Fundamentals of Artificial Intelligence

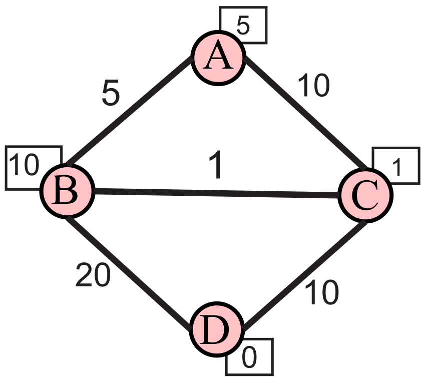
Exercise 2b: Informed Search – Solutions Winter Semester 2022/23

Sebastian Mair

1 Presented Problems

Problem 2.5: Heuristics for Informed Search

Consider the following graph. We start from node A and our goal node is D. The path costs are shown on the arcs and the heuristic values are shown at each node.



Problem 2.5.1: What are the requirements on a heuristic for A* tree search and graph search to be optimal?

Solution:

- A* tree search requires the heuristic to be admissible.
- A* graph search requires the heuristic to be consistent.

Solution: For its definitions, we use $h(n)$ as the heuristic value at node n and $g(n, n')$ as the path cost from node n to node n'.

Solution: An admissible heuristic always under-approximates the actual cost:

$$h(n) \leqslant g(n, goal) \tag{1}$$

Solution: A consistent heuristic fulfills this triangle inequality for all nodes:

$$h(n) \leqslant g(n, n') + h(n') \tag{2}$$

Problem 2.5.2: Is the given heuristic admissible? Is it consistent? If not, why not?

Solution: Admissible? : Yes

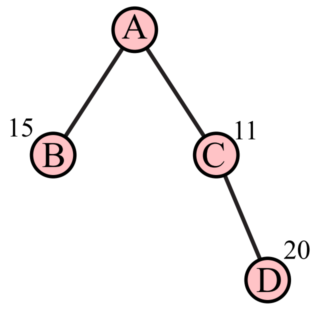
Consistent? : No, since for nodes B and C the triangle inequality does not hold:

$$h(B) \nleqslant c(B, C) + h(C) \tag{3}$$

Problem 2.5.3: Perform A* graph search and tree search to verify your previous answers

Solution: Recall that for A*, $f(n) = g(n) + h(n)$. In the frontier, each node n stores $f(n)$ (as this specifies the priority, i.e. always the node with the smallest value for $f(n)$ is expanded next), and additionally the parent node is memorized for reconstructing the solution path as soon as a goal node is found. The first row always shows the node n that is currently expanded; it contains the costs leading to n (i.e. $g(n)$, rather than $f(n)$ like in the frontier), as we need $g(n)$ for calculating $g(n')$ for each child node n' of n (i.e. $g(n') = g(n) + c(n, n')$ where $c(n, n')$ is the branch costs from n to n')

- A* graph search:



node with $g(n)$ and parent	A(0)	C(10, A)	B(5, A)	D(20,C)

frontier with $f(n)$ and parent	C(10+1=11, A)
	B(5+10=15,A)
		B(5+10=15,A)
		D(20+0=20,C)
			D(20+0=20,C)	

