Traversing Shortest Paths in Minneapolis with A*

Matthew Vincent

April 2019

1 Abstract

The heuristics of uniform cost, euclidean distance, 2x weighted euclidean distance, Manhattan distance, 2x weighted Manhattan distance, diagonal, and 2x weighted diagonal distance were used in A* to find the shortest paths in a Minneapolis road network. Minneapolis's road network was setup as a graph state space with each street corner represented as a node, each street as an edge, and the distance for each street represented by the cost. The factors of speed limit and time was not taken into consideration for computing the shortest path in the state space representation of the road network for Minneapolis. The characteristics of complexity, completeness, admissiblity, and optimality were observed and found to be that each algorithm was not complete when used with the simple rendition of A* created by scratch trying to handle one way streets in the directed graph state representation of Minneapolis's road network. Euclidean distance was found to be the only admissible and optimal heuristic while Manhattan distance was confirmed to provide the best results of lowering complexity at the price of increasing path length. The results concerning diagonal heuristics showed that it is likely not a good option when applied to road networks, however, more future work should be done to better test this. Weighted heuristics were also observed to reduce the complexity of each heuristic, however, at the cost of increasing path length.

2 Introduction

This paper approaches the problem of finding shortest paths in the road network of Minneapolis by using the search algorithm A* driven by differing heuristics to guide the search. This is important to individuals using road networks in order to minimize their time spent either walking or driving in the city as taking a shortest path reduces the resources spent such as energy or time to get from one place to another. If the road network is already provided then the next step to finding the shortest path using A* is to select a suitable version of it and heuristic that accommodates to a few characteristics of the road map. The road map may differ in characteristics of size, being directed or undirected depending on one-way roads and two-way roads, what method of transportation the map is going to be used for such as walking, driving, or biking, and many other characteristics. The case for Minneapolis road network provided will not taken into consideration the differing speed limits of roads, however, it will take into consideration one-ways and two way roads.

In order to optimize A* for this problem and possibly for other road networks, multiple heuristics must be compared to figure out which heuristic provides the most favorable result depending on the algorithms factors of complexity, completeness, admissiblity, and optimality. Reducing complexity may be a concern for road networks that contain a lot of roads or corners as an algorithm that opens

too many nodes may be too costly as far as computing power or memory is concerned to find the desired shortest path. Reducing admissiblity and optimality by a small amount may also be a concern for road networks that contain a lot of roads or corners as using a heuristic that is a bit more greedy than a euclidean distance heuristic may allow for a near optimal path, however, speeding up and reducing complexity for the search algorithm by reducing the number of nodes requried to be opened if the goal is reached faster. Once each heuristic is tested against each other on multiple routes the differences each factor may be observed to apply the appropriate heuristic along A* to more efficiently find the shortest paths in the road network. The case for Minneapolis road network will require a heuristic that does favor directed graphs, may use a near optimal solution to reduce complexity with less nodes opened to reduce computing power if the algorithm was to be used in some application. The results observed from this problem may be then applied to other possibly similar road networks to have a basis to start of off when attempting to apply A* to different road networks while having an idea of which heuristic may better suit the factors desired in finding a shortest path.

3 Background

An important area of research for computer science and helping increase the efficiency of everyday life is finding the shortest path to take to get from a starting position to an ending position. Finding the shortest path to a goal reduces the cost used during the trip to the goal whether those resources be time, gas for cars, or energy. The shortest path problem has been tackled by many different algorithms and approaches ranging from graph searching with depth first search, using genetic algorithms [2], employing hub labeling [1] and many others solving different variants of the problem. One of the earliest approaches, Dijkstra's algorithm [4], solves the problem in almost linear time [1] traversing through directed graphs with unbounded non-negative weights. Dijkstra's algorithm formed a basis for more best-first search methods [3] to build off of such as A* and it's many variants in which each algorithm tends to try to optimize some aspect of the solution whether it be completeness, admissibility, optimality, or reducing complexity while traversing.

