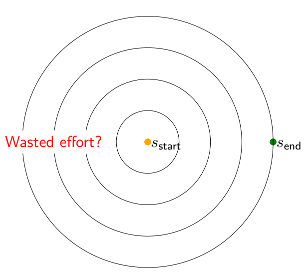
Search: A*

A* algorithm

UCS in action:

A* in action:

Can uniform cost search be improved?



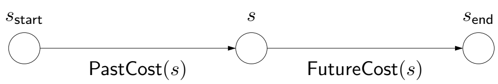
Problem: UCS orders states by cost from s_{start} to s

Goal: take into account cost from s to s_{end}

- Now our goal is to make UCS faster. If we look at the UCS algorithm, we see that it explores states based on how far they are away from the start state. As a result, it will explore many states which are close to the start state, but in the opposite direction of the end state.
- Intuitively, we'd like to bias UCS towards exploring states which are closer to the end state, and that's exactly what A* does.

Exploring states

UCS: explore states in order of $\text{PastCost}(s)$



Ideal: explore in order of $\text{PastCost}(s) + \text{FutureCost}(s)$

A*: explore in order of $\text{PastCost}(s) + h(s)$

Definition: Heuristic function

A heuristic $h(s)$ is any estimate of $\text{FutureCost}(s)$.

- First, some terminology: $\text{PastCost}(s)$ is the minimum cost from the start state to s, and $\text{FutureCost}(s)$ is the minimum cost from s to an end state. Without loss of generality, we can just assume we have one end state. (If we have multiple ones, create a new official goal state which is the successor of all the original end states.)
- Recall that UCS explores states in order of $\text{PastCost}(s)$. It'd be nice if we could explore states in order of $\text{PastCost}(s) + \text{FutureCost}(s)$, which would definitely take the end state into account, but computing $\text{FutureCost}(s)$ would be as expensive as solving the original problem.
- A* relies on a **heuristic** $h(s)$, which is an estimate of $\text{FutureCost}(s)$. For A* to work, $h(s)$ must satisfy some conditions, but for now, just think of $h(s)$ as an approximation. We will soon show that A* will explore states in order of $\text{PastCost}(s) + h(s)$. This is nice, because now states which are estimated (by $h(s)$) to be really far away from the end state will be explored later, even if their $\text{PastCost}(s)$ is small.

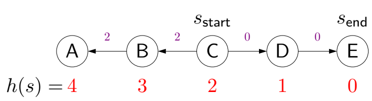
A* search

Algorithm: A* search [Hart/Nilsson/Raphael, 1968]

Run uniform cost search with **modified edge costs**:
$$\text{Cost}'(s,a) = \text{Cost}(s,a) + h(\text{Succ}(s,a)) - h(s)$$

Intuition: add a penalty for how much action a takes us away from the end state

Example:



$\text{Cost}'(C,B) = \text{Cost}(C,B) + h(B) - h(C) = 1 + (3-2) = 2$

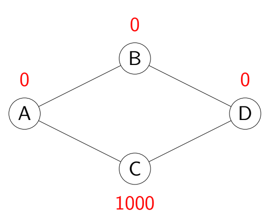
- Here is the full A* algorithm: just run UCS with modified edge costs.
- You might feel tricked because we promised you a shiny new algorithm, but actually, you just got a refurbished version of UCS. (This is a slightly unorthodox presentation of A*. The normal presentation is modifying UCS to prioritize by $\text{PastCost}(s)+h(s)$ rather than $\text{PastCost}(s)$.) But I think the modified edge costs view shows a deeper connection to UCS, and we don't even have to modify the UCS code at all.
- How should we think of these modified edge costs? It's the same edge cost $\text{Cost}(s, a)$ plus an additional term. This term is difference between the estimated future cost of the new state $\text{Succ}(s, a)$ and that of the current state s. In other words, we're measuring how much farther from the end state does action a take us. If this difference is positive, then we're penalizing the action a more. If this difference is negative, then we're favoring this action a.
- Let's look at a small example. All edge costs are 1. Let's suppose we define $h(s)$ to be the actual $\text{FutureCost}(s)$, the minimum cost to the end state. In general, this is not the case, but let's see what happens in the best case. The modified edge costs are 2 for actions moving away from the end state and 0 for actions moving towards the end state.
- In this case, UCS with original edge costs 1 will explore all the nodes. However, A* (UCS with modified edge costs) will explore only the three nodes on the path to the end state.

An example heuristic

Will any heuristic work?

No.

Counterexample:



Doesn't work because of **negative modified edge costs**!

