**AI – Heuristic and informed search – 04**

***Heuristic* h(n)**

A heuristic is a function h(n) is a guess to produce a feasible solution in a reasonable time span

**h(n) = estimated cost of the cheapest path from the state at node n to a goal state**

This means that we are trading off on solution optimality (e.g. the shortest path), but we rather gain on time, and **a** solution is guaranteed to be found

***Best-first search (example of informed search)***

Best-first search refers to a **class** of search algorithms, in which the most promising (according to predefined rules) is chosen to be explored first, usually implying the usage of **distance as heuristic** via setting a **priority queue**

To search the graph space, the BFS\* method uses two lists for tracking the traversal:

* + An ‘Open’ list that keeps track of the current ‘immediate’ nodes available for traversal
  + A ‘Closed’ list that keeps track of the nodes already traversed.

A natural question would be “**how does this differ from Dijkstra’s algorithm?”**

BFS\* fails on weighted graphs, as it will choose **distance** over **cost of edge traversal**

***Greedy best-first search***

Greedy BFS\* will always expand to the most promising node (i.e. nearest), as said before, the example in the slides uses **straight line distance** (which is, trivially, **less** than the actual distance)as heuristic for the example, and it certainly finds a solution, just not the best

The usage of backtracking is considered, but Greedy BFS\* focuses on exploring the most promising path **without considering alternative paths**

It can lead to dead ends and/or be suboptimal, and will need backtracking if we want more out of it; The heuristic is:

**f(n) = h(n)**

***A\* algorithm***

**Optimal for positive costs**

A\* algo comes from the fact that heuristic is often misleading, and operates by having the **evaluation function** (aka fitness number) to minimize:

**f(n) = g(n) + h(n)**

where:

* n is the next node on the path
* g(n) is the cost of the path from the start node to n
* h(n) is a heuristic that estimates the cost of the cheapest node from n to the goal

***Admissibility and consistency of heuristic*** (see Manhattan distance)

Having talked about A\*, we introduce two concepts that make a heuristic “good”:

* **Admissibility**

h(n) **never** overestimates the cost of reaching the goal, so it’s not higher than the lowest possible one (this property makes A\* optimal)

* **Consistency**

Graphical user interface, text, application

Description automatically generatedh(n) is **always** less than or equal to the estimated distance from any neighbour vertex to the goal, plus the cost to reach that neighbour

**A consistent heuristic is always admissible, while the converse is not true**

When talking about consistency, we focus solely on the heuristic function

c(N,P) is aka c(n, a, n’), AX is Axiom, H is hypothesis, P is proposition, TH is thesis, L is lemma

Proof : **Consistency** ⇒ **Admissibility**

H1: c(n,a,n’) > 0 All costs are non-negative

H2: h(n) <= c(n,a,n’) + h(n’) Triangle inequality

**TH: h(n) <= h\*(n) Never overestimate (admissible)**

Where h\*(n) is the optimal heuristic, the exact cost to get to the goal through the optimal path

Let W\*(n) be the optimal path from n to n\_goal (goal state); trivially,

**h\*(n) = c(n,a,n’) + h\*(n’)**

* Base for induction:

**h(n0) = h(n\_goal) = 0**

as I’m already there, qed for the base case

Taking the definition of h\*(n),

**h\*(n1) = c(n1,a1,n0) + h\*(n0) = c(n1,a1,n0)** and, trivially:

**h(n1) <= c(n1,a1,n0) +h(n0) <= c(n1,a1,n0)**

as h\*(n0) = h(n0) = 0;

**h\*(n2) = c(n2,a2,n1) +h\*(n1)** and, trivially:

**h(n2) <= c(n2,a2,n1) +h(n1)**

We found a recurring pattern, and we can conclude the proof by induction

* Inductive hypothesis:

**h(n i-1) <= h\*(n i-1)**

**h(n i) <= c(n i, a, n i-1) + h(n i-1)**

**<= c(n i, a, n i-1) + h\*(n i-1) := h\*(n i)**

Then **h(n i) <= h\*(n i)** qed

Proof: **Optimality of A\* with consistent heuristic and graph search (Lemma 1)**

AX1: g(n’) = c(n,a,n’) + g(n) Path cost increases

AX2: f(n) = g(n) + h(n)

**L1: f(n’) >= f(n) for successor n’, provided consistency**

AX2 → P1: f(n’) = g(n’) + h(n’)

= c(n,a,n’) + g(n) + h(n’) substituted using AX1

H2 (previous proof) → P2:

h(n’) >= h(n) – c(n,a,n’)

P1∧P2 → P3: f(n’) >= g(n) + h(n) substituted h(n’), some terms cancel out

Using AX2: **f(n’) >= f(n)** qed

Proof: **Optimality of A\* with consistent heuristic and graph search (Lemma 2)**

(see pdf)

Proof: **Optimality of A\* with admissible heuristic**

Delta = h\* - h or g\*(goal) – h = C\* - h(n)