Week 3 T2

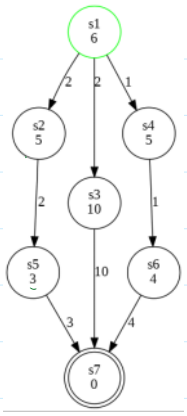
Tuesday, 9 August 2022 1:04 PM

Heuristics

Estimates remaining cost of reaching goal state from current state.

- Can specify manually for each state

More useful to use a general function



Definition (Safe/Goal-Aware/Admissible/Consistent). *Let Π be a planning task with state space $\Theta_\Pi = (S, L, c, T, I, S^G)$, and let h be a heuristic for Π. The heuristic is called:*

- *safe if $h^*(s) = \infty$ for all $s \in S$ with $h(s) = \infty$;*
- *goal-aware if $h(s) = 0$ for all goal states $s \in S^G$;*
- *admissible if $h(s) \leq h^*(s)$ for all $s \in s$;*
- *consistent if $h(s) \leq h(s') + c(a)$ for all transitions $s \xrightarrow{a} s'$.*

→ **Relationships?**

Safe : Heuristic function thinks that it is impossible to reach goal from current state ONLY when it is actually impossible

- $h(s) = \infty \Rightarrow h^*(s) = \infty \quad \forall s \in S$
- Does not necessarily mean that every 'dead end' state will have $h(s) = \infty$, but if $h(s) = \infty$, then state must be a 'dead end'
- Safe from risk of missing potentially viable path.

Dominance

h_1 dominates h_2 if $h_1(s) \geq h_2(s)$ for all s

Goal Aware : $h(s) = 0$ for all goal states

- Can also be $h(s) = 0$ for non goal states

Admissible : Never overestimates remaining cost (Optimistic)

Consistent: heuristic of a state should 'make sense' given the heuristics of its neighbouring states and the transition costs to them

Goal Aware + Consistent => Admissible

Admissible => Safe + Goal Aware

Common Heuristic Functions

- Null heuristic : $h(s) = 0 \quad \forall s \in S$
- Manhattan heuristic : distance in x direction + distance in y direction
- Euclidean heuristic : distance of straight line
- Goal counting : # goals remaining

Informedness of heuristic

- How good of an estimate is h to h^* (the true remaining cost)
- Greater informedness tends to result in better search efficiency (fewer nodes expanded) but calculation is slower

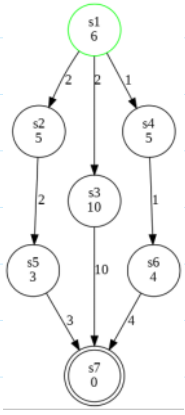
Informed Search Algorithms

Greedy Best First

- Chooses node with smallest $h(s)$
- Open set as priority queue for $h(s)$

A*

- Chooses node with smallest $f(s)$
- $f(s) = g(s) + h(s)$ (cost to reach state s + estimated remaining cost to goal)
- Open set as priority queue for $f(s)$



Weighted A*

- Weight parameter w
- $f(s) = g(s) + w \times h(s)$
- $w = 0$: Dijkstra's
- $w \to \infty$: GBFS

Dijkstra's Algorithm:

- Chooses node with smallest $g(s)$
- Open set as priority queue for $g(s)$
- Blind Search Algorithm