Artificial Intelligence Principles

6G7V0011 - 1CWK100

Dr. Peng Wang

Email: p.wang@mmu.ac.uk

Department of Computing and Mathematics

Tuesday, Nov. 12th, 2024

Outline

Outline

UCS Quiz and Recap

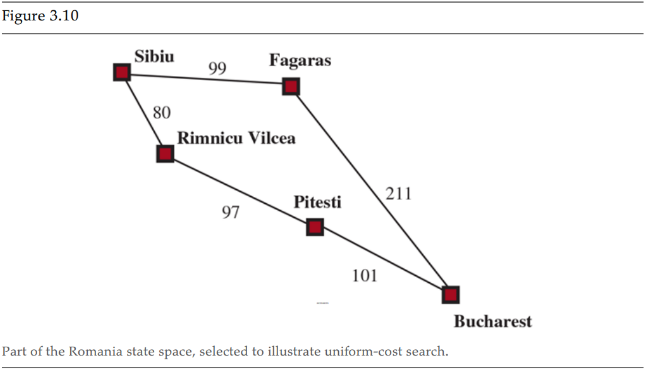


Figure 1: Map for UCS Quiz

- Q1: Could you draw the search tree using UCS for this partial Romanian map?
- Q2: Could you write down how openset and closedset being changed?

UCS Quiz and Recap - continued

O = {'Sibius':0},
C = {}.

O = {},
C = {'Sibius':0},
current = 'Sibius'.

O = {'Fagaras':99, 'Rimnicu Vilcea':80},
C = {'Sibius':0}.

O = {'Fagaras':99},
C = {'Sibius':0, 'Rimnicu Vilcea':80}, current = 'Rimnicu Vilcea'.

O = {'Fagaras':99, 'Pitesti':177},
C = {'Sibius':0, 'Rimnicu Vilcea':80}.

O = {'Pitesti':177},
C = {'Sibius':0, 'Rimnicu Vilcea':80, 'Fagaras':99},
current='Fagaras'.

UCS Quiz and Recap - continued

O = {'Pitesti':177, 'Bucharest':310},
C = {'Sibius':0, 'Rimnicu Vilcea':80, 'Fagaras':99}.

O = { 'Bucharest':310},
C = {'Sibius':0, 'Rimnicu Vilcea':80, 'Fagaras':99, 'Pitesti':177},
current='Pitesti'.

O = { 'Bucharest':278},
C = {'Sibius':0, 'Rimnicu Vilcea':80, 'Fagaras':99, 'Pitesti':177}.

O = { },
C = {'Sibius':0, 'Rimnicu Vilcea':80, 'Fagaras':99, 'Pitesti':177, 'Bucharest':278},
current=Bucharest.

Experiences/Heuristics I

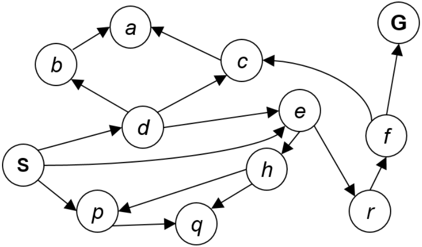


Figure 2: Same cost for each edge

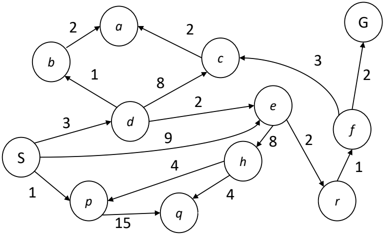


Figure 3: Different cost for each edge

Recap: In uninformed search, including BFS, DFS, and UCS

- Agents do not have goal information, i.e., no idea where the *goal* is
- Agents try to minimise the cumulative path cost from *start* to *goal*
- Given a *node* n, agents evaluate the path cost by the evaluation function $f(n)$, which can be distance, time, etc. that you 'care' the most!

Experiences/Heuristics II

- **Question:** $f(n)$ **for BFS (depth of node), DFS (negative/inverse/reciprocal of the depth), and UCS (path cost)?**

We now define the (cumulative) path cost of a *node* n as $g(n)$, then for uninformed search, we have

$$f(n) = g(n) \tag{1}$$

What if

- Agents still do not know any goal information
- But, as designers, we can provide our experience (expert knowledge) of the environment to the agents?

