Doctoral Course in Speech Recognition

May 2007

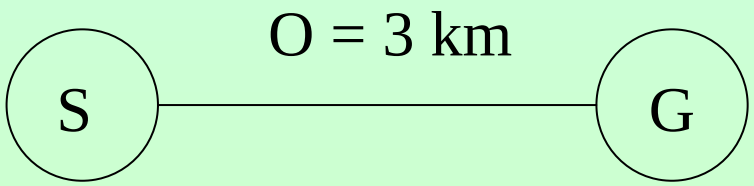
Kjell Elenius

CHAPTER 12

BASIC SEARCH ALGORITHMS

State-based search paradigm

- Triplet S, O, G
  - S, set of initial states
  - O, set of operators applied on a state to generate a transition with corresponding cost to another state
  - G, set of goal states



12.1.1 General Graph Searching Procedures

- Dynamic Programming is powerful but cannot handle all search problems, e.g. NP-hard problems
- NP hard problems
  - Nondeterministic Polynomial-time hard
  - Definition: The complexity class of decision problems that are intrinsically harder than those that can be solved by a deterministic Turing machine in polynomial time.
- Examples
  - The traveling salesman problem
    - Visit all cities once, find shortest distance
  - The 8 Queen problem
    - Place 8 Queens on a chessboard so no-one can capture any of the other

Simplified Salesman Problem

- Find shortest path from S to G

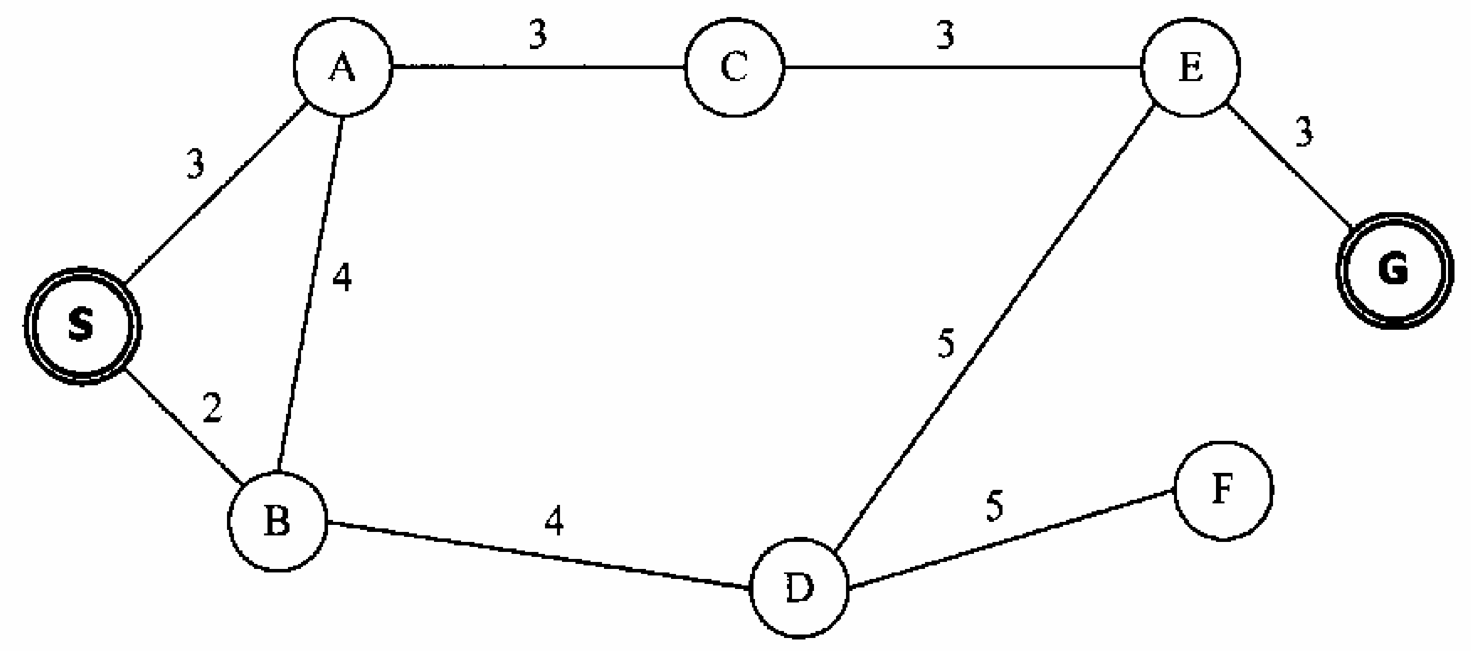
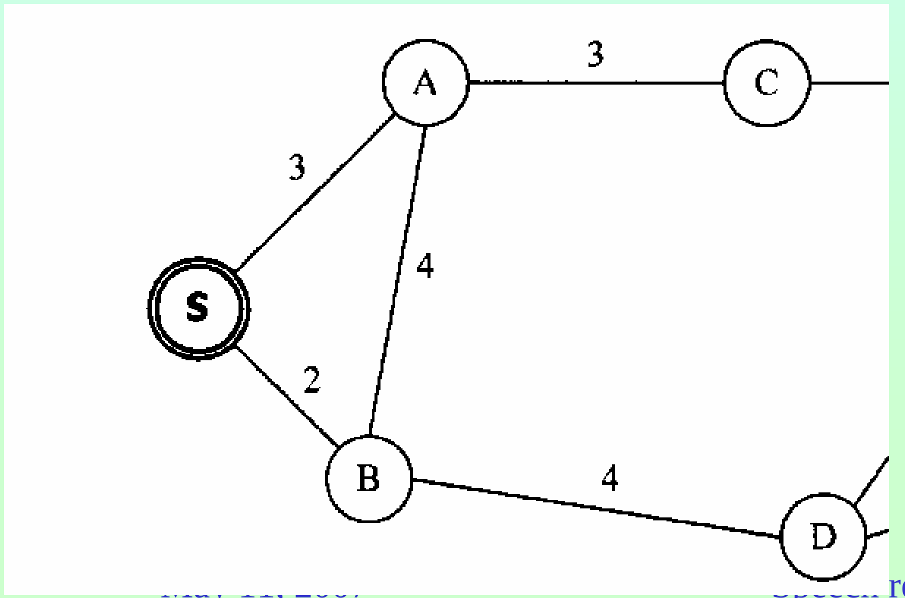
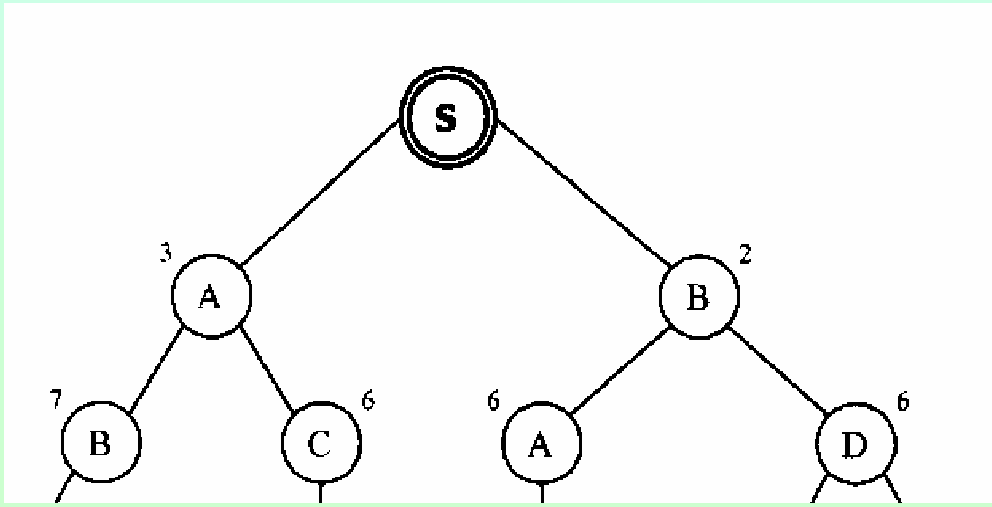


Figure 12.1 A highway distance map for cities S, A, B, C, D, E, F, and G. The salesman needs to find a path to travel from city S to city G [42].

Expand paths

- The successor (move) operator
  - generates all successors of a node and computes all costs associated with an arc
- Branching factor
  - average number of successors for each node





Fully expanded Search Tree (Graph)

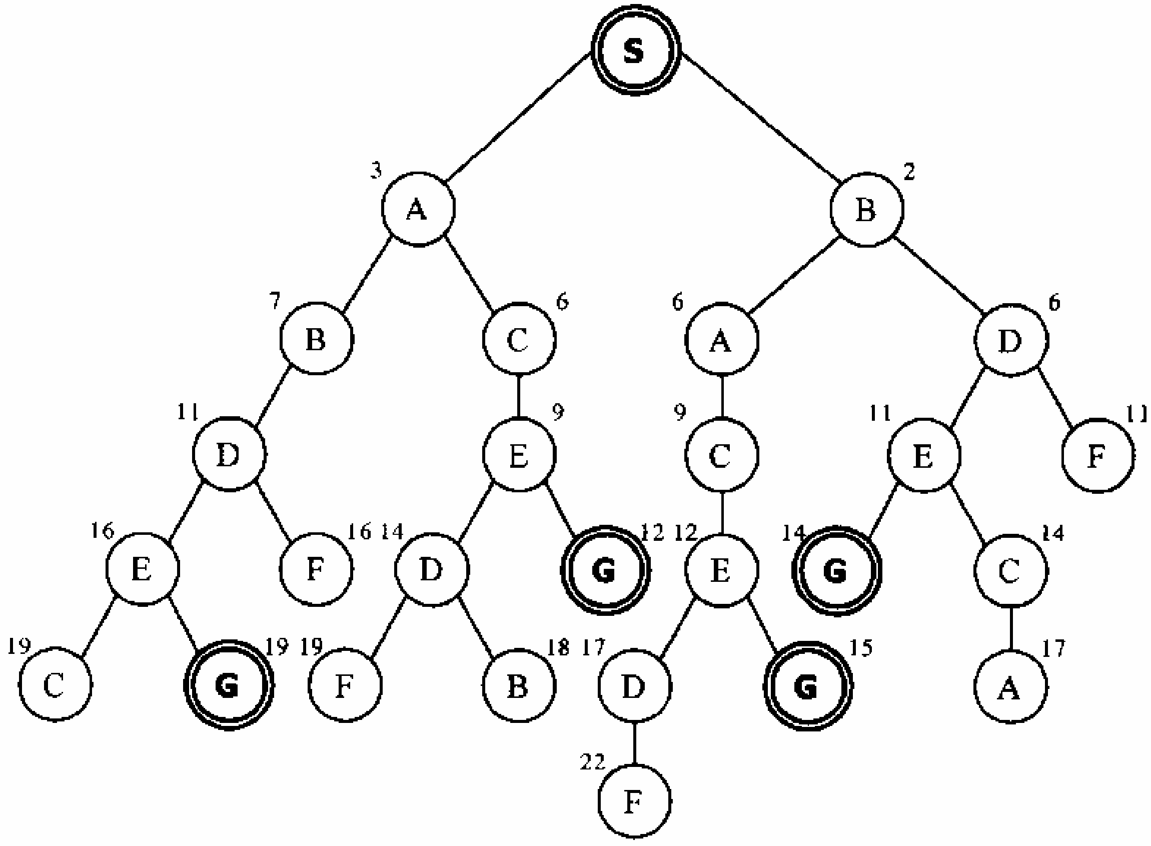


Figure 12.2 The search tree (graph) for the salesman problem illustrated in Figure 12.1. The number next to each node is the accumulated distance from start city to end city [42].

