Artificial Intelligence

SCALAB

Grupo de Inteligencia Artificial

Universidad Carlos III de Madrid

2017-2018

Heuristic Search – Exercises

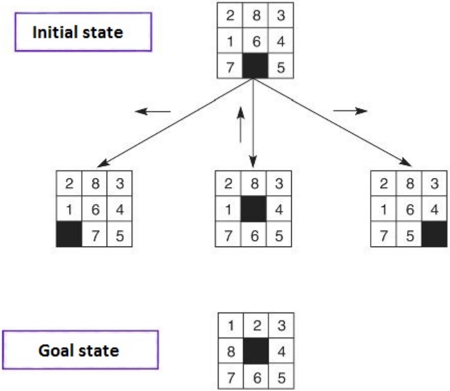
Recap: Heuristic search/ Informed methods

- They have information about the proximity of each state to the goal state, guiding with this the search towards the more “promising” path
- They do not guarantee to find the solution even if it exists
- If they find a solution it may not be always optimal
- Sometimes they do find a very good solution

Recap: Heuristic = h'

- Improves the search
- Uses knowledge about the domain
- Guidance about the state order of successors in the search
- Sometimes they do find a very good solution
- Link each state to a numeric amount that evaluates how promising is that state to reach a goal. It can be:
  - An estimation of the quality of the state
  - An estimation of how near is the state to the solution/goal (distance, cheapest cost,..). We will use this one

Heuristics for 8 puzzle



Initial state

2	8	3
1	6	4
7		5

2	8	3
1	6	4
	7	5

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7	5	

Goal state

1	2	3
8		4
7	6	5

Heuristics for 8 puzzle II

- ha = Sume of the digits' distance to their correct position in the goal state. Manhattan distance. Example: 1+1+0+0+01+1+2=6

2	8	3
1	6	4
■	7	5

1	2	3
8	■	4
7	6	5

- hb = Number of misplaced digits. Example: 5. Simple, but does not use cost information (number of movements)

Heuristics for 8 puzzle III

- The previous heuristics do not give importance to the difficulty of positions' inversion: If two digits are adjacent and they must exchange positions, putting them in the right place takes more than 2 movements
- hc = double of the number of digits to swap.
  - This is also a poor heuristic because it focuses too much on a special kind of difficulty
  - A lot of states will have value 0 even if their are not the solution
  - Usually added to other heuristics to form a final one. In the example: 2*0=0

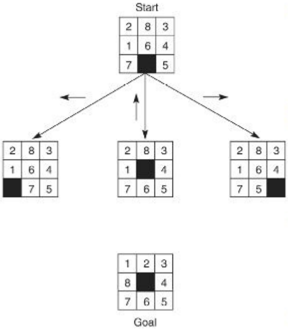
1	2	3
8	■	4
7	5	6

1	2	3
8	■	4
7	6	5

Heuristics for 8 puzzle IV

hd (n) = ha(n) + hc(n)

- Better heuristic. However it requires more computation. Heuristics must be easy to compute
- It could be improved



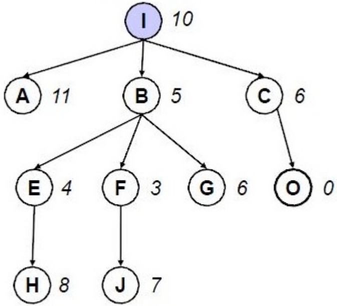
	h'_a	h'_b	h'_c	h'_d
2 8 3 1 6 4 ■ 7 5	6	5	0	6
2 8 3 1 ■ 4 7 6 5	4	3	0	4
2 8 3 1 6 4 7 5 ■	6	5	0	6

Greedy Search

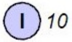
- f’(n)=h’(n). Represents the minimum estimated cost for a given node n to reach the goal (cheapest path)
- It tries in each step to get closer to the solution
- Not optimal. Not complete, like DFS.