Example:

Experiences/Heuristics III

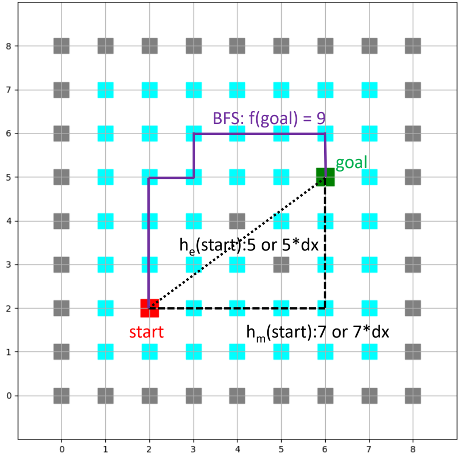


Figure 4: Different heuristics for a grid environment

Experiences/Heuristics - continued

Given two points **s** and **g** of dimension N, we have

Euclidean distance

$E(\mathbf{s}, \mathbf{g}) = \sqrt{\sum_{i=0}^{N} (s_i - g_i)^2}$

Manhattan distance

$M(\mathbf{s}, \mathbf{g}) = \sum_{i=0}^{N} |s_i - g_i|$

1. Heuristic 1: $h_e(start)$ $(h_1(\mathbf{s}))$: The Euclidean distance between *start* and *goal* is 5*dx, or simply 5 in cases where $f(n)$ is calculated same way, though it crosses the obstacles
2. Heuristic 2: $h_m(start)$ $(h_2(\mathbf{s}))$: The Manhattan distance between *start* and *goal* is 7*dx, or simply 7. Manhattan distance is tricky, 7 corresponds to more than one path.

When there are more choices, heuristic becomes informative!

Experiences/Heuristics - continued

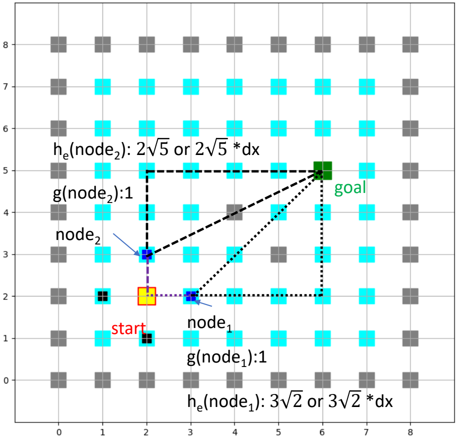


Figure 5: Heuristics when there are more than one choices

Experiences/Heuristics - continued

Heuristic function $h(n)$

Functions such as $h_e(n)$ or $h_m(n)$ that helps estimate the cost of the cheapest path from the state at node n to a goal state.

What we can observe from the previous Figure:

1. $n_1 \leftarrow \text{node}_1$, $n_2 \leftarrow \text{node}_2$
2. $g(n_1) = 1$, $g(n_2) = 1$
3. $f(n_1) = g(n_1) = 1$, $f(n_2) = g(n_2) = 1$
4. $h_e(n_1) = \sqrt{18} < h_e(n_2) = \sqrt{20}$

Since $h_e(n_1) < h_e(n_2)$, can we simply choose n_1 over n_2 because it is 'closer' to the goal according to our experiences (heuristic)?

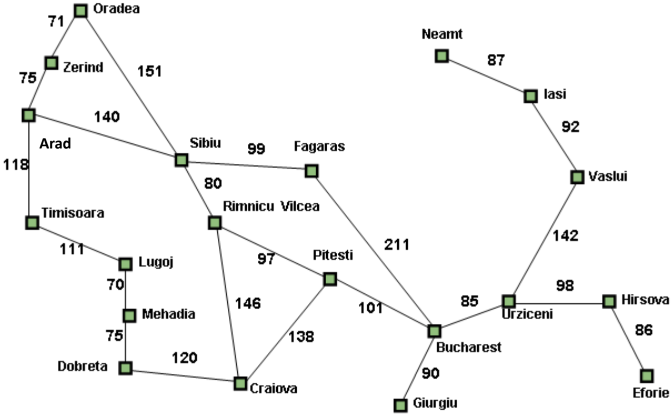
Yes, we can! Greedy best-first search.

Greedy Best-first Search

What is it?

