Tree Search: The Code

willie

Announcements

Assignment 2 updates

Assignment 2 due Monday

Assignment 3 released Monday

Midterm Monday on 11/5

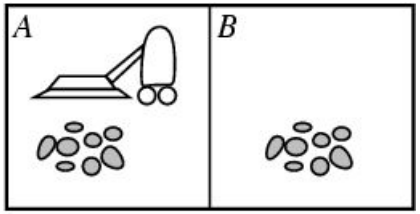
- Last year’s exam will be distributed the week before
- Friday before will have some review

Also 11/5: Bilge Mutlu

First: Defining the Problem

What are the elements of a problem?

First: Defining the Problem



1. **Formulate problem and goal**
2. Search for a sequence of actions that will lead to the goal (the policy)
3. Execute the actions one at a time

Well-defined problem:

(State space)

Initial state

Goal test

Actions/Successor function

Path cost

Problem code

Try it with simple math

Problem

- Initial: some integer
- Goal: some integer
- Actions: +1, *5, *2
- Path cost: 1

Let’s look at it

Tree Search Algorithm

1. Add the initial state (root) to the <fringe>
2. **Choose a node (curr) to examine from the** <fringe>
   (if there is nothing in <fringe> - FAILURE)
3. Is curr a goal state?
   If so, SOLUTION
   If not, continue
4. Expand curr by applying all possible actions (add the new resulting states to the <fringe>)
5. Go to step 2

Now the code

How to do

DFS?

BFS?

Greedy best-first?

A*?

Fringe

DFS

- LIFO Stack

BFS

- FIFO Queue

Informed

- PriorityQueue
- (and need evaluation function)

Try it with simple math

Problem

- Initial: some integer
- Goal: some integer
- Actions: +1, *5, *2
- Path cost: 1

Let’s look at it

Good ole 8-puzzle

Problem

Initial: arrangement of 8 pieces in 3x3: e.g., [4,1,2,0,5,6,7,8]

Goal: pieces in order: [0,1,2,3,4,5,6,7,8]

Actions: up, down, left, right (not all always available)

Path cost: 1

Let’s look at it

A simplistic social model

Problem

- Initial: integer values for acquaintance, positivity, synchrony, information
    - [0, 0, 0, 0]
- Goal: some information acquired, some positivity
    - [0, 1, 0, 10]
- Path cost: 1

Actions:

- greet: increase acquaintance
- well-being inquiry: increase positivity
- pause: increase synchrony
- info exchange: increase information

Adversarial

How do we extend this for adversarial problems?

- How do we represent the problem?
- How do we modify the algorithm?