Greedy Search II

State Space

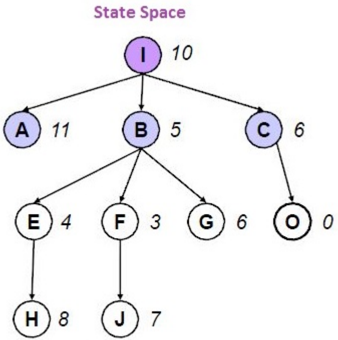


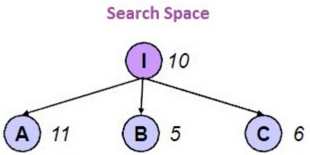
Search Space



Priority list of open nodes : I (10)

Greedy Search III

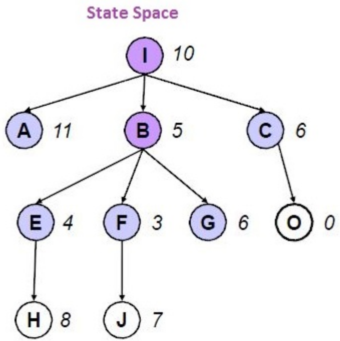


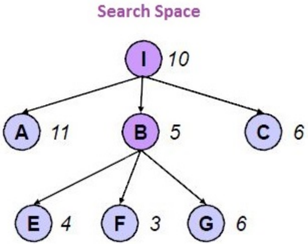


Priority list of open nodes : A (11) C (6) B (5)

Expanded nodes (in order) : I

Greedy Search IV

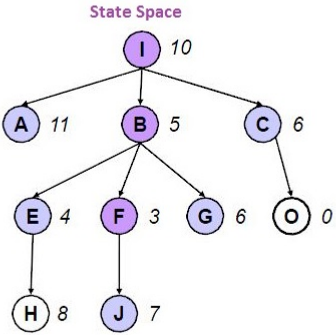


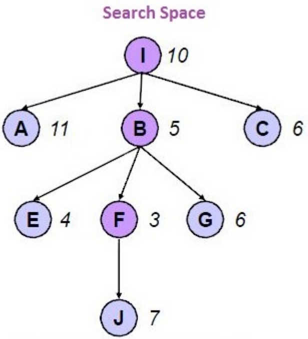


Priority list of open nodes : A (11) G(6) C (6) E (4) F (3)

Expanded nodes (in order) : I B

Greedy Search V

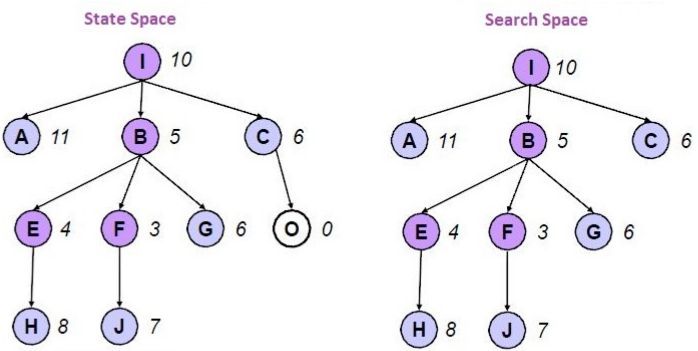




Priority list of open nodes : A (11) J (7) G (6) C (6) E (4)

Expanded nodes (in order) : I B F

Greedy Search VI



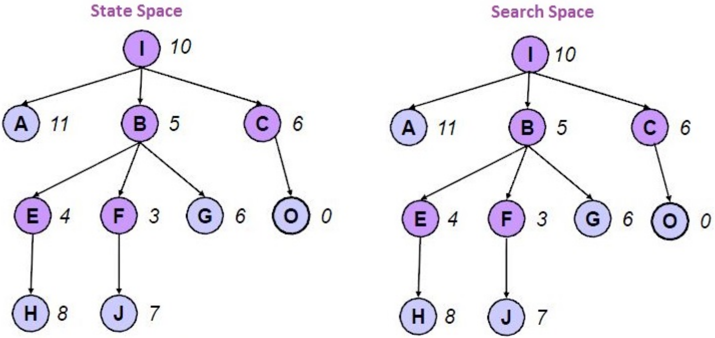
State Space

Search Space

Priority list of open nodes : A (11) H (8) J (7) G (6) C (6)