- A form of best-first search that expands first the node with the lowest $h(n)$ value
- Node n appears to be closest to the goal (?)
- Likely leads to a solution quickly (?)
- Evaluation function $f(n) = h(n)$
  - No path cost counted

Greedy Best-first Search



Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 6: Heuristics on the Romania routing problem

Heuristic $h(n)$: Straight-line distance

Greedy Best-first Search

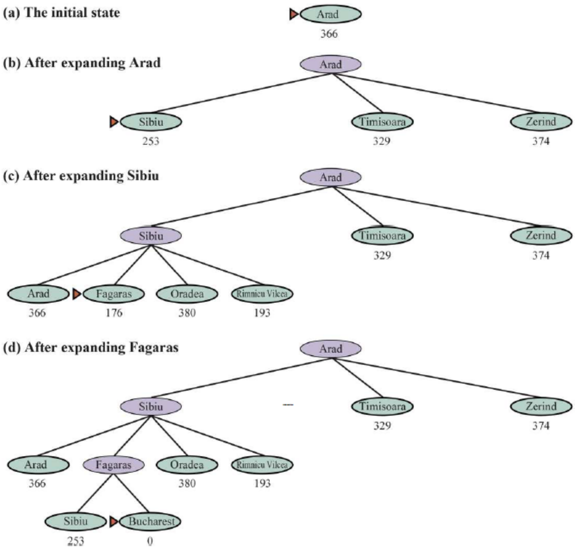
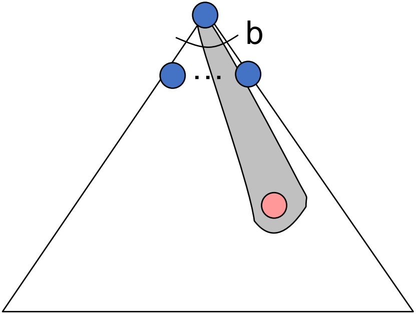
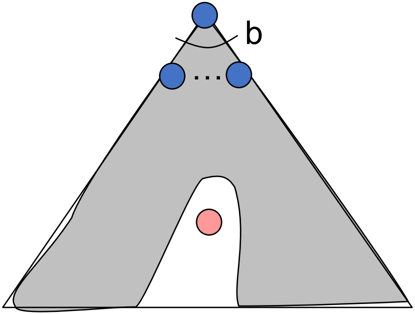


Figure 7: How greedy best-first search works on Romanian problem

Greedy Best-first Search

Strategy: always expand a node that 'you' tell the agent is closest to a goal state. Not necessarily the best solution (Romania problem).





- Takes you straight to the goal, with a sub-optimal solution
- Like a badly-guided DFS, perfectly misses goal(s)

Outline

A Star (A^*) search

An agent is planning a path from start **s** to a goal **g**. Let's look at *node n*:

- UCS has the cost from **s** to n, which is the path cost $g(n)$, and it is the minimum in frontier
- It is backward cost from n's perspective

- Greedy best-first considers only proximity to **g**, which is heuristic cost $h(n)$, it is minimum in frontier
- It is forward cost from n's perspective

Figure 8: Consider both forward and backward cost while planning (image from internet)

A Star (A^*) search

Recap the pros and cons of UCS and Greedy search:

- UCS searches all directions $\rightarrow$ no goal information
- It is optimal

- Greedy best-first is not optimal. Worst-cast: badly-guided DFS
- Could takes the agent to goal straightly

Figure 9: Can agents compromise?

What about a new evaluation function $f(n)$?

$$f(n) = g(n) + h(n) \tag{2}$$

A Star (A^*) Search

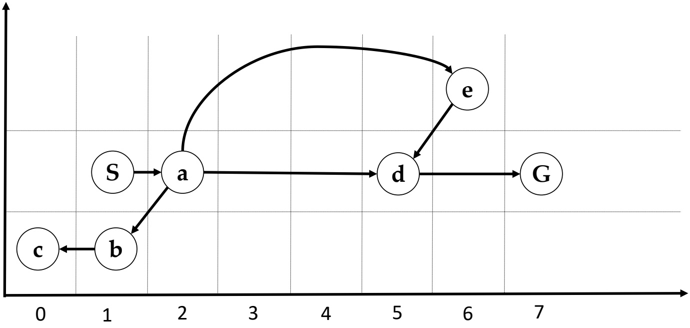


Figure 10: State graph with spatial information

- A state graph with a start Ⓢ and a goal Ⓖ, and other nodes
- Arrows indicate reach-ability, one direction
- Grids indicate spatial relationships $\leftarrow$ Hops, Manhattan distance
- Adjacent grids may not reach each other directly, e.g., Ⓢ and ⓑ