Explicit search impractical for large problems

- Use Graph Search Algorithm
  - Dynamic Programming principle
  - Only keep the shortest path to a node
- Forward direction (reasoning) normal
- Backward reasoning
  - more initial states than goal states
  - backward branching factor smaller than the forward one
- Bi-directional search
  - start from both ends simultaneously

A bad case for bi-directional search

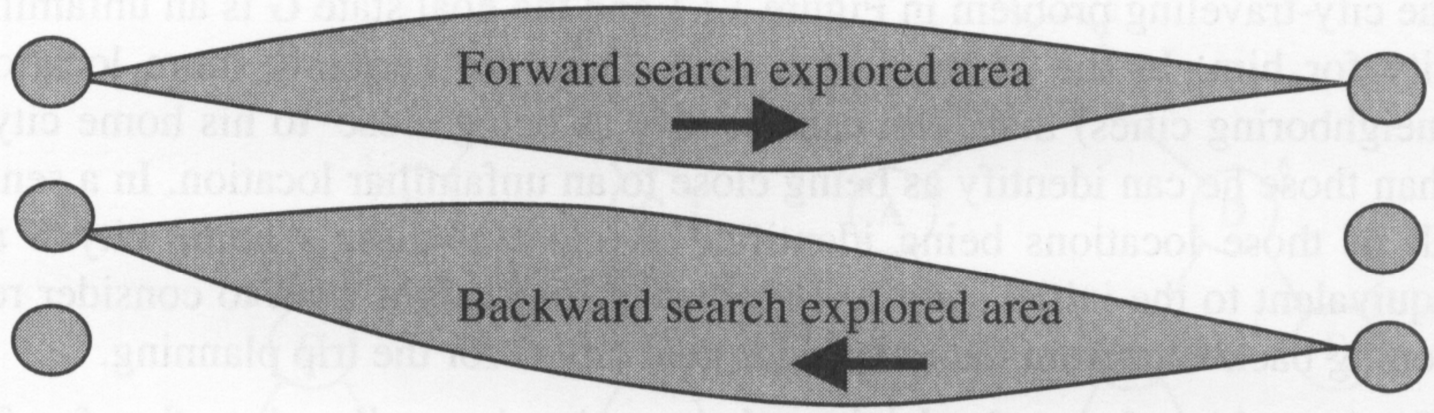
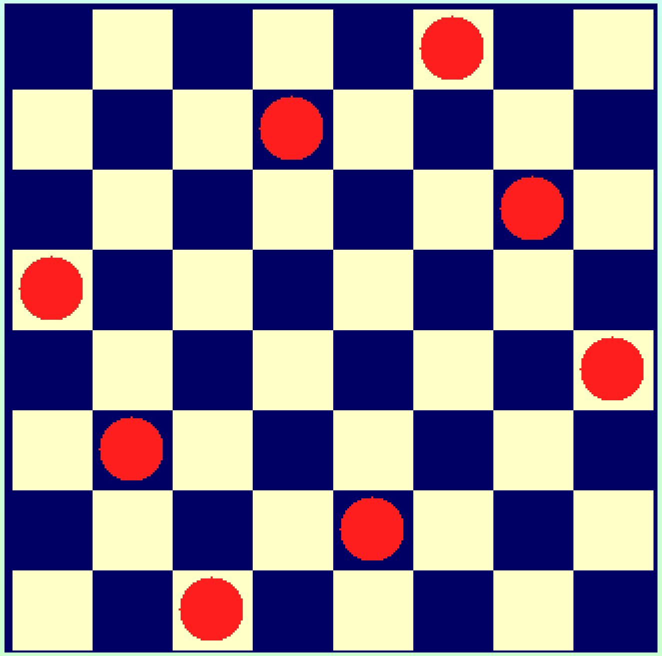


Figure 12.3 A bad case for bi-directional search, where the forward search and the backward search crossed each other [42].

The 8 Queen problem

- 1 of 12 solutions



12.1.2 Blind Graph Search Algorithms

- Find an acceptable path – need not be the best one
- Blindly expand nodes without using domain knowledge
- Also called Uniform search or Exhaustive search

Depth-First Search

- Deepest nodes are expanded first
- Nodes of equal depth are expanded arbitrarily
- Backtracking
  - If a dead-end is reached go back to last node and proceed with another one
- If Goal reached, exit

Depth-First Search

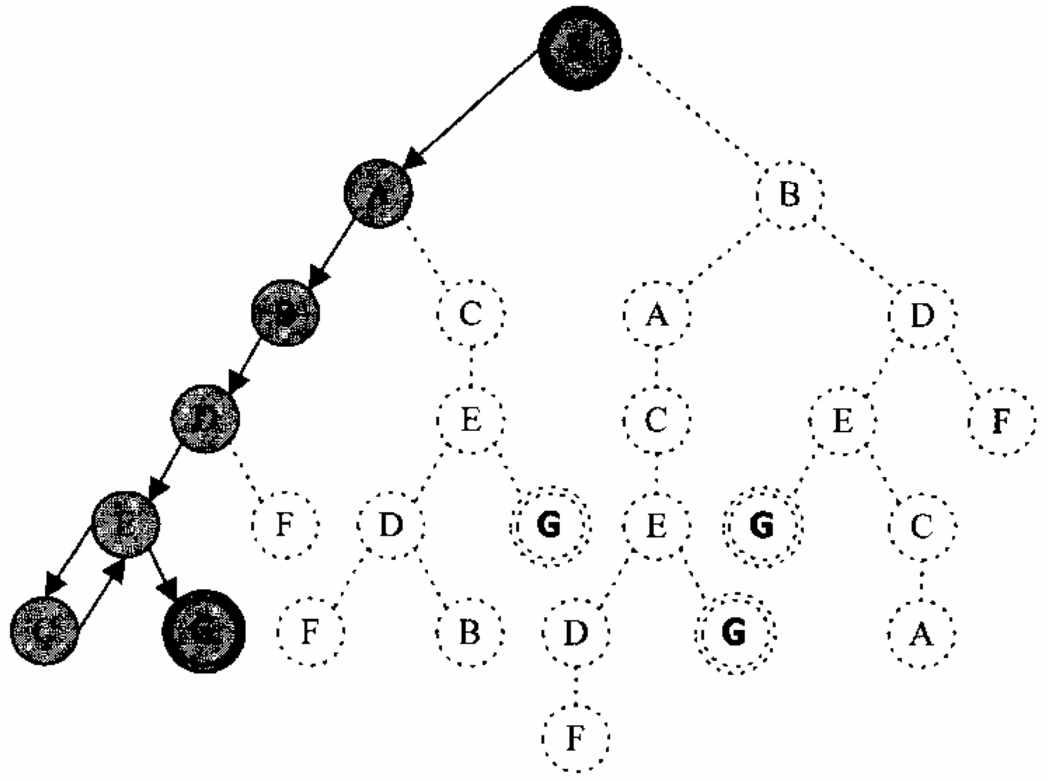


Figure 12.4 The node-expanding procedure of the depth-first search for the path search problem in Figure 12.1. When it fails to find the goal city in node *C*, it backtracks to the parent and continues the search until it finds the goal city. The gray nodes are those that are explored. The dotted nodes are not visited during the search [42].

Breadth-First Search

- Same level nodes are expanded before going to the next level
- Stop when goal i reached
- Guaranteed to find a solution if one exists
- Can find optimal solution after all solutions have been found - brute-force search

Breadth-First Search

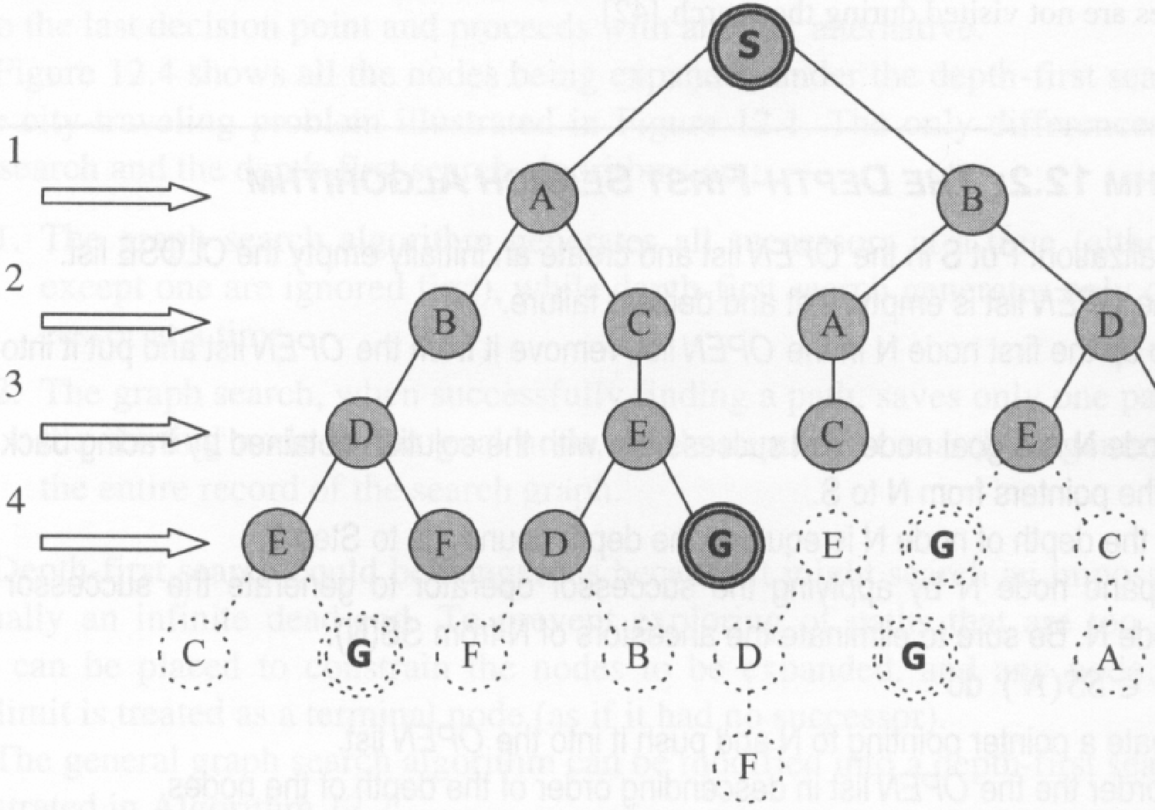


Figure 12.5 The node-expanding procedure of a breadth-first search for the path search problem in Figure 12.1. It searches through each level until the goal is identified. The gray nodes are those that are explored. The dotted nodes are not visited during the search [42].

