

COMP3620 – Search Tutorial

1. From *Russell and Norvig Exercise 3.7, Page 90*

Give the state representation, initial state, goal state and the successor function for each of the following. Choose a formulation that is precise enough to be implemented.

- a) You have to colour a map of the World using only four colours, in such a way that no two adjacent countries have the same colour.

Solution: *State can be represented by a list of (country, colour) pairs. Initial state: an empty list. Goal state: all countries coloured. Successor function: colour one country, subject to the constraint that the colour is different to its neighbours.*

- b) A 3-foot tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two 3-foot-high crates (A and B), which the monkey can stack move and climb.

Solution: *State can be represented by the booleans:*

- *Is crate A on crate B*
- *Is crate B on crate A*
- *Is monkey on top of crate A?*
- *Is monkey on top of crate B?*
- *Is crate A under banana?*
- *Is crate B under banana?*

Initial state: all booleans set to false. Goal state: monkey on crate B, crate B under banana, crate B on crate A (also the same but with A and B interchanged).

Successor function has the following actions:

- *Move crate A*
- *Move crate B*
- *Put crate B on top of crate A*
- *Put crate A on top of crate B*
- *Climb to top of crate A*
- *Climb to top of crate B*

- c) You have a program that outputs the message “Illegal input record” when fed a certain file of input records. You know that processing of each record is independent of the other records and there is exactly one illegal record. You want to discover which record is illegal.

Solution: State is the number of records n . Initial state: $n = 0$. Goal state: the program **does not** output “Illegal input record” when the first n records are given to it. Once we reach the goal state, we know that the illegal record is the n -th record in the file. Successor function: increment n . Alternatively the successor function could do binary search if the maximum value for n is added to the state.

- d) You have three jugs, measuring 12 litres, 8 litres and 3 litres, and a water tap. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure exactly 1 litre.

Solution: State can be represented by 3 integers (a , b and c) that store the number of litres in each jug. Initial state: $a = b = c = 0$. Goal state: $a == 1 \vee b == 1 \vee c == 1$.

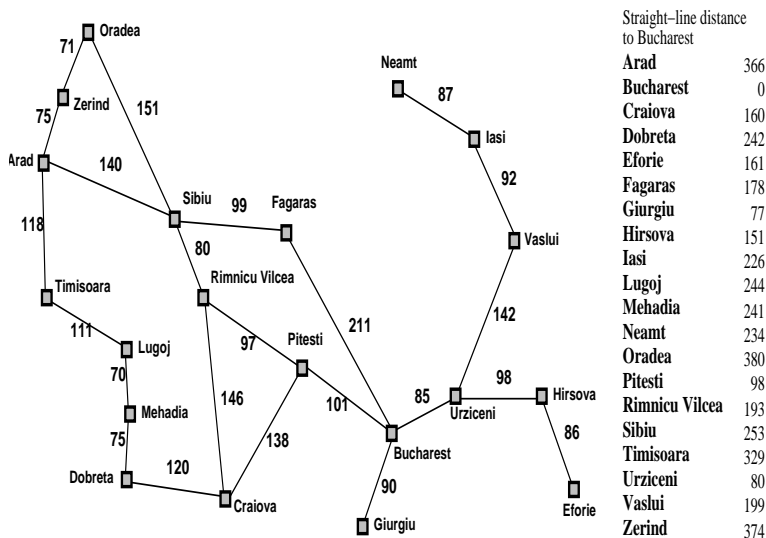
Successor function has the following actions:

- Pour from jug X to jug Y until either X is empty or Y is full
- Fill jug X from tap
- Empty jug X

2. From *Russell and Norvig Exercise 4.1, Page 134*

Trace the operation of A^* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f , g , and h score for each node.

Solution: Nodes considered for expansion are shown in bold.



Iteration	Fringe	f	g	h
1	Lugoj	244	0	244
2	Mehadia	311	70	241
	Timisoara	440	111	329
3	Dobreta	387	145	242
	Timisoara	440	111	329
4	Craiova	425	265	160
	Timisoara	440	111	329
5	Pitesti	501	403	98
	Rimnicu Vilcea	604	411	193
	Timisoara	440	111	329
6	Pitesti	501	403	98
	Rimnicu Vilcea	604	411	193
	Arad	595	229	366
7	Rimnicu Vilcea	604	411	193
	Arad	595	229	366
	Bucharest	504	504	0

We know that we can stop at Iteration 7, because the goal node (Bucharest) has the lowest f -value. In this case we know that the goal node is optimal, but it may not be true in other situations (see the A* example from the lecture notes).

3. Blame Phil

In the *Jobshop scheduling problem* we have n jobs, each of which has m parts. Part j of job i takes processing time t_{ij} on machine j . (That is, part

j can only be done on machine j). For all jobs, part j must be completed before part $j + 1$ (precedence constraints), and once we start part j we must complete it (no pre-emption). We want to find a schedule that minimizes “makespan” – the time the last job is complete.

a) Describe an admissible heuristic for the job-shop problem.

Solution: *To find an admissible heuristic it is useful to think about possible relaxations of the original problem. For example:*

- *Relax the precedence constraints. Unfortunately, the resulting problem is still in NP.*
- *Relax the pre-emption constraints. Unfortunately, the resulting problem is still in NP (but easier). This method is often used, because good approximate answers can be obtained in reasonable time.*
- *Relax the precedence AND pre-emption constraints. Resulting problem can be solved by inspection (basically the time remaining on each machine is the sum of processing times of all incomplete jobs). This one works, but the estimate is pretty poor.*

This is an example where extra time spent finding the lower bound pays off in the the long run.

b) Thinking about a local search approach, define one or more neighbourhoods for the problem.

Solution: *A neighbourhood is a set of potential successor states from a given state. The most common neighbourhood is defined by swapping the position of two jobs on one machine (and re-arranging start times so that precedence is maintained).*

One problem with this approach is that a swap will often have no effect on the objective - there are large flat areas. Several swaps are sometimes needed to reduce the objective. Alternatively, neighbourhoods can be defined that include several swaps in a single “move”. A completely different approach is to define a “surrogate objective function”. This acts similarly to an Eval function in adversarial search, since it rewards “good features” of a solution. Note that domain knowledge is required to know what those features are.