CS 337, Fall 2023
A* Search

Scribes: Vijay Balsubramaniam*, Ginjala Lohith Reddy*, Dheer Harshad Banker*,
G Chanikya Prakash, Koyi Sree Nikhil, Olivia Sommer Droob, Naman Soni
Edited by: Vishal Tapase

October 30, 2023

Recap

We are exploring Search Algorithms (*pun intended*), to help an *agent* look for the optimal *path* (with respect to some *cost function*) to a goal state s_{goal}, in a problem which is characterized by the following defining features:

- The state space S which is a set of states
- The goal state $s_{goal} \in S$ that the *agent* attempts to attain
- The start state $s_{start} \in S$ from where the *agent* begins its journey
- A (deterministic) transition function $T : S \times A \to S$ where A is the set of actions (i.e., choices offered to the agent at each state)

Here, a *path* is merely a sequence of states in S. We additionally associate the problem with a *Cost Function*, Cost : $S \times A \to \mathbb{R}$, where, for some $s \in S$ and an action $a \in A$, Cost(s, a) denotes the cost for the *agent* in moving from s by taking the action a. We will assume a perspective of the problem as a graph, with S forming the set of nodes and T corresponding to the edges.

At any stage in the search procedure, the agent has an associated set of explored states, a set of unexplored states, and a frontier set of states which separates the explored and the unexplored. The agent iteratively picks a state from the frontier state and moves it to the explored set. It moves the unexplored children of a node in the frontier(these selections are all on some criteria: this is what differs in the search algorithms!) and moves it into the frontier.

Five algorithms were introduced:

Depth First Search

Expanding on the deepest node with a LIFO Stack as frontier

*Larger credit goes to these scribes for the final notes.

Breadth First Search

Expanding on the shallowest node with a FIFO Queue frontier

Uniform-Cost Search (UCS)

This search strategy explores a problem's state space by considering the cost of each path and selecting the path with the **minimum cumulative cost** at each step. In UCS, the frontier is managed as a **priority queue**, where states are ordered based on their path cost, with the least costly states explored first. The evaluation function is as follows:

$$f(n) = g(n) \quad \text{(where g(n) is the lowest cost from the start state to n)}$$

Greedy Best-First Search

Greedy Best First search tries to expand the node that is **closest to the goal**, on the grounds that this is likely to lead to a solution quickly. This shows why the algorithm is called "greedy"—at each step it tries to get as close to the goal as it can. However, this strategy is **not guaranteed to give an optimal solution**. The evaluation function is as follows:

$$f(n) = h(n) \quad \text{(heuristic estimate of distance from n to the goal state)}$$

A* Search

Expanding on the optimization function $f(s) = g(s) + h(s)$

1 A* Search

1.1 What Motivates Greedy Best First Search and A* Search?

Uniform Cost Search is an uninformed search. Its frontier grows based solely on the cost of states from the start state. It uses no knowledge of how far it is from s_{goal}. Intuitively, this leads to wasteful exploration of the frontier in a direction which will potentially not lead to the goal. Greedy Best First Search tries to combat this problem by growing its frontier in the direction which minimises a suitable heuristic estimate of the cost from s_{goal}.

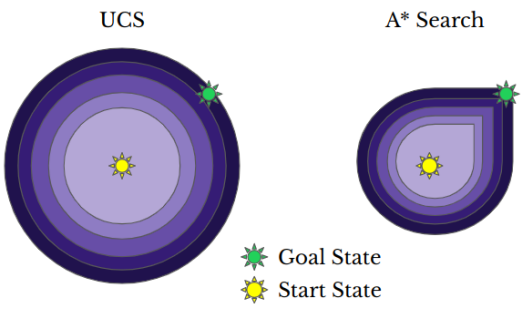

Figure 1: Contours (of equal cost from s_{start}) demonstrating growth of the frontier

Greedy Best First Search also faces a challenge due to its myopic focus on growing in the (perceived) direction of s_{goal} which may result in wasteful frontier growth due to the inaccuracies of the heuristic estimate's modeling of the true cost to s_{goal}. A* Search aims to get the best of both the worlds.

Uniform Cost Search (UCS) explores states in the order of $g(s)$. Greedy Best-First Search explores states in order of $h(s)$, but A* explores states in the order of $g(s) + h(s)$. Ideally, we should search in the order of $g(s) + h^*(s)$.

