MS4: Algorithm Report

NPS2001B: Eryn Lim En Yi (A0265301R), Hamamoto Sayuri (A0265893L), Ho Kai Liang Austin (A0251821M), Pandrangi Tamrita (A0257397L), Tang Ser Jen, Tiffany (A0255729R)

1. Purpose and choice of algorithm

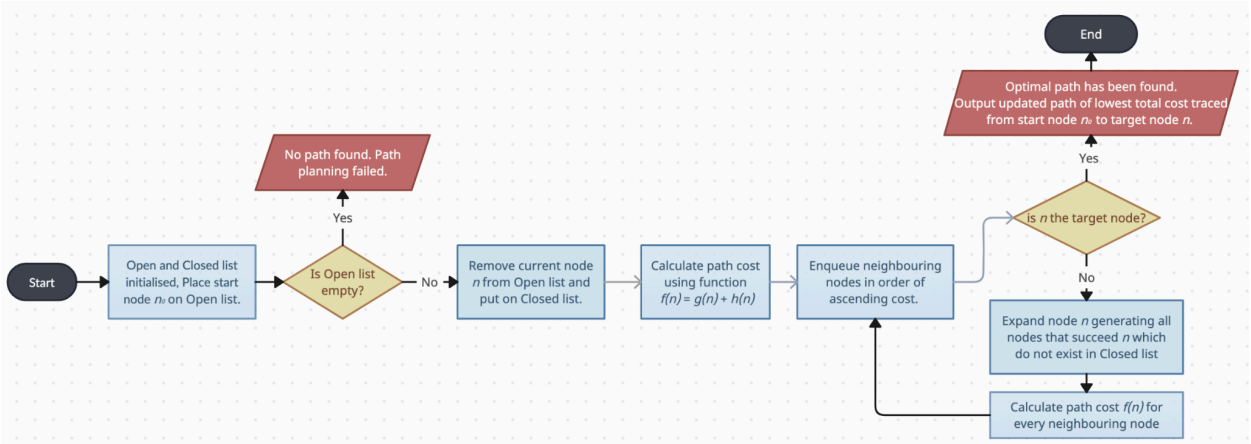
The purpose of our app's algorithm is to search for the optimal accessible route from one point on the map of the Faculty of Art and Social Sciences (FASS) to another (i.e. the shortest path from one node to another on a weighted graph), particularly routes accessed by people with mobility constraints. Our chosen algorithm, the A* search algorithm, is not the only algorithm capable of performing this task; other search algorithms include Dijkstra's algorithm and the Bellman-Ford algorithm.

The A* algorithm was chosen as it improves on Dijkstra's algorithm by using a heuristic to inform the search—while Dijkstra's algorithm considers all possible paths, the A* algorithm prioritises processing nodes nearer to the goal than others (Hart et al., 1968), which gives a sense of "goal-directedness". The Bellman-Ford algorithm, which can handle negative edge weights (Cormen et al., 2002, p. 612), is unsuitable as our graph does not have negative weights (since time and distance are non-negative); using it would only unnecessarily increase the time complexity. As such, the A* search algorithm was selected as a good middle ground in terms of complexity and suitability for our intended task.

2. How the algorithm works

The algorithm works much like Dijkstra's algorithm except in processing a node: the total cost of a route passing through a given node n, $f(n)$, is the sum of the cost to reach n, $g(n)$, and the cost estimated by a heuristic function to get from n to the goal, $h(n)$ (Hart et al., 1968, p. 102). For a heuristic to be admissible, it must never estimate a cost higher than the true cost from a node to a goal, for all nodes (Hart et al., 1968). Thus, we are using the Euclidean or straight-line distance between two points as a heuristic, since it is the shortest possible distance between two nodes and will never overestimate the real cost of traversal; it is given by $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, where x_1, x_2 and y_1, y_2 are the longitudinal and latitudinal geographical coordinates of any two arbitrary points 1 and 2, respectively.

As such, the inputs required are **nodes**—all possible start and end points, and points of traversal, e.g. access points like lifts and the classrooms and lecture theatres of FASS; and **weights**—the mobility constraints and Manhattan distance between two nodes given by $|x_1 - x_2| + |y_1 - y_2|$, using their geographical coordinates. The **output** is the lowest-cost path from the Start node to the Target node. The sequence of steps is illustrated in the flowchart below, adapted from Zidane & Ibrahim (2018):

