Shortest Path Comparison between the A\* and Dijkstra’s Algorithm

By

Steven Gantz

Intro to Dijkstra and A\*

I. Introduction to History of A\* and Dijkstra’s Algorithms

In June of 1959, Edgar Dijkstra approached the problem of finding an algorithm that could find the shortest path through a graph in polynomial time. He approached this problem to demonstrate the capabilities of a new computer called ARMAC. Dijkstra finally formulated the algorithm one day while attempted to generate a transportation map of 64 cities in the Netherlands. [2]

He noted his findings in a paper titled A Note on Two Problems in Connexion with Graphs. The paper suggested that the aforementioned algorithm was valid only when at least one path exists between any two nodes in a graph. The paper describes exactly 2 problems the algorithm is able to solve.

The first is the following:

*Construct the tree of minimum total length between the n nodes. (A tree is a graph with one and only one path between every two nodes) [1]*

The second is the following:

*Find the path of minimum total length between two given nodes P and Q[1]*

This algorithm was more effective than the previous algorithm which Dijkstra noted was, *“The solution given above is to be preffered to the solution by L. R. Ford as described by C. Berge for, irrespective of the number of branches, we need not store the data for all branches simultaneously but only those for the branches in sets I and II, and this number is always less than n. Furthermore, the amount of work to be done seems to be considerably less.”*[1]

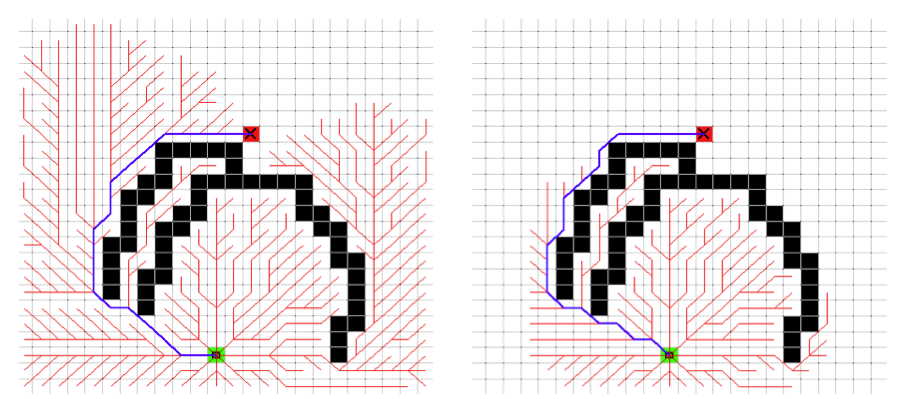
This was the algorithm used primarily by Computer Scientists for relevant applications until a requirement arose for a faster algorithm with a smaller space and time complexity. This manifested itself in a path planning algorithm.

In 1968, Nils Nilsson was attempting to improve the path planning done by Shakey the Robot. An early prototype robot that could navigate through an obstacle filled room. The algorithm he implemented, was titled A1. Soon after, Bertram Raphael suggested significant improvement upon the algorithm, with the pair retitling it to A2. Finally, Peter Hart made some final changes to A2 before testing and concluding that it was the best possible algorithm for finding shortest paths.

Later that year, in June of 1968, the trio published a paper titled A Formal Basis for the Heuristic Determination of Minimum Cost Paths. The paper attempts to bring together two different ways of approaching the shortest path problem. They note the methods as; Mathematical approach, and Heuristic approach [3]. They proceed to define the algorithm with examples and proofs. The algorithm has since been classically called the A\* algorithm. It enjoys widespread use throughout computer systems and continues to be used in many applications around the world.

II. Overview of Computations

The A\* algorithm is remarkably similar to Dijkstra’s original algorithm. The main difference in the process is the inclusion of a heuristic to each path calculation. Where Dijkstra’s Algorithm keeps track only of the current path traveled thus far, comparing each node to find the shortest path forward, A\* deviates by also adding in a heuristic analysis to this comparison. Therefore there is a more specific weight given to each path that is more likely to represent to paths distance so far as the heuristic should favor nodes that are literally closer to the final goal node. It can be said that A\* places preference on nodes literally closer to the goal node. Effectively reducing Dijkstra’s Algorithms extra expansions on nodes that may further away from the goal node than other more favorable nodes. This is shown visually in Figure 1 below:

Fig. 1

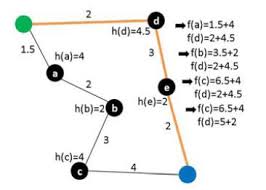
[6]

Dijkstra’s Algorithm is located on left, which travels in equal distance in every direction until it eventually finds the source node. Comparatively the A\* algorithm attempts to travel only closer to the goal node using the heuristic to estimate the remaining distance.

This heuristic estimates the distance to the goal node from each checked node, giving a different final weight than would be found by Dijkstra’s Algorithm.

Fig. 2

[5]



As seen in figure 2 above, where Dijkstra’s Algorithm would only count the length of the current path to build its tree, the heuristic adds an extra level of detail that sets the algorithm on a more direct course. Under certain well-defined conditions, the A\* algorithm will always be the optimal method for traversing through the shortest path in the graph.

The heuristic being the primary defining factor between Dijkstra’s Algorithm and it’s A\* counterpart, it becomes obvious that though Dijkstra’s was developed nearly ten years before the A\* algorithm was first implemented, it is now commonly accepted that most general variants of Dijkstra’s Algorithm are in fact a special case of A\* where the heuristic has a zero value for all nodes.

How algorithms work

Differences between Dijkstra and A\*

A\* admissibility

Project created to display A\* vs Dijkstra

Algorithm used for project

A\* used in gaming

A\* used in AI

**References**

[1] Dijktra, E. W.: A Note on Two Problems in Connexion with Graphs. Pg 1 (1959)

[2] Frana, Phil: “An Interview with Edsger W. Dijkstra” (<http://dl.acm.org/citation.cfm?doid=1787234.1787249>)

[3] Hart Peter E., Nilsson Nils, J, Raphael Bertram: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. Pg 1 (1968)

[4]

[5] [http://digilib.its.ac.id/public/ITS-Undergraduate-19142-Paper-519446.pdf Pg 2](http://digilib.its.ac.id/public/ITS-Undergraduate-19142-Paper-519446.pdf%20Pg%202)

[6] <http://gamedev.stackexchange.com/questions/61850/in-a-star-how-does-the-heuristic-help-determine-your-path> - User: <http://gamedev.stackexchange.com/users/998/bobobobo>

Bibliography:

<http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf> – original paper by Dijkstra

<http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/astar.pdf> - original A\* paper

<http://matwbn.icm.edu.pl/ksiazki/cc/cc35/cc3536.pdf>

<http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>

https://en.wikipedia.org/wiki/Artificial\_Intelligence:\_A\_Modern\_Approach

Find alternative sources:

<http://stackoverflow.com/questions/1332466/how-does-dijkstras-algorithm-and-a-star-compare>

<http://gamedev.stackexchange.com/questions/15/how-does-a-pathfinding-work>

<https://en.wikipedia.org/wiki/Admissible_heuristic> (formulation and search algorithms)

<https://www.cs.rit.edu/~ark/winter2012/730/team/1/presentation1.pdf> (basically my project...)