CS 356 - Artificial Intelligence

State Space Search

Dr. Stephen P. Carl

Quote of the Day

[Chess is] the drosophila melanogaster

of machine intelligence.

- Donald Michie

State Space Representations

A *state space* can be represented with a tree or, more generally, a graph. Each arc in the graph is a *move* from one step toward a solution to another; each state represents a partial solution.

Each such graph has one or more start states, the initial problem or configuration, and one or more goal states, each of which is a possible solution.

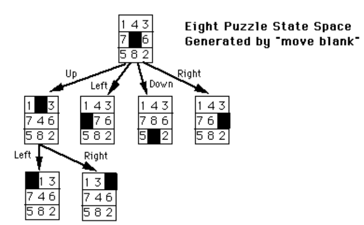
Nodes and arcs may have a label describing the state or move. *State space search* is the process of finding a (hopefully optimal) path from the start state to a goal state.

SEWANEE
The University of the South

Example Representations

For any problem one must determine what the start, goal, and intermediate states represent, and what actions are legal moves or steps of the solution.

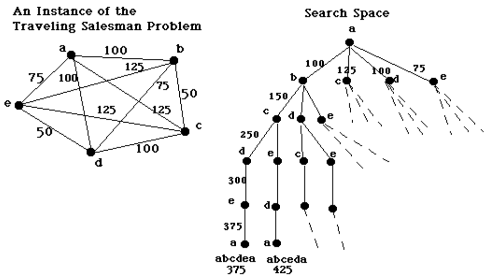
Example: In the 8-Puzzle, the goal is for tiles 1 through 8 to be arranged into non-decreasing order.



According to Russell, optimal solution of the *n-* puzzle family is NP-hard.

SEWANEE
The University of the South

Classic Traveling Salesman Problem



- Each city to visit is a node in a graph, and the start state is the home city. State space is a tree.
- Arcs are moves between cities (roads, rail links, flights) and are labeled with the distance between them.
- The goal is not a state, but a property of the entire path: find the least total distance traveled.

Search Space Issues

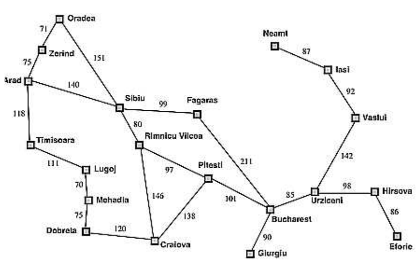
For some problems, the state space forms a non-acyclic graph, which can cause search to loop indefinitely. Cycles occur when a series of moves returns to an already-visited state. Search algorithms must detect and remove cycles.

For example, a search of the state space for the 8-puzzle has cycles, because the tiles can easily be moved into a previously-searched state.

For other problems, the state space forms a tree (directed acyclic graph). The TSP solver is one example, because no moves can return to a previous state. The tic-tac-toe solver is another. For these problems the overhead of cycle detection is not needed.

Romania Travel Problem

On vacation in Romania, I'm in Arad and need to be in Bucharest for the flight home tomorrow. What's the quickest route?



Romania Travel Problem

Cast as a search problem, we need four things:

- **Initial state:** I'm at Arad
- **Successor function:** *S(x)* is a set of action-state pairs of the form *S(Arad) -> {(Arad->Zerind, Zerind), (Arad->Sibiu, Sibiu), etc.}*
- **Goal test:** *x = I'm at Bucharest*
- **Path cost:** sum of distances

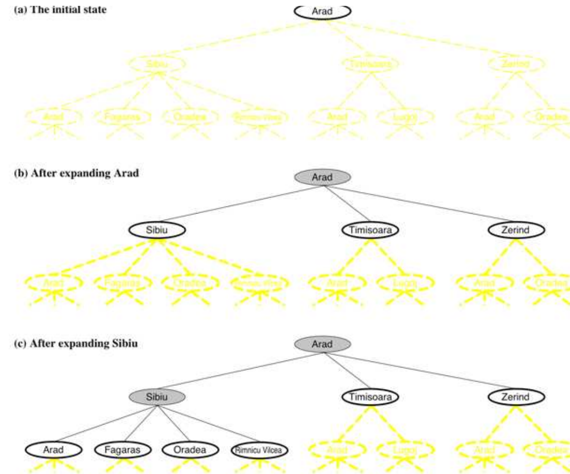
The solution is the sequence of actions that take us from an initial state to a goal state.

Basic Tree Search algorithm

The idea behind the family of tree-search algorithms is to simulate exploration of the state space. For each goal visited, the algorithm generates a set of successor goals using the Successor function.

```
function TreeSearch(problem, strategy)
   search-tree <- initial goal
   while no solution do
     if no node can be expanded, return fail
     choose leaf node to expand according to strategy
     if node is a goal-state return solution
     else expand node and add resulting nodes to tree
```


Romania Travel Search



Really Hard Problems

Chess:

- Each board configuration is a state; the start state is the initial position of game pieces.
- Arcs are legal moves of the various pieces. A goal state might be a board cleared of opponent pieces, or checkmate.
- Legal moves may revisit previous states, so the state space is a graph.
- How many possible states are generated by a given state?

References

8-Puzzle Image: Ralph Morelli, Trinity College

Romania Problem from *Artificial Intelligence: A Modern Approach* by Stuart Russell and Peter Norvig

Romania map courtesy Stuart Russell from *http://aima.cs.berkeley.edu/slides-pdf*