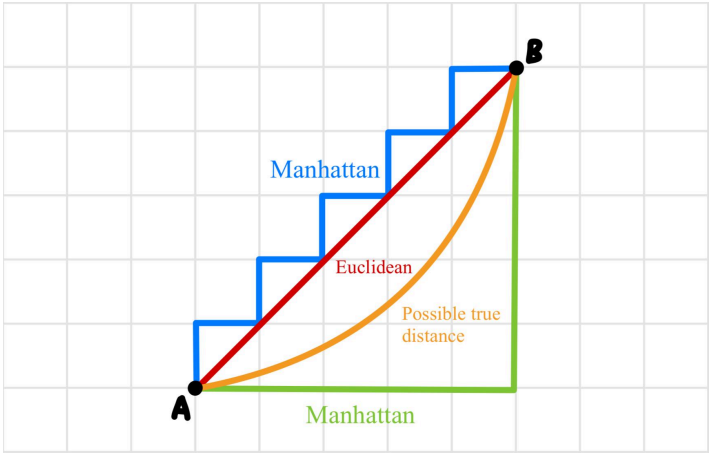


1. Firstly, we initialise an **Open** and **Closed List**. The Open List contains all the nodes that have yet to be examined, acting as a queue which prioritises the lowest cost node for processing (Ori, 2021). The Closed List collects all processed nodes to avoid revisiting them (Ori, 2021).
2. Beginning with the Start node, we process nodes in the Open List by calculating their neighbours' costs using the function $f(n) = g(n) + h(n)$, and enqueueing them in order of ascending cost.
3. The current node n is then removed from the Open List and placed in the Closed List. As long as the current node n is not the Target node, we continue processing nodes on the Open List, if any.
4. This process is terminated once the current node n is the Target node, and the path is returned by tracing the parent nodes from the Target node back to the Start node. Alternatively, if the Open List is empty, no path is returned since no possible path is found.

3. Limitations of the algorithm

Space complexity. One of the primary limitations of using the A* algorithm is its space complexity, fixed at $O(b^d)$ where "b is the branching factor—the average number of edges from each node, and d is the number of nodes on the resulting path" (Cox, 2020). This complexity arises from the need to constantly maintain and thus store lists of all nodes in memory (Cox, 2020), which is exacerbated since d is in the exponent. One way to manage this limitation is to minimise the number of nodes in our graph by excluding those which are irrelevant (Cox, 2020). Since locations in FASS are relatively rather dense, we can achieve this by clustering groups of locations which are nearby (e.g. classrooms side-by-side on the same corridor) into a single node on the graph, possibly using an unsupervised learning algorithm. Should user feedback or activity reflect that any node is not relevant, we could consider removing it.

Finding a good heuristic function. A difficulty that may arise with using the A* algorithm is devising a good heuristic function, due to the need to balance the criterion of **admissibility** and desideratum of **dominance**. An admissible heuristic never overestimates the true cost from a node to a goal, for all nodes (Hart et al., 1968). On the other hand, if a heuristic $h_1(n)$ estimates a higher cost than another heuristic $h_2(n)$ for all nodes, $h_1(n)$ dominates $h_2(n)$ since the former's cost is closer to the true cost—it is thus more informed and expands fewer nodes (López, 2018). Devising a good heuristic that balances both is difficult because we would have to get a heuristic that is as close to the true cost as possible without overestimating it, for *all* nodes.



For instance, while the Manhattan distance dominates the Euclidean distance as a heuristic since the former always produces a higher cost than the latter (Jha, 2021), the former might not be admissible as not all pathways in FASS intersect at right angles, meaning the real cost of travel is likely to be lower than the Manhattan distance. As such, in the absence of better alternatives, we might have to settle for the Euclidean distance as a heuristic.

Code for A* algorithm, with test cases: MS4 A* Algorithm with Test Cases.ipynb

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.

Cox, G. (2020, August 13). *A* Pathfinding Algorithm | Baeldung on Computer Science*. https://www.baeldung.com/cs/a-star-algorithm

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–7. https://doi.org/10.1109/TSSC.1968.300136

Jha, R. (2021, January). *What is the difference between Manhattan and Euclidean distance measures?* Quora. https://www.quora.com/What-is-the-difference-between-Manhattan-and-Euclidean-distance-measures

López, C. L. (2018, November 23). *Answer to 'Does domination rule apply for inadmissible heuristics?'*. Computer Science Stack Exchange. https://cs.stackexchange.com/a/100456

Ori. (2017, August 8). *Answer to 'A* star algorithm open and closed lists'*. Stack Overflow. https://stackoverflow.com/a/45577852

Zidane, I., & Ibrahim, K. (2018). Wavefront and A-star algorithms for mobile robot path planning. *Advances in Intelligent Systems and Computing*, 69–80. https://doi.org/10.1007/978-3-319-64861-3_7