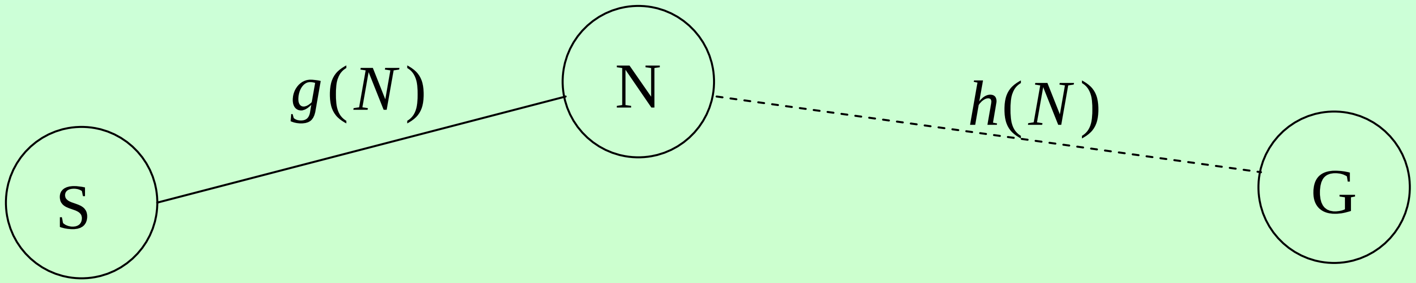
12.1.3 Heuristic Graph Search

- Use domain-specific (heuristic) knowledge to the guide search

$h(N)$ Heuristic estimate of remaining distance from node N to G

$g(N)$ The distance of the partial path from root S to node N

$f(N) = g(N) + h(N)$ Estimate of the total distance from S to N



Best-First (A* Search)

- A search is said to be admissible if it can guarantee to find an optimal solution if one exists
- If h(n) is an underestimate of the remaining distance to G the best first search is admissible. This is called A* search.

City travel problem

- Use straight-line distance to goal as heuristic information (bold digits)

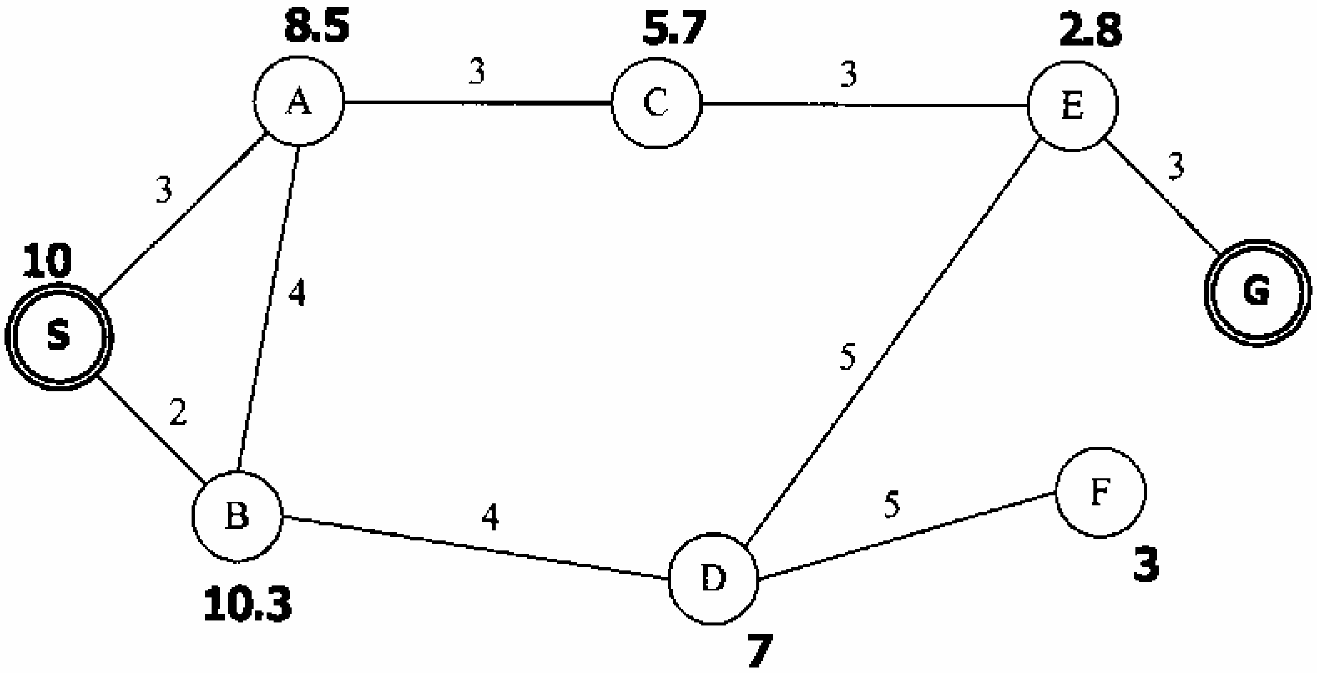
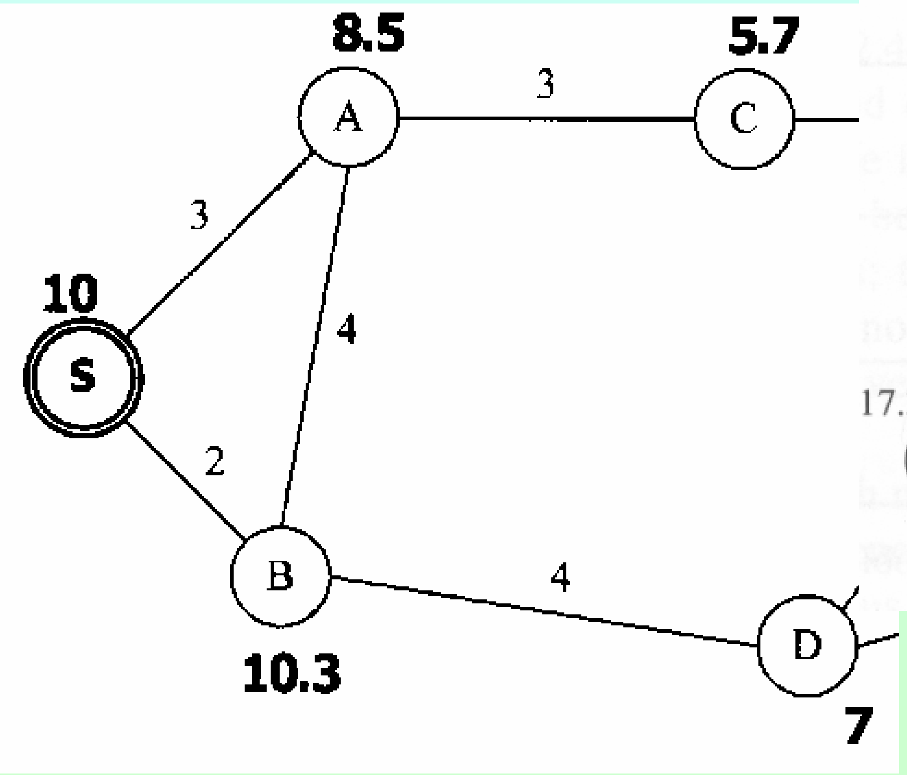
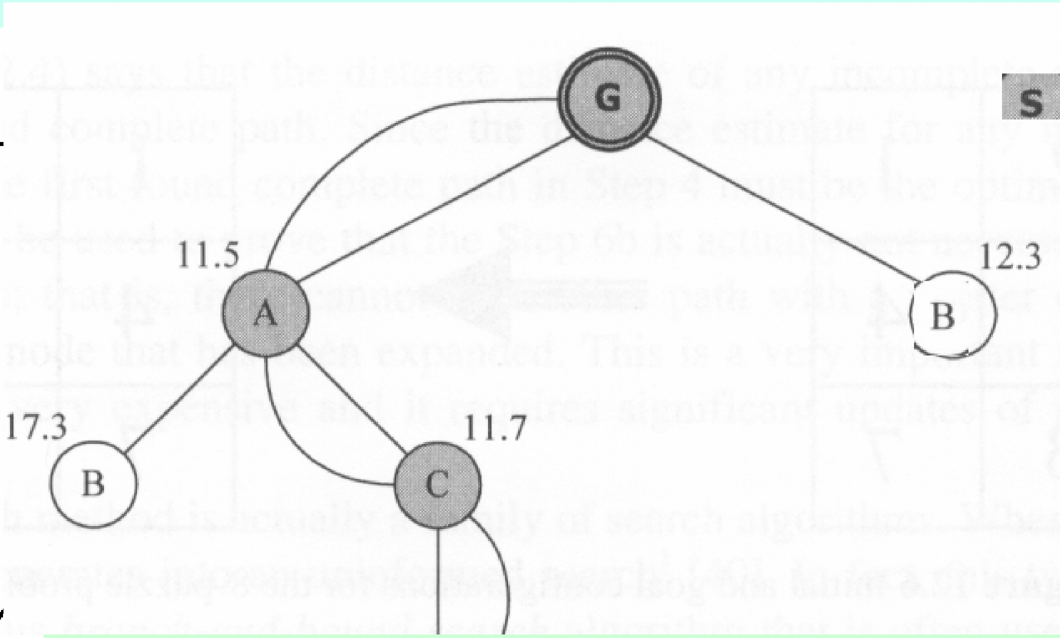


Figure 12.7 The city-travel problem augmented with heuristic information. The numbers beside each node indicate the straight-line distance to the goal node G [42].