- So far, we've just said that $h(s)$ is just an approximation of $\text{FutureCost}(s)$. But can it be any approximation?
- The answer is no, as the counterexample clearly shows. The modified edge costs would be 1 (A to B), 1002 (A to C), 5 (B to D), and -999 (C to D). UCS would go to B first and then to D, finding a cost 6 path rather than the optimal cost 3 path through C.
- If our heuristic is lying to us (bad approximation of future costs), then running A* (UCS on modified costs) could lead to a suboptimal solution. Note that the reason this heuristic doesn't work is the same reason UCS doesn't work when there are negative action costs.

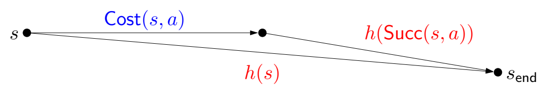
Consistent heuristics

Definition: consistency

A heuristic h is **consistent** if

- $\text{Cost}'(s,a) = \text{Cost}(s,a) + h(\text{Succ}(s,a)) - h(s) \geq 0$
- $h(s_{\text{end}}) = 0$.

Condition 1: needed for UCS to work (triangle inequality).



Condition 2: $\text{FutureCost}(s_{\text{end}}) = 0$ so match it.

- We need $h(s)$ to be **consistent**, which means two things. First, the modified edge costs are non-negative (this is the main property). This is important for UCS to find the minimum cost path (remember that UCS only works when all the edge costs are non-negative).
- Second, $h(s_{\text{end}}) = 0$, which is just saying: be reasonable. The minimum cost from the end state to the end state is trivially 0, so just use 0.
- We will come back later to the issue of getting a hold of a consistent heuristic, but for now, let's assume we have one and see what we can do with it.

Correctness of A*

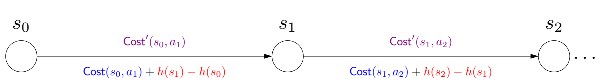
Proposition: correctness

If h is consistent, A* returns the minimum cost path.

- The main theoretical result for A* is that if we use any consistent heuristic, then we will be guaranteed to find the minimum cost path.

Proof of A* correctness

- Consider any path $[s_0, a_1, s_1, \ldots, a_L, s_L]$:



- Key identity:

$$\underbrace{\sum_{i=1}^{L} \text{Cost}'(s_{i-1}, a_i)}_{\text{modified path cost}} = \underbrace{\sum_{i=1}^{L} \text{Cost}(s_{i-1}, a_i)}_{\text{original path cost}} + \underbrace{h(s_L) - h(s_0)}_{\text{constant}}$$

- Therefore, A* (finding the minimum cost path using modified costs) solves the original problem (even though edge costs are all different!)

- To show the correctness of A*, let's take any path of length L from $s_0 = s_{\text{start}}$ to $s_L = s_{\text{end}}$. Let us compute the modified path cost by just adding up the modified edge costs. Just to simplify notation, let $c_i = \text{Cost}(s_{i-1}, a_i)$ and $h_i = h(s_i)$. The modified path cost is $(c_1 + h_1 - h_0) + (c_2 + h_2 - h_1) + \cdots + (c_L + h_L - h_{L-1})$. Notice that most of the h_i's actually cancel out (this is known as **telescoping sums**).
- We end up with $\sum_{i=1}^{L} c_i$, which is the original path cost plus $h_L - h_0$. First, notice that $h_L = 0$ because s_L is an end state and by the second condition of consistency, $h(s_L) = 0$. Second, h_0 is just a constant (in that it doesn't depend on the path at all), since all paths must start with the start state.
- Therefore, the modified path cost is equal to the original path cost plus a constant. A*, which is running UCS on the modified edge costs, is equivalent to running UCS on the original edge costs, which minimizes the original path cost.
- This is kind of remarkable: all the edge costs are modified in A*, but yet the final path cost is the same (up to a constant)!

Efficiency of A*

Theorem: efficiency of A*

A* explores all states s satisfying
PastCost$(s) \leq$ PastCost$(s_{\text{end}}) - h(s)$

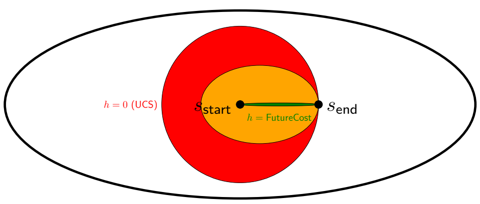
Interpretation: the larger $h(s)$, the better

