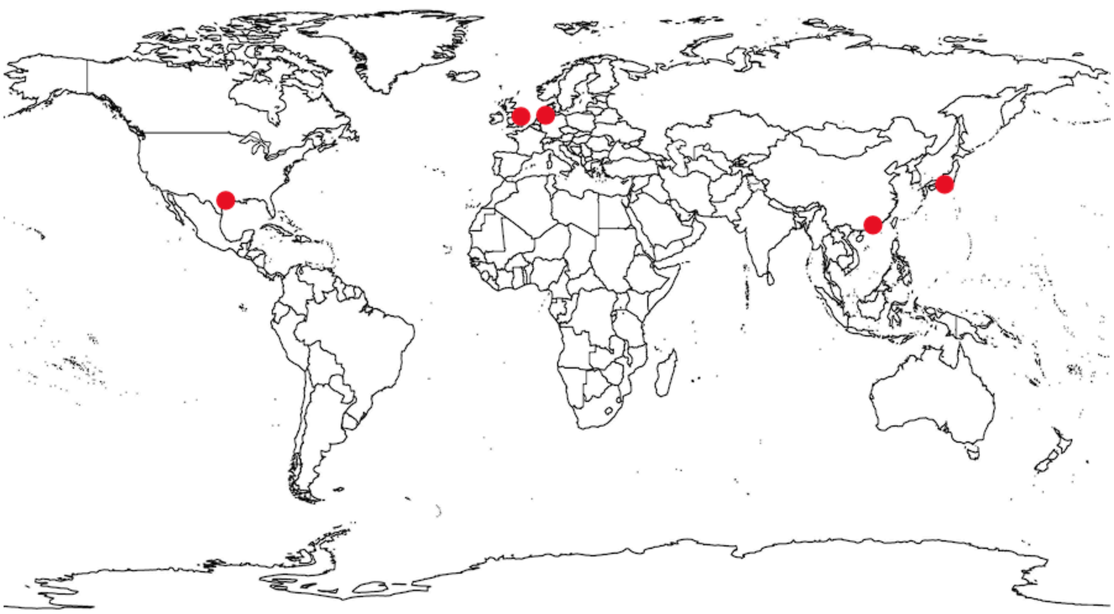
Optimising United State’s Trade Routes Using Dijkstra’s Algorithm

1.

(a) Trade involves the movement of goods and products across the borders of different countries. Over 90% of trade is conducted via shipments that require large vessels for the goods to be transported[1]. Consequently, fuel consumption must be considered, measured by the distance travelled by the vessel. It is possible to minimise fuel consumption and massively increase the efficiency of trade shipments in the US by applying Dijkstra’s Algorithm. The top 4 largest exporters globally will be investigated, to find the optimal routes from the US to other countries using Dijkstra’s Algorithm.

(b) To convert the situation into a mathematical model, the distance between the countries (edges) and the countries themselves (nodes) will be measured, to determine the most minimal route between each country. The countries being looked at include: China, the United States, Germany, Japan, and the United Kingdom.



The points on the map in each country were carefully selected upon viewing the ports with the highest trade volume being shipped. Using an online vessel calculator found at:https://sea-distances.org/, it was possible to determine the exact nautical miles required to travel from one country to another as shown in the table below.

Country Shipment Path	Distance (Nautical Miles)

China	United States	10138
China	Germany	10778
China	Japan	933
China	United Kingdom	10456
United States	United Kingdom	4983
Germany	Japan	11311
Germany	United Kingdom	360
Japan	United Kingdom	11009

Limitations

It should be noted that the US-Germany and US-Japan distance has been omitted from the calculations since direct distances (upon viewing the distances) are the shortest and would make Dijkstra's algorithm inapplicable in this case.

The calculation mentioned above of nautical miles doesn't explicitly state the methodology of how the data is compiled, meaning, several possible factors that may affect the total distance (edges), from one vertex to another, haven't been considered. These include

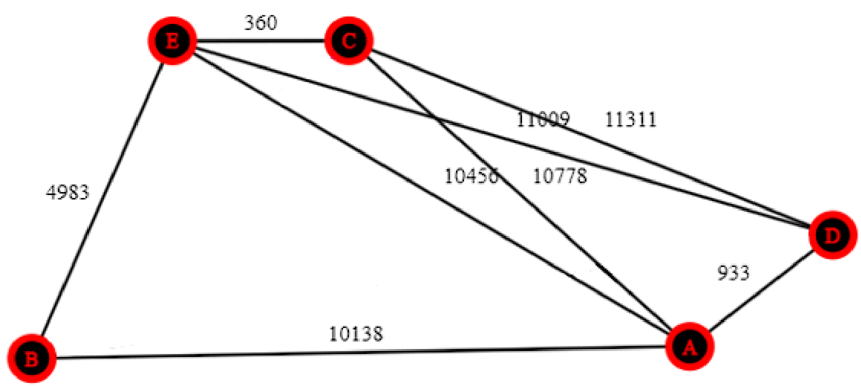
Weather events (storms and rough seas) that are unpredictable and will cause an increase in the distance travelled, to avoid them.

Navigational routes are also predetermined or set paths to avoid other shipping lanes and hazards that may increase distance.

Port layovers are also frequent happenings with shipments, in which a vessel may stop at another location before heading to the original destination, this will alter the overall distance travelled. Hence, the assumption is that the data accurately represents the directional distance between the US and other countries and can be optimised.