A* takes on a similar approach as Dijkstra's best-first search method by expanding nodes toward the target node, however, A* implements an informed best search using the heuristics to guide the direction of node expansion while pruning other paths by using an evaluation function to achieve better performance than Dijkstra's algorithm. A* is a powerful family of algorithms that when supplied with different heuristics may be optimized for solving many problems presented in the AI community. A* if supplied with an admissible heuristic will find the optimal solution to the shortest path problem, however, the admissibility [3] of A* can generally lead to harsh exponential complexity in some variations of the problem.

Multiple variants of A* have been proposed in previous papers in order to reduce the complexity problem by trading off with some other aspects of A*. Hart, Nilsson, and Raphael initially proposed the method of weighted A* [6] using a heuristic somewhat larger than defined to reduce computational effort while finding a satisfactory, near optimal solution by sacrificing admissibility of A*. Ghallab and Allard proposed another variant of weighted A*, Aε [5], seeking an near optimal solution within a factor $(1 + \varepsilon)$ of the optimal solution also lowering the average complexity of A*. Real time heuristic search [10] has also been implemented similarly to weighted A* by Rivera, Baier, and Hernández yields some promising results compared to other weighted search algorithms in improving performance. The reduction of complexity of A* is an important factor for consideration when attempting to find shortest paths on road networks as this problem is undertaken by many GPS navigation systems in applications used today. A near optimal solution will spare computing resources for phones using GPS apps while

the tradeoff is potentially only missing out on the opportunity of arriving at destinations potentially a little earlier by seconds or minutes depending on the scope of the problem. Sacharidis and Bouros further developed the algorithm SimplestNearFastest-A* (SNF-A*) [11] to help decide routes through road networks when providing routing directions. SNF-A* uses bounds on the complexity to provide simpler routes for GPS users to understand at the expense of not always being the fastest route possible. The potential for this is to provide simpler routes in road networks for individuals such as [11] tourists or people in emergency that don't have time to remember all as many turns required in unfamiliar or stressful environments.

Multiple other methods derived off of Dijkstra have also been proposed recently such as hub labeling, spatial queries, and keyword covering queries which may be used for solving shortest path problems based on road networks. Hub-based labeling has been researched by Microsoft's research teams [1] to solve the problem resulting in the best time bounds compared to other methods when implemented. Spatial queries methods have been used and multiple characteristics of implementing them have been tested more recently in 2018 such as testing euclidean distance with road networks [7] by Hua, Xie, and Tanin. Other variants of the shortest path problem with road networks such as shortest keyword covering routes have been important and studied recently. The problem of needing to go through multiple points of interest such as running errands like filling up on gas and stopping by an atm before arriving home from work while finding the shortest path to take satisfying each point of interest. This problem has been recently studied more using keyword covering queries [8] by Kaffes, Belesiotis, Skoutas, and Skiadopoulos.

There are also variants to the shortest path problem with road networks where more adaptive algorithms take suite better as a solution. For example, the problem of navigating road networks with AI such as self driving cars is becoming more prevalent today and is a possibility for navigating the road networks of Minneapolis as well. Multiple algorithms built off of A* have been implemented to be used to solve these problems in real time where interaction with a changing environment is sometimes evident. One variation, Generalized Adaptive A* [12], has been implemented by Sun, Koenig, and Yeoh for solving search problems faster as it updates the heuristic values over time in environments where action costs can increase or decrease. This is relevant to the problem of self navigation and route planning if traffic is applied to the path cost or if it is applied to dealing with surrounding cars. Another variation which performs similarly in a designated time cycle that is useful for self driving cars in road networks is ARA* Anytime A* [9] which has worked on by Likhachev, Gorden, and Thrun. ARA* Anytime A* operates by using a large heuristic to find a quick solution in a route then refines the solution by lowering the heuristic and repeating the search as much as possible within a given time limit. ARA* AnyTime A* proves useful for obtaining quick responses and putting a time bound on A* allowing for the complexity of A* by itself to not be a bottleneck on finding quick responses when a reaction is required such as handling interactions with other traffic. These methods involved with self driving cars on road networks is not as relevant to the heuristics tested in this paper, however, they are important to bring up if the work in this paper is to be expanded on and brought more into the scope the algorithms are relevant for.

