Search – Chapter 3

◊ Problem-solving agents
◊ Problem types
◊ Problem formulation
◊ Example problems
◊ Basic search algorithms
Goal Oriented Search – Ex: 1

On holiday in Romania; currently in Arad.
Flight leaves tomorrow from Bucharest

Formulate goal:
be in Bucharest

Formulate problem:
states: various cities
operators: drive between cities

Find solution:
sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest
Problem Type

Deterministic, accessible $\rightarrow$ single-state problem

Deterministic, inaccessible $\rightarrow$ multiple-state problem

Non-deterministic, inaccessible $\rightarrow$ contingency problem
- must use sensors during execution
- solution is a tree or policy
- often interleave search, execution

Unknown state space $\rightarrow$ exploration problem ("online")
Single State Problem Definition

A problem is defined by four items:

initial state  e.g., “at Arad”

operators (or successor function $S(x)$)
  e.g., Arad $\rightarrow$ Zerind  Arad $\rightarrow$ Sibiu  etc.

goal test, can be
  explicit, e.g., $x = \text{“at Bucharest”}$
  implicit, e.g., $\text{NoDirt}(x)$

path cost (additive)
  e.g., sum of distances, number of operators executed, etc.

A solution is a sequence of operators leading from the initial state to a goal state
State Space Definition

Real world is absurdly complex
⇒ state space must be *abstracted* for problem solving

(Abstract) state = set of real states

(Abstract) operator = complex combination of real actions
e.g., “Arad → Zerind” represents a complex set
of possible routes, detours, rest stops, etc.
For guaranteed realizability, any real state “in Arad”
must get to *some* real state “in Zerind”

(Abstract) solution =
set of real paths that are solutions in the real world

Each abstract action should be “easier” than the original problem!
Eight State Puzzle

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

Start State

Goal State

states??
operators??
goal test??
path cost??
Robotic Arm

states??: real-valued coordinates of robot joint angles parts of the object to be assembled

operators??: continuous motions of robot joints

goal test??: complete assembly with no robot included!

path cost??: time to execute
Converting the Problem to a Graph
Basic Uninformed Search

Basic idea:

offline, simulated exploration of state space
by generating successors of already-explored states
(a.k.a. expanding states)

```plaintext
function GENERAL-SEARCH(problem, strategy) returns a solution, or failure
initialize the search tree using the initial state of problem
loop do
  if there are no candidates for expansion then return failure
  choose a leaf node for expansion according to strategy
  if the node contains a goal state then return the corresponding solution
  else expand the node and add the resulting nodes to the search tree
end
```
A state is a (representation of) a physical configuration. A node is a data structure constituting part of a search tree. It includes parent, children, depth, path cost $g(x)$. States do not have parents, children, depth, or path cost!

The \texttt{Expand} function creates new nodes, filling in the various fields and using the \texttt{Operators} (or \texttt{SuccessorFn}) of the problem to create the corresponding states.
Search Strategies and Effectiveness

A strategy is defined by picking the order of node expansion

Strategies are evaluated along the following dimensions:
- completeness—does it always find a solution if one exists?
- time complexity—number of nodes generated/expanded
- space complexity—maximum number of nodes in memory
- optimality—does it always find a least-cost solution?

Time and space complexity are measured in terms of
- $b$—maximum branching factor of the search tree
- $d$—depth of the least-cost solution
- $m$—maximum depth of the state space (may be $\infty$)
Uninformed Search

*Uninformed* strategies use only the information available in the problem definition.

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search