Ss
State-space Search

State Spaces

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2018

Ss
State-space Search

Outline

- Motivation: A foundation, plus problem solving
- Example 1: The Missionaries and Cannibals puzzle
- Example 2: Towers of Hanoi
- State, operator, state space
- Operators, preconditions, moves
- State space as a tree
- Example 3: The Traveling Salesman Problem
- State space as a graph
- Blind search methods: DFS, BFS.
- Example 4: Farmer, Fox, Chicken and Grain puzzle
- Combinatorics of the Painted Squares Puzzle

Ss
State-space Search

Motivation

We begin our technical coverage of AI with the *classical theory of problem solving*, which forms a foundation on which most other AI techniques rely.

A useful form of intelligence is the ability to solve problems. The standard AI approach to solving a problem is to formulate it as a search through a space of possible solutions or a space of partial solutions and then systematically search the space.

This was the idea behind the "General Problem Solver" program created by A. Newell, H. Simon in 1961.

Ss
State-space Search

"Looking Ahead"

A key idea in search is "looking ahead."

An agent should answer the question:

What will be the consequences of different sequences of actions or possible decisions?

Problem solving via search is sometimes called "planning."

Missionaries and Cannibals Puzzle

http://www.learn4good.com/games/puzzle/boat.htm

Missionaries and Cannibals Puzzle

MMM
CCC
B

| R |

| R |

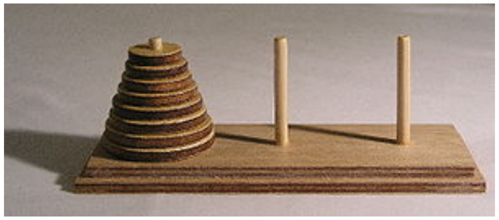
MMM
CCC
B

Missionaries and Cannibals Puzzle

What sequence of legal moves takes us from the initial state to the goal state?

This sequence is a *solution*.

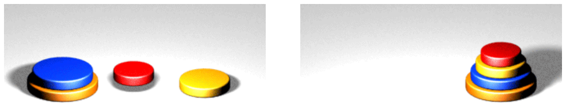
Towers of Hanoi



What sequence of legal moves takes us from the initial state to the goal state?

http://en.wikipedia.org/wiki/Tower_of_Hanoi

Towers of Hanoi



(a) An intermediate state, (b) goal state.

http://en.wikipedia.org/wiki/Tower_of_Hanoi

States

A *state* consists of a complete description or snapshot of a situation that can be arrived at during the solution of a problem.

Initial state: the starting position or arrangement, prior to any problem-solving actions being taken.

Goal state: the final arrangement of elements or pieces that satisfies the requirements for a solution to the problem.

State Space

The set of all possible states – the arrangements of the elements or components to be used in solving a problem – forms the space of states for the problem. This is known as the *state space*.

We will often use the symbol Σ to represent state space.

Moves

A *move* is a transition from one state to another.

An *operator* is a representation of a partial function* (from states to states) that can (sometimes) be applied to a state to produce a new state, and also, implicitly, a move.

A sequence of moves that leads from the initial state to a goal state constitutes a *solution.*

*A partial function is a mapping from a domain to a range, but which might not be defined for all elements of the domain. If we restrict the domain to elements on which the mapping is defined, then the mapping is a function; i.e., there is a single unambiguous range element associated with each domain element. Example: (Real) square root is a partial function of the real numbers, because it is undefined on negative values, but it acts as a function on non-negative values.

Operator Preconditions

We will represent operators as triples:
(name, precondition_function, state_transformation_function)

Precondition: A necessary property of a state in which a particular operator can be applied. (implemented as a predicate, i.e., Boolean function)

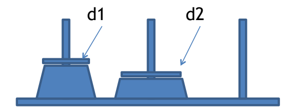
Example: In checkers, a piece may only move into a square that is vacant. Thus, Vacant(place) is a precondition on moving a piece into place.

Example: In Chess, a precondition for moving a rook from square A to square B is that all squares between A and B be vacant. (A and B must also be either in the same row or the same column.)

A conjunction of such preconditions that is sufficient to make the application of an operator legal can serve as "the" precondition for the operator.

Operator Representation

Example, in Towers of Hanoi puzzle:



Operator:

Name: "Move-1-2"
Purpose: Move a disk from Peg 1 to Peg 2
Precondition (predicate):
There is a disk d1 on Peg 1, and
d1 is the topmost disk on Peg 1,
and either there is no disk on Peg 2 or
there is a disk d2 on Peg such that
d2 is the topmost disk on Peg 2,
and diameter of d2 is greater than diameter of d1.
State transformation (function):
Remove disk d1 from Peg 1 and
put it on top of disk d2 on Peg 2.

Problem Spaces

A state space Σ, together with a set of operators Φ defines a *problem space* (Σ, Φ).

Note that the same state space can be part of multiple problem spaces.
Let $\Sigma = Z^+$. (non-negative integers)
Let Φ_a = {add1}, Φ_b = {add4, subtract3}

Problem-Space Graphs

The problem-space graph for a problem space (Σ, Φ) consists of one node s_i for each $\sigma_i \in \Sigma$, and an edge (s_i, s_j, k) whenever $\varphi_k(\sigma_i) = \sigma_j$.