This paper tackles the single source shortest path (SSSP) problem applied to a road network of Minneapolis converted into a directed graph. The prospect of using different variants of A* by varying the heuristic is tested in this paper to gain a better understanding of performance differences between multiple heuristics for future reference of which heuristic to use while traversing Minneapolis's road network. The factors of complexity, completeness, admissibility, and optimality are to be tested between the heuristics for A*. The multiple variations of weighted A* and finding simplest paths are the main methods of altering the heuristics to observe less complexity which is most relevant to this

paper. The heuristics of using Manhattan distance, Diagonal distance, and Euclidean distance will be tested to observe the difference in performance for A* when applied to road maps as well. The results of this are important as understanding the differences in complexity may allow for choosing a heuristic which may use the least computing power while still finding a near optimal solution which is important for saving resources, however, the trade off is generally sacrificing admissibility. The observed differences in the heuristics may be applied to solving future road network shortest path problems and expanded on for more research on road networks of other cities or applied to other areas in search problems using heuristic searches.

4 Approach

The problem will be approached by using the coding language processing to develop a state space in which the road network of Minneapolis will be constructed using street corners as nodes and directed edges as streets. The cost of each edge will represent the length of each street from one street corner, node, to another street corner. One way and two way streets will be established by whether an edge is bidirectional or not in which for one ways the vector from the starting node toward the ending node will represent the direction of the one way street. The street infrastructure from Fig.1 with associated data to represent each street corner as a node with a coordinate system will be used as the reference when constructing the state space. The state space's start and goal will be established at different street corners, nodes, allowing for an individual to pick a corner on the Minneapolis road network to start at and then designate another street corner as the goal.

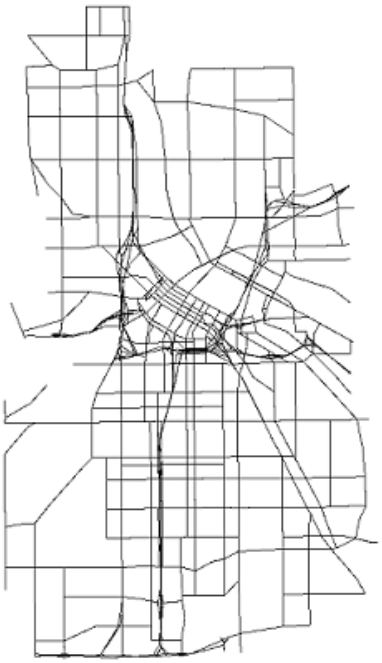

Figure 1: Map of Minneapolis street infrastructure

A simple implementation of A* will be developed to be used on the graph state space to find the shortest path from the starting node to the goal node. The code will be developed from scratch with pseudo code from online to guide the creation of the A* algorithm. The programming language processing will allow the development of a graphical display of the road network to be made illustrating the paths found by A* when given different starting nodes, goal nodes, and heuristics to be used on the Minneapolis road network. Multiple heuristics for A* will be tested among different routes through the road network to compare heuristics to determine the best heuristic to be used. Processing will allow for easy visualization of how each heuristic influences the shortest path found by A* highlighting the path found in red lines, the starting node with a green circle, the goal node with a blue circle, while also displaying the distance for each path, which heuristic is used, the number of nodes in the closed set for A*, and the number of nodes in the open set for A*. The results for each path found may then be exported in text files to obtain numerical results which are convenient to compare.

