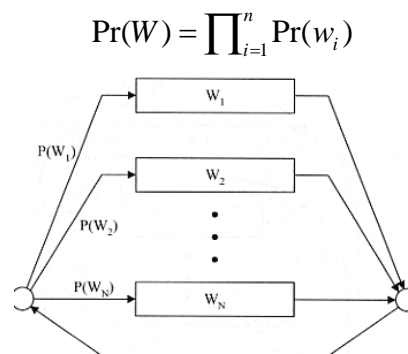


## Language model states

- Given a search of language model states:
  - The network of acoustic models is implicitly defined:
    - Concatenation of sub-word models to form each word.
    - Connections between words.
  - We only need search the language model states and consider the Pr of each word as given by the acoustic model.

## Unigram search space

- Memoryless
- Similar to our previous two word toy example.

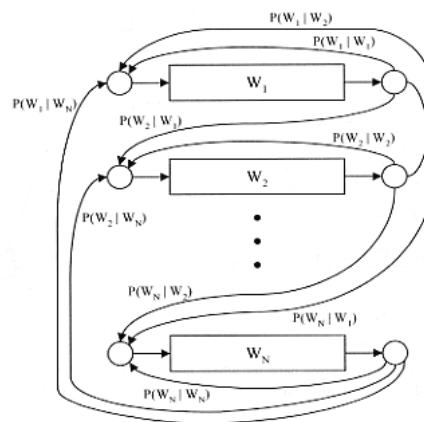


Huang et al. p. 617

## Bigram search space

- Naive representation requires  $N^2$  nodes.
- Total number of states is proportional to  $N$
- Due to the small size, bigrams frequently serve as the first pass in multiple-pass strategies.

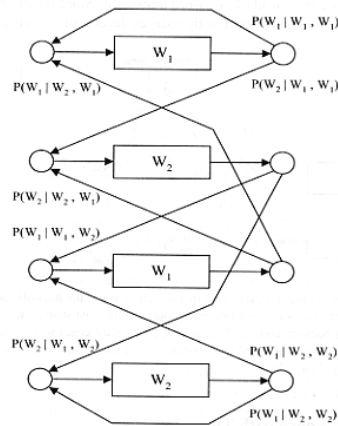
$$\Pr(W) = \Pr(w_1 | < s >) \prod_{i=2}^n \Pr(w_i | w_{i-1})$$



## Trigram search space

$$\Pr(W) = \Pr(w_1 | < s >) \Pr(w_2 | < s >, w_1) \prod_{i=3}^n \Pr(w_i | w_{i-2}, w_{i-1})$$

- As the model depends upon the previous 2 states, the number of grammar states increases to  $N^2$ .
- Graph is generated dynamically do to its size.

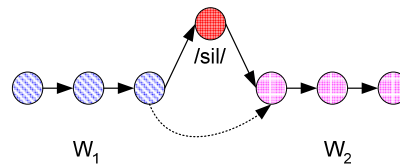


## Silence may be golden, but it still must be accounted for.

- Silence occurs frequently at phrasal boundaries and in general can occur between words.
- The “silence” may consist of non-speech sounds (i.e. breath, lip smacks, etc.) and we construct a general model to recognize these types of events.

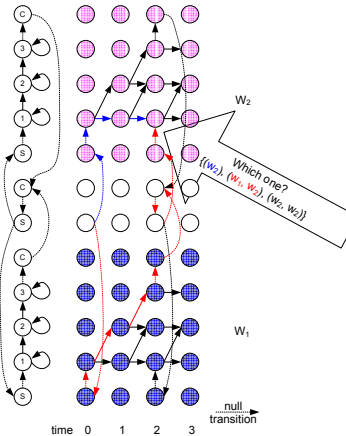
## Silence models

- The silence model is inserted in between every pair of words.
- A null arc permits the skipping of the silence if it does not occur.



## Decoding subtleties

- Recall that the states of our search space are sequences of words.
- Different sequences may imply different states.
- As a consequence, we cannot combine paths that have different word histories.



## Decoding subtleties

- To perform a full decoding of  $\Pr(X|W)$  we would need to extend the forward algorithm with additional bookkeeping to avoid merging incompatible paths.
- One alternative is the time-synchronous Viterbi beam search

## Time-synchronous Viterbi Beam Search

- Provides an efficient way of handling the multiple word sequences by only keeping the best possible word sequence at any given HMM state.
- Time-synchronous vs. time-asynchronous search

$$\frac{\Pr(X | W) \Pr(W)}{\Pr(X)}$$

- synchronous:  $P(X)$  constant & can be ignored
- asynchronous:
  - Length  $X$  inversely proportional to  $P(X)$
  - Normalization required when comparing differing lengths

## Outline of Viterbi beam search

For each time  $t$  {

For each active state of the search graph {

Execute standard Viterbi algorithm when the previous node is from the same word.

