Solving Problems by Searching - Uninformed Search

1. Introduction

All AI tasks involve searching.

**General idea:**
- You know *where you are* (initial state) and *where you want to be* (target states)
- You know the available actions that you could take to solve your problem
- You don't know which sequence of actions will get you to a solution
- You can search through all the possible sequences to find one that will give a solution.

**The scenario:**
Initial state
Target (Goal) states
A set of intermediate states
A set of operations that move the agent from one state to another.

**The task:**
Find a sequence of operations that will move the agent from the initial state to a target state. *Solved in terms of searching a graph*

The set of all states: **search space**
**Actions:** may be physical actions (such as move from town A to town B, or put block C on the table) or may be more abstract actions, like theorem proving steps (we may be searching for a sequence of steps that will allow us to prove X from the set of facts S).

**Examples:**

Puzzles: Find a sequence of actions to solve the puzzle
Chess: Find the sequence of moves that will result in winning the game.
Travel: Find a route from the starting place to the destination.

2. **Problem-solving agents**

Decide what to do by finding a sequence of actions that lead to desirable states

2.1. **Goal formulation**

The desirable states constitute the goal of the agent.
The task here for the developer is to decide how the desirable states (the target states) would be described.
- by listing
- by property

2.2. **Problem formulation**

Problem formulation is the process of deciding **which actions and states to consider**.
Questions:
  - Given the initial state and a set of desirable states, how can we describe the intermediate states?
  - Given the description of states, what are the actions that will take us from one state to another?
  - Given a set of states and a set of actions, what are the applicable conditions, i.e. for a given state which actions are applicable?

2.3. **Search for a solution**

A search algorithm has:
  - **input:** the problem as formulated in step 2.2.
  - **output:** sequence of actions that lead from the initial state to a target state.
2.4. State Space Search

**Basic idea:** 1. You have to decide on a representation of
   - the state of the world,
   - the available actions in the domain,

2. Go systematically through all possible sequences to find a sequence of actions that will get you from your initial to target state.

- **initial state** - where the agent is
- **operator set** - set of action to go from one state into another
- **goal test** - a test to see if the agent is in a goal state
- **path cost** - the cost of the path from the initial to the goal state - the sum of the costs of each edge in the path.

**State space** - the set of states generated by the set of operators.

**Measuring the performance**
- Does the agent find a solution
- Is the solution good
- What was the cost of the search

**Search tree - built during the search process**

- **Root** - corresponds to the initial state
- **Nodes**: correspond to intermediate states (two different nodes may correspond to one and the same state)
- **Links** - correspond to the operators applied to move from one state to the next state

**Node** description:
- The corresponding state
- The parent node
- The operator applied
- The length of the path from the root to that node
- The cost of the path

To expand a node means to apply operators to the node and obtain next nodes in the tree i.e. to generate its children.

**successor nodes**: obtained from the current node by applying the operators.
Note the distinction between a node and a state. A node is part of the search tree. A state is part of the agent’s world. One and the same state can be reached via different paths in the search tree. Several nodes may correspond to the same state.

2.5. Search strategies

A. Uninformed vs. informed
   - Uninformed search (blind search)- no information about the number of steps.
   - Informed search (heuristic search) - choose "the best " direction in the search tree using some heuristics.

B. Depth-first vs. breadth-first
   - Depth-first search: explore the tree in depth, (use a stack)
   - Breadth-first search: generate the successors for all nodes at a given level before proceeding with the next level of the tree (use a queue)

Comparison of depth-first and breadth-first search

**Length of path:** breadth-first finds the shortest path first.
**Memory:** depth-first uses less memory
**Time:** If the solution is on a short path - breadth first is better, if the path is long - depth first is better.

3. Generate and test method

Initially stack/queue is empty
Store root (initial state) in stack/queue

While there are unexpanded nodes in stack/queue DO:
   Take a node
   While there are applicable operators DO:
      Generate next state/node by applying a relevant operator
      Test to see if the goal state is reached.
      If yes - stop
      Test to see if the state is on the path from the root to the generated node
      If no - add the generated state/node into a stack/queue

To get the solution: Print the path from the root to the found goal state
Comments:
   The second test is absolutely necessary in depth-first search, it may be omitted in breadth first search
4. Toy problems and real world problems

**Toy problems** are used mainly for the purpose of testing search algorithms. They have a small amount of states and operators and it is relatively easy to describe them.

Real world problems are much more difficult to describe. Some real world problems are:
- Route finding
- The travelling salesperson problem
- Robot navigation

Sources of difficulty:
- Large number of states, and /or
- Large number of operators
- Uncertainty in the outcome of an action

**Examples of toy problems:**

**Problem 1:**
```
``You are given two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring markers on it. There is a tap that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug''.
```

We have to decide:
- representation of the problem state, initial and final states
- representation of the actions available in the problem, in terms of how they change the problem state.
- what would be the cost of the solution
  - Path cost
  - Search cost

For the example:
- problem state: pair of numbers (X,Y): X - water in jug 1, Y - water in jug 2.
  - Initial state: (0,0)
  - Final state: (2, X)
Available actions (operators):

<table>
<thead>
<tr>
<th>Description</th>
<th>Pre-conditions on (X,Y)</th>
<th>Action (Post-conditions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1. Fill A</td>
<td>X &lt; 3</td>
<td>(3, Y)</td>
</tr>
<tr>
<td>O2. Fill B</td>
<td>Y &lt; 4</td>
<td>(X,4)</td>
</tr>
<tr>
<td>O3. Empty A</td>
<td>X &gt; 0</td>
<td>(0,Y)</td>
</tr>
<tr>
<td>O4. Empty B</td>
<td>Y &gt; 0</td>
<td>(X,0)</td>
</tr>
<tr>
<td>O5. Pour A into B</td>
<td>a. X &gt; 4 - Y</td>
<td>(X + Y - 4, 4)</td>
</tr>
<tr>
<td></td>
<td>b. X ≤ 4 - Y</td>
<td>(0, X + Y)</td>
</tr>
<tr>
<td>O6. Pour B into A</td>
<td>a. Y &gt; 3 - X</td>
<td>(3, X + Y - 3)</td>
</tr>
<tr>
<td></td>
<td>b. Y ≤ 3 - X</td>
<td>(X + Y, 0)</td>
</tr>
</tbody>
</table>

The actions can be applied only if the conditions hold.

Generate and test method:

Take a node,
Generate next nodes by applying relevant operators (not all operators are applicable for a given node!)
Test to see if the goal state is reached.
If yes - stop
If no - add the generated nodes into a stack/queue (do not add if the node has already been generated on the path to the root) and repeat the procedure.

To get the solution:
Record the paths
**Problem 2:** A farmer has to move a goat, a cabbage and a wolf from one side of a river to the other side using a small boat. The boat can carry only the farmer and one more object (either the goat, or the cabbage, or the wolf).

If the farmer leaves the goat with the wolf alone, the wolf would kill the goat.
If the goat is alone with the cabbage, it will eat the cabbage.

How can the farmer move all his property safely to the other side of the river?

1. Identify the initial and the target states and the goal test
2. Choose appropriate representation of the states.
3. Decide on operators to change the states. Describe how each operator changes the states.
4. Identify restrictions: are all operators applicable to all states?
5. Decide on how the cost would be measured.
6. Solve the problem

**Problems with Simple Search Techniques:** Combinatorial Explosions with high branching factor of the state space representation