Figure 2: Showcase of "roadnetwork search" processing program using A* to find red line shortest path from starting green node to goal blue node. (https://github.com/vince145/minneapolis-a-star)

5 Experimental Evaluation

The processing program implemented was setup to consider twenty different routes all starting from the same node and going to a different goal node such as an example of an individual leaving home to go to different areas in the city. For each of the twenty routes evaluated each heuristic was tested with the simple implementation of A* to find the shortest path from the start node to the goal node. The distance traveled (the f(goal) function for A*), the number of street corners passed (number of nodes in the obtained shortest path), the number of nodes in the closed set for A* (number of nodes visited), and the number of nodes in the open set for A* (number of neighbors evaluated not including nodes visited) were measured for each heuristic A* as found the shortest path for each route. The differences in these characteristics of each heuristic for each shortest path were then evaluated to obtain a better understanding of how each heuristic performs in the road network of Minneapolis. The trends of differences between each heuristic were then evaluated to signify which heuristic was

better for optimizing the factors of complexity, completeness, admissiblity, and optimality.

The heuristics of Euclidean, 2x Weighted Euclidean, Manhattan, 2x Weighted Manhattan, Diagonal, 2x Weighted Diagonal, and a heuristic of zero equivalent to running uniform cost was tested using the twenty predetermined goals for A* on the road network of Minneapolis finding the shortest path showcasing the characteristics for each heuristic.

Heuristic	AVG f(goal)	AVG Path Size	AVG ClosedSet Size	AVG OpenSet Size
Uniform Cost	2255.46	26.5	537.45	27.15
Euclidean	2255.46	26.5	148.45	30.3
2x Weighted Euclidean	2381.65	26	71.25	25.25
Manhattan	2315.04	25.65	90.5	27
2x Weighted Manhattan	2424.90	25.1	69	24.05
Diagonal	2262.26	26.25	433.9	32.85
2x Weighted Diagonal	2407.55	27.55	392.15	32.1

Figure 3: Average characteristic results of heuristics used with A* to find shortest paths to twenty predetermined goal nodes from the same starting node on the Minneapolis road network. (https://github.com/vince145/minneapolis-a-star/blob/master/MinneSearch/results.xlsx)

Heuristic	STDev f(goal)	STDev Path Size	STDev ClosedSet Size	STDev OpenSet Size
Uniform Cost	968.89	8.68	252.17	12.53
Euclidean	1015.73	7.81	137.56	8.60
2x Weighted Euclidean	1014.42	8.18	132.92	9.10
Manhattan	1067.44	7.79	132.03	10.05
2x Weighted Manhattan	1022.82	8.46	290.50	11.27
Diagonal	1021.42	9.52	267.93	10.62
2x Weighted Diagonal	1035.75	9.31	265.88	9.80

Figure 4: Standard deviation of characteristic results of heuristics used with A* to find shortest paths to twenty predetermined goal nodes from the same starting node on the Minneapolis road network.

(Norm h) - (2x Weighted h)	AVG f(goal)	AVG Path Size	AVG ClosedSet Size	AVG OpenSet Size
Euclidean Differences	-126.19	0.50	77.20	5.05
Manhattan Differences	-109.86	0.55	21.50	2.95
Diagonal Differences	-145.29	-1.30	41.75	0.75

Figure 5: Difference between average characteristic results of normal heuristics and 2x weighted version of normal heuristics used with A* to find shortest paths to twenty predetermined goal nodes from the same starting node on the Minneapolis road network.

(Uni. Cost. H) - Heuristic	AVG f(goal)	AVG Path Size	AVG ClosedSet Size	AVG OpenSet Size
Euclidean	0.00	0.00	389.00	-3.15
2x Weighted Euclidean	-126.19	0.50	466.20	1.90
Manhattan	-59.58	0.85	446.95	0.15
2x Weighted Manhattan	-169.44	1.40	468.45	3.10
Diagonal	-6.80	0.25	103.55	-5.70
2x Weighted Diagonal	-152.09	-1.05	145.30	-4.95

Figure 6: Difference between average characteristic results of uniform cost search (h=0) and other tested heuristics used with A* to find shortest paths to twenty predetermined goal nodes from the same starting node on the Minneapolis road network.