City travel problem with heuristics





City travel problem with heuristics

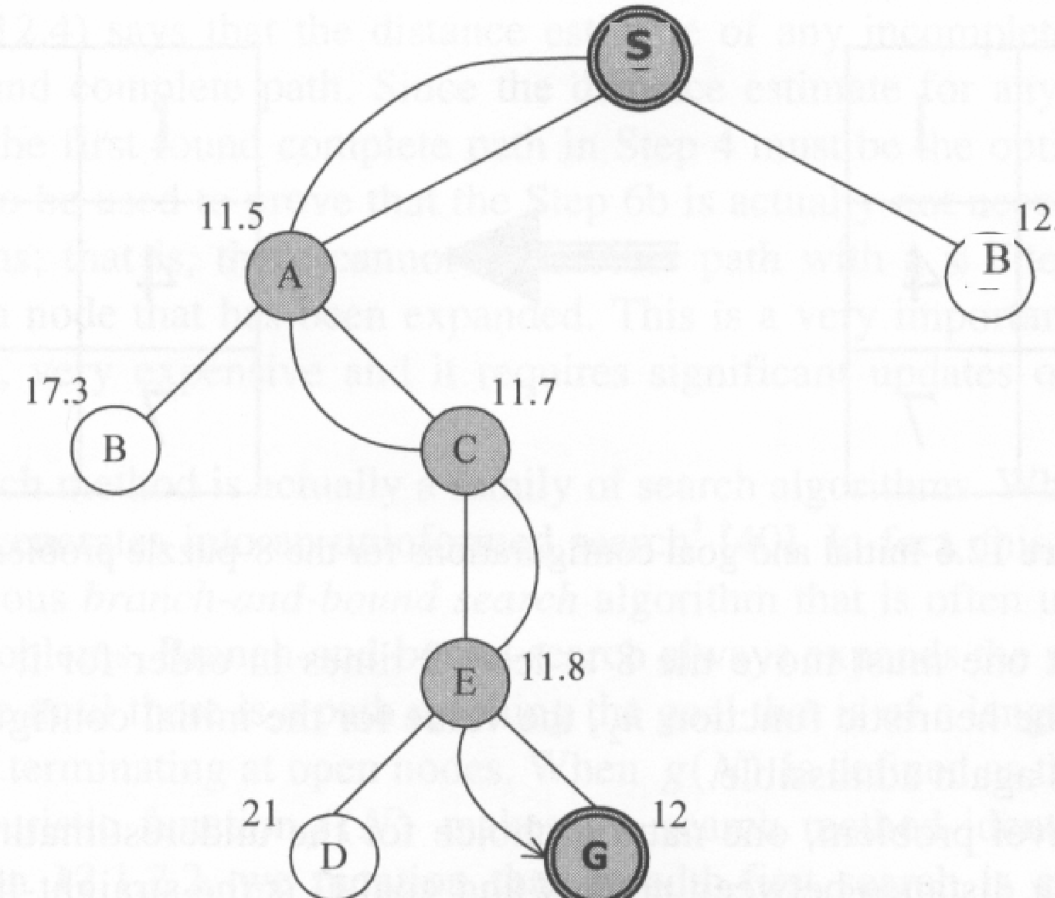


Figure 12.8 The search progress of applying A* search for the city-travel problem. The search determines that path S-A-C-E-G is the optimal one. The number beside the node is *f* values on which the sorting of the *OPEN* list is based [42].

Beam Search

- Breadth-first type of search but only expand paths likely to succeed at each level
- Only these nodes are kept in the beam and the rest are ignored, pruned
- In general a fixed number of paths, w, are kept at each level

Beam Search

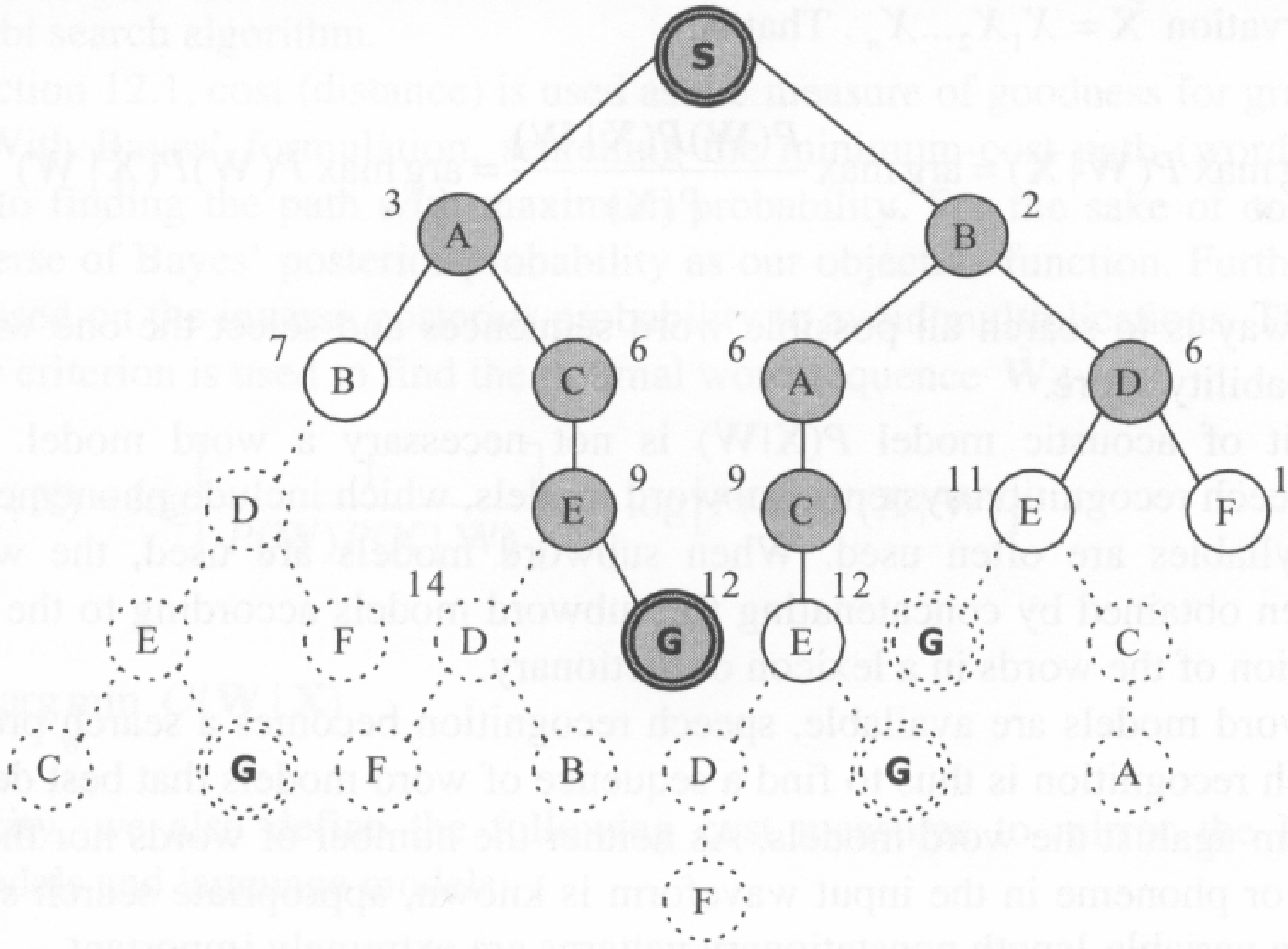


Figure 12.9 Beam search for the city-travel problem. The nodes with gray color are the ones kept in the beam. The transparent nodes were explored but pruned because of higher cost. The dotted nodes indicate all the savings because of pruning [42].

12.2 Search Algorithms for Speech Recognition

- Goal of speech recognition
  - Find word sequence with maximum posterior probability
- Change to minimum criterion function C for consistency with search
  - use inverse of Bayes posterior probability and use log to avoid multiplications

$$\hat{W} = \arg\max_{w} P(\mathbf{W}|\mathbf{X}) = \arg\max_{w} \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})} \propto \arg\max_{w} P(\mathbf{W})P(\mathbf{X}|\mathbf{W})$$

$$C(\mathbf{W}\,|\,\mathbf{X}) = \log\left[\frac{1}{P(\mathbf{W})P(\mathbf{X}\,|\,\mathbf{W})}\right] = -\log\left[P(\mathbf{W})P(\mathbf{X}\,|\,\mathbf{W})\right]$$

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{w}} C(\mathbf{W}|\mathbf{X})$$

12.2.2 Combining Acoustic and Language Models

- To balance the language model probability with the acoustic model probability a Language model Weight LW is introduced
  - thus we get the language model $P(W)^{LW}$
- Since generally LW > 1 every new word gets a penalty
  - if penalty is large the decoder prefers few long words else many short words
- If LW is used primarily to balance the acoustic model a special Insertion Penalty IP may be used
  - thus we get the language model $P(W)^{LW}\ IP^{N(W)}$

12.2.3 Isolated Word Recognition

- Boundaries known
- Calculate P(X|W) using forward algorithm or Viterbi
- Chose W with highest probability
- When subword models (monophones, triphones, …) are used HMMs may be easily concatenated

12.2.4 Continuous Speech Recognition

- Complicated
  - no reliable segmentation of words
  - each word can theoretically start at any time frame
  - the search space becomes huge for large vocabularies

Simple continuous task with 2 words

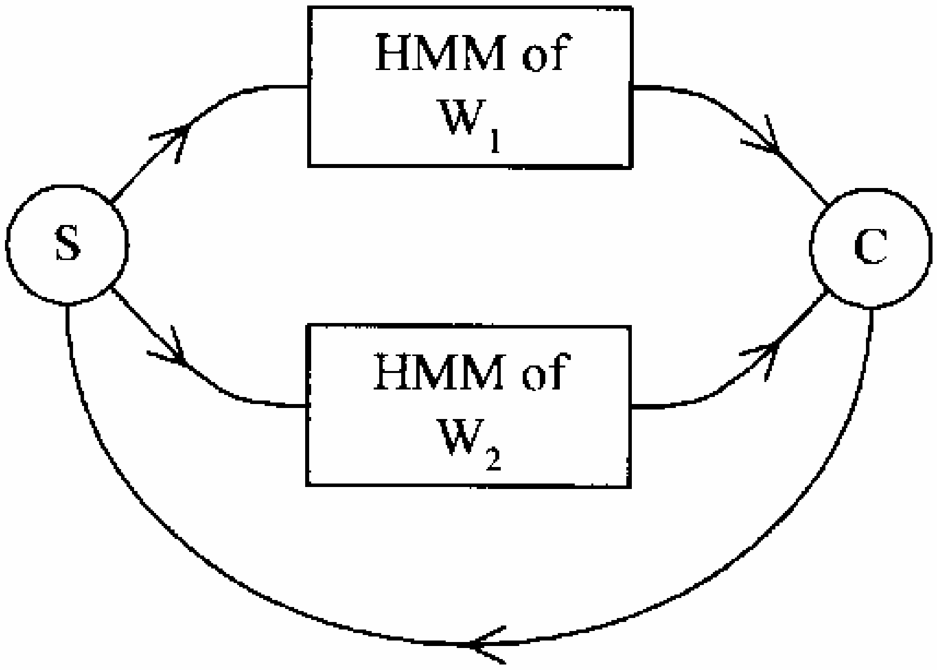


Figure 12.10 A simple example of continuous speech recognition task with two words w_1 and w_2. A uniform unigram language model is assumed for these words. State S is the starting state while state C is a collector state to save fully expanded links between every word pair.