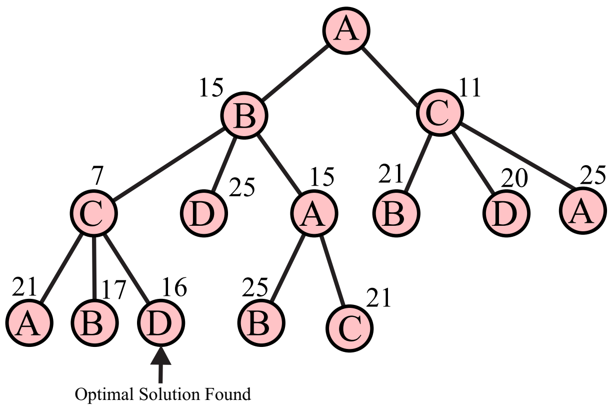
explored	A	AC	ACB	ACB

Note that when exploring C in step 2, we do not add its child B to the frontier, since its $f(n) = 21$ is higher than the value of f of the B already on the frontier.
The solution path is A – C – D. It can be reconstructed by following the parent nodes at the time of expansion (i.e. in the first line of the table), starting from the goal node D: The parent of D in the first line is C, the parent of C is A, which is already the start node, resulting in the path A – C – D.
A* graph search did not find the optimal solution: when expanding B in the 3rd step, we did not add C to the frontier, as it was already expanded at that time; by that, the optimal path A-B-C-D is "missed". If the heuristic was consistent, this would not have happened, as then B would be expanded before C. To see this, consider the triangle inequality of consistent heuristics for B and C: $h(B) \leq c(B, C) + h(C)$. If this held, then we would get the following results for the f-values of B and C after the first step:

$$\begin{aligned}
&f(B) = g(B) + h(B) = 5 + h(B) \\
&\leq 5 + c(B, C) + h(C) \;\; \text{(inserting the triangle inequality)} \\
&= 5 + 1 + h(C) = 6 + h(C) \\
&< 10 + h(C) \\
&= f(C)
\end{aligned}$$

which means that the f-value of B would be smaller than the one of C, giving that B would be expanded first.

- A* tree search:



node with $g(n)$ and parent	A(0)	C(10,A)	B(5,A)	C(6,B)

frontier with $f(n)$ and parent	C(10+1=11, A)
	B(5+0=15,A)
		B(5+0=15,A)
		D(20+0=20,C)
		B(11+10=21,C)
		A(20+5=25,C)
			C(6+1=7,B)
			A(10+5=15,B)
			D(20+0=20,C)
			B(11+10=21,C)
			A(20+5=25,C)
			D(25+0=25,B)
				A(10+5=15,B)
				D(16+0=16,C)
				B(7+10=17,C)
				D(20+0=20,C)
				A(16+5=21,C)
				B(11+10=21,C)
				A(20+5=25,C)
				D(25+0=25,B)

A(10,B)	D(16,C)

D(16+0=16,C)
B(7+10=17,C)
D(20+0=20,C)
A(16+5=21,C)
B(11+10=21,C)
C(20+1=21,A)
A(20+5=25,C)
B(15+10=25,A)
D(25+0=25,B)
	B(7+10=17,C)
	D(20+0=20,C)
	A(16+5=21,C)
	B(11+10=21,C)
	C(20+1=21,A)
	A(20+5=25,C)
	B(15+10=25,A)
	D(25+0=25,B)

A* tree search found the optimal solution: the path is A – B – C – D, again reconstructed by following the parents from the goal node.

Problem 2.6: Application of Search Algorithms: Train Journey

We want to travel from Aberdeen to Birmingham. The (semi-fictional) map of the British rail system is available in Fig. 1. Routes have the same cost in both directions for both time and price. If there is no clear preference for the next node to explore, we explore first the node which is first alphabetically.

Problem 2.6.1: In the table below, we are given a heuristic for cost in time, which is based on the straight-line distances to Birmingham and the maximum speed of the train being no more than 120km/h. Perform Greedy Best-First and A* Search for time cost using graph search. For each step, write down the node which is currently expanded and the nodes in the frontier set, each with its value of the evaluation function $f(n)$ and the parent node.

node n	A	G	E	C	Y	P	M	W	S	D	B
heuristic function $h(n)$	259	203	197	138	86	76	56	55	31	28	0

