State Spaces

1 State spaces
State space - a set of states

Two types of search problems

- Global search problems - solutions are paths (state spaces are states and transitions. Get from initial to goal state by a sequence of transitions. Each transition has a step cost. Different actions can be used for the same transitions. The aim is to find a path from initial to goal state where the total step cost for the path is either max or min depending on the problem)
- Local search problems - Have state space and state transitions but no actions or goal states. Each state has a cost associated called an objective cost. Find the state of either smallest or largest cost depending on if the problem is to minimise or maximise

How can you deal with an infinite state space? - Only ever retain in memory what has already been visited in the state space. Build states when you have to

2 Real world problems
2.0.1 8-Puzzle Problems

- Move tiles into the empty cell to reach a specific tile configuration
- Not always a solution for n problem, but can find if there is a solution in P
- If there is a solution, finding one is in P
- Finding a solution with the fewest moves is in NP

What is a state? - A particular tile configuration
How might we represent this state?

- As a **tuple** $x \in \{0,1,...,8\}^9$ so that all components are distinct (sequence of 9 integers in a list, numbers between 1 and 9)
- As a 3×3 **matrix** M over $\{0,1,...,8\}$ so that all components are distinct (just a matrix looking like the grid)

What is a state transition - Moving a tile into the empty space
How might we represent this transition?

- (x,y) is a transition iff
  - $x_i = y_i \ \forall i \in \{1,2,...,9\}$ except $\exists j,k$ s.t. $0 = x_j \neq y_j$ and $x_k \neq y_k = 0$
- (M_1, M_2) is a transition iff
  - $M_1[1,j] = M_2[i,j] \ \forall i,j \in \{1,2,3\}$ except $\exists (a,b),(c,d)$ s.t.$0 = M_1[a,b] \neq M_2[a,b]$ and $M_1[c,d] \neq M_2[c,d] = 0$
  - Either ($a = c$ and $|b = d| = 1$) or ($b = d$ and $|a - c| = 1$)
- The initial state and the goal state are as required

2.0.2 Rush hour

- Move car to the exit
- Can generalise to $n \times n$
- Finding if there is a solution is PSPACE-complete - harder than NP complete
- If there are no lorries, still PSPACE-complete

2.1 Path Problems

2.1.1 Travelling Salesman Problem

- Given a set of cities and the distance between each pair of cities, find a shortest tour of all the cities
- "cities" don't have to be cities and "distances" don't have to be distances. Could model anything
- City-to-city distances don't have to be symmetric ($A \to B$ doesn't have to be $B \to A$ (might have a hill)) nor Euclidean (where you place them on the plane doesn't have to be scaled and stuff)
  - Computing the optimal solution - **NP Hard**
  - Computing the optimal solution for the symmetric Euclidean TSP - **NP Hard**

Modelled as a search problem
What is a state?

- A list of distinct cities (a partial tour)
- A list containing each of the n cities exactly once

How might we represent this state

- As a list of i cities, where $0 \leqslant i \leqslant n$
- As a list of n cities

What might the initial state be?

- An empty list?

What is a state transition

- An ordered pair of states so that the second state is the first state incremented with an as yet unvisited city
- An ordered pair of states so that the second state is the first state except that the cities in two locations have been swapped

How might we represent this transition?

- An ordered pair of lists (p, q) so that $p = (p_1, p_2, ..., p_i)$, for some $0 \leqslant i < n$ and $q = (p_1, p_2, ..., p_i, c)$, where $c \neq p_j$, $\forall j \in \{1, 2, ..., i\}$ (add unvisited cities)
- An ordered pair of lists (p, q) so that $q = (q_1, q_2, ..., q_n)$ and $p = (p_1, p_2, ..., p_n)$ where $p_i = q_i$, $\forall i \in \{1, 2, .., n\}$ except $\exists j \neq k$ s.t. $p_j = q_k$ and $p_k = q_j$

What are the initial state and a goal state?

- *Idea 1*
  - Initial state: the empty list
  - Goal State: A list of all cities
- *Idea 2*
  - Initial state: list of cities in an arbitrary order
  - Goal state: Every state

What is the cost to move from n-1 to n cities?
The cost of moving from n-1 to nth cities, and the cost from the nth city back to the 1st

2.1.2 Disjoint Connecting Paths

- In a labyrinth of rooms connected by corridors, each person p_i, where $i = 1, 2, ..., k$ has to reach their own exit-room t_i starting at their own start-room s_i
- Start and exit rooms can only appear as terminal rooms on paths and no other room lies on more than one person's path

3 Some considerations (and subtleties)

The state space might be very large or even infinite

- There are $n!$ states in an instance of the n-puzzle problem
- There are $> 2n!$ and $n!$ states, respectively, in our two n-city TSP abstractions

So how do we store the state space?