A Star (A^*) Search

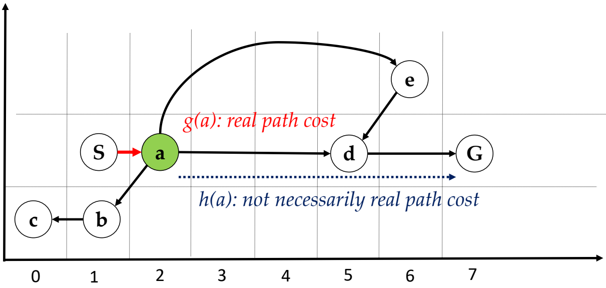


Figure 11: Costs at node a

- Backward: path cost $g(a)$
- Forward: heuristic (hops) $h(n)$ from ⓐ to Ⓖ

1. Can be anything, i.e., number of blocks.
2. Not necessarily real path cost from ⓐ to Ⓖ

- $f(n) = g(n) + h(n)$ **is an estimate of the full 'path cost'**

A Star (A^*) Search - example

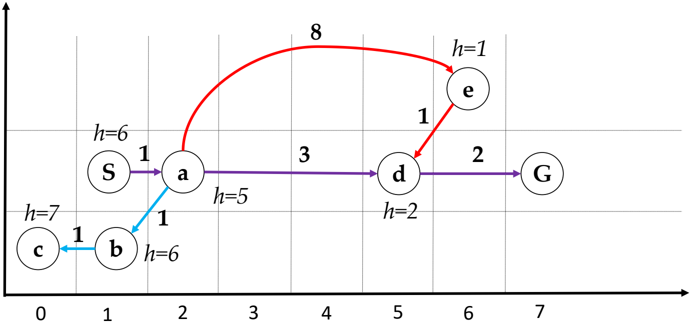


Figure 12: State graph with cost and heuristic

- Edge cost is the real distance
- Heuristic is hops. **Adjacency does not mean reach-ability**
- Can you calculate the hops and Manhatten distance, and what's the difference?

A Star (A^*) Search - example

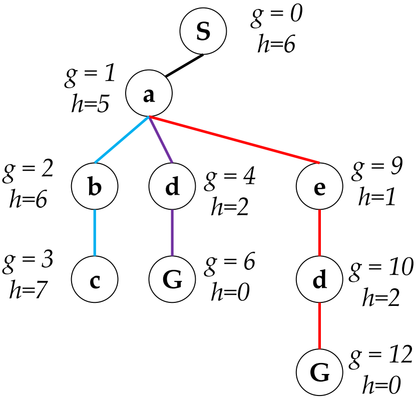
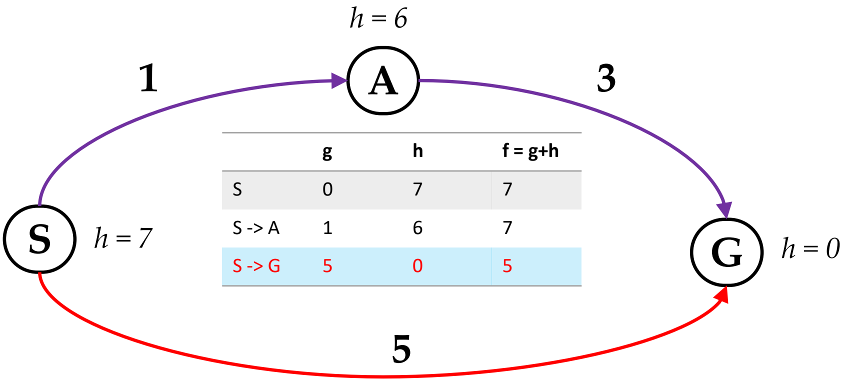


Figure 13: Search tree with path cost $g(n)$ and heuristic $h(n)$

Next step?

1. UCS goes to ⓑ, why? 2. Greedy search goes to ⓔ, why? 3. A^* goes to ⓓ, why?

Is A^* Optimal?



	g	h	f = g+h
S	0	7	7
S -> A	1	6	7
S -> G	5	0	5

Figure 14: A sub-optimal path is picked by A^*

- UCS CAN pick the optimal route, **how?**
- Greedy search will not
- **Can we conclude the Greedy search contribution caused the issue?**

Admissible Heuristics

Admissible Heuristic

A heuristic $h(\cdot)$ is **admissible** (optimistic) if for a node $\cdot$

$$0 \leq h(n) \leq h^*(n), \qquad (3)$$

where $h^*(\cdot)$ is the true cost of a node to a nearest goal.