Handle as a special case the null transitions which lead into new words

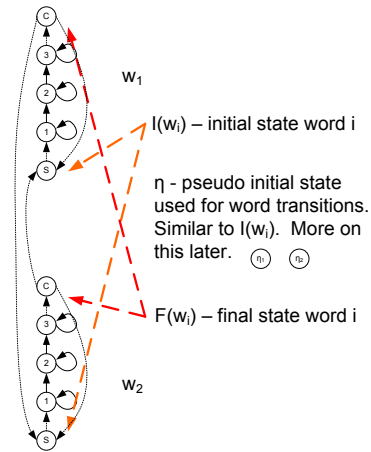
}

Find lowest cost path and prune paths which are outside of the beam.

}

## Viterbi Beam Search Notation

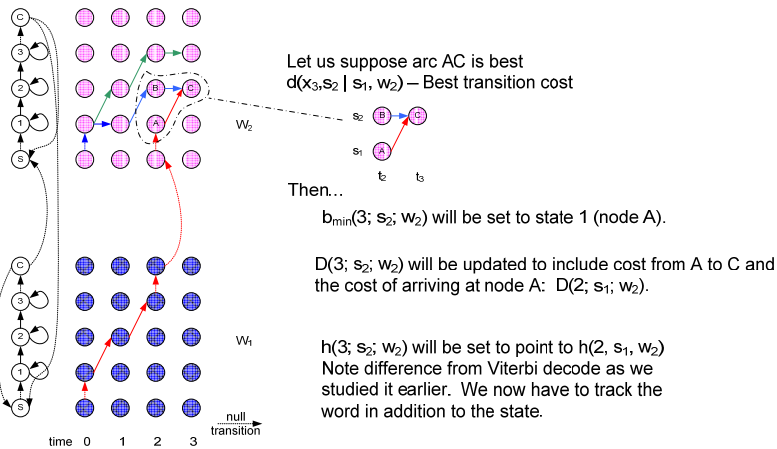
- $D(t; s_t; w)$  – Cost of best path up to time  $t$  ending in state  $s_t$  with grammar word state  $w$ .
- $h(t; s_t; w)$ 
  - State history of  $D(t; s_t; w)$
  - Permits recovery of words and best path (more later)



## Viterbi Beam Search Notation

- $d(x_t, s_t | s_{t-1}, w)$  = Cost of transitioning from  $s_{t-1}$  to  $s_t$  and observing  $x_t$ .
- $b_{\min}(t; s_t; w)$  = Best path backpointer.
  - Simply  $\arg \min d(x_t, s_t | s_{t-1}, w)$  and used for updating the state history  $h(t; s_t; w)$ .

## Illustration – Intra word



## Intra-word transitions

- Find lowest cost transition  $t-1 \rightarrow t$ .

$$D(t; s_t; w) = \min_{s_{t-1}} \{d(x_t, s_t | s_{t-1}; w) + D(t-1; s_{t-1}; w)\}$$

- Word history becomes that of the lowest cost transition.

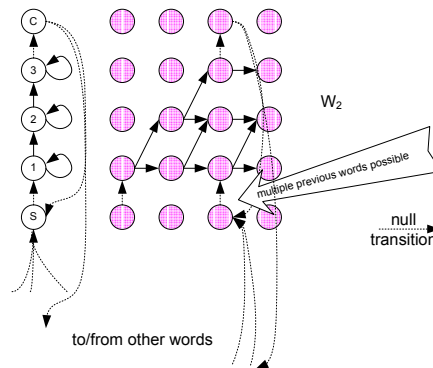
$$b_{\min}(t; s_t; w) = \arg \min_{s_{t-1}} \{d(x_t, s_t | s_{t-1}; w) + D(t-1; s_{t-1}; w)\}$$

- Record the grammar state associated with best partial path.

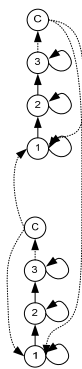
$$h(t; s_t; w) = h(t-1, b_{\min}(t; s_t; w), w)$$

## Inter-word transitions

- When paths enter from a different word, we need to determine which path we should keep and how we will track it.