Expanded nodes (in order) : I B F E

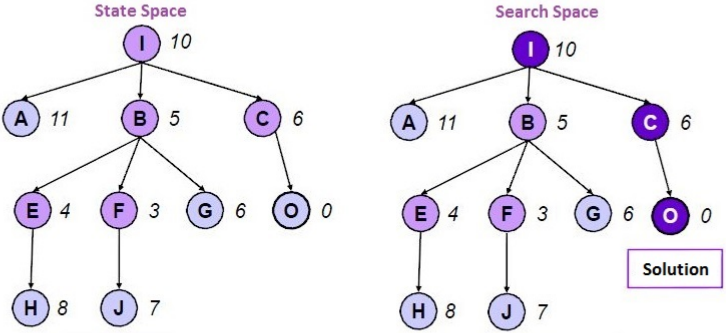
Greedy Search VII



Priority list of open nodes : A (11) H (8) J (7) G (6) O (0)

Expanded nodes (in order) : I B F E C

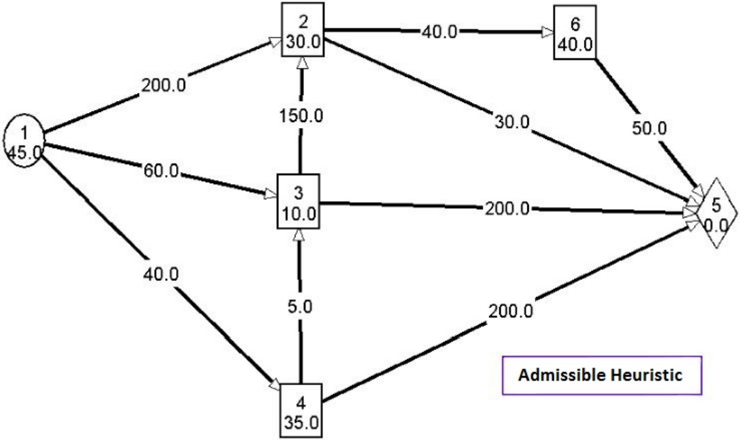
Greedy Search VIII



Priority list of open nodes : A (11) H (8) J (7) G (6)

