STATE SPACE SEARCH & CASE STUDY ON IUI

Dr Pradipta Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
http://cpam.iisc.ernet.in/PBiswas.htm

STATE-SPACE SEARCH

Many problems can be stated in the form of STATE SPACE SEARCH.

- The goal is to search through the possible states to find the solution.
- We can only decide what to do (i.e., what to do) by considering the possible moves from the current state and trying to look ahead as far as possible.
- For example, it is a very efficient backtracking search problem.

STATE-SPACE MODEL

- Initial State
- Operators: maps a state into a new state
- Actions: succession of states
- Goal predicate: test to see if goal achieved
- Optional
  - Cost of operators
  - Cost of solution

WHAT YOU SHOULD KNOW

- Create a state-space model
- Ensure number of states
- Identify goal or objective function
- Identify operators
UNINFORMED SEARCH STRATEGIES

- UNINFORMED SEARCH
  - **You have no clue whether one non-goal state is better than any other. Your search is blind. You don’t know if your current exploration is likely to be fruitful.**

- INFORMED SEARCH
  - **Breadth-first search**
  - **Uniform-cost search**
  - **Depth-first search**
  - **Breadth-first expanding search** (generally preferred)
  - **Bidirectional search** (preferred if applicable)

BREADTH-FIRST SEARCH

- **Expand shallowest unexpanded node**
- **Frontier (or fringe):** nodes in queue to be expanded
  - **Frontier is a FIFO queue**, i.e., new successors go at end of queue
  - **Goal-test when inserted**
BREADTH-FIRST SEARCH

- Expand shallowest unexpanded node
- Frontier is a FIFO queue, i.e., new successors go at end

PROPERTY OF BREADTH-FIRST SEARCH

- Complete (i.e., always reaches a goal if 1 is time)
- Time = b^d + 1 (where b = branching factor, d = depth)
- Optimal solution (minimizes time and space)
- Space is a bigger problem (more than time)

\[ F(G) \iff F(G) \leq \text{cost}(D) \geq F(D) \]
DEEPTh-First SEARCH

- Expand deepest unexpanded node
- Frontier = Last In First Out (LIFO) queue, i.e., new successors go at the front of the queue.
- Goal-test when inserted.

1. Is A a goal state?
2. Forgotten/reclaimed = black nodes
3. Expanded/active = gray nodes
4. Future = green dotted circles
5. Frontier = [B, C]
6. Put B, C at front of queue.
7. Expand A to B, C.
8. Frontier = [D, E, C]
10. Expand D to H, I.
11. Frontier = [H, I, E, C]
13. Expand H to D.
DEPTH-FIRST SEARCH

- Expand deepest unexpanded node
- Frontier = UFO queue, i.e., put successors at front

Frontier = \[I, E, C\]

Forget H.
Expand H to no children.

Forgotten/reclaimed = black nodes
Expanded/active = gray nodes
Frontier = white nodes
Future = green dotted circles

Put J, K at front of queue.
Is J or K a goal state?
Expand E to J, K.

Forget D, I.
Expand I to no children.
DEPTH-FIRST SEARCH

• Expand deepest unexpanded node
• Frontier = LIFO queue, i.e., put successors at front

EXPAND K to re-children
Future = C, E, K
Remain = [C]

Forgotten/reclaimed= black nodes
Expanded/active=gray nodes
Frontier=white nodes
Future= green dotted circles

PROPERTIES OF DEPTH-FIRST SEARCH

• Complete: Not fail in infinite/finite spaces
• Can modify to avoid unexplored states and/or nodes
• Does not require more memory than depth-first
• Can use graph search: remember all nodes found
• Requires more search space if unfruitful, may find
• Still fails in infinite spaces (may find goal) framework
• Tree-Depth with an unchanging depth of space
• tree-depth p is much larger than q
• Breadth-first is more efficient, may be much faster than DFS
• Space: O(b^d), i.e., linear space
• Remember a single path + expanded unexplored nodes
• Optimal: Not if it may find a non-optimal goal first

DEPTH-FIRST VS. BREADTH-FIRST

ADVANTAGES OF DEPTH-FIRST:
• Never require small memory for storing the expansion
• tree-depth with an unchanging depth of space
• Requires more search space if unfruitful, may find
• Still fails in infinite spaces (may find goal) framework

DISADVANTAGES OF DEPTH-FIRST:
• Can sometimes fail to find a solution
• Not guaranteed to find an optimal solution
• Can take a lot longer to find a solution

ADVANTAGES OF BREADTH-FIRST:
• Guaranteed to find a solution if one exists
• Depending on the problem, can be guaranteed to find an optimal solution

DISADVANTAGES OF BREADTH-FIRST:
• Needs a lot of memory time exploring the same space as the search space is a high branching factor.
To avoid the infinite depth problem of DFS, only search until depth \( L \), i.e., we don't expand nodes beyond depth \( L \).

- What if solution is deeper than \( L \)? Increase \( L \) iteratively.
- This inherits the memory advantage of Depth-first search.
- Better in terms of space complexity than Breadth-first search.

**Depth-Limited Search**

**Iterative Deepening Search**

This inherits the memory advantage of Depth-first search.