That is, whenever there is a valid move from σ_i to σ_j using operator φ_k, there is a directed edge from the node for σ_i to the node for σ_j and the edge is labeled with the name or index of φ_k.

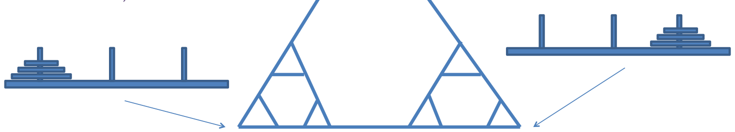
A TOH Problem-Space Graph

Consider a Towers of Hanoi puzzle.
In this puzzle, each operator has an inverse operator, and so the graph is essentially undirected (although the labels for the edges in each direction are different).

The TOH graphs have cycles. Nice layouts can be constructed for them.
Here is an example, with NDISKS = 3.

(For completeness, there should be labels on each edge here, but they are not shown in this illustration.)

This layout is related to the Sierpinski triangle



Farmer, Fox, Chicken & Grain

A farmer has to get his fox, chicken, and grain across the river.

The boat can hold only the farmer and one item.

The fox cannot be left alone with the chicken.

The chicken cannot be left alone with the grain.

How does the farmer get everything across?

States?

Operators?

Preconditions ?

Draw the problem-space graph.

Search Trees

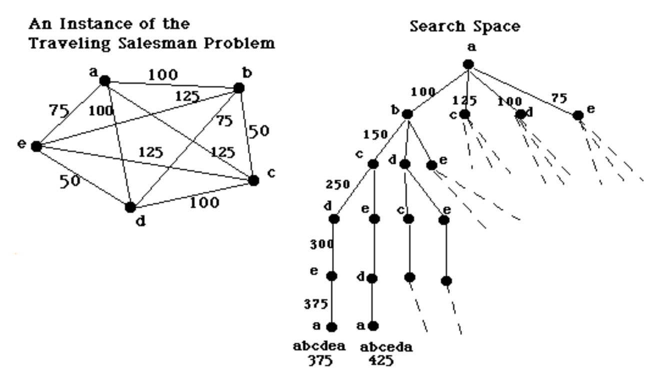
By applying operators from a given state we generate its children or *successors*.

Successors are *descendants* as are successors of descendants.

If we ignore possible equivalent states among descendants, we get a tree structure.

Depth-First Search: Examine the nodes of the tree by fully exploring the descendants of a node before trying any siblings of a node.

Example: Traveling Salesman Problem

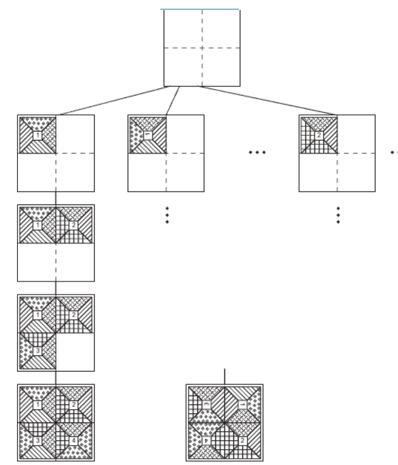


http://www.cs.trincoll.edu/~ram/cpsc352/notes/search.html

How Large Is a State Space?

- The size of a state space impacts the amount of time that might be required to search it.
- It also can impact the amount of memory required for the search (depending on which algorithm is used)
- When determining the size of a state space, we often use *combinatorics*, the branch of discrete mathematics that deals with how to count elements of various kinds of sets.

Tree of States for a 2x2 Painted Squares Puzzle



Combinatorics of the Painted Squares Puzzle

Consider placements to be unconstrained.

Branching factor:

$$b = n\_pieces\_left \cdot n\_places\ left \cdot n\_orientations$$

At the root: $b = 4 \cdot 4 \cdot 4 = 64$
At ply 1: $b = 3 \cdot 3 \cdot 4 = 36$
At ply 2: $b = 2 \cdot 2 \cdot 4 = 16$
At ply 3: $b = 1 \cdot 1 \cdot 4 = 4$

Total leaf nodes (including repetitions): 64 · 36 · 16 · 4 = 147,456.
Total nodes: 1 + 64 + 2304 + 36864 + 147456 = 186,689.

Combinatorics of the Painted Squares Puzzle

Number of filled boards using the 4 pieces, allowing violations of the side-matching constraints:

$n\_permutations \cdot n\_orientations\text{^}n\_pieces$

$4! \cdot 4^4 = 24 \cdot 256 = 6144$

If we constrain piece placements to go to the next available space on the board, then this is the number of leaf nodes.

Note that dividing 147,456 by 4! gives 6144, too.

The Combinatorial Explosion

Ss
State-space
Search

Assume the branching factor is constant.
Suppose a search process begins with the initial state.

Then it considers each of *b* possible moves. Each of those may have b possible subsequent moves.

In order to thoroughly look n steps ahead, the number of states that must be considered is
$1 + b + b^2 + . . . + b^n$.

For $b > 1$, the value of this expression grows exponentially as *n* increases. This is known as the *combinatorial explosion*.