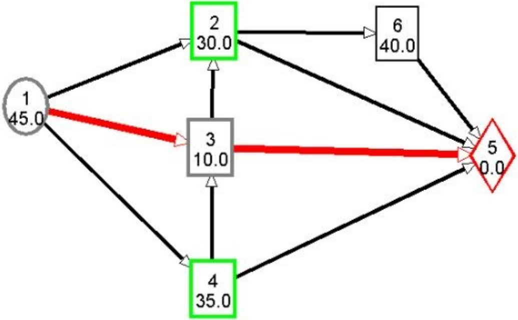
Expanded nodes (in order) : I B F E C O

Greedy Search IX



Admissible Heuristic

Greedy Search X

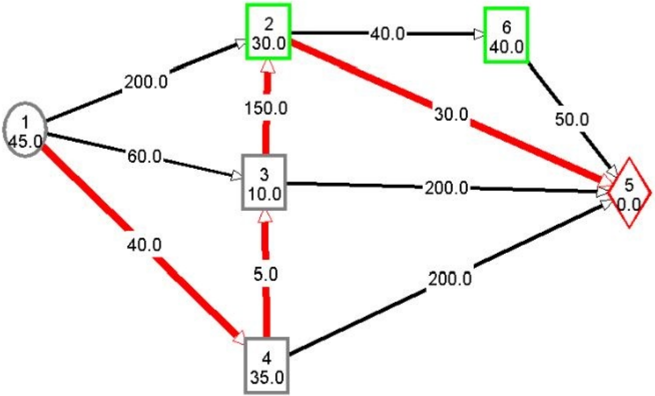


Greedy solution : 1-3-5 **Cost (not taken into account)** : 60+200 = 260

A*

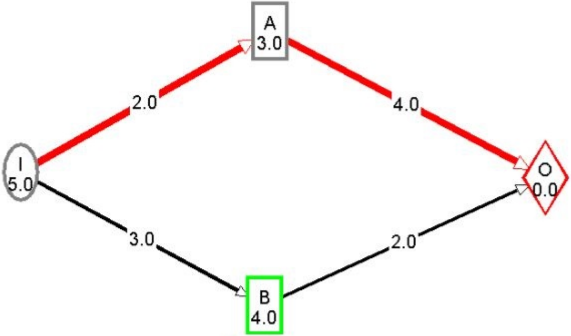
- Optimal search: f'(n) = g(n) + h'(n).
    - f'(n)= estimation of the total minimum cost (from initial state to goal state) of a solution that goes through n
    - g(n)= real cost of the path to n
    - h'(n)= estimation of the minimum cost from n to a goal state
    - If h' = 0 => non heuristic search. If g=0 => greedy search
- h' is an admissible heuristic if $h'(n) \leq h(n)$ where h(n) is the real cost of going from n to a goal state choosing the path with the minimum cost.
- Only if h' is admissible, we call this search A* otherwise it is only A.

A* Solution



A* Solution : 1-4-3-2-5 **Minimum Cost** : 40+5+150+30 = 225

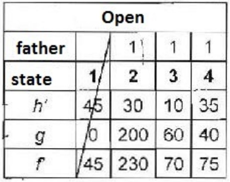
A Solution, not A* (not admissible heuristic)



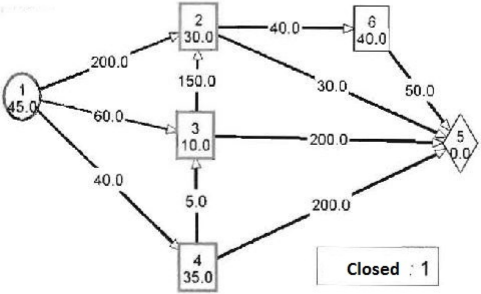
Non admissible heuristic doesn't guarantee minimum cost; It is not A*

A*: step by step

We expand 1



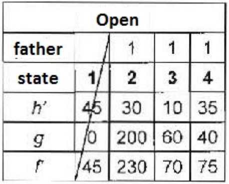
Open				
father		1	1	1
state	~~1~~	2	3	4
h′	~~45~~	30	10	35
g	~~0~~	200	60	40
f	~~45~~	230	70	75



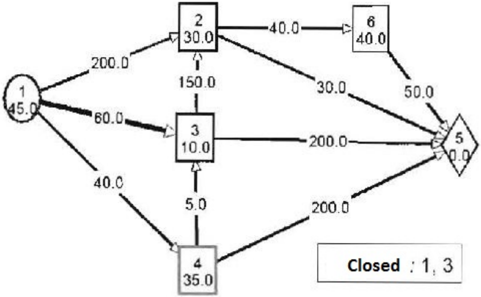
Closed : 1

A*: step by step

We expand 3 (the one with minimum cost in the open list)



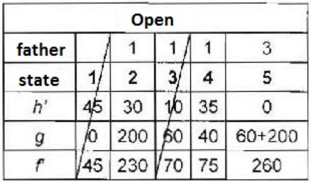
Open				
father		1	1	1
state	~~1~~	2	3	4
h'	~~45~~	30	10	35
g	~~0~~	200	60	40
f	~~45~~	230	70	75



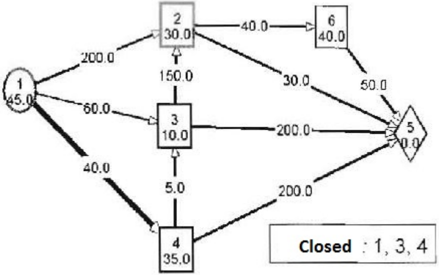
- Path: 1-3. Cost: 60
- Successors of 3: 2 and 5
    - 5 is new: we add it to open
    - 2 is already in open
    - Is it better this new path 1-3-2 than 1-2? No: it has more cost (60+150 against 200). Hence we stick with what we had.