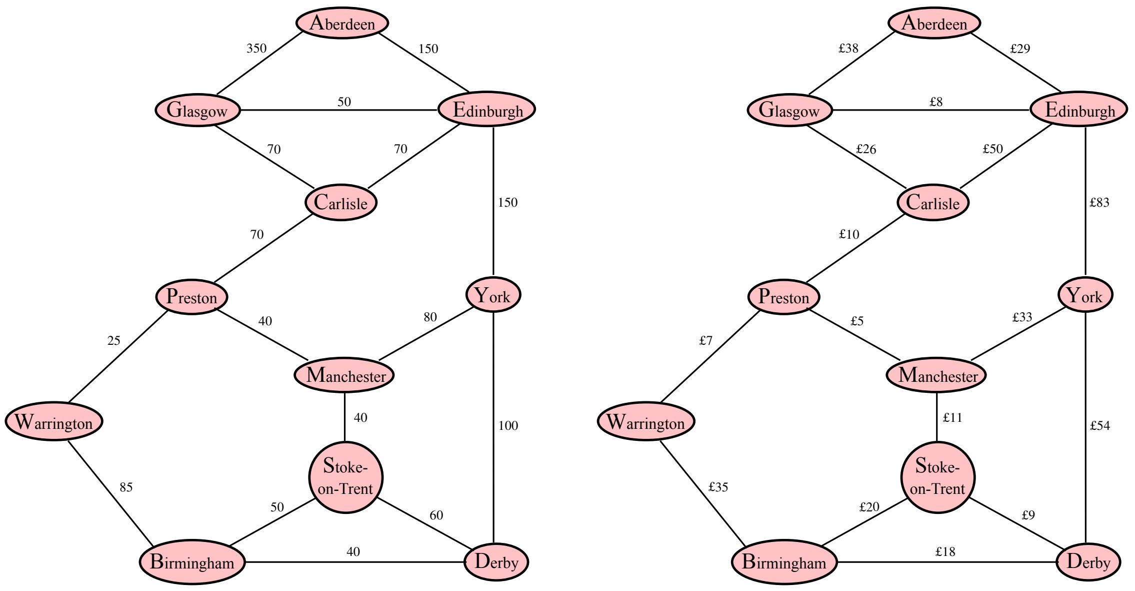


Figure 1: Rail map: left, cost in time (minutes); right, cost in price (£ sterling)

Solution: First of all, let us examine whether the given heuristic is consistent by only looking at how it is defined rather than the concrete values for each node. According to the description in the problem statement, the heuristic h is defined by $h(X) = \frac{d(X,B)}{v_{\max}}$ with $v_{\max} = 120km/h$, i.e. the time the train would need when going the straight-line distance to Birmingham from X with maximum speed. Let us define $d(X,Y)$ as the straight-line distance between X and Y. Then it clearly holds, that $d(X,Z) \leq d(X,Y) + d(Y,Z)$. By dividing the inequality by $v_{\max}$ and inserting B for Z, it follows that

$$\frac{d(X,B)}{v_{\max}} \leq \frac{d(X,Y)}{v_{\max}} + \frac{d(Y,B)}{v_{\max}}$$

which implies that

$$h(X) \leq \frac{d(X,Y)}{v_{\max}} + h(Y)$$

If X and Y are connected, then we know that the time costs are $c(X,Y) = \frac{d(X,Y)}{v}$ with $v \leq v_{\max}$, resulting in $c(X,Y) \geq \frac{d(X,Y)}{v_{\max}}$. With that, the inequality above can be transformed to

$$h(X) \leq c(X,Y) + h(Y)$$

proving that the heuristic is consistent.

Let us now apply the search algorithms.
Greedy Best-First graph search (recall that $f(n) = h(n)$): The solution is A-E-Y-D-B with cost in time of 440 minutes and in price of £184. Note that for the currently expanded node, we only need the parent node in brackets (and not also the g-value, as we do not use path costs in Greedy Bes-First search).

node with parent	A	E(A)	Y(E)	D(Y)	B(D)
frontier with $f(n)$ and parent	E(197, A), G(203, A)	Y(86, E), C(138, E), G(203, A)	D(28, Y), M(56, Y), C(138, E), G(203, A)	B(0, D), S(31, D), M(56, Y), C(138, E), G(203, A)	S(31, D), M(56, Y), C(138, E), G(203, A)
explored	A	A, E	A, E, Y	A, E, Y, D	A, E, Y, D, B

Solution: A* graph search (recall that $f(n) = g(n) + h(n)$): A* finds the optimal solution, which is A-E-C-P-W-B with cost in time of 400 minutes and in price of £131.

node with $g(n)$ and parent	A(0)	E(150, A)	C(220, E)
frontier with $f(n)$ and parent	E(347 = 150 + 197, A) G(553 = 350 + 203, A)	C(358 = 220 + 138, E) Y(386 = 300 + 86, E) G(403 = 200 + 203, E)	P(366 = 290 + 76, C) Y(386 = 300 + 86, E) G(403 = 200 + 203, E)
explored	A	A, E	A, E, C

