Agents that Search
Chapter 3: Problem Solving by Searching

• In which we see how an agent can find a sequence of actions that achieves its goals when no single action will do.

• Such agents must be able to:
  – Formulate a goal
  – Formulate the overall problem
  – Find a solution
Single-State Problem Formulation

• A problem is defined by six items:
  1. A set of states
  2. Start/initial state
  3. Available actions for each state
  4. Successor/action function (which actually defines all reachable states)
  5. Goal test
  6. Path cost (additive)
     e.g., sum of distances, number of actions executed, etc.
     \( C(x,a,y) \) is the step cost, assumed to be \( \geq 0 \)
Appropriate environment for Searching Agents

- Fully Observable
- Deterministic
- Episodic
- Static
- Discrete
- Single-Agents
- Partially Observable
- Stochastic
- Sequential
- Dynamic
- Continuous
- Multi-Agent
Appropriate environment for Searching Agents

- Fully Observable
- Deterministic
- Episodic
- Static
- Discrete
- Single-Agents

- Partially Observable
- Stochastic
- Sequential - Normally
- Dynamic
- Continuous
- Multi-Agent - Either
A Problem

• Three IT directors and three hackers
• Want to cross a river using one canoe.
• Canoe can hold up to two people.
• Can never be more hackers than IT on either side of the river.

• Aim: To get all safely across the river without any IT losing their data.

• States?? Actions?? Goal test?? Path cost??
Single-State Problem Formulation

- A problem is defined by six items:
  1. A set of states
  2. Start/initial state
  3. Available actions for each state
  4. Successor/action function (which actually defines all reachable states)
  5. Goal test
  6. Path cost (additive)
     - e.g., sum of distances, number of actions executed, etc.
     - $C(x,a,y)$ is the step cost, assumed to be $\geq 0$
States

• We can show the number of IT, hackers, and canoes on each side of the river
  – [1,1,0,2,2,1]
Problem Representation

• States
  – However, since the system is closed, we only need to represent one side of the river, as we can deduce the other side.
  – We will represent the starting side of the river, and omit the ending side.
  – So start state is:
    • [3,3,1]
Problem Representation

• Goal State(s)
  – [0,0,0]
  – TECHNICALLY also [0,0,1]
Problem Representation

• Successor Function (5 actions)
  1. Move one IT.
  2. Move one hacker.
  3. Move two IT.
  4. Move two hackers.
  5. Move one of each.
Problem Representation

• **Successor Function**
  
  – To be a little more mathematical/computer like, we want to represent this in a true successor function format…
    
    S(state) ⇒ state
    
    1. Move one IT across the river.
      
      S([x,y,1]) ⇒ [x-1,y,0]
      
      S([x,y,0]) ⇒ [x+1,y,1]

[Note, this is a slight simplification.
We also require, 0 ≤ x, y, [x+1 or x-1] ≤ 5 ]
Problem Representation

• Successor Function

\[
S([x,y,0]) \Rightarrow [x+1,y,1] \quad //1 \text{ IT} \\
S([x,y,1]) \Rightarrow [x-1,y,0]
\]

...

\[
S([x,y,0]) \Rightarrow [x,y+1,1] \quad //1 \text{ hacker} \\
S([x,y,1]) \Rightarrow [x,y-1,0]
\]

...

\[
S([x,y,0]) \Rightarrow [x+1,y+1,1] \quad //1 \text{ of each} \\
S([x,y,1]) \Rightarrow [x-1,y-1,0]
\]
Problem Representation

• Path Cost
  - One unit per trip across the river.
Let's look at the solution
Problem Solving Agents
Tree Search Example
Tree Search Example
General Graph Search: Text

1: procedure \texttt{Search}(G, S, \text{goal})
2: \hspace*{1em} \textbf{Inputs}
3: \hspace*{1em} \(G\): graph with nodes \(N\) and arcs \(A\)
4: \hspace*{1em} \(s\): start node
5: \hspace*{1em} \text{goal}: Boolean function of nodes
6: \hspace*{1em} \textbf{Output}
7: \hspace*{1em} path from \(s\) to a node for which \(\text{goal}\) is true
8: \hspace*{1em} or \(\bot\) if there are no solution paths
9: \hspace*{1em} \textbf{Local}
10: \hspace*{2em} \text{Frontier}: set of paths
11: \hspace*{2em} \text{Frontier} := \{\langle s \rangle\}
12: \hspace*{2em} \textbf{while} \text{Frontier} \neq \{\} \textbf{do}
13: \hspace*{3em} \textbf{select and remove} \langle n_0, \ldots, n_k \rangle \text{ from Front}er
14: \hspace*{3em} \textbf{if} \text{goal}(n_k) \text{ then}
15: \hspace*{4em} \text{return} \langle n_0, \ldots, n_k \rangle
16: \hspace*{3em} \text{Frontier} := \text{Frontier} \cup \{\langle n_0, \ldots, n_k, n \rangle : \langle n_k, n \rangle \in A\}
17: \hspace*{2em} \text{return} \bot

Figure 3.4: Search: generic graph searching algorithm
General Graph Search: Alternate

```plaintext
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 solution
    add the node to the explored set
    expand the node, adding the resulting nodes to the frontier
      iff not in the explored set [or Frontier]
  loop do
```