HMM trellis for 2 word continuous rec.

- Viterbi search
  - stochastic finite state network with transition probabilities and output distributions

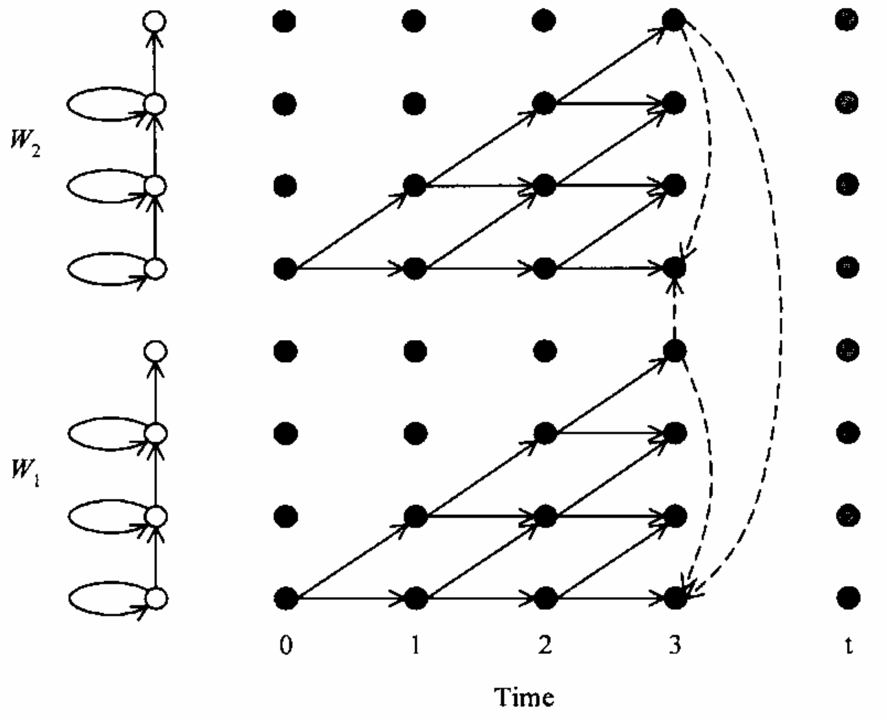


Figure 12.11 HMM trellis for continuous speech recognition example in Figure 12.10. When the final state of the word HMM is reached, a null arc (indicated by a dashed line) is linked from it to the initial state of the following word.

HMM trellis for 2 word continuous rec.

- Viterbi search
  - the computations are done *time-synchronously* from left to right, i.e. each cell for time t is computed before proceeding to time t+1

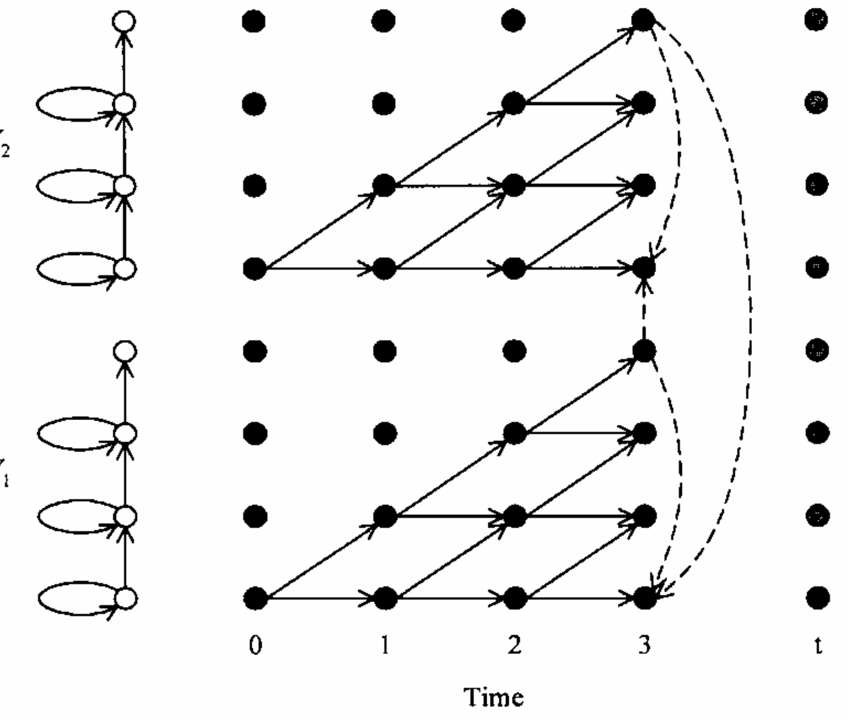


Figure 12.11 HMM trellis for continuous speech recognition example in Figure 12.10. When the final state of the word HMM is reached, a null arc (indicated by a dashed line) is linked from it to the initial state of the following word.

12.3 Language Model States

- Search Space with FSM and CFG
- Search space with Unigram, Bigrams and Trigrams
- How to Handle Silence Between Words

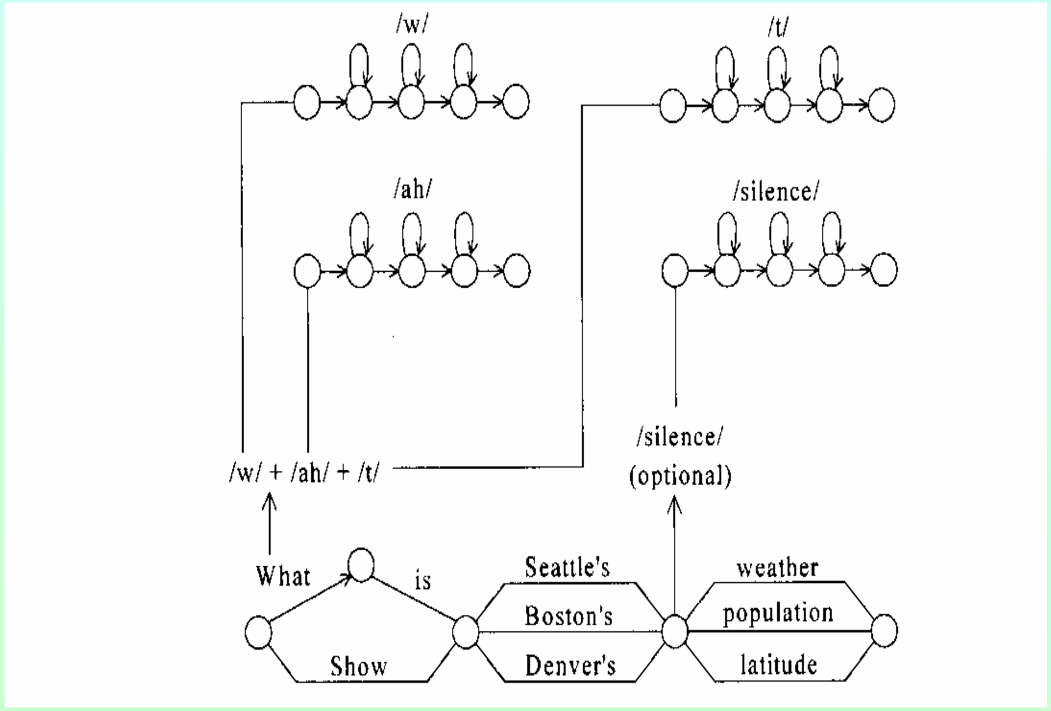
12.3.1 Search Space with FSM and CFG

- FSM – Finate State Machine
  - word network expanded into phoneme network (HMMs)
  - sufficient for simple tasks
  - very similar to CFG when using sub-grammars and word classes
- CFG – Context Free Grammar
  - set of production rules expanding non-terminals into sequence of terminals (words) and non-terminals (e.g. dates, names)
  - Chart parsing not suitable for speech recognition which requires left-to-right processing
  - Formulated with Recursive Transition Network (RTN)

Finite-State Machines, FSM

- Word network expanded into phoneme network (HMMs)
- Search using time-synchronous Viterbi
- Sufficient for simple tasks
- Similar to CFG when using sub-grammars and word classes

FSM

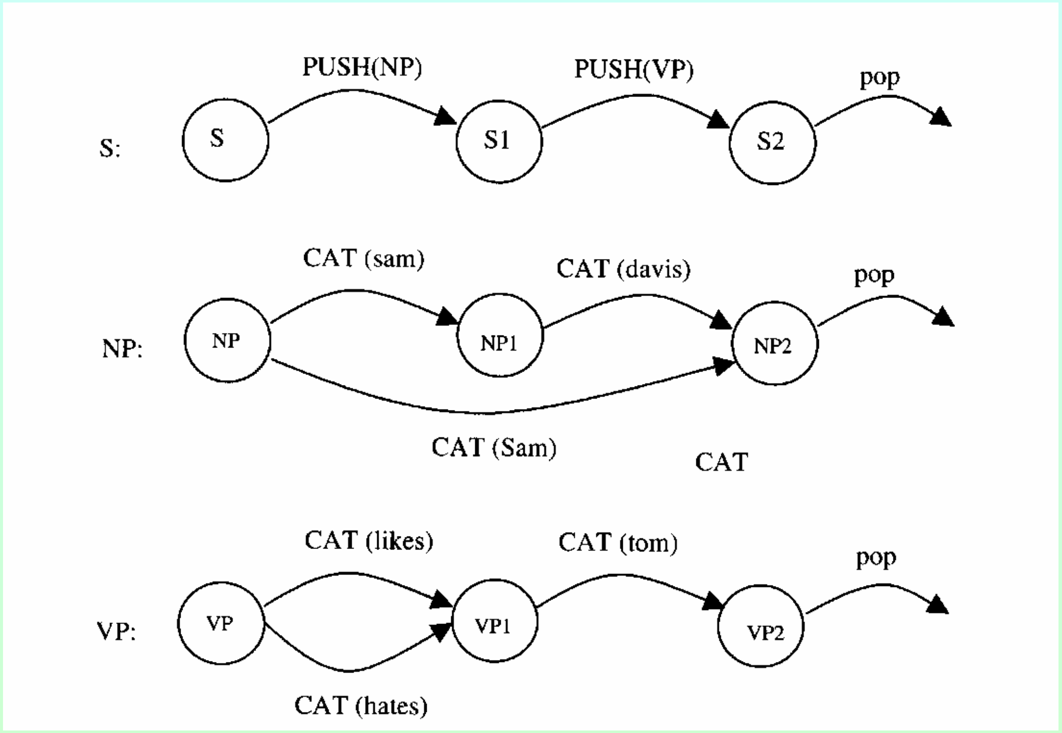


/w/
/ah/
/t/
/silence/
/w/ + /ah/ + /t/
/silence/ (optional)
What
is
Show
Seattle's
Boston's
Denver's
weather
population
latitude