A*: step by step

We expand 4



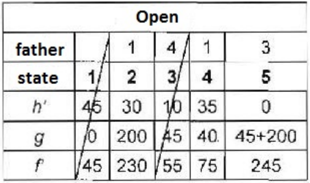
Open					
father		1	~~1~~	1	3
state	~~1~~	2	~~3~~	4	5
h'	~~45~~	30	~~10~~	35	0
g	~~0~~	200	~~60~~	40	60+200
f	~~45~~	230	~~70~~	75	260



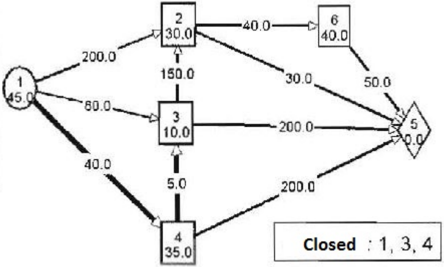
Closed : 1, 3, 4

- We change paths. Path: 1-4. Cost: 40
- Successors of 4: 3 and 5
  - 3 is in closed (we had already expanded it)
  - Is it better this new path 1-4-3 than 1-3? Yes: it has less cost (40+5 against 60). Hence we stick with this one and we propagate this change to the successors of 3 (2 and 5).

A*: step by step



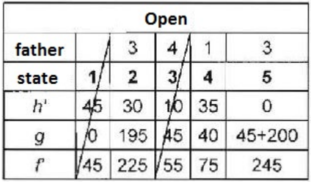
Open					
father		1	~~4~~	1	3
state	~~1~~	2	~~3~~	4	5
h'	~~45~~	30	~~10~~	35	0
g	~~0~~	200	~~45~~	40	45+200
f	~~45~~	230	~~55~~	75	245



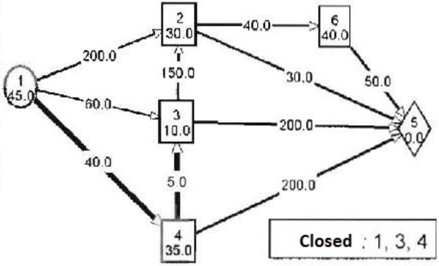
Closed : 1, 3, 4

- We update 3 and its son 5
- We are missing its son 2
  - It was in open as son of 1
  - Is it better 1-4-3-2 than 1-2? Yes: it has less cost (40+5+150 against 200). We change to this new path.

A*: step by step



Open					
father		3	~~4~~	1	3
state	~~1~~	2	~~3~~	4	5
h'	~~45~~	30	~~10~~	35	0
g	~~0~~	195	~~45~~	40	45+200
f	~~45~~	225	~~55~~	75	245



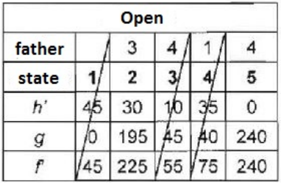
Closed : 1, 3, 4

We are missing 5 as successor of 4

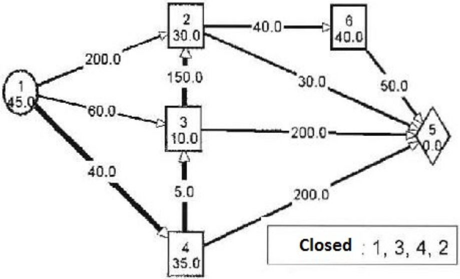
- 5 is already in open
- Is it better 1-4-5 than 1-4-3-5? Yes: it has less cost (40+200 against 245). We change to this new path.

A*: step by step

We expand 2



Open					
father		3	~~4~~	~~1~~	4
state	~~1~~	2	~~3~~	~~4~~	5
h'	~~45~~	30	~~10~~	~~35~~	0
g	~~0~~	195	~~45~~	~~40~~	240
f	~~45~~	225	~~55~~	~~75~~	240

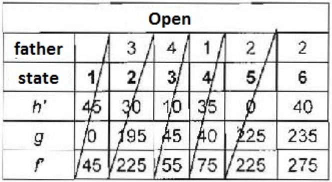


Closed : 1, 3, 4, 2

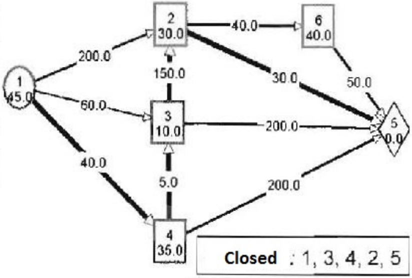
- Path: 1-4-3-2. Cost: 195
- Successors of 2: 6 and 5
  - 5 is in open and 6 is new (we add it to open)
  - Is it better 1-4-3-2-5 than 1-4-5? Yes: it has less cost (195+30 against 240). Hence we change to this new path.

