Chapter 3. Problem Solving by Informed Search

Topic 3.1. AI problems and Heuristic Search Techniques

a) Chess playing / planning robot navigation / finding language structures / visual perception: searching through a state space for a goal state

b) Goal–based agents are considered.

Basic Concepts:

1. **Initial state**
   Description of environment that the agent starts with

2. **Goal state**
   Description of environment that the agent wants to achieve

Classroom / Toy example:
8-puzzle problem

Any placement of tiles, S_0

6	3	4
7		5
2	1	8

An ordered placement of tiles, G_i

1	2	3
4	5	6
7	8	

3. **Successor function**

That takes a state as argument and returns a set of ⟨action, state⟩ pairs

SUCCESSOR(S_0) = {⟨up, S_1⟩, ⟨left, S_2⟩, …}

6		4
7	3	5
2	1	8

6	3	4
7	5	
2	1	8

6	3	4
	7	5
2	1	8

6	3	4
7	1	5
2		8

4. **State Space**

Set of states reachable from the initial state;

Initial state + successor function

S_0, S_1, S_2, …

S_0, SUCCESSOR

5. **Path**

A sequence of states connected by actions

S_0 - up - S_1 - right - S_{12} …

6. **Goal test**

Test to determine whether a given state is a goal state or not

Matching with a given possible goal state

7. **Path cost**
Sum of step costs
Uniform / non-uniform

Number of steps / actions in the path; Uniform step cost, 1;
$c(n, a, n') = c(S_0, \text{up}, S_1) = 1$

8. **Solution**
A path from the initial state to a goal state

A path in a directed graph or tree constructed by SUCCESSOR, from S_0 to G_i.

9. **Optimal Solution**
Solution that has the lowest path cost

A specific path, P_s found in the directed graph.

c) Conventional vs Heuristic search

- Search space may be very big
- Efficient algorithms needed
- Uninformed or blind search not usable directly
- Informed or Heuristic search helps
  - Additional prior knowledge
  - Estimation or problem specific rule
  - Helps sometimes, but not always
  - Heuristics: ‘to find’, ‘to discover’, ‘rule of thumb’ or ‘judgmental technique’

d) Heuristic Functions

Heuristics is often expressed as a function of a state:

h(s) = <u>Estimated cost</u> of the <u>cheapest path</u> from s to a goal state.

Examples:

i) 8-puzzle problem

Select the generated state s with

$h_1(s)$ = number of tiles not matching those in the goal.

6	3	4
7		5
2	1	8

Initial

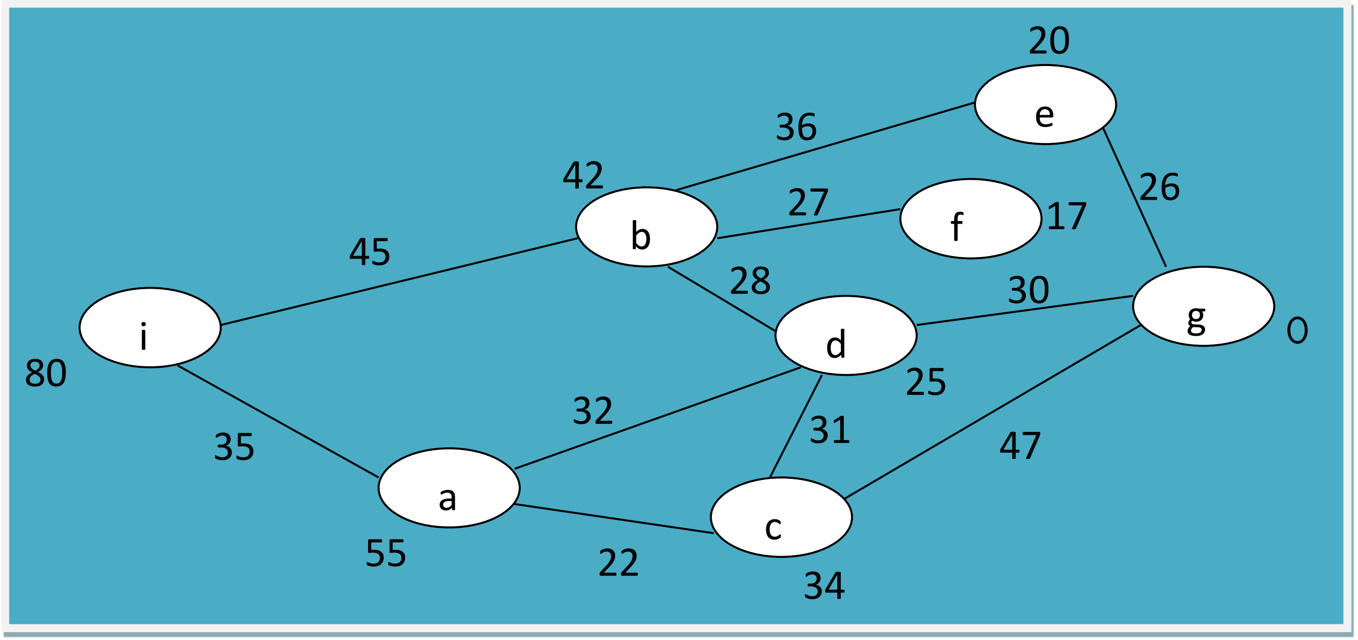
	1	2
3	4	5
6	7	8

Goal

h_1(Initial) = 6

ii) For finding the shortest path between two towns intended for traveling through other towns:

$h_2(s)$ = Straight-line distance from town s to the destination.



i: Initial state/node (source)

g: Goal state/node (destination)

$h_2(i) = 80$

e) Good heuristic functions

- Finding a good heuristic function is a big problem.
- Statistical and probabilistic methods help.
- May require additional computational cost (as for h_1), or prior knowledge (as for h_2).
- Good heuristic functions are said to be admissible or consistent.

f) Admissibility of heuristic functions

❑ An admissible heuristic function never over estimates the cost to reach the goal:

$h(s) \leq$ Actual cost to reach the goal from s

❑ h_1 and h_2 seen above are admissible:

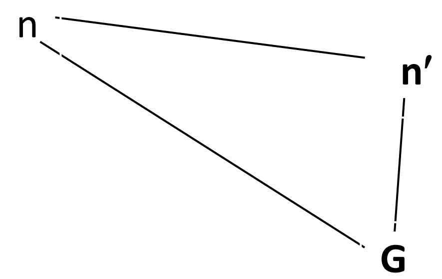
h_1 – any tile that is out of place, must be moved at least once

h_2 – straight line is the shortest of any other lines

g) Monotonicity or Consistency of heuristic functions

$h(n) \le c(n, a, n') + h(n')$, for any node n and every successor n′ of n

It is from triangular inequality: one side is shorter than the sum of the other two.

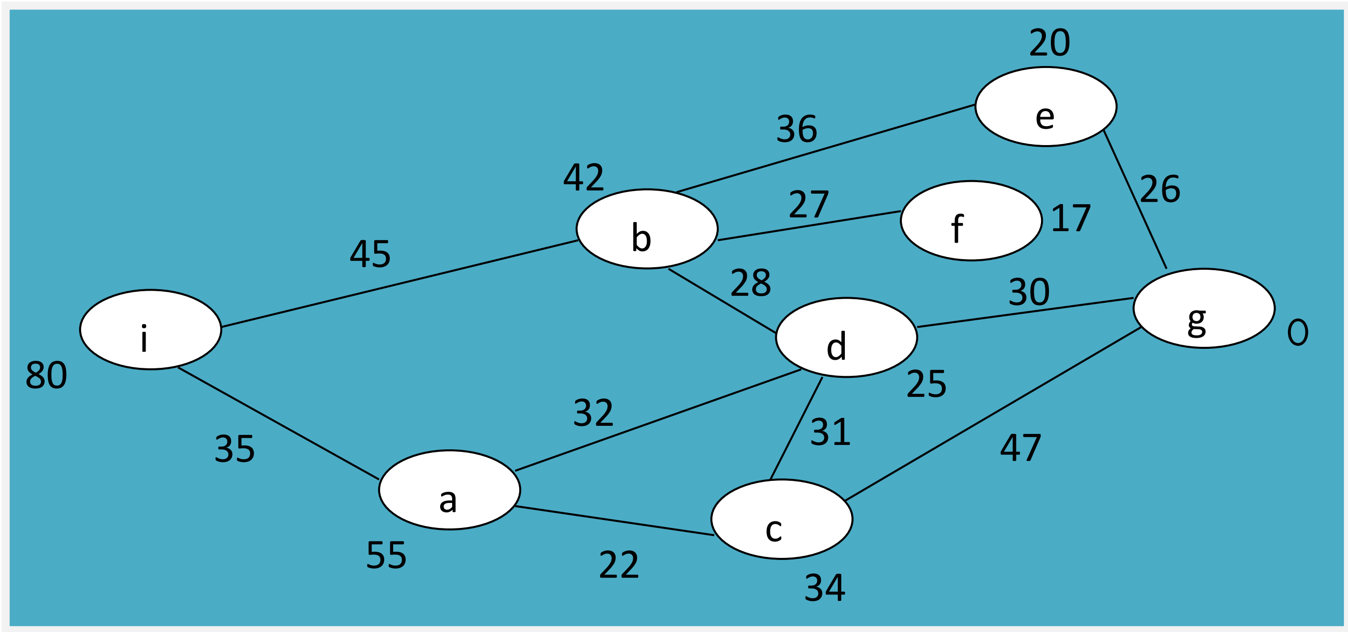


❖ h_1 and h_2 seen above are also consistent.

❖ Every consistent heuristic function is also admissible, but vice-versa is not always true.

[May work very well on n′, while not so well on n.]

h) Checking for admissibility and consistency of heuristic functions



h(s) ≤ Actual cost to reach the goal from s, for any node s

h(i)=80 ≤ cost(i-b-e-g)=107 | cost(i-b-d-g)=103 | …;

…

h(n) ≤ c(n, *a*, n′) + h(n′), for any node n and every successor n′ of n

h(i)=80 ≤ c(i, →, b) + h(b)=45+42, h(i)=80 ≤ c(i, →, a) + h(a)=35+55;
h(b)= …; …