Search with CFG

- Example formulation with RTN

```
S→NP VP
NP → sam|sam davis
VP → VERB tom
VERB → likes | hates
```



CAT arcs can be expanded to HMMs and searched

12.3.2 Search Space with Unigrams

- The simplest n-gram is the memoryless unigram

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i)$$

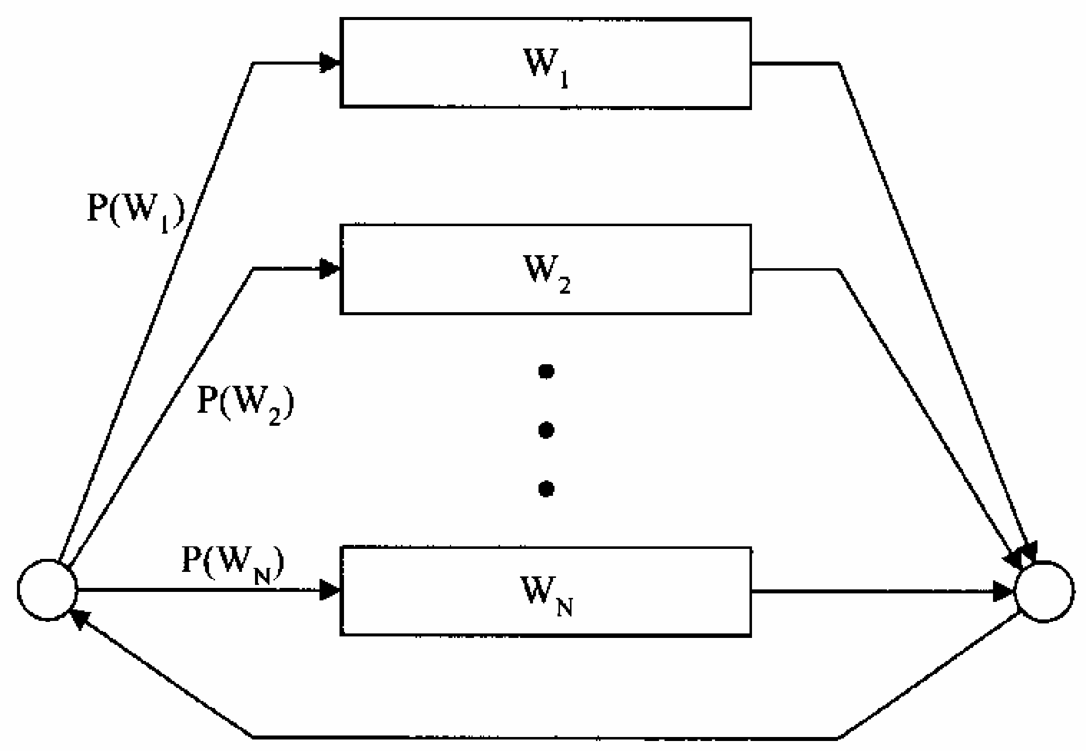


Figure 12.14 A unigram grammar network where the unigram probability is attached as the transition probability from starting state *S* to the first state of each word HMM.

12.3.3 Search Space with Bigrams

N states, N^2 word transitions

$$P(\mathbf{W}) = P(w_i \mid <s>) \prod^{n} P(w_i \mid w_{i-1})$$

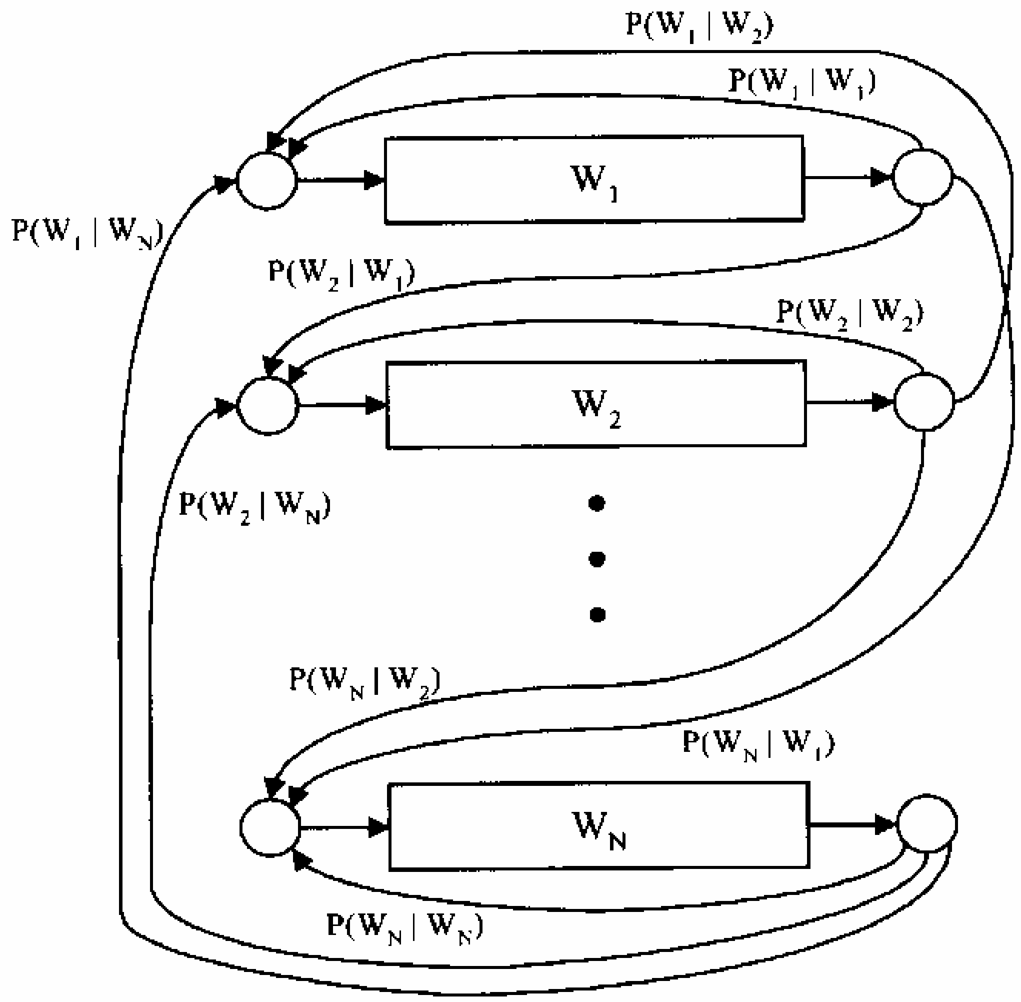


Figure 12.15 A bigram grammar network where the bigram probability $P(w_j \mid w_i)$ is attached as the transition probability from word w_i to w_j [19].

12.3.3.1 Backoff Paths

For an unseen bigram $P(w_j|w_i)=\alpha(w_i)P(w_j)$ where $\alpha(w_i)$ is the backoff weight for word w_i

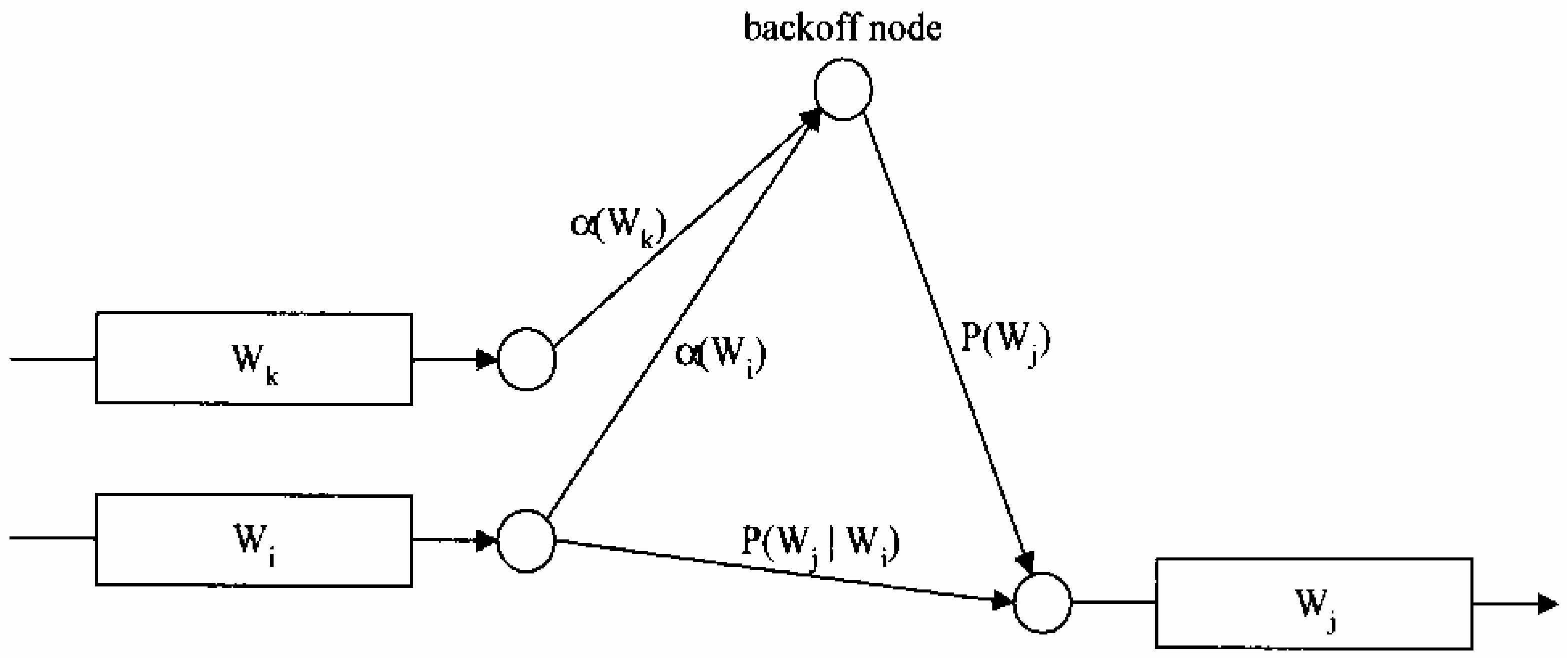
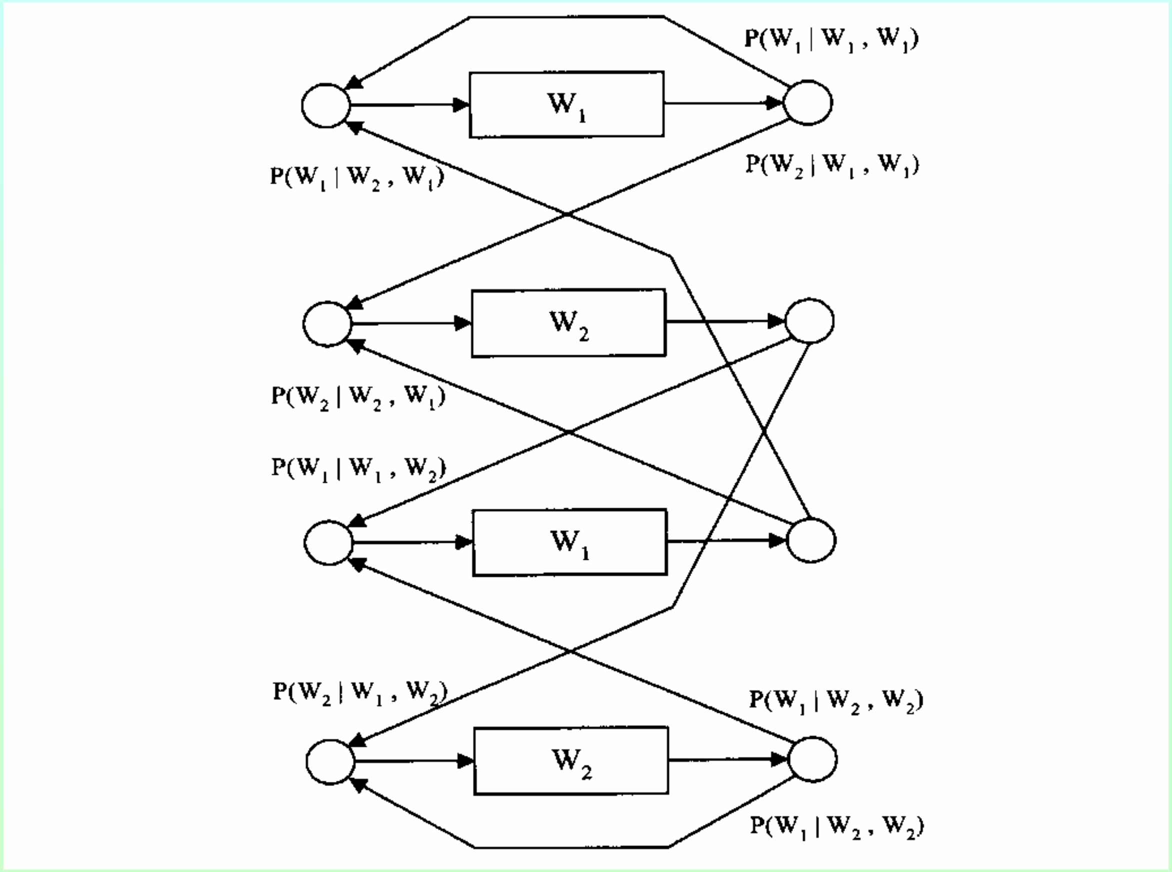