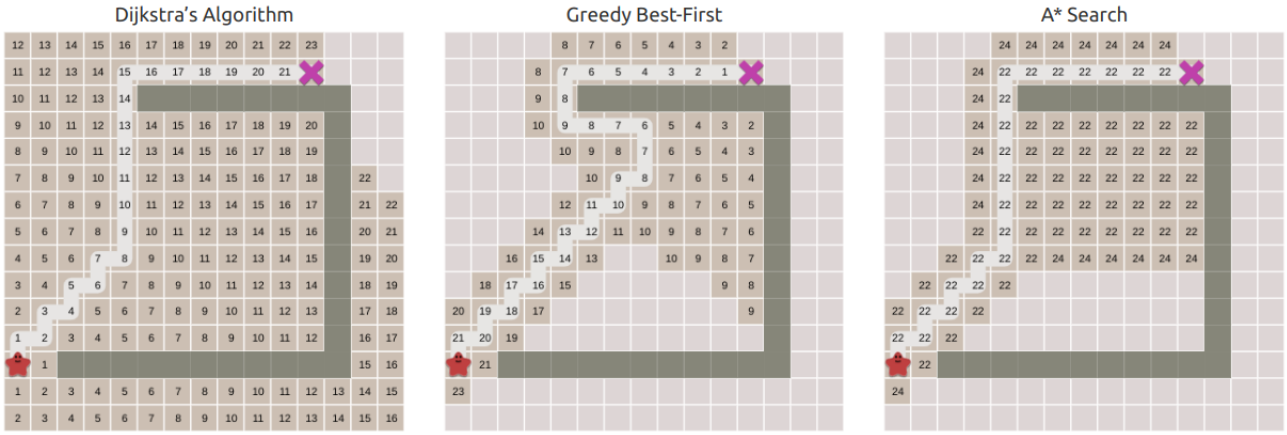

Figure 2: Comparision of different search algorithms

1.2 A* Algorithm

It is pretty much the same as UCS, but instead of using the original edge costs, we modified them as follows:

$$cost'(s, a) = cost(s, a) + h(s') - h(s)$$

The added term i.e., $h(s') - h(s)$ is the penalty on an action $\boldsymbol{a}$ since it indicates how much the search algorithm drifted from reaching the goal state. Also ensure that $h(s_{goal}) = 0$. Since we are running UCS, the cost of edges must be non-negative for the search to terminate i.e.,

$$cost'(s, a) \geq 0$$

$$h(s) \leq h(s') + cost(s, a) \text{ (\textbf{Consistency} of a heuristic)}$$

A heuristic is said to be **admissible** if $h(s) \leq h^*(s)$ (True minimum cost)

1.3 Why is A* optimal?

Claim: If h is a consistent heuristic then A* search is guaranteed to return the optimal path.

Proof: Consider a path of length n actions, say s_0, s_1, ..., s_n where $s_0 = s_{\text{start}}$ and $s_n = s_{\text{goal}}$.
(shorthand) Let $h_i = h(s_i)$ and $c_i = \text{Cost}(s_{i-1}, a_i)$, where $h(s_i)$ is heuristic estimate of cost from state s_i to s_{goal} and $\text{Cost}(s_{i-1}, a_i)$ is cost of moving from state s_{i-1} to s_i by action a_i.

Imagine that the agent plans to perform a UCS; then, the agent expands its frontier on the state in the frontier with the minimum Path Cost (recall this is given by $g(s_k) = \sum_{i=1}^{k} c_i$) from s_0 to s_k. Suppose we

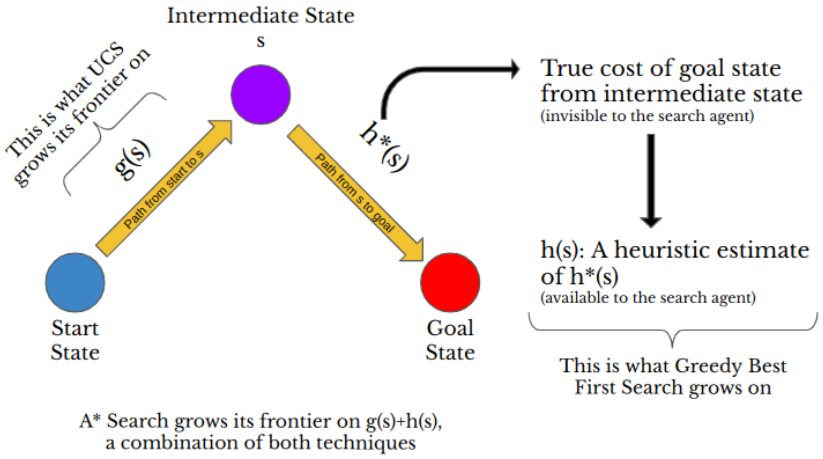

Figure 3: A simple diagram (as drawn on the blackboard) illustrating the intuition behind A* search