Proof: A* explores all s such that

$$\text{PastCost}(s) + h(s)$$
$$\leq$$
$$\text{PastCost}(s_{\text{end}})$$

- We've proven that A* is correct (finds the minimum cost path) for any consistent heuristic h. But for A* to be interesting, we need to show that it's more efficient than UCS (on the original edge costs). We will measure speed in terms of the number of states which are explored prior to exploring an end state.
- Our second theorem is about the efficiency of A*: recall that UCS explores states in order of past cost, so that it will explore every state whose past cost is less than the past cost of the end state.
- A* explores all states for which PastCost$'(s)$ = PastCost$(s) + h(s) - h(s_{\text{start}})$ is less than PastCost$'(s_{\text{end}})$ = PastCost$(s_{\text{end}}) + h(s_{\text{end}}) - h(s_{\text{start}})$, or equivalently PastCost$(s) + h(s) \leq$ PastCost(s_{end}) since $h(s_{\text{end}}) = 0$.
- From here, it's clear that we want $h(s)$ to be as large as possible so we can push as many states over the PastCost(s_{end}) threshold, so that we don't have to explore them. Of course, we still need h to be consistent to maintain correctness.
- For example, suppose PastCost$(s_1) = 1$ and $h(s_1) = 1$ and PastCost$(s_{\text{end}}) = 2$. Then we would have to explore s_1 $(1 + 1 \leq 2)$. But if we were able to come up with a better heuristic where $h(s_1) = 2$, then we wouldn't have to explore s_1 $(1 + 2 > 2)$.

Amount explored



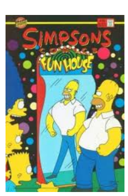
- If $h(s) = 0$, then A* is same as UCS.
- If $h(s)$ = FutureCost(s), then A* only explores nodes on a minimum cost path.
- Usually $h(s)$ is somewhere in between.

- In this diagram, each ellipse corresponds to the set of states which are explored by A* with various heuristics. In general, any heuristic we come up with will be between the trivial heuristic $h(s) = 0$ which corresponds to UCS and the oracle heuristic $h(s)$ = FutureCost(s) which is unattainable.

A* search

Key idea: distortion

A* distorts edge costs to favor end states.



- What exactly is A* doing to the edge costs? Intuitively, it's biasing us towards the end state.

Admissibility

Definition: admissibility

A heuristic $h(s)$ is admissible if
$$h(s) \leq \text{FutureCost}(s)$$

Intuition: admissible heuristics are optimistic

Theorem: consistency implies admissibility

If a heuristic $h(s)$ is **consistent**, then $h(s)$ is **admissible**.

Proof: use induction on FutureCost(s)

- So far, we've just assumed that FutureCost(s) is the best possible heuristic (ignoring for the moment that it's impractical to compute). Let's actually prove this now.
- To do this, we just have to show that any consistent heuristic $h(s)$ satisfies $h(s) \leq \text{FutureCost}(s)$ (since by the previous theorem, the larger the heuristic, the better). In fact, this property has a special name: we say that $h(s)$ is **admissible**. In other words, an admissible heuristic $h(s)$ **underestimates** the future cost: it is optimistic.
- The proof proceeds by induction on increasing FutureCost(s). In the base case, we have $0 = h(s_{\text{end}}) \leq \text{FutureCost}(s_{\text{end}}) = 0$ by the second condition of consistency.
- In the inductive case, let s be a state and let a be an optimal action leading to $s' = \text{Succ}(s, a)$ that achieves the minimum cost path to the end state; in other words, $\text{FutureCost}(s) = \text{Cost}(s, a) + \text{FutureCost}(s')$. Since $\text{Cost}(s, a) \geq 0$, we have that $\text{FutureCost}(s') \leq \text{FutureCost}(s)$, so by the inductive hypothesis, $h(s') \leq \text{FutureCost}(s')$. To show the same holds for s, consider: $h(s) \leq \text{Cost}(s, a) + h(s') \leq \text{Cost}(s, a) + \text{FutureCost}(s') = \text{FutureCost}(s)$, where the first inequality follows by consistency of $h(s)$, the second inequality follows by the inductive hypothesis, and the third equality follows because a was chosen to be the optimal action. Therefore, we conclude that $h(s) \leq \text{FutureCost}(s)$.
- Aside: People often talk about admissible heuristics. Using A* with an admissible heuristic is only guaranteed to find the minimum cost path for tree search algorithms, where we don't use an explored list. However, the UCS and A* algorithms we are considering in this class are graph search algorithms, which require consistent heuristics, not just admissible heuristics, to find the minimum cost path. There are some admissible heuristics which are not consistent, but most natural ones are consistent.