Figure 12.16 Reducing bigram expansion in a search by using the backoff node. In addition to normal bigram expansion arcs for all observed bigrams, the last state of word w_i is first connected to a central backoff node with transition probability equal to backoff weight $\alpha(w_i)$. The backoff node is then connected to the beginning of each word w_j with its corresponding unigram probability $P(w_j)$ [12].

12.3.4 Search Space with Trigrams

- The search space is considerably more complex
  - N^2 grammar states and from each of these there is a transition to the next word



12.3.5 How to Handle Silences between Words

- Insert optional silence between words

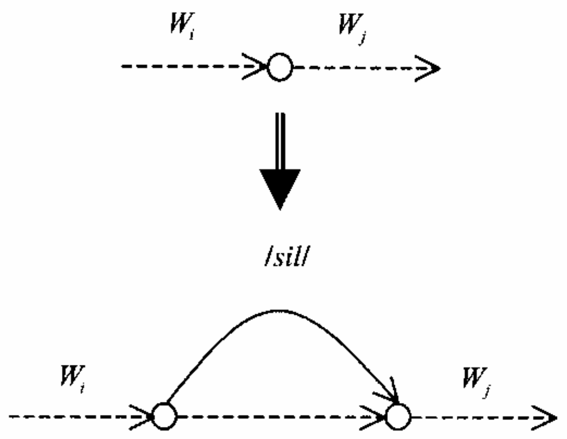


Figure 12.18 Incorporating optional silence (a non-speech event) in the grammar search network where the grammar state connecting different words is laced by two parallel paths. One is the original null transition directly from one word to the other, while the other first goes through the silence word to accommodate the optional silence.

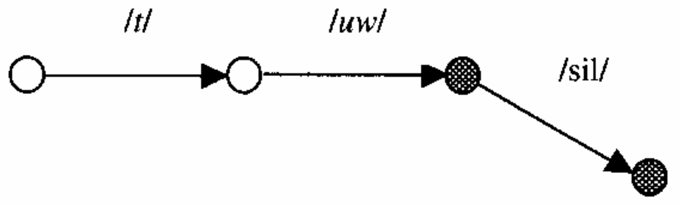


Figure 12.19 An example of treating silence as of the pronunciation network of word TWO. The shaded nodes represent possible word-ending nodes: one without silence and the other one with silence.

12.4 Time-Synchronous Viterbi Beam search

- The Viterbi approximation
  - The most likely word sequence is approximated by the most likely state sequence
  - For time t each state is updated by the best score of time t-1
  - e.g. time synchronous Viterbi search
  - record backtracking pointer at each update
  - better to only record word history at end of each word
    - then we need only 2 successive time slices for the Viterbi computations

12.4.1 The Use of Beam

- The search space for Viterbi search is O(NT) and the complexity $O(N^2T)$ where
  - N is the total number of HMM states
  - T is the length of the utterance
- For large vocabulary tasks these numbers are astronomically large even with the help of dynamic programming
- Prune search space by Beam Search
- Calculate lowest cost D_{min} at time *t*
- Discard all states with cost larger than D_{min} + B before moving on to the next time sample *t+1*

12.4.2 Viterbi Beam Search

- Empirically a beam size of between 5% and 10% of the total search space is enough for large-vocabulary speech recognition.
- This means that 90% to 95% can be pruned off at each time *t*.
- The most powerful search strategy for large vocabulary speech recognition

12.5 Stack Decoding A* Search

- Variety of A* algorithm based on the forward algorithm
  - Gives the probability of each word or subword not just an approximation as Viterbi search
- Consistent with the forward-backward training algorithm
- Can search for the optimal word string rather than the optimal state sequence
- Can, in principle, accommodate long-range language models

Search tree for a 3 word vocabulary

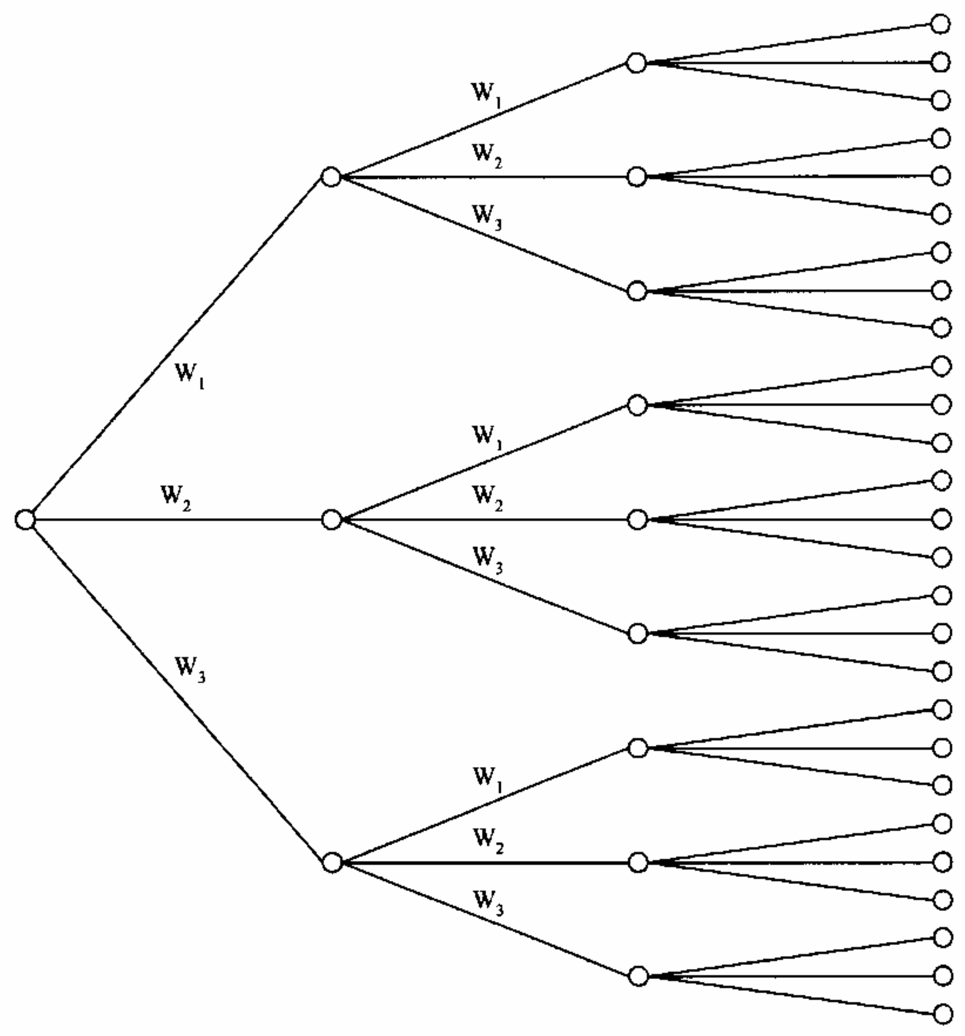


Figure 12.20 A stack decoding search tree for a vocabulary size of three [19].

