# CS 520 : Notes on $A^*$

16:198:520

The goal in class was to motivate $A^*$ (*prioritizing the fringe based on estimated total distance from start to goal through a given node,* $f(n) = g(n) + h(n)$) and present some of its nice properties (optimality, efficiency, etc), but here I want to look in more detail at why and how it works the way it does. The point of these notes is to address some of the more technical aspects of $A^*$ in a more rigorous way than was presented in class.

Consider a search tree with start node $s$, and let $G$ be the set of goal nodes. Assume that any incremental cost between nodes is positive. For any node $n$, let $g(n)$ be the distance traversed from $S$ to $n$, and let $h(n)$ be an estimate of the minimum remaining distance from $n$ to any node in $G$. Let $C^*(x, y)$ be the minimum cost from $x$ to $y$, and let $C^* = C^*(s, G)$. For the purposes of these notes, consider implementing $A^*$ *without* a closed list, so nodes can be entered into the priority queue potentially multiple times at various priorities, based on the length of the discovered path from $S$ to $n$. Note, when a goal node is discovered with total cost $C^*$, it will be inserted into the priority queue with priority $C^*$.

**Claim:** $A^*$ will return a goal node if it is possible to reach the goal from $S$.

**Proof:** This is relatively straightforward. $A^*$ is going to continue to run until one of two things happen - either the fringe is completely empty and there are no more nodes to explore in which case there is no path from $S$ to $G$, or it has explored the connections of enough nodes from the fringe that it has discovered a path to $G$ and terminates before then. Independent of priority and ordering, $A^*$ will run until one of these two exit conditions is met.

If we are willing to make some assumptions about the heuristic $h$, in particular that the estimate of the remaining distance is in some sense 'good' or accurate, we can say more about the behavior of $A^*$:

**Definition:** A heuristic $h$ is **admissible** or **optimistic** if for any node $n$,

$$h(n) \leq C^*(n, G). \tag{1}$$

In other words, $h$ is optimistic if it never overestimates the true or optimal remaining cost from $n$ to $G$.

**Claim:** If $h$ is **admissible**, then the first time $A^*$ returns a goal node represents the discovery of the optimal path from $s$ to $G$.

**Proof:** Because $h$ is optimistic, the priority of any node $x$ on the fringe represents a conservative estimate of the true cost from start to goal through $x$, i.e.,

$$f(x) \leq C^*(s, x) + C^*(x, G). \tag{2}$$

Any goal node placed in the fringe has a priority given by the actual distance from $s$ to that goal node - no estimation is required. The first time that $A^*$ removes a goal node from the fringe, it has a total cost *less than or equal to all current priorities*, i.e., the true cost of that goal node is less than or equal to the estimated cost to the goal through any remaining node in the fringe. Because of the above bound, we have that the true cost of the returned goal node is *less than or equal to the true cost through any remaining node in the fringe.* As such, there can be no shorter path through any remaining node, and the path discovered is optimal.

Note: suppose you discover a goal node, but the cost of that goal node is *greater* than the priority of some node in the fringe. At that point, while you have discovered a path to the goal, you cannot be sure that a shorter path doesn't exist through one of the remaining fringe nodes (based on the estimates available). Hence, $A^*$ inserts that goal node into the priority queue at the appropriate point, and continues exploring.

**Claim:** If $h$ is optimistic, $A^*$ will expand no node of total estimated cost $f(n) > C^*$.

**Proof:** It is worth considering the question - what nodes will $A^*$ actually expand. At the very least, we know that $A^*$ will expand every node along a shortest path from $s$ to $G$. For any such node $n$ along this path (including, importantly, immediate neighbors of $s$), we must have that $f(n) = g(n) + h(n) \leq C^*(s, n) + C^*(n, G) = C^*$, i.e., every node along this path will be given a priority less than or equal to $C^*$. As such, all the nodes along this path will be given a priority less than (i.e., will be expanded before) any node with a total estimated cost greater than $C^*$. Since the goal will be discovered at total cost $C^*$ when all the nodes along this path are expanded, $A^*$ will terminate before ever getting to a node with total estimated cost greater than $C^*$.

To further refine this, we refine slightly what it means for a heuristic to be good:

**Definition:** A heuristic $h$ is **consistent** if for any node $n$ and any child $n'$ of $n$,

$$h(n) \leq C^*(n, n') + h(n'). \tag{3}$$

This can be interpreted as saying the "estimated cost to $G$ from $n$" is no worse than the "estimated cost to $G$ from $n$ *through* $n'$" - the added restriction of passing through $n'$ only makes things worse. This is a very natural property for $h$ to have in a lot of contexts, and essentially enforces that not only is the estimate optimistic in the previous sense, but *all estimates are consistent with each other.*

**Claim:** If a heuristic $h$ is consistent, then $h$ is optimistic.

**Proof:** The proof proceeds inductively, working backwards from the goal set. Assume $h$ is consistent, and let $n$ be a node with a child in $G$. We have then that for any child $n'$ in $G$ (observing that $h(n') = 0$),

$$h(n) \leq C^*(n, n') + h(n') \leq C^*(n, G) = C^*(n). \tag{4}$$

Hence $h$ is optimistic for any node one step back from the goal set $G$. We can extend this argument inductively backwards. Consider any node $n$, and let $n'$ be the next node on the shortest path from $n$ to $G$. Using consistency, and an assumption of admissibility on the closer node, we have

$$h(n) \leq C^*(n, n') + h(n') \leq C^*(n, n') + C^*(n') = C^*(n), \tag{5}$$

and the result of admissibility percolates outwards from the goal set one step at a time.