P(290, C)	W(315, P)	M(330, P)
W(370 = 315 + 55, P) M(386 = 330 + 56, P) Y(386 = 300 + 86, E) G(403 = 200 + 203, E)	M(386 = 330 + 56, P) Y(386 = 300 + 86, E) B(400 = 400 + 0, W) G(403 = 200 + 203, E)	Y(386 = 300 + 86, E) B(400 = 400 + 0, W) S(401 = 370 + 31, M) G(403 = 200 + 203, E)
A, E, C, P	A, E, C, P, W	A, E, C, P, W, M

Y(300, E)	B(400, W)
B(400 = 400 + 0, W) S(401 = 370 + 31, M) G(403 = 200 + 203, E) D(428 = 400 + 28, Y)	S(401 = 370 + 31, M) G(403 = 200 + 203, E) D(428 = 400 + 28, Y)
A, E, C, P, W, M, Y	A, E, C, P, W, M, Y, B

Problem 2.6.2: For the problem with cost in price (see Fig. 1 right), would a heuristic based on distance be valid?

Solution: A consistent heuristic for the cost in price based on the distance could be defined in the following way: Firstly, we compute the minimum price per distance (denoted by $p_{\min}$) that the rail company charges by iterating over all edges in the rail map and diving the price of an edge by its distance (assuming that we have the distances given), and then taking the minimum among these values. Then for $p_{\min}$, it holds that $p_{\min} \leq \frac{c(X,Y)}{d(X,Y)}$ for any connected nodes X and Y, where $d(X,Y)$ is again the straight-line distance from X to Y (which is defined for any nodes X and Y, not only connected ones). The heuristic can now be defined by $h(X) := p_{\min} * d(X,B)$. This heuristic is consistent, which is fairly easy to see and is proven in the following.
For any nodes X and Y with a connecting edge, the following holds:

$$\begin{aligned}
h(X) =& p_{\min} * d(X, B) && \text{definition of our heuristic}\\
&\le p_{\min} * (d(X,Y) + d(Y,B)) && \text{with } d(X,B) \le d(X,Y) + d(Y,B)\\
&= p_{\min} * d(X,Y) + p_{\min} * d(Y,B) &&\\
&= p_{\min} * d(X,Y) + h(Y) && \text{with } p_{\min} * d(Y,B) = h(Y)\\
&\le c(X,Y) + h(Y) && \text{with } p_{\min} \le \frac{c(X,Y)}{d(X,Y)}
\end{aligned}$$

Thus, overall it holds that $h(X) \le c(X,Y) + h(Y)$ for any connected nodes X and Y, proving that our heuristic h is consistent.
However, the heuristic might only have limited benefit, as the price is not necessarily correlated with distance. In this case, our heuristic might often significantly underestimate the true costs from a city to Birmingham, and therefore only perform poorly.

Problem 2.6.3: Which search algorithm would we use if we want to minimize train changes (assuming that we change train at every station)?

Solution: To minimize train changes, we have to find the shallowest solution. Breadth-First search finds this solution.

Problem 2.6.4: Is bidirectional search a good option for the train journey search?

Solution: Yes, because

- there is only one goal node, and it is known.
- actions are reversible, i.e. for a certain node X, we know exactly the nodes that can reach X and the actions that these nodes have to apply to reach X.

The backwards search can be performed just like the forward search.

