

1 Administrivia

If you wish to turn in anything (e.g., reading summaries, homeworks) electronically, you are free to do so, but please use a portable/nonproprietary format like PDF, PostScript, HTML, or ASCII. Please do *not* use MS Word, WordPerfect, etc.

2 Heuristic Search, Concluded: A*

- Last time we discussed best-first search – basically, always expand node that looks best according to the current heuristic.
- We proposed a couple of heuristics for the 8-puzzle that depend only on the current state description – e.g., the sum of Manhattan distances between displaced tiles and their goal locations.
- Problem with this is that you don't account for how long it took you to find a particular node (more precisely, the cost to get to that node)
- You might end up finding the goal, but only by the longest possible path
- Small modification improves this problem dramatically
- Replace simple evaluation function f with

$$f(x) = g(x) + h(x)$$

Where $g(x)$ measures the distance from the start to x and $h(x)$ estimates the distance from x to the goal. (Require that $g(s) = 0$ for start node s)

- Clearly, the true $g(x)$ is easy to get ahold of; $h(x)$ is harder
- In our previous nomenclature, $f * (x) = g * (x) + h * (x) = g(x) + h * (x)$
- You can interpret this as “the optimal path from s to t that is constrained to pass through x ”
- Note that this is a *nonlocal* measure – you can't calculate $f(x)$ *purely* with information about state x – you also need to know something about the *rest* of the search space. In particular, you have to know what the shortest route to get to x from s is.
- A best-first search employing *this* heuristic is called “algorithm A” — see slide
- Note that there are also a bunch of variant algorithms: AO, B, Z, etc.
- If this algorithm provably always returns the *optimal* path to goal t (or to any node), then it is called “A*”.
- (You should be getting the picture that AI folk often use “foo*” to denote “the optimal version of foo”.

- So when *is* this provably optimal?
- When the heuristic h is **admissible**
- **Definition** a heuristic $h(x)$ is admissible iff for all $x \in \mathcal{S}$, $h(x) \leq h^*(x)$
- I.e., h is admissible if it always *never overestimates* the cost from x to the goal.
- That is, $h(x)$ should always be *optimistic* about how long it takes to get from x to t
- What's the simplest admissible heuristic? A: $h(x) = 0$ for all x
- $h(x) = 0$ is not very helpful, though it *does* produce optimal paths to the goal. We'd like something more *informative*.
- **Definition** Heuristic $h'(x)$ is said to be *more informative* than $h(x)$ iff $h'(x) \geq h(x) \forall x \in \mathcal{S}$
- Clearly, if h' is both more informative than h and admissible, then $h(x) \leq h'(x) \leq h^*(x)$
- So why does this ensure that you actually get optimal solution paths out?
- First, need to know that A* terminates with a path to goal when such a path exists (this is called **completeness**).
- Proof: A* (and any best-first search) terminates only when (a) a goal is found or (b) OPEN is empty. Suppose that a path to goal existed, but A* terminated without finding it. That implies that there's some node n on that path that generates no successors on the path to t . But that blatantly contradicts our assumption that n is on a goal path. Contradiction. Therefore, A* terminates only by finding a path to goal when such a path exists. (I.e., A* is complete).
- Note: if no (finite) path to goal exists, then A* can run indefinitely (in an infinite graph).
- Note also that we haven't used admissibility of h yet. Need that to show that the path A* finds must be optimal.
- **Exercise:** Prove that A* (graph search with an admissible heuristic) returns only optimal solution paths (when solutions exist). Hints: (a) until termination, there must always be some n on the optimal path to the goal in OPEN and (b) consider, by contrast, some other OPEN node n' on a sub-optimal path to the goal.
- **Proof:** Let the cost of the optimal path to the goal be C^* . If n is on OPEN and is on optimal path to goal, then

$$f(n) = g(n) + h(n) = g^*(n) + h(n) \leq g^*(n) + h^*(n) = f^*(n) = C^*$$

Now consider n' . Since n' is on a sub-optimal path to the goal, at some point (possibly just before t), $g(n') > C^*$. Therefore, A* will expand n before n' and will terminate with the optimal path to the goal before finding the sub-optimal path to the goal.

- Note that this *requires* that we actually know the optimal path to any point on the fringe. This is why we have to have the elaborate CLOSED/OPEN bookkeeping. If we could *guarantee* that we find the shortest path to n the *first* time we see n , then there would be no question – we wouldn't have to bother keeping things on CLOSED.
- This happens when h is **monotonic** (a.k.a., **consistent**): if for all children n' of n ,

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

(this is a form of the triangle inequality).

- When you can *guarantee* that h is monotonic, then you can do away with a lot of the book-keeping, b/c you know that you've always found the shortest path to a node the first time you see it.
- Note: monotonic implies admissible, but not vice versa!
- It turns out that A* is also *optimally efficient* for a fixed h , in the sense that no algorithm that expands *fewer* nodes than A* can be guaranteed to find the goal. Thus, the attention is largely on how good your heuristic is.
- When you have monotonicity, f is nondecreasing along any path from the start. Thus, you can draw “contours” through the space representing fringes of constant cost. Everything within the fringe has been expanded and the edge is on the fringe.
- When $h(x) = 0$, then contours are “circular”. As h approaches h^* , they become more elliptical (pointing toward goal). This is why more informative heuristics are better: they find the goal with a smaller “volume” contour (i.e., they expand fewer nodes).
- Note that A* expands *all* nodes with $f(n) < C^*$, so you want there to be as few of those as possible. Nodes with $f(n) > C$ are *pruned*
- Want heuristics that are admissible, but more informative. Turns out that A* is still exponential unless h is *very* informative.
- Specifically, A* is sub-exponential iff

$$|h(n) - h^*(n)| \leq O(\log h^*(n))$$

That is, when the absolute error between h and the true path length grows only with the log of path length. Usually, you can't do this. (Thought exercise: do any of the heuristics we've proposed for the 8-puzzle have this property?)