

Simple Search

Search	2
Search Basics	2
Search Agent	3
The Search Problem	4
Example Search Problems	5
Inputs for Search Algorithms	6
Structures in Search Algorithms	7
Breadth-First Search	8
Depth-First Search	9
Iterative Deepening Search	10
More Search Algorithms	11
Performance Characteristics	12
Search Performance	13
Other Issues	14

Search

Search Basics

- *Search* is finding a sequence of *actions* that achieve a *goal* from an *initial state*.
- Idea: An agent that can predict the results of its actions can choose better actions.
- Assumptions: Actions are deterministic.
Relevant features of states can be perceived.
Whether a state satisfies the goal can be detected.
What kind of environment is assumed?

Search Agent

```
function SEARCH-AGENT()  
  static: goal, actions  
  action-sequence ← null  
  loop  
    percept ← perceive environment  
    s ← DETERMINE-STATE(percept)  
    exit if s satisfies goal  
    if action-sequence does not go from s to goal  
    then find action-sequence from s to goal  
       fail if no action-sequence is found  
    action ← remove first action from action-sequence  
    perform action on environment
```

The Search Problem

- *Initial state.* A description of the initial conditions.
- *Actions.*
- *Transition Model.* Given a state and an action, what is the *successor* state? *Expanding* a given state finds all of its successors. The *state space* is the *graph* where states are nodes and actions are edges.
- *Goal test.* Determines if a state is a goal state.
- *Path cost.* The cost of performing a sequence of actions from the initial state.
- *Solution.* A path from the initial state to a goal state.

CS 3793 Artificial Intelligence

Simple Search – 4

Example Search Problems

8-puzzle, 15-puzzle	Tower of Hanoi
n -queens	Monkey and bananas
Cryptarithmic (TWO + TWO = FOUR)	Airline booking
Missionaries and cannibals	Traveling salesman
Rubik's cube	Device assembly
Blocks-world	Calculus, algebra problems

CS 3793 Artificial Intelligence

Simple Search – 5

Inputs for Search Algorithms

- *Initial state.*
- *Successor/Expand Function.* Return the states that can be reached from a given state by performing the actions.
- *Goal test function.* Determine if a given state is a goal state.
- Optional: *Path/edge cost function.* Determine the cost of a given path/edge.
- Optional: *Resource limit.* When to give up.

CS 3793 Artificial Intelligence

Simple Search – 6

Structures in Search Algorithms

A search algorithm needs to keep track of:

- *Fringe, frontier.* States that have been generated, but not yet processed. Search algorithms can be characterized by how the fringe is managed.
- *Closed states.* Processed states.

CS 3793 Artificial Intelligence

Simple Search – 7

Breadth-First Search

```
function BFS( $s_{\text{initial}}$ , EXPAND, GOAL)
 $q \leftarrow$  NEW-QUEUE()
ENQUEUE( $s_{\text{initial}}$ ,  $q$ )
while  $q$  is not empty
do  $s_{\text{current}} \leftarrow$  DEQUEUE( $q$ )
  if GOAL( $s_{\text{current}}$ ) then return solution
  for each  $s_{\text{successor}}$  in EXPAND( $s_{\text{current}}$ )
    do ENQUEUE( $s_{\text{successor}}$ ,  $q$ )
return failure
```

CS 3793 Artificial Intelligence

Simple Search – 8

Depth-First Search

```
function DFS( $s_{\text{current}}$ , EXPAND, GOAL, bound)
if GOAL( $s_{\text{current}}$ ) then return solution
if bound = 0 then return failure
for each  $s_{\text{successor}}$  in EXPAND( $s_{\text{current}}$ )
do DFS( $s_{\text{successor}}$ , EXPAND, GOAL, bound - 1)
  if DFS succeeds then return solution
return failure
```

CS 3793 Artificial Intelligence

Simple Search – 9

Iterative Deepening Search

```
function IDS( $s_{\text{initial}}$ , EXPAND, GOAL)
  for  $depth \leftarrow 0$  to depth of search graph
    do DFS( $s_{\text{initial}}$ , EXPAND, GOAL,  $depth$ )
      if DFS succeeds then return solution
  return failure
```

CS 3793 Artificial Intelligence

Simple Search – 10

More Search Algorithms

- Uniform Cost (Dijkstra's Algorithm):
Select least costly state from fringe.
Implement: modify BFS with priority queue.
- Bidirectional Search:
Search from both initial state and goal state.
Implement: dual BFS with efficient equality.
Assumes single (or few) goal state(s).
Assumes actions can be inverted.

CS 3793 Artificial Intelligence

Simple Search – 11

Performance Characteristics

Assume the state space is *tree-structured* with:
 $b = \text{branching factor}$,
 $d = \text{depth of closest solution}$,
 $m = \text{maximum depth of state space}$,
 $l = \text{depth bound}$

CS 3793 Artificial Intelligence

Simple Search – 12

Search Performance

Search Method	States Visited	States Memory
Breadth-First	$O(b^d)$	$O(b^d)$
Depth-First	$O(b^m)$	$O(bm)$
DFS (bounded)	$O(b^l)$	$O(bl)$
Iterative Deep.	$O(b^d)$	$O(bd)$

BFS and IDS return optimal (shortest) solution.

Ratio of states visited is $\frac{\text{IDS}}{\text{BFS}} \approx \frac{b}{b-1}$

CS 3793 Artificial Intelligence

Simple Search – 13

Other Issues

- Repeated states:
Do not want to search the same state twice.
In increasing effectiveness and overhead:
 1. Avoid going back to a state's parent, or
 2. Avoid circular paths, or
 3. Avoid any state previously generated.
- Constraint satisfaction search:
Assign values to variables satisfying constraints.
Choose a variable and branch on possible values.

CS 3793 Artificial Intelligence

Simple Search – 14