(Heuristic)/(Uni. Cost. H)*100-100	AVG f(goal)	AVG Path Size	AVG ClosedSet Size	AVG OpenSet Size
Euclidean	0.000	0.000	-72.379	11.602
2x Weighted Euclidean	5.595	-1.887	-86.743	-6.998
Manhattan	2.642	-3.208	-83.161	-0.552
2x Weighted Manhattan	7.513	-5.283	-87.162	-11.418
Diagonal	0.302	-0.943	-19.267	20.994
2x Weighted Diagonal	6.743	3.962	-27.035	18.232

Figure 7: Percent difference between average characteristic results of uniform cost search (h=0) and other tested heuristics used with A* to find shortest paths to twenty predetermined goal nodes from the same starting node on the Minneapolis road network.

6 Analysis

The results for the seven different heuristics after being tested with A* for the twenty different goal nodes were averaged according to heuristic used in figure 3 and the standard deviation for each one was calculated in figure 4. The averages allow for analysis of how each heuristic compares among each other in the four different measurements taken to allow for observations on the complexity, optimality, amd admissiblity of each heuristic when used with our simple version of A* applied to the Minneapolis road network. The difference of the weighted versions and unweighted versions of each heuristic tested was also calculated in figure 5 to observe how multiplying the weight of each heuristic affects their characteristics. The difference of uniform cost, where h = 0, and each other heuristic tested was also calculated in figure 6 to observe the difference between algorithms such as uniform cost and dijkstra's while also allowing for easier observations of the differences between algorithms and how much they affect A*. The percent difference of uniform cost and each other heuristic tested was also calculated in figure 7.

The usuage of uniform cost search as a heuristic allows for us to observe if each other heuristic is optimal as uniform cost search is guaranteed to final an optimal solution while some heuristics for A* may overshoot the optimal path when searching. Uniform cost search results in an average f(goal) of 2255.46 for the tests performed while the euclidean distance algorithm also resulted in the same average leading to the euclidean distance algorithm being optimal in this scenario. The other five heuristics resulted in paths that were larger than uniform cost search's path with the unweighted heuristics being 0.302 and 2.642 percent larger while the weighted heuristics led to much larger percent differences of 5.595 for weighted euclidean, 7.513 for weighted manhattan, and 6.743 percent

for weighted diagonal leading to them not being optimal, however, finding near optimal solutions. The unweighted euclidean distance heuristic proves to be admissible while the others overestimate the goal cost when being applied to the road network of Minneapolis.

The usuage of uniform cost search also allows for the observation of complexity with how many nodes need to visited to find a shortest path with the characteristics of how many nodes are in the closed set and how many nodes are in the open set. The closed set allows for the analysis of how many nodes have been visited and opened with their neighbors added to the open set while the open set is the remaining neighbors to be evaluated when the goal was found. Having nodes remaining in the open set affects the amount of comparisons that need to be done when evaluating which node should be closed next adding more computations to be performed. The six heuristics when compared to uniform cost search reduced the number of nodes closed by as little of 19.267 percent less to 87.162 percent less leading to all heuristics resulting in better complexity than uniform cost search in terms of nodes explored. This observation does not include the difference in computation complexity for each calculation done when computing the individual heuristics which may cause a large difference for squaring functions and absolute functions. Diagonal heuristics are made to be more efficient than euclidean in the regards that the squaring function does not need to be computed but rather only a maximum function, so the difference here is not regarded but should be noted. The diagonal heuristics ended up reducing the amount of nodes closed the lowest with 19.267 percent and 27.035 while each other heuristic reduced the amount of nodes closed by percents over 72 percent. The Manhattan distance while generally providing the largest h values reduced the amount of nodes closed by the largest with 83.161 and 87.162 percent. The weighted version of each heuristic resulted in better complexity on average by 8.711 percent, however, the difference path cost was increased on average by 8.333 percent. When all the heuristics are compared it appears that Manhattan distance as confirmed by many other papers regarding road networks proves to be the best of both worlds when providing a reduction in complexity with less nodes closed by 83.161 percent for the price of the path being 2.642 percent larger.

