Fundamentals of Artificial Intelligence
Exercise 2b: Informed Search

Sebastian Mair

Technical University of Munich

November 10th, 2023

Recap: Informed Search

Informed Search Algorithms: use **domain-specific knowledge** about the location of goals, represented by **heuristic function** $h(n)$ that

- ... is problem-specific
- ... must be non-negative and fulfill $h(\hat{n}) = 0$ for goal node $\hat{n}$

→ informed search can be more efficient

→ informed search algorithms are instances of **Best-First search**, where $f(n)$ is based on $h(n)$

Recap: Greedy Best-First Search (GBFS)

f(n)=h(n) $\rightarrow$ Idea: always expand the node next that is closest to the goal using just the heuristic function

$\rightarrow$ not optimal, as heuristic can be misleading

Recap: A*-Search

$\boxed{\mathbf{f(n)=g(n)+h(n)}}$ $\rightarrow$ combines ideas from UCS and GBFS

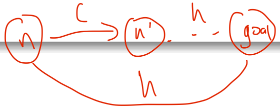
- always expand the node next that might be part of an optimal path
- optimal, if the heuristic is admissible:

Admissible Heuristic

$h(n)$ is admissible if it always underestimates the actual cost to the goal.

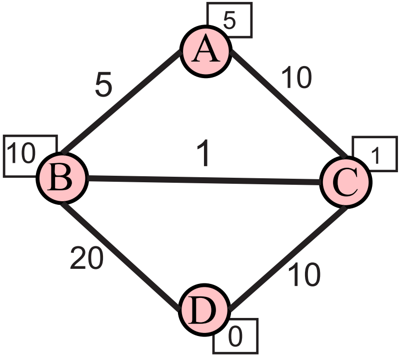
Consistent Heuristic

$h(n)$ is consistent, if the triangle inequality is fulfilled for all nodes n and its successors n' :
$h(n) \leq c(n, n') + h(n')$.



Problem 2.5: Heuristics for Informed Search

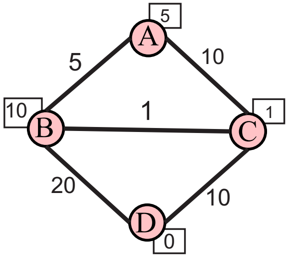
Consider the following graph.
We start from node A and our goal node is D. The path costs are shown on the arcs and the heuristic values are shown at each node.



Problem 2.5.1: Heuristics for Informed Search

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

Tweedback code: **zbq7** (twbk.de/zbq7)

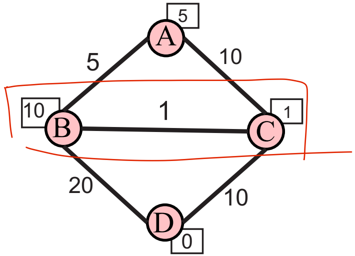


- Admissible:

Problem 2.5.1: Heuristics for Informed Search

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

Tweedback code: **zbq7** (twbk.de/zbq7)



- Admissible: ✓
- Consistent:

$h(n) \leq c(n, n') + h(n')$

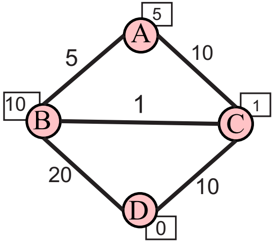
$h(B) \quad c(n, n') + h(C)$

$10 \quad 1 + 1$

Problem 2.5.2: Heuristics for Informed Search

Perform A* search: $f(n) = g(n) + h(n)$ (g(n), parent, f(n))

Node	Children	Frontier	Reached
—	—	A (0, -, 5)	A (0, -)
A (0, -)	B (5, A, 15) C (10, A, 11)	~~C (10, A, 11)~~ B (5, A, 15)	B (5, A) ~~C (10, A)~~
C (10, A)	~~A (20, C, 25)~~ ~~B (11, C, 21)~~ D (20, C, 20) Check	~~B (5, A, 15)~~ D (20, C, 20)	~~D (20, C)~~
B (5, A)	~~A (10, B, 15)~~ C (6, B, 7) ~~D (25, B, 25)~~	~~C (6, B, 7)~~ D (20, C, 20)	C (6, B) low cost D (16, C)
C (6, B)	~~A (16, C, 21)~~ ~~B (7, C, 17)~~ D (16, C, 16) Check	D (16, C) replace	
D (16, C)			

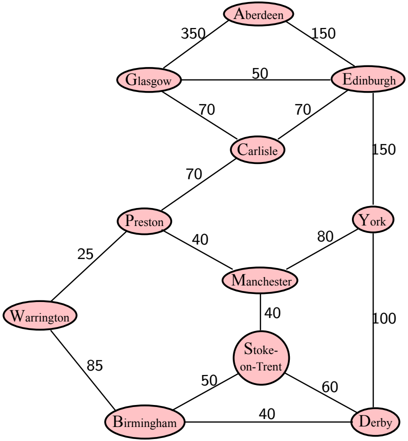


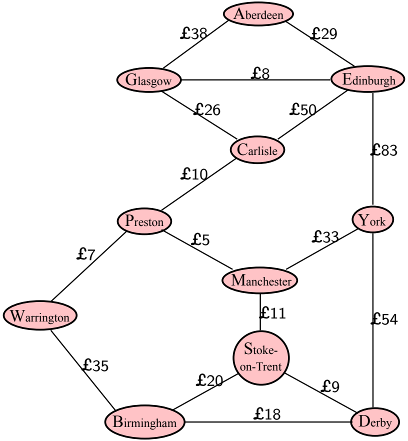
Tweedback code: **zbq7** (twbk.de/zbq7)

D ⇐ C ⇐ B ⇐ A

Problem 2.6: Application of Search Algorithms: Train

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





Problem 2.6.1: Application of Search Algorithms: Train

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.

