Search

George Konidaris gdk@cs.duke.edu



Spring 2016

(pictures: Wikipedia)

Search



5	3			7				
6			1	9	5			
	9	8					6	
8				6				З
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Basic to problem solving:

• How to take action to reach a goal?

Search



Specifically:

- Problem can be in various states.
- Start in an *initial state*.
- Have some actions available.
- Each action has a cost.
- Want to reach some *goal*, minimizing cost.

Happens in simulation. Not web search.

Formal Definition



Set of states ${\boldsymbol{S}}$

Start state $s \in S$

Set of actions A and action rules $a(s) \rightarrow s'$

Goal test $g(s) \rightarrow \{0, 1\}$

Cost function $C(s, a, s') \to \mathbb{R}^+$

So a search problem is specified by a tuple, (S, s, A, g, C).

Find a sequence of actions $a_1, ..., a_n$ and corresponding states $s_1, ..., s_n$



$$s_0 = s$$

 $s_i = a_i(s_{i-1}), i = 1, ..., n$
 $g(s_n) = 1$

$$\sum_{i=1}^{n} C(s_{i-1}, a, s_i)$$





Find a sequence of actions $a_1, ..., a_n$ and corresponding states $s_1, ..., s_n$

... such that:

$$s_0 = s$$

 $s_i = a_i(s_{i-1}), i = 1, ..., n$
 $g(s_n) = 1$

start state

$$\sum_{i=1}^{n} C(s_{i-1}, a, s_i)$$



Find a sequence of actions $a_1, ..., a_n$ and corresponding states $s_1, ..., s_n$

... such that:

$$s_0 = s$$

 $s_i = a_i(s_{i-1}), i = 1, ..., n$
 $g(s_n) = 1$

start state legal moves

$$\sum_{i=1}^{n} C(s_{i-1}, a, s_i)$$



Find a sequence of actions $a_1, ..., a_n$ and corresponding states $s_1, ..., s_n$

... such that:

$$s_0 = s$$

 $s_i = a_i(s_{i-1}), i = 1, ..., n$
 $g(s_n) = 1$

start state legal moves end at the goal

$$\sum_{i=1}^{n} C(s_{i-1}, a, s_i)$$



Find a sequence of actions $a_1, ..., a_n$ and corresponding states $s_1, ..., s_n$

... such that:

$$s_0 = s$$

 $s_i = a_i(s_{i-1}), i = 1, ..., n$
 $g(s_n) = 1$

start state legal moves end at the goal

while minimizing:

 $\sum_{i=1}^{n} C(s_{i-1}, a, s_i) \quad \text{minimize sum of costs - rational agent}$

Example

<u>Sudoku</u>

States: all legal Sudoku boards.

Start state: a particular, partially filled-in, board.

Actions: inserting a *valid* number into the board.

Goal test: all cells filled and no collisions.

Cost function: I per move.

9 5 9 8 2 8 1 9





Flights - e.g., ITA Software.

States: airports.





Start state: RDU.

Actions: available flights from each airport.

Goal test: reached Tokyo.

Cost function: time and/or money.

The Search Tree



Classical conceptualization of search.



The Search Tree





Important Quantities



Breadth (branching factor)



The Search Tree



Depth





 $O(b^d)$ leaves in a search tree of breadth b, depth d.

The Search Tree



Expand the tree one node at a time.

Frontier: set of nodes in tree, but not expanded.



which node to expand next?

How to Expand?

DUKE COMPUTER SCIENCE

Uninformed strategy:

nothing known about likely solutions in the tree.

What to do?

- Expand deepest node (depth-first search)
- Expand closest node (breadth-first search)

Properties

- Completeness
- Optimality
- Time Complexity (total number of nodes visited)
- Space Complexity (size of frontier)





Expand deepest node







Expand deepest node







Expand deepest node /













 $O(b^{d+1} - b^{d-m}) = O(b^{d+1})$











Depth-First Search



Properties:

- Completeness: Only for finite trees.
- Optimality: No.
- Time Complexity: $O(b^{d+1})$
- Space Complexity: O(bd)

Here *m* is depth of found solution (not necessarily min solution depth).

The deepest node happens to be the one you most recently visited easy to implement recursively OR manage frontier using LIFO queue.

























 $O(b^{m+1})$

Properties:

- Completeness:Yes.
- Optimality: Yes for constant cost.
- Time Complexity: $O(b^{m+1})$
- Space Complexity: $O(b^{m+1})$

Better than depth-first search in all respects except *memory* cost - must maintain a large frontier.

Manage frontier using FIFO queue.

Iterative Deepening Search



Combine these two strengths.

The core problems in DFS are a) *not optimal*, and b) *not complete* ... because it fails to explore other branches.

Otherwise it's a very nice algorithm!

Iterative Deepening:

- Run DFS to a fixed depth z.
- Start at z=1. If no solution, increment z and rerun.



IDS

Optimal for constant cost! Proof?

How can that be a good idea? It duplicates work.

Sure but:

- Low memory requirement (equal to DFS).
- Not many more nodes expanded than BFS. (About twice as many for binary tree.)

IDS (Reprise)

IDS

Key Insight:

• Many more nodes at depth m+1 than at depth m.

MAGIC.

"In general, iterative deepening search is the preferred uninformed search method when the state space is large and the depth of the solution is unknown." (R&N)

Next Week

Informed searches ... what if you know something about the the solution?

What form should such knowledge take?

How should you use it?