2 Additional Problems

Problem 2.7: Graph Search

Consider the graph of Problem 2.1 (of Exercise 2a). With the heuristic given in the table below, perform Greedy Best-First and A* Search, each with tree search and with graph search. For each step, write down the node which is currently expanded and the nodes in the frontier set, each with its value of the evaluation function $f(n)$ (and the nodes in the explored set in case of graph search).

node n	A	B	C	D	E	F
heuristic function $h(n)$	4	5	2	1	7	0

Solution: Greedy Best-First tree search:

node with parent	A	D(A)	F(A)
frontier with $f(n)$ and parent	D(1, A), C(2, A), B(5, A)	F(0, A), C(2, A), A(4, D), B(5, A)	C(2, A), A(4, D), B(5, A)

Solution: A* tree search:

node with $g(n)$ and parent	A(0)	C(2, A)	D(4, A)	F(5, D)
frontier with $f(n)$ and parent	C(4 = 2 + 2, A), D(5 = 4 + 1, A), B(6 = 1 + 5, A)	D(5 = 4 + 1, A), B(6 = 1 + 5, A), A(8 = 4 + 4, C)	F(5 = 5 + 0, D), B(6 = 1 + 5, A), A(8 = 4 + 4, C), A(12 = 8 + 4, D)	B(6 = 1 + 5, A), A(8 = 4 + 4, C), A(12 = 8 + 4, D)

Solution: Greedy Best-First graph search:

node with parent	A	D(A)	F(D)
frontier with $f(n)$ and parent	D(1, A), C(2, A), B(5, A)	F(0, D), C(2, A), B(5, A)	C(2, A), B(5, A)
explored	A	AD	AD

Solution: A* graph search:

node with $g(n)$ and parent	A(0)	C(2, A)	D(4, A)	F(5, D)
frontier with $f(n)$ and parent	C(4 = 2 + 2, A), D(5 = 4 + 1, A), B(6 = 1 + 5, A)	D(5 = 4 + 1, A), B(6 = 1 + 5, A)	F(5 = 5 + 0, D), B(6 = 1 + 5, A)	B(6 = 1 + 5, A)
explored	A	AC	ACD	ACD

Problem 2.8: General Questions on Search

Problem 2.8.1: Can Uniform-Cost search be more time-effective or memory-effective than the informed search algorithms mentioned in the lectures? What about cost-effectiveness?

Solution: Greedy Best-First is not optimal, so may be less cost-effective than Uniform-Cost, and may be less time-effective and memory-effective, since the heuristic may lead the algorithm onto paths that appear to be dead ends. A* is optimal, so will give the same (optimal) cost as Uniform-Cost. Similarly, with a bad heuristic (e.g. $h = 0$ for all nodes), A* can perform as badly in memory or time as Uniform-Cost, but not worse (since for $h \geq 0$, any nodes explored in A* will be explored in Uniform Cost, but not vice-versa.)

Problem 2.9: Application of Search Algorithms in Daily Life

Which search algorithm (out of those covered in lectures 3 and 4) do you think your brain uses in the following cases, and what are the advantages of this algorithm? (Note that there are no hard-and-fast solutions for these; it is a matter of personal preference. The purpose of this exercise is to think about situations where you use search algorithms in daily life.)

a. Planning a route from Arad to Bucharest with a map.

Solution: Since you can see the whole map, you are informed and have some sort of heuristic (you would tend to prefer the roads leading towards Bucharest, for example). Therefore, you might first use Greedy Best-First, as you do not know the exact path cost; perhaps also using Bidirectional Search. To find the optimal solution, you might use A* or Uniform-Cost afterwards.

b. Finding an Easter egg:

i. in a building you are unfamiliar with, on your own.
ii. in a building you are familiar with, on your own.
iii. in a building you are familiar with, with a team of people who all have mobile phones.
iv. if the Easter egg has a bell attached (which constantly rings).

Solution: On your own, only nodes close together can be explored (you cannot jump halfway across the building to explore another node). Therefore, some sort of Depth-First search would be sensible.
If you do not know the building, you may opt for Depth-Limited Search (as you do not want to get lost) or Iterative Deepening. If you do know the building, you may opt for Depth-First Search, Depth-Limited Search, or Iterative Deepening.

If you have a team of people, nodes all along the frontier can be explored. You could opt for Breadth-First Search.

With the attached bell, you have a heuristic - how loud or soft the sound is. On your own, perhaps you would prefer Greedy Best-First, as it limits the amount of backtracking you need to do. If you have a team of people, you could perform A* Search.

c. Playing a strategy game, e.g. chess.

Solution: This is very individual! It is left to you to decide what you do. Personally, I use some kind of Depth-Limited Search.