redefine this Path Cost function and modify it. The Modified Path Cost is given by:

$$\text{Modified Cost } = (c_1 + h_1 - h_0) + (c_2 + h_2 - h_1) \ldots + (c_n + h_n - h_{n-1}) \tag{1}$$

$$= \sum_{i=1}^{n} c_i + h_n - h_0 \tag{2}$$

$$= \sum_{i=1}^{n} c_i - h_0 \qquad (\because h_n = h_{goal} = 0) \tag{3}$$

As we can see above, the Modified Path Cost of any path (that ends on s_{goal}) in the new regime is simply the original Path Cost (to s_{goal}) of UCS subtracted by a constant (h_0).
h_0 is constant because it is not path dependent. This means that the ordering of all paths from s_{start} to s_{goal} based on their modified path costs will be same as with their original path costs. Now comes the punch line:

Running UCS on original edge costs

$\iff$ Returns the path from s_{start} to s_{goal} with the least original path cost

$\iff$ Returns the path from s_{start} to s_{goal} with the least modified path cost($\because$ ordering of paths is same!)

$\iff$ Running UCS on modified edge costs

But running UCS on the modified costs of edges is exactly what A* search is. Therefore, A* search always returns the optimal cost path.

Q. In the claim we required a consistent heuristic but where was consistency invoked?
A: The whole idea is running UCS on modified edge costs. This implies the new edge costs should be positive.(Because UCS algorithm assumes its edge costs to be positive)

$$c_i + h_i - h_{i-1} \geq 0 \implies h(s_{i-1}) \leq \text{Cost}(s_i, a_i) + h(s_i) \tag{4}$$

As we can see from equation 4, the heuristic is consistent.

1.4 Consistency implies Admissibility

A heuristic h is said to be **admissible** if it never overestimates the true cost to s_{goal}. The heuristic is said to be **consistent** if when going from neighboring states a to b, the heuristic difference/step-cost never overestimates the actual step cost. We continue to assume $h(s_{\text{goal}}) = h^*(s_{\text{goal}}) = 0$. We also assume that the true cost is additive over the edges of the state graph.

$$\text{Admissibility}: \quad h(s) \leq h^\star(s) \;\; \forall s \in S \tag{5}$$

$$\text{Consistency}: \quad h(s) \leq h(s') + \text{Cost}(s, a) \tag{6}$$

For brevity in our further discussion, we define $\tau : S \times \mathbb{N} \to 2^S$, as follows. $\tau(s, k)$ represents the set of states from which the goal state s_{goal} is reachable by taking a sequence of k actions. It follows that $\tau(s_{\text{goal}}, 1)$ denotes the set of states from which s_{goal} is reachable by taking a single action. We will perform an induction on k.

Base Case: For any state $s \in \tau(s_{\text{goal}}, 1)$ from which an action a leads directly to s_{goal}, consistence implies:

$$h(s) \leq h(s') + \text{Cost}(s, a) \tag{7}$$

But we earlier assumed $h(s) = h^\star(s)$ and so:

$$h(s) \leq h^\star(s') + \text{Cost}(s, a) \tag{8}$$

$$\implies h(s) \leq h^*(s) \quad (\because \text{Cost}(s, a) \geq 0) \tag{9}$$

Inductive Hypothesis: For any state $s \in \tau(s_{\text{goal}}, k) \forall$, we assume that $h(s) \leq h^\star(s)$ holds.
Induction: For any state $s \in \tau(s_{\text{goal}}, k+1)$ there is a sequence of $k+1$ actions which leads to s_{goal}, by definition of τ. Let a be the first such action, then, taking action a from s will take us to some state s' which must belong to $\tau(s_{\text{goal}}, k)$, since taking the remainder k actions in the sequence leads the agent to s_{goal}. So, the inductive hypothesis implies, $h(s') \leq h^\star(s')$ as $s' \in \tau(s_{\text{goal}}, k)$
Consistency grants:

$$h(s) \leq h(s') + \text{Cost}(s, a) \tag{10}$$

$$\implies h(s) \leq h^*(s') + \text{Cost}(s, a) \tag{11}$$

Since $h^*(s)$ is the true optimal cost to move the agent from s to s_{goal}, we have

$$h^*(s) = h^*(s') + \text{Cost}(s, a) \tag{12}$$

and therefore from the relations 11 and 12:

$$h(s) \leq h^*(s) \tag{13}$$

Thus, consistency of a heuristic implies admissibility.