A*: step by step

We expand 5



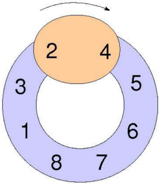
Open						
father		3	4	1	2	2
state	1	2	3	4	5	6
h'	45	30	10	35	0	40
g	0	195	45	40	225	235
f	45	225	55	75	225	275



Closed : 1, 3, 4, 2, 5

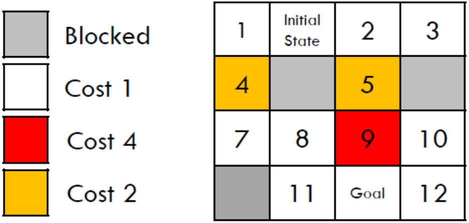
- Path: 1-4-3-2-5. Cost: 225
- 5 is goal state
  - We reach a solution
  - The heuristic is admissible hence it is optimal : we have found the path with less cost.

Exercise 1: top spin



- Permutation game in which we rotate the base to exchange the order of numbers
- Goal state: numbers are ordered from 1 to 8 with the pair (1 2) in the base
- Operators:
  - Move numbers to the right
  - Move numbers to the left
  - Rotate base to switch order of the numbers in it
- Define a heuristic for this game
- Apply Hill Climbing search using the heuristic function

Exercise 2 – Maze



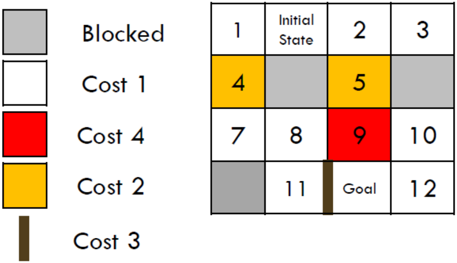
▨	Blocked
□	Cost 1
🟥	Cost 4
🟨	Cost 2

1	Initial State	2	3
🟨 4	▨	🟨 5	▨
7	8	🟥 9	10
▨	11	Goal	12

Use the numbers written on each cell as a cell reference in the solution

- The aim is to find the path with minimum cost to the goal considering the cost of stepping on each cell and the order of operators: move Up, move Left, move Down, move Right
- Use Manhattan Distance as a heuristic function (the minimum number of movements to get to the goal without considering the blocks and costs) and apply Hill Climbing and A* to the problem. Compare the results

Exercise 3 – Maze with door



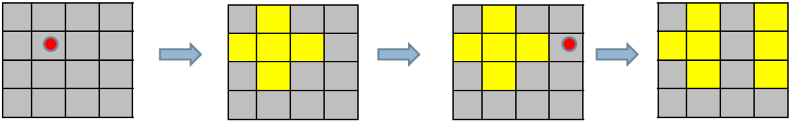
▨	Blocked
□	Cost 1
🟥	Cost 4
🟨	Cost 2
▮	Cost 3

1	Initial State	2	3
🟨 4	▨	🟨 5	▨
7	8	🟥 9	10
▨	11	Goal	12

- Consider that there is now a door between cell 11 and Goal. Passing through this door costs 3
- Solve the problem with A*

Exercise 4 – Lights Out

- The game consists of an NxN grid of lights. When the game starts, a random number of these lights is switched on. Pressing any of the lights will toggle it and the four adjacent lights. The goal of the puzzle is to switch all the lights off, preferably in as few button presses as possible



- How do you represent the states and operators for this problem?
- Define an evaluation function (the cost function + the heuristic function) for A* algorithm