State-based search paradigm

- Triplet S, O, G (or quadruple S, O, G, N)
  - 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
  - N, set of intermediate states. Can be preset or generated by O. (not mentioned in the course book)
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
  - Definition: The complexity class of decision problems that are intrinsically harder than those that can be solved by a Non-deterministic Turing machine in Polynomial time.
  - E.g. exponential time
- Examples
  - The traveling salesman problem
    - Leave home, Visit all cities once, Return home
    - Find shortest distance
  - The 8 Queen problem
    - Place 8 Queens on a chessboard so no-one can capture any of the other
- Use heuristics to avoid combinatorial explosion

---

The 8 Queen problem

- 1 of 12 solutions
Simplified Salesman Problem

- Will illustrate different search algorithms
- Find shortest path from S to G
- Not required to visit all cities

Expand paths

- We can expand the graph to an explicit tree with all paths specified
- 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
- Inhibit cyclic paths
  - No path progress
Fully expanded Search Tree (Graph)

No cyclic paths

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 may be more effective if
  - more initial states than goal states
  - backward branching factor smaller than the forward one
- Bi-directional search
  - start from both ends simultaneously
A good case for bi-directional search

- The increase of the number of hypotheses in one search direction can be limited by the hypotheses of the opposite direction

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].
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 and Breadth-First
- Can find optimal solution after all solutions have been found
  - Brute-force search or British Museum 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
- Dangerous if infinite dead-end!
  - Introduce bound on depth
Depth-First Search

- Same level nodes are expanded before going to the next level
- Stop when goal is reached
- Guaranteed to find a solution if one exists
12.1.3 Heuristic Graph Search

- Blind search methods spend time searching in hopeless directions
- Use domain-specific (heuristic) knowledge to guide the search

\[
\begin{align*}
h(N) & \quad \text{Heuristic estimate of remaining distance from node } N \text{ to } G \\
g(N) & \quad \text{The distance of the partial path from root } S \text{ to node } N \\
f(N) = g(N) + h(N) & \quad \text{Estimate of the total distance from } S \text{ to } N
\end{align*}
\]
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

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].
Best-First (A* Search)

- Can also be used to find the n-best solutions
- Not suited for real-time incremental speech recognition
  - Incremental recognition: the initial part of the sentence is recognized before the utterance is complete
  - The estimate of h(n) requires information on the remainder of the utterance

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 width)
Beam Search

• Unlike A* search, beam search is an approximate heuristic search method that is not admissible.
• However, it has a number of unique merits. Because of its simplicity in both its search strategy and its requirement of domain-specific heuristic information, it has become one of the most popular methods for complicated speech recognition problems.
12.2 Search Algorithms for Speech Recognition

- Objective of a speech recognition system
  - 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

\[
\hat{W} = \arg \max_w P(W|X) = \arg \max_w \frac{P(W)P(X|W)}{P(X)} = \arg \max_w P(W)P(X|W)
\]

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

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

Simplification

- When HMMs are used for acoustic models, the acoustic model score (likelihood) used in search is by definition a summation of the scores of all possible state sequences (forward probability).
  - Computationally very costly
- The Viterbi Approximation
  - The most likely word sequence is approximated by the most likely state sequence.
12.2.2 Combining Acoustic and Language Models

- The acoustic models are observed at a higher rate than the language models.
- The acoustic observations are correlated.
- Gives the acoustic model higher weight than the language model.
- To balance the language model probability with the acoustic model probability a language model weight $LW$ is introduced.
  - Thus we get the language model $P(W)^{LW}$.
- Side effect: penalty for many words in the utterance.
  - Every new word lowers $P(W)$ ($LW > 0$).
  - $P(W)$ biased towards higher values for few (long) words than for many (short) words.
  - If $LW$ is high, the importance of the language model will increase compared with the acoustic model. The bias will have stronger effect on the recognition result.
- 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

- Added complexity from isolated word rec.
  - unknown word boundaries
  - each word can theoretically start at any time frame
  - the search space becomes huge for large vocabularies

Simple continuous speech recognition task

2 word vocabulary

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 cont. 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 cont. rec.