## Inter-word transitions

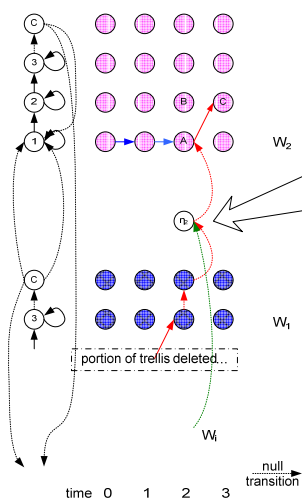


- In the example HMMs
  - Each HMMs start state
    - null transition into state
    - no self transition implies that we cannot have an active path.
  - Not always the case
    - Sample network to the left is valid and problematic.

## Inter-word transitions

- We introduce a pseudo initial state  $\eta$ .
- In the beam search, we will consider all incoming words as if they came into state  $\eta$ .
- Allows us to not have to worry about active paths in the initial state.

## Illustration – Inter word



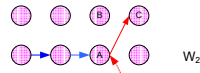
Compute  $D(2, s_1, w_2)$  as normal.

Of the words which are terminating, determine which one is best with pseudo-state  $\eta$ . In this example, we assume that  $w_1$  was the cheapest path.

Check if the cost through the best word is lower than any existing path. If so, use it. Otherwise keep old best path.

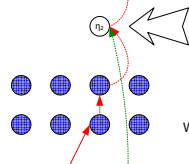
## Inter-word transitions

portion of trellis deleted...



Multiple words may have ended, so we find the best match amongst each terminating word  $v$  and assign this as the cost of each next possible word's pseudo-initial state.

$$D(t; \eta; w) = \min_v \{-\log \Pr(w | v) + D(t; F(v); v)\}$$



Only the best end of word transition will be retained.

portion of trellis deleted...



$$v_{\min} = \arg \min_v \{-\log \Pr(w | v) + D(t; F(v); v)\}$$

time 0 1 2 3 null transition



41

## Inter-word transitions

Take the new state-space and append the old one to maintain a running history.

$$h(t; \eta; w) = \langle v_{\min}, t \rangle :: h(t, F(v_{\min}); v_{\min})$$

$\langle v_{\min}, t \rangle ::$  indicates word  $v_{\min}$  recognized at time  $t$ .



42

## Inter-word transition

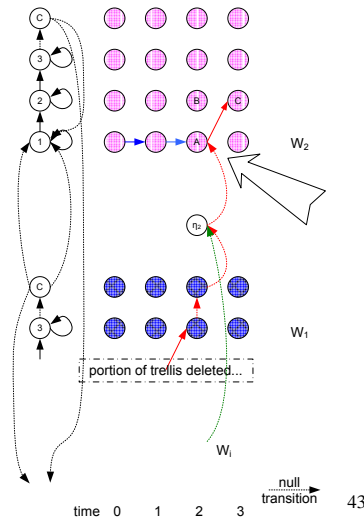
- Only use recognized word if it has lower cost than the best intra-word path

if  $(D(t; \eta; w) < D(t; I(w); w))$

$$D(t; I(w); w) = D(t; \eta; w)$$

$$h(t; I(w); w) = h(t; \eta; w)$$

end



## Viterbi beam search - initialization

- Initialize
  - For each possible starting state, set the cost  $D$  to zero and the word sequence  $h$  empty.

$$D(0; I(w); w) = 0$$

$$h(0; I(w); w) = \text{null}$$

## Viterbi beam search – induction

for t=1:T

for each active intra-word transition

$$D(t; s_t; w) = \min_{s_{t-1}} \{d(x_t, s_t | s_{t-1}; w) + D(t-1; s_{t-1}; w)\}$$

$$h(t; s_t; w) = h(t-1, b_{\min}(t; s_t; w), w)$$

where :

$$b_{\min}(t; s_t; w) = \arg \min_{s_{t-1}} \{d(x_t, s_t | s_{t-1}; w) + D(t-1; s_{t-1}; w)\}$$

end

/\* t loop continues \*/

## Viterbi beam search – induction

/\* continuing for t loop \*/

for each active inter-word transition

$$D(t; \eta; w) = \min_v \{-\log \Pr(w | v) + D(t; F(v); v)\}$$

$$v_{\min} = \arg \min_v \{-\log \Pr(w | v) + D(t; F(v); v)\}$$

$$h(t; \eta; w) = \langle v_{\min}, t \rangle : h(t, F(v_{\min}); v_{\min})$$

if ( $D(t; \eta; w) < D(t; I(w); w)$ )

$$D(t; I(w); w) = D(t; \eta; w)$$

$$h(t; I(w); w) = h(t; \eta; w)$$

end

## Viterbi beam search – induction

/\* continuing for t loop \*/

Find cost of best path and decide beam threshold

Prune unpromising hypotheses

end

## Viterbi beam search – termination

- Pick best path among all final states of grammar at time T.
- Use back pointer to find word sequence.

“Jack be nimble...[Jack the new?!?]”