It is worth noting here from a practical perspective that while consistency implies admissibility, consistency is a stronger notion (making a statement about how all the various estimates related to one another) and is frequently harder to prove for a given heuristic. However, most of the admissible heuristics you will come across in practices are additionally consistent.

**Claim:** If $h$ is consistent, $A^*$ will expand every node with total estimated cost $f(n) < C^*$.
[*Worth noting: if a node is unreachable from the start node, we would take* $g(n) = \infty$.]

**Proof:** Let $n$ be a node with estimated cost $f(n) < C^*$, and let $n'$ be the immediately preceding node on the shortest path between $s$ and $n$. Note that, applying consistency of the heuristic, we have that

$$f(n') = g(n') + h(n') \leq g(n') + C^*(n', n) + h(n) = g(n) + h(n) = f(n) < C^*. \tag{6}$$

Working backwards inductively in this way, we see that of $f(n) < C^*$, *every node along the shortest path from $s$ to $n$ must have a priority of at most $C^*$*, including the start node $s$. Because $s$ starts on the fringe and is expanded, with all its children added to the fringe, every node along this path must be added to the fringe and expanded before $A^*$ discovers the optimal path of cost $C^*$ and terminates. Therefore, if $f(n) < C^*$, $A^*$ will expand $n$.

At this point, we have established the following claims about executing $A^*$ with a consistent heuristic:

- $A^*$ will discover a goal node if one is reachable.
- The first path to a goal node that comes off the fringe under $A^*$ will be optimal.
- $A^*$ expands all nodes with $f(n) < C^*$.
- $A^*$ expands no node with $f(n) > C^*$.

For the case of $f(n) = C^*$, there is some flexibility as to how $A^*$ will execute, depending on the order with which these nodes are discovered and loaded into the queue - essentially coming down to how the priority queue handles ties in priority.

One important question remains: how does $A^*$ rank in efficiency against any other optimal algorithm? Consider a proposed algorithm, $B^*$, that has access to the same heuristic that $A^*$ does. The algorithm $B^*$ may structure its

fringe and decision making process in any kind of way - we want to make as few assumptions as possible about $B^*$. While $B^*$ may be more efficient on a specific search problem or tree than $A^*$, $B^*$ *cannot expand fewer nodes than* $A^*$*, universally for any search problem.*

We justify this with the following observations:

- Suppose that for a given search problem, $B^*$ returns a path of cost $C$ to the goal set, but fails to expand a node $n$ with $f(n) < C$. As noted previously, every node along the shortest path from $s$ to $n$ will also have a value of $f$ that is less than $C$. Based on the information that $B^*$ has to work with (in particular, the heuristic $h$) it cannot rule out the possibility that any one of these nodes has a *true* cost from $s$ to $G$ of something less than $f$, i.e., less than $C$. Again, for this *specific* problem it is possible that the path $B^*$ returns is optimal. But $B^*$ has not explored enough to justify this with confidence - in fact, by modifying some of the costs and heuristic values that follow $n$ en route to the goal set, we can construct instances of a search problem where $B^*$ will perform exactly the same, returning the same path, but in fact the true optimal path lies through $n$.
- Suppose that for a given search problem, $B^*$ returns a path of cost $C$, but expanded a node $n$ with $f(n) > C$. Recall again that because the heuristic is consistent, every node along the path returned is going to have an estimated total cost of $f$ *at most* $C$. By reordering the explorations (i.e., bringing $B^*$ more in line with $A^*$), we could have still discovered the path of cost $C$ by expanding each node along this path, and have discovered the path of cost $C$ without ever taking the time to expand node $n$. This would've generated the same final result but been more efficient in terms of nodes explored.

Hence we see that any algorithm $B^*$, however it is structured, can be made more efficient by not exploring nodes of higher estimated cost than those observed, or is in some sense *too* efficient and avoided expanding nodes it needed to ensure that the algorithm will universally return an optimal path. The only way to avoid both these potential traps is if $B^*$ expands every node that $A^*$ does, and avoids every node that $A^*$ does. *[Again, observing some flexibility with regards to the* $f(n) = C^*$ *case.*]

Hence, among optimal algorithms based on a given consistent heuristic $h$, $A^*$ expands as few nodes as possible.