A\* and Alternative Methods of Efficiency

Adam Donaldson

Harrisburg University of Science and Technology

A\* and Alternative Methods of Efficiency

A\* search allows for a significant increase in efficiency over non heuristic alternatives by allowing for basic decision making weights. Despite A\* ability to greatly increase the speed at which a goal state is found it is not perfect and largely dependent not only on the way the heuristic is assigned but also the structure of the data. If all data were structured the same there would be no reason for searching through it as the goal would be obvious every time so the question must be asked, is there a way to alter the A\* algorithm such that it can find the goal state with greater efficacy? Two possible solutions, landmark A\* search and bidirectional search, exists to this issue but they both present their own unique drawbacks and advantages.

A\* an intelligent search algorithm that can find its way through a set of data using the relative distances between the nodes of the data to find the shortest path to the end state. This algorithm is very similar to Djikstra’s with one key difference: it uses heuristics to guarantee that the first time it reaches the goal state it is the shortest possible path to the goal state.This major improvement in efficiency comes from the fact that heuristics give algorithms a way to ‘see’ how close they are to the end state that they are attempting to reach. The data set I have used is a simple maze with a start point and an endpoint with a number of dead ends within the maze to throw the algorithm off. Djikstra’s would spread out in a simple search pattern going to the shortest path then the next shortest path and then the next and the next and the next. When visualising this approach it appears to move steadily through the maze trying every available path. When a heuristic value is introduced, in this case I used euclidean distance, to weight the nodes of the maze in relation to the end of the maze this significantly speeds up the process as the algorithm can know tell when it is getting closer and will ignore paths that do not bring it closer the the end state (A\* Search, 2018).

This may seem trivial when thinking about it from a human perspective but from a computing perspective this is a big deal. A\* however is not perfect and suffers from some key issues notably when a path is perfectly straight with no deviations or alternate routes then Djikstra’s will be better as it does not need to assign heuristic values. This extra amount of computation can be minimized by simply ensuring that the heuristic assignment does not take more than constant time so that each time the algorithm is run it will take n number of steps to assign heuristics values with n being the number of nodes. A\*’s biggest flaw is that it has no way to avoid or identify false paths to the end goal and will go down them if they are perceived to be closer to the desired end goal. There are other variations on the idea of A\* that run faster than the normal algorithm and can eliminate some of the traversal down false paths (A\* Search, 2018).

A\* landmark search takes the idea of A\* search and expands on it by adding landmarks into the mix to speed up the efficiency and reduce the likelihood of A\* exploring false paths. A landmark simply put is a marker placed within the dataset that the algorithm is looking for each landmark requires that the nodes of the dataset have a heuristic value computed for each landmark as well as the end goal. Once these landmarks are established the algorithm will then work from landmark to landmark on its way to the end goal that the algorithm is trying to find. These landmarks are expressly useful in that they help local navigation within the dataset and when properly placed provide and easy way to avoid the pitfalls that normal A\* falls into. A\* landmark does however suffer from its own slew of issues that normal A\* search does not suffer from namebly that of placement (Richter 2012).

The entire purpose of a search algorithm is to have a way to find a desired end goal that would be difficult or otherwise time consuming for humans to do. Land marks work best with knowledge of where they should be placed. These opposing factors mean that placing landmarks can be exceedingly difficult or impossible in a great many cases. Take for instance the idea of randomly placing a number of landmarks this could in theory increase the runtime of the algorithm or it could have disastrous consequences and greatly increase the run time by unintentionally leading it down false paths. Some human input when it comes to the placement of landmarks can be extremely useful but this completely eliminates the entire purpose of the algorithm being able to replace a human in the searching process. A well structured dataset that needs to be passed through consistently such as a rail network or road system could possibly keep records of most passed through places and use these as landmarks but this could unintentionally detour some routes. So this begs the question how to place the landmarks? The answer unfortunately is not easy and largely depends upon the nature of the data set in question (Richter 2012).

The next possible option that could be explored as an option is bidirectional search. On the surface bidirectional search does not seem like a vast improvement over A\* search as it runs two simultaneous A\* searches starting from the end point and going to the startpoint and from the start point to the end point. It does however offer major improvements in avoiding false paths. This avoidance is not direct like landmark A\* instead it comes from the fact that only roughly half the area needs to be searched by each algorithm giving it less opportunities to pursue false paths. This improves the run time from O(b^d) for A\* search to O(b^(d/2) + b^(d/2)) for bidirectional search which can further be simplified to O(b^(d/2)). This does not mean that bidirectional search is perfect however as it has a few caveats that must be considered similar to A\* landmark search (Bidirectional, 2020).

The biggest downside to bidirectional search is that it doubles the number of heuristic values that need to be computed from the normal A\* search as every node now needs to consider a heuristic value for both algorithms. There is also the issue of needing a little bit of extra code to recognise when the two algorithms have reached the midpoint at which they can then stop. These issues are minor when compared to some of the potential improvements offered by landmark but so is the increased performance from bidirectional search. When testing this algorithm on my dataset I ran into the distinct issue that it actually made the runtime worse on my relatively small data set by exploring every single false data path. I managed to fix this error after I realised an issue in my data set but the fact that this is possible at all is concerning (Bidirectional, 2020).

In the future I would like to further explore a number of topics that were either outside the scope of my research or that I did not have enough time to implement. The first which I decided to cut for time was combining landmark A\* search and bidirectional search into one singular search which I have dubbed bidirectional landmark. While I did not get to implement this fully I think that it would marginally improve over what both algorithms can do on their own open as they both seek to eliminate similar issues. As for outside the scope of my research I would like to further delve into landmark assignment and see if I could devise a way to place landmarks that was not cost prohibitive in terms of run time. I think such a method could be extremely useful in speeding up pathfinding in complex data sets given that the method of finding landmarks could be done in constant time or at least a small enough amount of processing that it did not increase runtime over not having landmark search.

Landmark A\* search and Bidirectional search both promise improvements over the traditional A\* search however they both come with significant drawbacks of their own. Landmark requires foresight of placement and also knowledge of the number of landmarks desired which can be difficult to near impossible to ascertain in a vacuum. Bidirectional’s improvement can really only begin to be felt on large datasets with its possible improvements being hard to see on a small set of data. Both work well on complex datasets and could even be used in conjunction to get even better performance out of the algorithm. Both bidirectional search and A\* landmark searches are interesting in useful tools in their own right but need to be correctly applied to the task at hand.

References

A\* Search Algorithm. (2018). Retrieved December 02, 2020, from https://www.geeksforgeeks.org/a-search-algorithm/

Bidirectional Search. (2020). Retrieved December 02, 2020, from https://www.geeksforgeeks.org/bidirectional-search/

Richter, S. (2012). Landmark-Based Heuristics and Search Control for Automated Planning