Foundations of Artificial Intelligence
CS472 Lecture #2
Bart Selman

Slide CS472–1

Today’s Lecture
Problem Solving as Search
Readings: R&N, Chapter 3.

Slide CS472–2
Human Problem Solving

Search is a central topic in AI
  — Automated reasoning is a natural search task
  — More recently: Given that almost all AI formalisms (planning, learning, etc.) are NP-Complete or worse, some from of search is generally unavoidable (no “smarter” alg. available).

Slide CS472–3

Defining a Search Problem

State space – described by an initial state and the set of possible actions available (operators). A path is any sequence of actions that lead from one state to another.

Goal test – applicable to a single state to determine if it is the goal state.

Path cost – relevant if more than one path leads to the goal, and we want the shortest path.

Note: very general formulation. Can be somewhat unnatural.

Slide CS472–4
Two Examples

Cryptarithmetic
(Newell and Simon 1972)

The 8-puzzle

Example I: Cryptarithmetic

\[
\begin{align*}
\text{SEND} & \\
+ \text{MORE} & \\
\hline
\text{MONEY} & 
\end{align*}
\]

Find substitution of digits for letters such that the resulting sum is arithmetically correct. Each letter must stand for a different digit.
Cryptarithmetic, cont.

**States:** a (partial) assignment of digits to letters.

**Operators:** the act of assigning digits to letters.

**Goal test:** all letters have been assigned digits and sum is correct.

**Path cost:** zero. All solutions are equally valid.

---

State Space Search

**Input:**
- Start state
- Goal state or goal test
- Operators

**Output:** legal sequence of nodes from initial node to goal node

Search space is generally not stored in its entirety.
example search space cryptarithmetic
DFS (depth-first search)

Cryptarithmetic
Is this (DFS) how humans tackle the problem?
And, if not, what do humans do?
Human problem solving appears much more sophisticated!

- For example, we derive new constraints on the fly.
- In a sense, we try to solve problems with little or no search!

In example, we can immediately derive that $M = 1$.
It then follows that $S = 8$ or $S = 9$. Etc. (derive more!)

---

Capturing such human problem solving strategies is surprisingly difficult. For example, how do we know to first consider assigning $M$?

Constraint programming techniques do provide some steps towards this kind of problem solving (next lecture).

Fortunately, computers are very good at fast search!
Search speed can compensate for lack of higher-level insights into the problem structure.
Example II: The 8-Puzzle

States: Specifies the location of each of the eight tiles in one of the nine squares

Operators: blank moves left, right, up, down

Goal test: state matches the goal configuration

Path cost: each step costs 1, so path cost = length of path

Slide CS472–13

Solving the 8-Puzzle

What would the search tree look like after the start state was expanded?

Slide CS472–14
Solving a Search Problem

Search problems are solved by searching the state space.
Search process builds up a search tree over the search space.

Root = the initial state
Leaves = states that do not have successors in the tree
  (none exist or node has not been expanded yet).
Search strategy = algorithm for deciding which leaf node
to expand next.

Generic Search Algorithm

L = make-queue/stack(initial-state)
loop
  node = remove-front(L)
  if goal-test(node) = true return( node )
  S = successors(node, operators)
  insert(S,L)
until L is empty
return failure
Search procedure defines a search tree
root node — initial state
children of a node — successors of the node
fringe of tree — L: nodes not yet expanded

stack: Depth-First Search (DFS).
queue: Breadth-First Search (DFS).
Aside: Actual implementation may not use stack/queue.