- We describe states succinctly using appropriate data structures
- State transitions are described using roles for application. So, states are built as and when we need them
- There is a tension between how we represent states and state transitions versus the "implementation cost" of building states and establishing state transitions

In an optimization problem, how do we know we have an optimal path to a goal state

- This will be down to the search algorithm that we employ. Maybe we'll have to settle for non-optimal solutions

We need to ensure that our abstraction of the real world problem is:

- Such that solutions to the search problem and real-world problem correspond
- Such that the level of real-world abstraction is appropriate. A balance between "abstraction" complexity and "computational" complexity

4 In summary

To formally describe a real-world problem, we need to undertake the following

1. Define a state space that contains representations of all possible essential configurations of the objects of the real-world problem
   - One need not enumerate all states, but one should be able to "build" a state as and when required
2. Specify an initial state corresponding to the configuration in which the real world problem starts
3. Specify goal states corresponding to the real-world configurations that are acceptable from which to derive solutions to the problem
4. Specify a set of rules to describe the transitions, possibly via actions between states in the state space, being sure to consider
   - All real world assumptions made in the abstraction of the problem
   - The generality of the rules in relation to the adequacy in yielding solutions
   - How much work should be represented within the rules
5. Establish a notion of cost to measure the relative quality of solutions.

5 Definition of a "global path-based" search problem - KNOW FOR EXAM

Every (search) problem has six essential components

1. An **initial state** in which a problem-solving agent starts
2. A description of the possible **actions** available to the agent, via a **transition**(or **successor**) function ϕ which for each state x in the **state space** and each action α, details a set of states (possibly empty) reachable from x via action α
   - The state space is implicit in the description of the transition function
   - The pair (y, α) is a successor of x if $y \in \phi(x, a)$
   - This is going from state x to state y by action α
3. A goal test which determines whether a state is a goal state

4. A non negative step-cost function σ which details the cost $\sigma(x, \alpha, y)$ of a transition from any state x to any state y via any action α (if possible)
5. *A solution to a problem*
   - A path from the initial state to a goal state
6. *An optimal solution*
   - A solution with the lowest/highest path-cost among all solutions where path-cost of some path is the sum of the constituent step-costs

6 Searching for solution paths - KNOW FOR EXAM

We're interested in search strategies that generate/reveal the search tree Initially, the search tree consists of a root-node

- The associated state is the initial state

A search tree is generated/revealed by iteratively expanding, one by one, the hitherto unexpanded leaves of the tree so that

- A leaf is augmented with new (tree-) nodes so that the states associated with the new leaves are successor states of the state associated with the old leaf

If $y \in \phi(x, \alpha)$ and $y \in \phi(x, \beta)$, where α and β are different actions

- Then a tree-node with associated state x will have two children each of whose associated state is y. One for the transition via α and one from the transition via β
- Each child node y will have an associated action - the action undertaken to get from its parent node

The unexpanded leaves form the fringe - the choice of which fringe leaf to expand is determined by the **search strategy** A goal-node is a tree node for which the associated state is a goal state

7 Generating/revealing the search tree

Generating the search tree

- Suppose in the state space $\phi(x_0, \alpha) = \{x_1, x_2\}, \phi(x_0, \beta) = \{x_2\}$ and $\phi(x_0, \gamma) = \emptyset$
- Expand the root node
- Suppose in the state space $\phi(x_1, \alpha) = \{x_1\}, \phi(x_1, \beta) = \emptyset$ and $\phi(x_1, \gamma) = \{x_3\}$
- Expand the left-most fringe node (because the search strategy says so)

Revealing the search tree

- Fringe equates to the tree-nodes that have been revealed

8 Search Tree vs State Space

The search tree can also be seen as the "unrolling" of the state space

- For every "walk" w in the state space from the initial state there is a path **p** in the tree starting at the root node so that the states associated with the tree-nodes of this path **p** form the walk **w**
- For every path **p** in the tree starting at the root node there is a walk **w** in the state space from the initial state so that the states of **w** form the sequence of states associated with the nodes of **p**