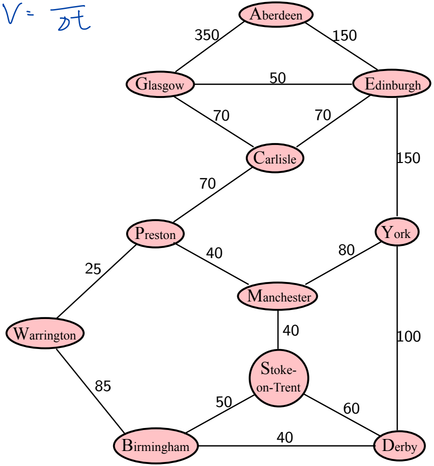
node n	A	G	E	C	Y	P	M	W	S	D	B
heuristic $h(n)$	259	203	197	138	86	76	56	55	31	28	0

Task: Perform Greedy Best-First and A* Search for time cost.

But first of all, let's have a look at the heuristic!

Problem 2.6: Application of Search Algorithms: Train

$V = \frac{\Delta s}{\Delta t}$



Our heuristic for cost in time is based on

- the **straight-line distances to Birmingham**, and
- the **maximum speed of the train**: 120km/h.

Based on this definition, is the heuristic admissible? Is it consistent?

Tweedback code: **zbq7** (twbk.de/zbq7)

$$\frac{d(x, Y)}{V} \geq \frac{d(x, Y)}{V_{max}}$$

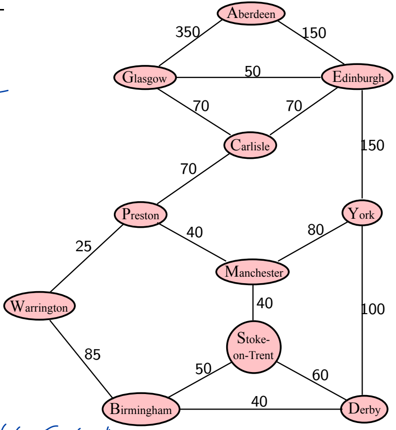
$$t_{rel} \geq t_{max}$$

Problem 2.6.1: Application of Search Algorithms: Train (g(n), pare, f(n))

Greedy Best-First: $f(n) = h(n)$

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

Node	Children	Frontier	Reached
–	–	~~A(0, –, 259)~~	A(0, –)
A(0, –)	E(150, A, 197) G(350, A, 203)	~~E(150, A, 197)~~ G(350, A, 203)	E(150, A) ~~G(350, A)~~
E(150, A)	A(~~300, E,~~ 259) G(200, E, 203) C(220, E, 138) Y(300, E, 86)	G(200, E, 203) C(220, E, 138) ~~Y(300, E, 86)~~	G(200, E) Y(300, E) D(400, Y)
Y(300, E)	E(~~450, Y,~~ 197) M(380, Y, 56) D(400, Y, 28)	D(~~400, Y,~~ 28) M(380, Y, 56)	M(380, Y) S(460, D)
D(400, Y)	Y(~~500, D,~~ 86) S(460, D, 31) B(440, D, 0)	B(~~440, D,~~ 0) S(460, D, 31)	B(440, D)
B(440, D)			



Tweedback code: **zbq7** (twbk.de/zbq7)

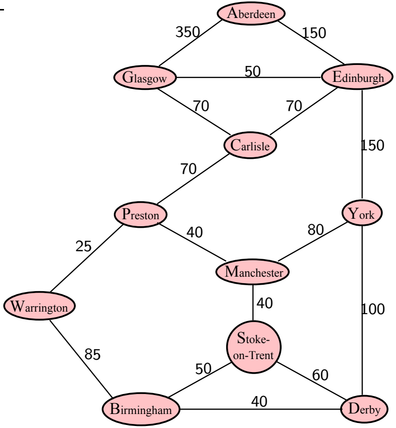
B ⇐ D ⇐ Y ⇐ E ⇐ A

Problem 2.6.1: Application of Search Algorithms: Train

A*: $f(n) = g(n) + h(n)$

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

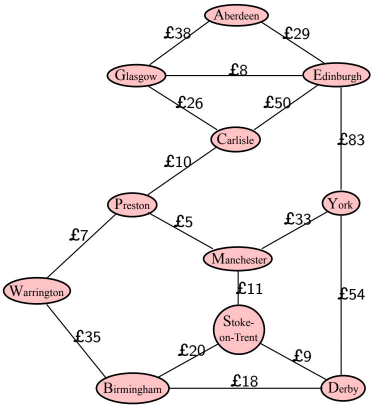
Node	Children	Frontier	Reached



Tweedback code: **zbq7** (twbk.de/zbq7)

Problem 2.6.2: Application of Search Algorithms: Train

For the problem with **cost in price**, would a **heuristic based on distance** be valid?



Tweedback code: **zbq7** (twbk.de/zbq7)

Problem 2.6.3: Application of Search Algorithms: Train

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

Tweedback code: **zbq7** (twbk.de/zbq7)

Problem 2.6.4: Application of Search Algorithms: Train

Is **bidirectional** search a good option for the train journey search?

Tweedback code: **zbq7** (twbk.de/zbq7)