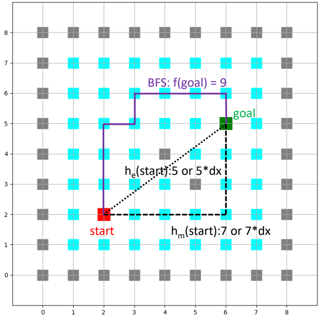
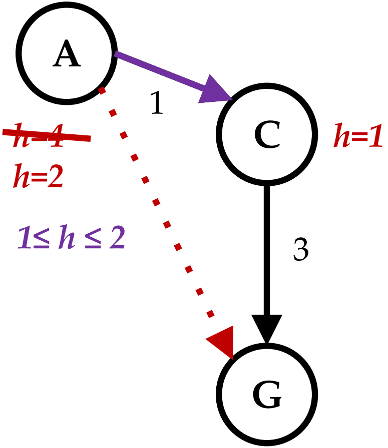


Figure 15: Admissible heuristic

Consistency of Heuristics

- Main idea: estimated heuristic costs $\leq$ actual costs
  - **Admissibility:** heuristic cost $\leq$ actual cost to goal
    1. $h(A) \leq$ actual cost from A to G
  - **Consistency:** heuristic edge cost $\leq$ actual cost for each edge
    1. $h(A) - h(C) \leq \text{cost}(A \text{ to } C)$
    2. **Triangle inequality:** $h(A) \leq \text{cost}(A \text{ to } C) + h(C)$
- Consequences of consistency:
  1. The f value along a path never decreases $h(A) \leq \text{cost}(A \text{ to } C) + h(C)$
  2. A^* graph search is optimal



Question: Is the condition $1 \leq h(.) \leq 2$ consistent?

A^* - Graph Search I

Algorithm 1: Pseudocode of A^* Search

Input: Initial state $\mathbf{s}$, goal state $\mathbf{g}$, evaluation function $f = g + h$
Output: A *node* helps to retrieve a solution (path) $\mathcal{P}$, or failure

node $\leftarrow$ with $\mathbf{s}$ as state
$\mathcal{O} \leftarrow$ *node*, an priority queue   % *node* with the least cost.
$\mathcal{C} \leftarrow \emptyset$
while $\mathcal{O}$!= $\emptyset$ **do**
  parent $\leftarrow$ the first node in $\mathcal{O}$ % 'first' due to the priority queue.
  if *parent*.state == $\mathbf{g}$ **then**
    return *parent*
  end if
  del *parent* from $\mathcal{O}$
  $\mathcal{C} \leftarrow$ *parent*
  for *child* **in** successor (of the current *parent*) **do**
    $\mathbf{v} \leftarrow$ *current child*

A^* - Graph Search II

```
        if v is not in C and child is not in O then
            add child to O        % found a new node.
        else if child is in O then
            if current pathcost of child < previous pathcost of child then
                add current child to O
            end if
        end if
    end for
end while
return failure
```

A^* Properties

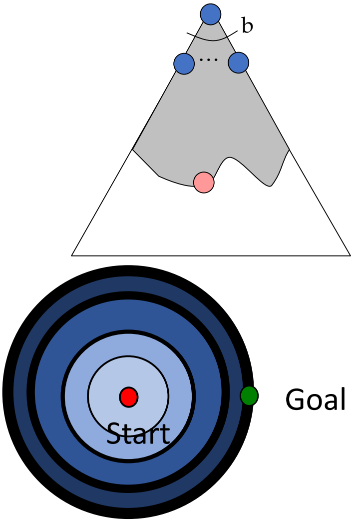


Figure 16: UCS

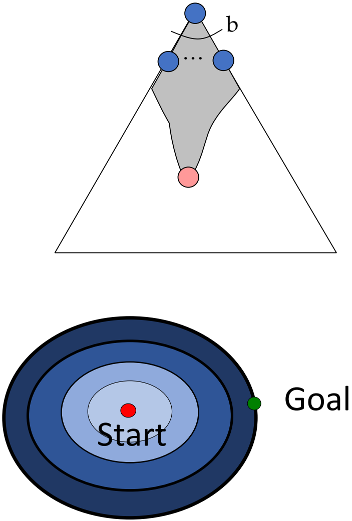


Figure 17: A^*

Summary

Graph search:

- A^* optimal if heuristic is consistent
- UCS optimal ($h = 0$ is consistent)

Consistency implies admissibility!

In general, most natural admissible heuristics tend to be consistent.