Key Points to Note

- UCS explores all states where the cost from the start state is less than or equal to the cost from start to the goal state i.e. $g(s) \leq g(s_{\text{goal}})$. This is true because UCS has a frontier in the form of a priority queue which allows the agent to pick the next state s with the least known cost from s_{start}. The UCS invariant mandates that this cost be the optimal cost of the path to the picked s. This in turn implies that any state picked before s_{goal} in the algorithm will have $g(s) \leq g(s_{\text{goal}})$.
- A* modifies costs and explores states whose $g'(s) = g(s) + h(s) - h(s_{\text{start}})$ is less than or equal to $g'(s_{\text{goal}}) = g(s_{\text{goal}}) + h(s_{\text{goal}}) - h(s_{\text{start}})$, and $g'(s) \leq g'(s_{\text{goal}})$ which implies $g(s) + h(s) \leq g(s_{\text{goal}})$ $(\because h(s_{\text{goal}}) = 0)$.

2 Why is A* often more efficient than UCS?

A* is typically more efficient than UCS in terms of the number of explored states before termination. Let's provide some mathematical intuition to demonstrate why A* can be more efficient than UCS in exploration.

From Box 1.4, it is clear that A* explores states s for which $g(s) + h(s) \leq g(s_{goal})$. If we choose the value of $h(s)$ to be sufficiently high, we can expect that some of the states that were explored by vanilla UCS (under the regime $g(s) \leq g(s_{goal})$) will go unexplored since the LHS will overshoot the value of $g(s_{goal})$. This is precisely what is visualised in figure 1, where the aforementioned overshooting helps A* redirect itself from points away from s_{goal}. This is also motivation to find h that is sufficiently large enough so that the $g(s_{goal})$ is exceeded, and we can obtain the benefits of lesser exploration.

Aside: Admissibility does not always entail Consistency

Here's an additive cost system on a graph which presents a counter example to the claim that admissible heuristics are consistent.

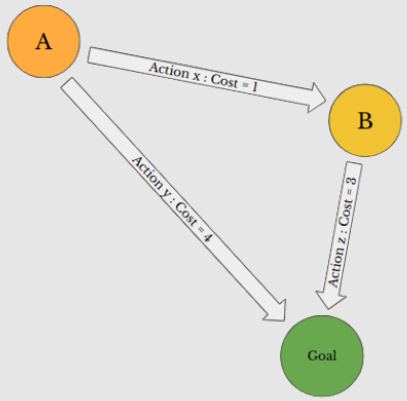

Figure 4: A simple graph to demonstrate admissibility but not consistence of a heuristic

The costs associated with each action is given in the figure. With $h^*(s_{\text{goal}}) = 0$ we have $h^\star(A) = 4$ and $h^\star(B) = 3$. Defining a heuristic h with $h(s_{\text{goal}}) = 0$, $h(A) = 4$ and $h(B) = 1$, it is evident that h is admissible, but h is inconsistent because $h(A) = 4 > \text{Cost}(A, x) + h(B) = 1 + 1 = 2$

3 Relaxation

How do we find consistent heuristics?

We'd like to be able to find $h^*(s)$ for the original problem , but instead we find $h^*_{relax}(s)$ for an easier relaxed problem. The relaxed problem can be obtained by removing constraints.This is equivalent to reducing the edge cost from inf to a finite cost. Below two examples are illustrating how to relax a problem.

Example 1: Sun to Moon: Obstacle-Free Grid Navigation

A simple example could be the following goal: "Move from the sun to the moon without moving into a wall or off the grid" Relaxing this problem would be to remove the walls. This results in a way easier problem with the closed-form solution that $h^*_{relaxed}(s)$ is equal to the Manhatten distance between the sun and the moon.