The completeness of each heuristic was not guaranteed for the simple rendition of A* when applied to the road network of Minneapolis which may be attributed to the algorithm not handling correctly the consideration of directed and undirected graphs to backtrack to find paths around one way streets. Each of the twenty goals were picked from the uniform cost search heuristic when observing every possible path using a for loop to run through the possibilities. There were many paths which returned null when attempting to find a path leading to each heuristic being incomplete for not being able to find a path in those scenarios. The seven heuristics were complete in the regards of the sample taken of the twenty goal nodes as each heuristic did find a path to the goal nodes.

7 Conclusion

The problem to determine the characteristics of different heuristics to use on the road network of Minneapolis when using the A* search algorithm to find shortest paths along one way and two way streets allowed for a deeper understanding of some common heuristics used in many search algorithms. The observation of the differences weighted algorithms can provide to help reduce complexity at the cost of obtaining a less near optimal solution was found to be that the weighted version of each heuristic resulted in better complexity on average by 8.711 percent, however, the difference path cost was increased on average by 8.333 percent. Manhattan distance provided the best results for reducing complexity at a lower price of average path cost increase. Euclidean distance appeared to be the only heuristic to be optimal while it still provided a reduction in nodes closed complexity of

72.379 percent compared to uniform cost search. There still needs to be future work detailing the completeness of each algorithm as the implementation of A* created does not handle backtracking well enough to work around the one way streets of Minneapolis. Future work should also be done concerning the optimality of time to perform each search for each heuristic and also observing deeper into the computation complexity concerning computing each heuristic and adding those h values to the search with how that differs for each heuristic. The diagonal distance heuristic would require this future work to be done to get a more accurate analysis of the difference it may make in complexity compared to other heuristics as it is based on removing the need to calculate square roots and powers in euclidean distance. The diagonal distance heuristic may not generally be good enough though as observed from these results to be used in place of the other heuristics when applied to road maps.

References

[1] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *Proceedings of the 10th International Conference on Experimental Algorithms*, SEA'11, pages 230–241, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur, and Frank Neumann. Computing single source shortest paths using single-objective fitness. In *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA '09, pages 59–66, New York, NY, USA, 2009. ACM.

[3] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32(3):505–536, July 1985.

[4] E Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[5] Malik Ghallab and Dennis G. Allard. A: An efficient near admissible heuristic search algorithm. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, pages 789–791, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[6] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[7] Hua Hua, Hairuo Xie, and Egemen Tanin. Is euclidean distance really that bad with road networks? In *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS'18, pages 11–20, New York, NY, USA, 2018. ACM.

[8] Vassilis Kaffes, Alexandros Belesiotis, Dimitrios Skoutas, and Spiros Skiadopoulos. Finding shortest keyword covering routes in road networks. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, SSDBM '18, pages 12:1–12:12, New York, NY, USA, 2018. ACM.

[9] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara* : Anytime a* with provable bounds on sub-optimality. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 767–774. MIT Press, 2004.

[10] Nicolas Rivera, Jorge A. Baier, and Carlos Hernandez. Weighted real-time heuristic search. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 579–586, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[11] Dimitris Sacharidis and Panagiotis Bouros. Routing directions: Keeping it fast and simple. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 164–173, New York, NY, USA, 2013. ACM.

[12] Xiaoxun Sun, Sven Koenig, and William Yeoh. Generalized adaptive a*. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume*

1, AAMAS '08, pages 469–476, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.