Forward trellis space for stack decoding

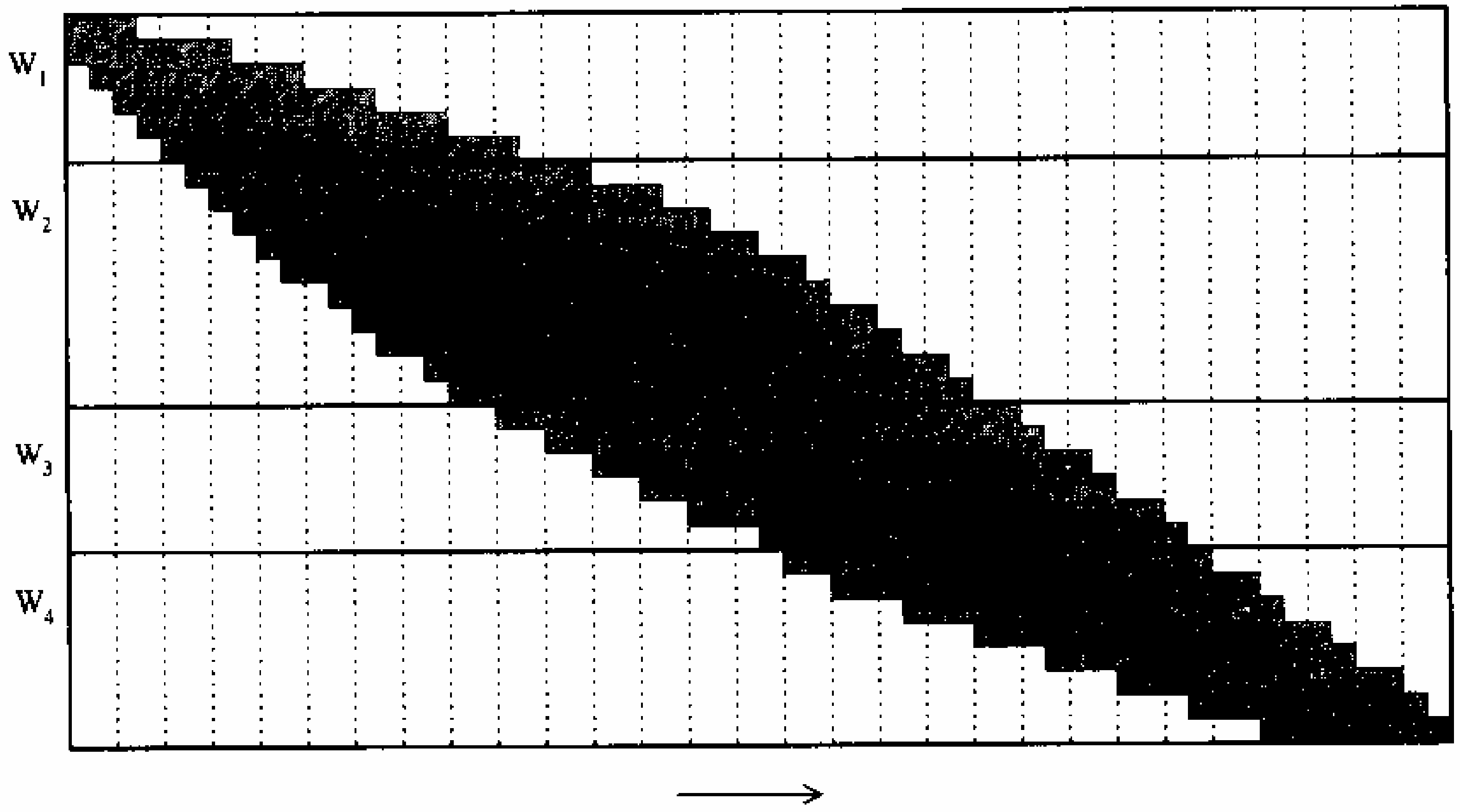


Figure 12.21 The forward trellis space for stack decoding. Each grid point corresponds to a trellis cell in the forward computation. The shaded area represents the values contributing to the computation of the forward score for the optimal word sequence $w_1, w_2, w_3, \ldots$ [24].

12.5.1 Admissible Heuristics for Remaining Path

- $f(t) = g(t) + h(T-t)$
- Calculate the expected cost per frame Ψ from the training set by using forced alignment
- $f(t) = g(t) + (T-t)\Psi$

Unnormalized cost

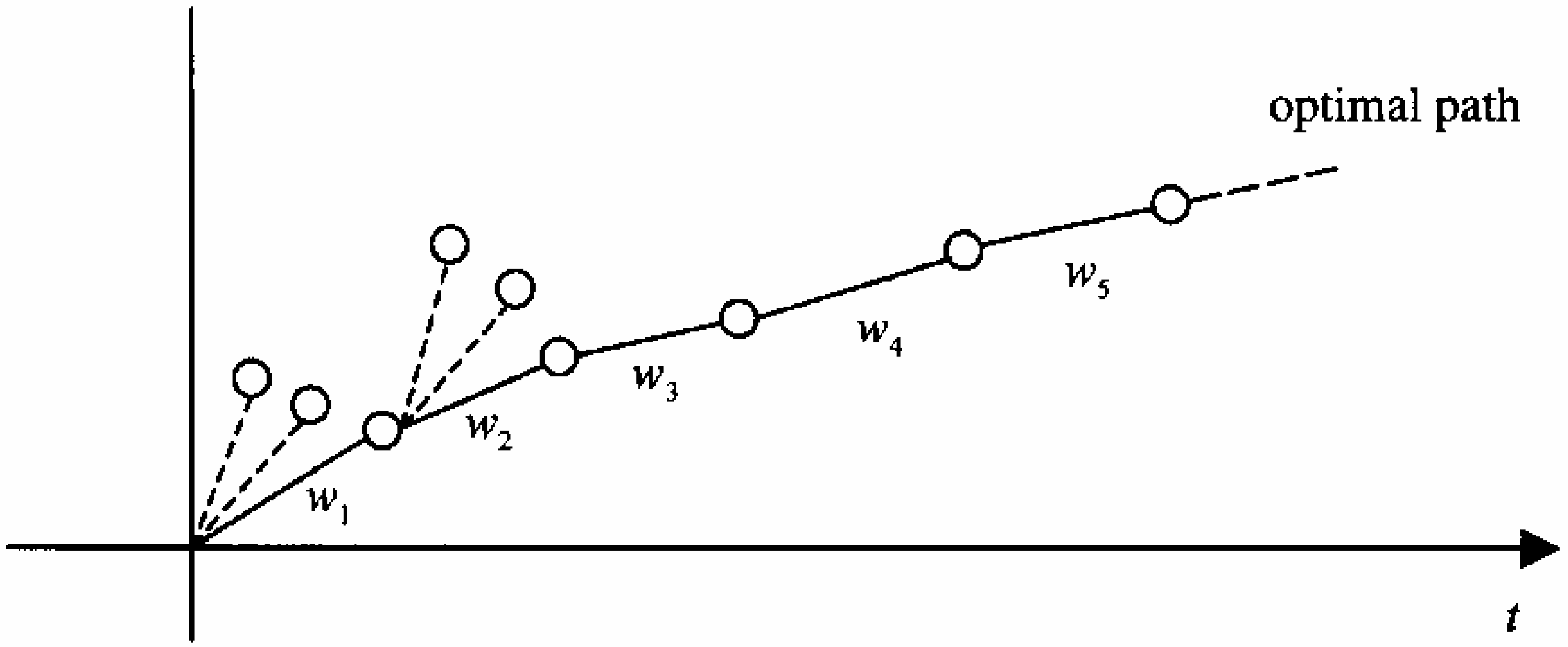


Figure 12.23 Unnormalized cost $C(\mathbf{x}_1^t, s_t \mid w_1^k)$ for optimal path and other competing paths as a function of time.

Normalized cost

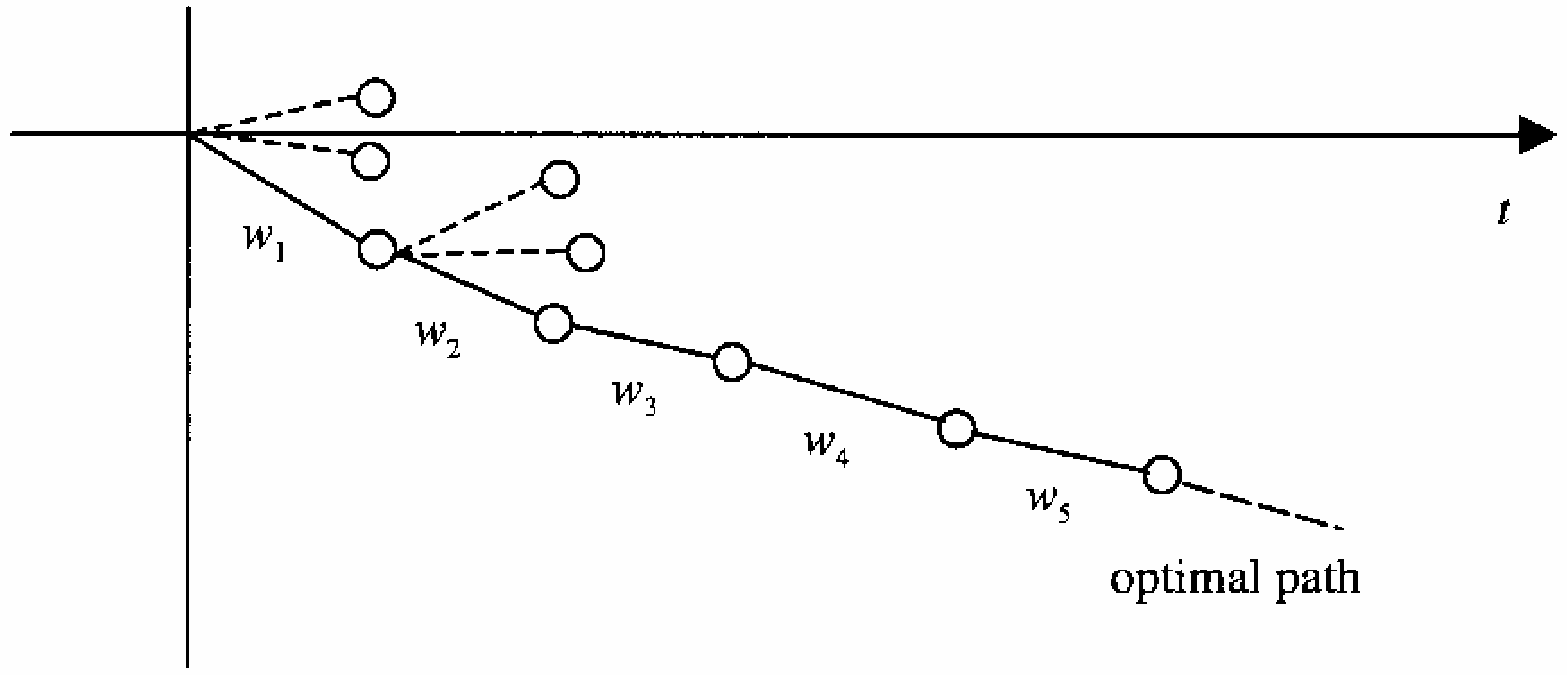


Figure 12.24 Normalized cost $\hat{C}(\mathbf{x}_1^t, s_t \mid w_1^k)$ for the optimal path and other competing paths as a function of time.

THE END