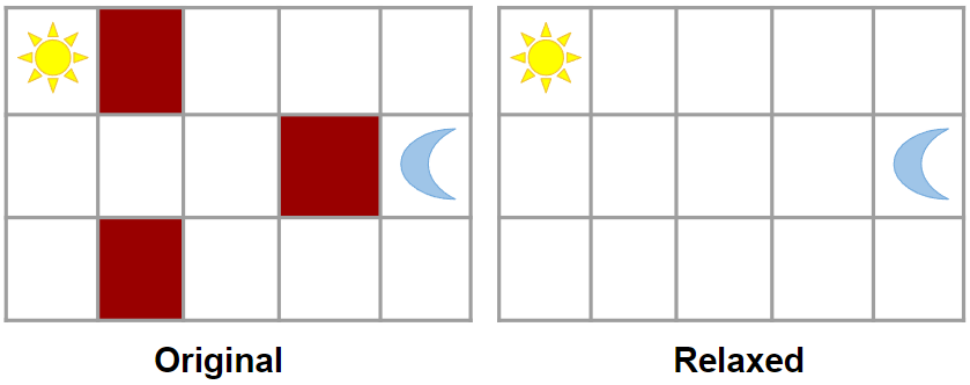

Figure 5: Obstacle Relaxation: Before and After

Example 2: 8-Tile Puzzle

Another example would be the 8-puzzle. Here the original goal is to slide around the tiles to make a specific configuration. No tiles are allowed to occupy the same position at any time. The way this problem can be relaxed is by allowing the tiles to overlap. This means that the problem now consists of 8 independent subproblems that can easily be solved.

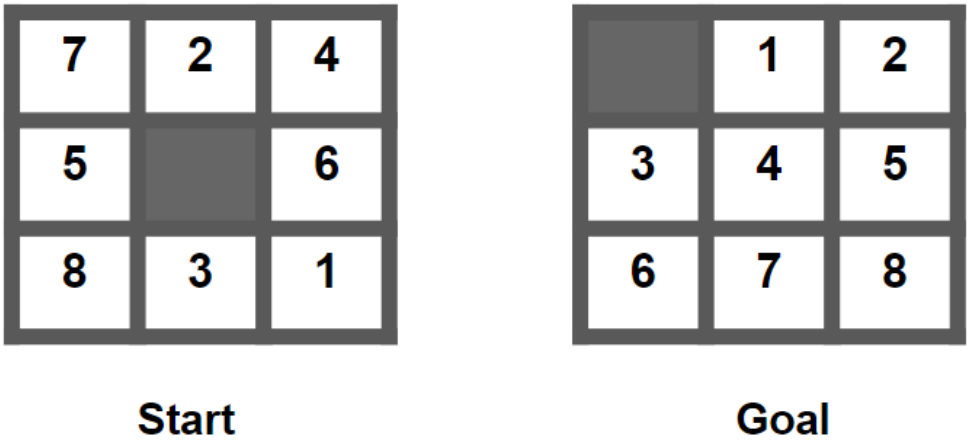

Figure 6

Now, for this relaxed problem let the cost be denoted by $cost_{relax}(s, a)$, then we have :

$$cost_{relax}(s, a) \leq cost(s, a)$$

We choose our heuristic $h(s)$ to be the $h^*_{relax}(s)$ of the relaxed problem, now we have to prove that our chosen heuristic is consistent. We can prove this by using the fact that $cost_{relax}(s, a) \leq cost(s, a)$ and $h(s) = h^*_{relax}(s)$

Proof.

$$h(s) \leq cost_{relax}(s, a) + h(s') \quad (h(s) = h^*_{relax}(s))$$
$$h(s) \leq cost(s, a) + h(s') \quad (cost_{relax}(s, a) \leq cost(s, a))$$

□

What if we have multiple consistent heuristics

We want our heuristic to be maximal in order to decrease the number of explored states by UCS and make the search fast. So the natural thought will be to take the maximal heuristic, we will now try to prove that if we have multiple consistent heuristics, then if for each state, we pick the heuristic with maximum value, then the resulting heuristic will also be consistent.

Consider multiple consistent heuristics $h_1, h_2, h_3, ..., h_n$, design new heuristic h' such that

$$h'(s) = max(h_1(s), h_2(s), h_3(s), ..., h_n(s))$$

Claim: h' is consistent
Proof: *Let* $h'(s) = h_i(s)$ *and* $h'(s') = h_j(s')$ *where* $1 \leq i, j \leq n$

Proof.

$$\begin{aligned}
h_i(s) &\leq cost(s,a) + h_i(s') && (h_i \text{ is consistent}) && (1)\\
h_i(s') &\leq h_j(s') && (Construction\ of\ h') && (2)\\
h_i(s) &\leq cost(s,a) + h_j(s') && (from\ 1\ and\ 2) && (3)\\
h'(s) &\leq cost(s,a) + h'(s') && (from\ 3\ )
\end{aligned}$$

□

Additional Point

- We found that A* returns the optimal path in a setting which features additive edge costs $Cost(s, a)$ and a consistent heuristic. The proof is generalizable to any case where the costs form a **Tropical Semiring**.