Better in terms of space complexity than Breadth-first search.
ITERATIVE DEEPENING SEARCH \( L = 3 \)

BIDIRECTIONAL SEARCH

- Data
  - What does searching backwards from \( G \) mean?
  - Need to keep track of the intersection of the two sets of paths
- Best
  - Use bidirectional search if possible
- Worst
  - Can still be exponential in the worst case

INFORMED SEARCH STRATEGIES

USE HEURISTIC KNOWLEDGE TO INCREASE EFFICIENCY OF SEARCH
- Select which node to expand next during search
- While expanding a node, decide which successors to generate and which to ignore
- Remove from the search space some nodes that have previously been generated — prune the search space

BEST-FIRST SEARCH

- Evaluate the information that can be obtained by expanding a node and its importance in closing the search
- The quality of a node is estimated by the heuristic search function \( h(n) \) for node \( n \)
- Greedy strategy — go to the best
ROMANIA WITH STRAIGHT-LINE DISTANCE

GREEDY BEST-FIRST SEARCH (OFTEN CALLED JUST “BEST-FIRST”)

- $h(n) = \text{ESTIMATE OF COST FROM } n \text{ TO GOAL}
- E.g., $h(n) = \text{STRAIGHT-LINE DISTANCE FROM } n \text{ TO BUCHAREST}

- Greedy best-first search expands the node that appears to be closest to goal.
- Priority queue function = $h(n)$
GREEDY BEST-FIRST SEARCH EXAMPLE

OPTIMAL PATH

PROPERTIES OF GREEDY BEST-FIRST SEARCH

1. The given can get stuck in loops.
2. Can be implemented in time space.
3. Can be implemented in time space.
4. A node heuristic can give significant improvement.
5. Too slow
6. Requires all nodes in memory

E.g., Arad → Sibiu → Ranoși Vilcea → Pitesti → Bucharest is shorter.
A* SEARCH

- Does avoid paths that are already open
- Generally a better choice than depth-first search
- Optimal when heuristic is admissible (consistent / optimistic)
- Evaluation function: \( f(S) = g(S) + h(S) \)
- \( g(S) \) = known cost so far to node \( S \)
- \( h(S) \) = heuristic estimate of cost to goal from node \( S \)
- \( f(S) \) = total cost to goal through node \( S \)
- Priority queue sort function = \( f(S) \)

Components of A*
**Contours of A* Search**

- A* expands nodes in order of increasing f value
- Gradually adds "a-contour" of nodes
- Contours has all nodes with f ≤ f
- Where f₁ < f₂

**Properties of A***

- **Complete**: Yes
  - If there are infinitely many nodes with f ≤ f, cannot happen if there are only a finite number of nodes
- **Inadmissable**: Exponential, O(n)
  - Excesses
- **Optimal**: Yes
  - With the optimal admissible heuristic:
  - Guaranteed optimal heuristic
- **Optimally Efficient**: Yes
  - No optimal algorithm with same heuristic is guaranteed to expand fewer nodes

---

**Multimodal HUD for Auto UI**

**Context**

**Issues with Dashboard**
- Drivers take eyes off road while operating dashboard
- Dashboard requires physical touch

**State of the Art**
- Direct Voice Input
- Gesture recognition system
- Translating movement

**Existing Problem**
- Accuracy of VUI changes for different languages and accent
- Voice to recognize set of devices or screen interactions
  - Does not tolerate latency in infrared sensors
EYE MOVEMENT

EYE GAZE CONTROLLED PROJECTED DISPLAY

IMPROVING GAZE TRACKING – HOT SPOTS

STATE SPACE SEARCH

- A STATE SPACE CONSISTS OF:
  - A POSE/PERMUTATION SET OF STATES
  - THE START STATE REPRESENTS THE INITIAL/INITIALIZATION
  - EACH STATE REPRESENTS STATE CONDITIONING INCIDENTS FROM THE START STATE
  - SOME STATES MAY BE GOAL STATES (ENDCONDS)
- A SET OF OPERATIONS
  - MOVING BETWEEN STATES TO OBTAIN STATES TO SELECT STATES IN THE STATE SPACE
USER STUDY

- Collected data from 11 participants, all with driving experience
- Driving task involved ISO 26002 Lane Changing Task
- Secondary task involved pointing and selection on a touch screen
  - Using TrackScreen
  - CAD Tracking HUD
  - CAD Tracking HUD with hotspots

SUMMARY

- Proposed a multimodal HUD for automotive environments
- Drivers need not to take their eyes off the road for the proposed systems
- Can be operated as fast as a touchscreen with improved driving performance
- Proposed a new algorithm to improve response times for gaze-controlled interactive systems

RESULTS FROM DUAL TASK STUDY

- Mean Deviation from Lane
  - Projected HS
  - Projected Touch
  - MEAN DEVIATION
  - Response Time (in msec)
  - RESPONSE TIME

TAKE AWAY POINTS

- HOT SPOT TRACKING
- SPATIOTEMPORAL TRACKING
- TRACKING
- Dot Rate
- Dot Rate
- Dot Rate