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

![HMM trellis](image)

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

- Content:
  - Search Space with Finite State Machines, FSM, and Context Free Grammars, CFG
  - Search space with Unigram, Bigrams and Trigrams
  - How to Handle Silence Between Words
12.3.1 Search Space with FSM and CFG

- FSM, Finite State Machine
  - word network expanded into phoneme network (HMMs)

- CFG, Context-Free Grammar
  - set of production rules expanding non-terminals into sequence of terminals (words) and non-terminals (e.g. dates, names)

Finite-State Machine (FSM)

- Word network expanded into phoneme network (HMMs)
- Search using time-synchronous Viterbi
- Sufficient for simple tasks (small vocabularies)
- Similar to CFG when using sub-grammars and word classes
FSM, Finite-State Machine

Context-Free Grammar (CFG)

- Set of production rules expanding non-terminals into sequence of terminals (words) and non-terminals (e.g. <date> and <name>)
- Chart parsing not suitable for speech recognition which requires left-to-right processing
- Formulated with Recursive Transition Network (RTN)
Recursive Transition Network

- There are three types of arcs in an RTN: CAT($x$), PUSH ($x$) and POP($x$).
- The CAT($x$) arc indicates that $x$ is a terminal node (which is equivalent to a word arc).

Search with CFG, Context Free Grammar

- Example formulation with RTN, Recursive transition network

$$S \rightarrow NP \ VP$$
$$NP \rightarrow \text{sam} | \text{sam davis}$$
$$VP \rightarrow \text{VERB} \ \text{tom}$$
$$\text{VERB} \rightarrow \text{likes} | \text{hates}$$

CAT arcs can be expanded to HMMs and searched
CFG and FSG vs n-grams

• The problem of connected word recognition by finite state or context-free grammars is that the number of states increases enormously when it is applied to more complex grammars.
• It is questionable whether FSG or CFG is adequate to describe natural languages or unconstrained spontaneous languages.
• Instead, n-gram language models are often used for natural languages or unconstrained spontaneous languages.

Finite State Transducers (FST)

• An FST is a finite state machine with an input and an output. The input is translated (transduced) into one or more outputs with probabilities assigned
• FSTs at different representation layers (e.g. syntax, lexicon, phoneme) are combined into a single FST
  – The combined FST can be minimized efficiently
  – Simplifies the search algorithm, which lowers the recognition time
• Popular for large vocabulary recognition
12.3.2 Search Space with Unigrams

- The simplest n-gram is the memoryless unigram

\[ P(W) = \prod_{j=1}^{n} P(w_j) \]

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

$$P(W) = P(w_i | s > \prod_{i} P(w_j | w_{i-1})$$

$$P(W_i | W_j)$$

$$P(W_j | W_i)$$

$$P(W_k | W_i)$$

$$P(W_n | W_i)$$

$$P(W_{n+1} | W_i)$$

Figure 12.15 A bigram grammar network where the bigram probability $P(w_j | w_i)$ 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 $w_i$ 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, $N^3$ word transitions

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 labeled 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 in 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 $t-1$
    - time synchronous Viterbi search
- The best-scoring state sequence can be found by back-tracking, but
  - Word sequence, not state sequence is requested
    - Backtracking pointer at each state not necessary
    - Sufficient to only record the complete word history at end state of each word
  - Then we need only 2 successive time slices for the Viterbi computations
- Dynamic construction of the search space during the search
  - Extend only the best-scoring hypotheses
  - Lowers the memory and computation requirements

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 $D_{\text{min}}$ at time $t$
- Discard all states with cost larger than $D_{\text{min}} + T$ 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 the 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
12.5.1 Admissible Heuristics for Remaining Path

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

Figure 12.23 Unnormalized cost \( C(x', s_i \mid w^t) \) for optimal path and other competing paths as a function of time.
Normalized cost

![Graph showing normalized cost](image)

Figure 12.24 Normalized cost $\hat{C}(x', s' | w')$ for the optimal path and other competing paths as a function of time.

Performance development
ARPA-project

Word error rate (%)

![Graph showing word error rate](image)

29/03/2012 Speech and speaker recognition 55

29/03/2012 Speech and speaker recognition 56

### Results 2005

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Speech type</th>
<th>Lexicon size</th>
<th>Word Error Rate (%)</th>
<th>Human Err. Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digit string (phone)</td>
<td>spontaneous</td>
<td>10</td>
<td>0.3</td>
<td>0.009</td>
</tr>
<tr>
<td>Resource Management</td>
<td>read</td>
<td>1000</td>
<td>3.6</td>
<td>0.1</td>
</tr>
<tr>
<td>ATIS</td>
<td>spontaneous</td>
<td>2000</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Wall Street Journal</td>
<td>read</td>
<td>64000</td>
<td>6.6</td>
<td>1</td>
</tr>
<tr>
<td>Radio News</td>
<td>mixed</td>
<td>64000</td>
<td>13.5</td>
<td>-</td>
</tr>
<tr>
<td>Switchboard (phone)</td>
<td>conversation</td>
<td>10000</td>
<td>19.3</td>
<td>4</td>
</tr>
<tr>
<td>Call Home (phone)</td>
<td>conversation</td>
<td>10000</td>
<td>30</td>
<td>-</td>
</tr>
</tbody>
</table>
The End