Problem Solving by Searching

Russell and Norvig, chapter 3.1-3.3
Puzzles!
The missionaries and cannibals problem

Goal: transport the missionaries and cannibals to the right bank of the river.

Constraints:
- Whenever cannibals outnumber missionaries, the missionaries get eaten
- Boat can hold two people and can’t travel empty
Formulating the problem

A state description that allows us to describe our state and goal:

$$(M_L, C_L, B)$$

- $M_L$: number of missionaries on left bank
- $C_L$: number of cannibals on left bank
- $B$: location of boat (L,R)

- Initial state: (3,3,L)  Goal: (0,0,R)
Problem solving agents

- Problem Formulation
  - States and actions (successor function).

- Goal Formulation
  - Desired state of the world.

- Search
  - Determine a sequence of actions that lead to a goal state.

- Execute
  - Perform the actions.

- Assumptions:
  - Environment is fully observable and deterministic
  - Agent knows the effects of its actions
Graph formulation of the problem

- Nodes: all possible states.
- Edges: edge from state \( u \) to state \( v \) if \( v \) is reachable from \( u \) (by an action of the agent)
- Edges for missionaries and cannibals problem?
- Problem is now to find a path from \((3,3,L)\) to \((0,0,R)\).
- In general, paths will have costs associated with them, so the problem will be to find the lowest cost path from initial state to the goal.
Formulating a search problem

- State space $S$ (nodes)
- Successor function: the states you can move to by an action (edge) from the current state
- Initial state
- Goal test: is a state $x$ a goal?
- Cost
State: 33L three missionaries and three cannibals on left bank, boat is on left bank
Actions (operators):
CCR - transport two cannibals to the right bank
CMR - transport a missionary and a cannibal to the right bank
Why no MR or MMR transition from this state?
The (partially expanded) search tree

Actions:
- **CCR** - transport two cannibals to the right bank
- **CML** - transport a missionary and a cannibal to the left bank
Repeated states

The search tree contains duplicate nodes, since the underlying graph in which we are searching is not a tree!
The state space graph

To address this, need to track previously explored states
Aside: Searching the graph

function GRAPH-SEARCH(problem) returns a solution, or failure

initialize the frontier using the initial state of problem
initialize the explored set to be empty

loop do

if the frontier is empty then return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state then return the corresponding solution

add the node to the explored set

expand the chosen node

add the resulting nodes to the frontier only if not in the frontier or explored

The difference between BFS and DFS is in how the node is chosen and/or how it is added to the frontier: queue versus stack.
Often it is not feasible (or too expensive) to build a complete representation of the state graph. A problem solver must construct a solution by exploring a small portion of the graph.
Searching the state space

Search tree
Searching the state space
Searching the state space
Searching the state space
Searching the state space
Searching the state space

Search tree
The 8 puzzle

- States?
- Initial state?
- Actions?
- Goal test?
- Path cost?
The 8 puzzle

- **States?** Integer location of each tile. How many of them are there?
- **Initial state?** Any state
- **Actions?** (tile, direction)
  - where direction is one of \{Left, Right, Up, Down\}
- **Goal test?** Check whether goal configuration is reached
- **Path cost?** Number of actions to reach goal
- **Is the search graph a tree?**
(n^2-1)-puzzle

\begin{array}{ccc}
8 & 2 & \\
3 & 4 & 7 \\
5 & 1 & 6 \\
\end{array}

\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & \\
\end{array}
The 15-puzzle

Sam Loyd offered $1,000 of his own money to the first person who would solve the following problem:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
But no one ever won the prize !!
Why?
A solution is a path connecting the initial node to a goal node (any one)
Solution to the Search Problem

- A solution is a path connecting the initial node to a goal node (any one)
- The cost of a path is the sum of the edge costs along this path
- An optimal solution is a solution path of minimum cost
- There might be no solution!
Path Cost

- An edge cost is a positive number measuring the “cost” of performing the action corresponding to the edge, e.g.:
  - 1 in the 8-puzzle example

- We will assume that for any given problem the cost $c$ of an edge always satisfies:
  
  $$c \geq \epsilon > 0,$$

  where $\epsilon$ is a constant

  - Why? Has to do with the cost of arbitrarily long paths
**Goal State**

- It may be explicitly described:

- or partially described:

- or defined by a condition, e.g., the sum of every row, of every column, and of every diagonal equals 30

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
```

(“a” stands for “any” other than 1, 5, and 8)
Another example: the 8 queens problem

- Incremental vs. complete state formulation:
  - Incremental formulation starts with an empty state and involves operators that augment the state description
  - A complete state formulation starts with all 8 queens on the board and moves them around
8 queens problem: representation is key

Incremental formulation
- States? Any arrangement of 0 to 8 queens on the board
- Initial state? No queens
- Actions? Add queen in empty square
- Goal test? 8 queens on board and none attacked
- Path cost? None
  $64 \times 63 \times \ldots \times 57 \sim 3 \times 10^{14}$ states to investigate

Is the search graph a tree?
A better representation

Another incremental formulation:

- **States?** n (between 0 and 8) queens on the board, one in each of the n left-most columns; **no queens attacking each other**.
- **Initial state?** No queens
- **Actions?** Add queen in leftmost empty column such that it does not attack any of the queens already on the board.
- **Goal test?** 8 queens on board

\[2057\] possible sequences to investigate
8 queens solved

Source:
https://sites.google.com/a/lclark.edu/drake/courses/ai/search-and-n-queens
n-queens problem

- A solution is a goal node, not a path to this node (typical of design problem)
- Number of states in state space:
  - 8-queens $\rightarrow$ 2,057
  - 100-queens $\rightarrow$ $10^{52}$
- But techniques exist to solve n-queens problems efficiently for large values of n
Path planning

What is the state space?
Path planning

What is the state space?

Formulation #1
Cost of one horizontal/vertical step = 1
Cost of one diagonal step = \(-2\)

Formulation #2
sweep-line

Optimal Solution
This path is the shortest in the discretized state space, but not in the original space.
An alternative formulation

Visibility graph

Cost of a step is the length of the step
An alternative formulation

The solution in this space is the same as in the continuous space!

Lozano-Pérez, Tomás; Wesley, Michael A. (1979), "An algorithm for planning collision-free paths among polyhedral obstacles", Communications of the ACM 22 (10): 560–570,
Assumptions in Basic Search

- The world is static
- The world is discretizable
- The world is fully observable
- The actions are deterministic

Search remains an important tool even if not all these assumptions are satisfied
Applications

- Search plays a key role in many AI systems, e.g.:
  - Route finding (google/mapquest, internet, airline)
  - VLSI Layout
  - Robot navigation.
  - Pharmaceutical drug design, protein design
  - Video games