Graph Notation

Below is the graph model[2], displaying the data, The nodes are the countries (China, United States, Germany, Japan and the United Kingdom) written as alphabets in that respective order and placed to resemble the world map. The weight of the edges is written above their respective edges.



(c) Dijkstra's algorithm states that to identify the shortest route from one vertex to another; begin at one vertex and determine the shortest weighted edges adjacent to it. Keep track of the lowest weighted edges as you progress through each vertex, allowing you to distinguish the shortest path.

Dijkstra's algorithm can be tabulated into the form below to allow for a more methodological approach.

Vertexes	A	B	C	D	E
B	10138_B	0_B	∞_B	∞_B	4983_B
E	10138_B	0_B	360_E	10778_E	4983_B
C	10138_B	0_B	360_E	10778_E	4983_B
A	10138_B	0_B	360_E	933_A	4983_B
D	10138_B	0_B	360_E	933_A	4983_B

The steps taken to fill out the table are:

1. Identifying which vertex to begin with, in this case, B is selected as it represents the US
2. Having started at B, the adjacent vertexes are A and E, and the weights of these edges are written down as 10138 and 4983 respectively (the subscripted letters indicate the vertex that has been selected as the minimal distance) the distance from B to itself is 0 and is highlighted to represent that this vertex has been selected. The direct connection between B to C and D is impossible as they are not adjacent and hence are written as infinite.
3. After completing the first row, the lowest number (excluding 0) is identified; in this case, vertex E, being 4983, is the lowest; hence, this will be the next vertex visited.
4. The heading of the next column will be E and filled out the same way. However, the columns B and E will be highlighted and remain the same as they are now fixed, as the sum of the weighted edges is the lowest. A vertex is "highlighted" or fixed when it has been visited.

5. The rows are filled once again, but since E-A is 10456, it is omitted, and the number above is kept as a lower number (10138); the subscript remains B. To reach vertex C, the weight of the edge is 360 from edge E. Hence, this is written in the column as now the edge is adjacent and has a definitive value (the infinity is left out). Similarly, from E-D, there is now an adjacent path of 10778, which is written under the column for D.
6. The numbers will then be filled out for row E, with the values from columns B and E highlighted. The next vertex to be selected will be C, which has the lowest weight (360). Hence, the next row heading is C.
7. Now, B, C and E will be highlighted, leaving A and D to be filled. However, since the path from C to A and D is larger than the current weights, they remain the same. With that in mind, A is selected as the next vertex as it is lower.
8. In the next row, only D remains unhighlighted as the others have been visited, the weight placed in D goes from 10778 to 933, as the adjacent path to D from A is lower than the existing path weight.
9. Moving to the next row concludes Dijkstra's algorithm as all the vertexes have been visited and all the columns in the last row can be highlighted.
10. The last row however determines the shortest path from B to any other vertex. For example, from B to A, the A column's last row is looked at, which shows 10138_B indicating that the direct path from B→A is the least weighted. Similarly, the other vertexes can be summarised:

B → A = 10138_B nautical miles (path is direct to China)
B → B = remains 0 nautical miles
B → C = $360_E + 4983_B = 5343$ nautical miles (path is US→United Kingdom→Germany)
B → D = $933_A + 10138_B = 11071$ nautical miles (path is US→China→Japan)
B → E = 4983_B nautical miles (path is direct from US→United Kingdom)

(d) The numbers received in part (c) after applying Dijkstra's algorithm may give an estimate of the overall distances of the other countries from the US but will not reflect actual or accurate nautical miles that might be covered by a vessel. This is because this algorithm doesn't consider the external factors that may cause a difference in distance travelled. For example, rough weather, updraft and backdraft winds, shipping lanes and layovers will cause an increase or decrease in distance travelled which the model doesn't account for. The model will be effective if there is a large compiled dataset with accurate average trip lengths from the US to other countries, however, for this task, a nautical miles calculator was used. Another limitation of the model is that; if all the countries had a direct travel path from the US, it would be the shortest path from the US to that country, making Dijkstra's algorithm inapplicable. This has been avoided by omitting the direct distances from the US to Germany and Japan.

An improvement to the model would include its ability to consider factors external to the situation. A model that could potentially factor in some of the weather data or perhaps the layovers based on past travels, would have the ability to predict future trips based on recent travels.

2.

(a) Optimising the intricate network of roads and paths to ensure arrival at a destination from a starting point means that millions of users on platforms such as Google Maps require up-to-date and timely information. However, it is unlikely for these platforms to be exactly accurate. Traffic conditions are dynamic and constantly changing and many unpredictable factors are involved when calculating the distance and time required to reach the destination. Commuters wish to maximise their time efficiency by reaching their destination faster, minimise their fuel consumption and hope to travel in less congested and shorter routes.

(b) Applying the basic Dijkstra algorithm becomes limited or ineffective when considering multiple factors. Traffic conditions are not singularly reliant on any factor, but their combinations can create several situations where there are multiple optimal routes. Some factors affecting traffic include:
- Congestion (from peak hours)
- Accidents or breakdowns
- Weather events
- Road constructions or detours
- School zones

This would suggest the edges of the algorithm are changing and not constant. The simple Dijkstra's algorithm becomes inoperative in this case as changing edges will mean there are several combinations of weighted edges that are the lowest. For each graph, the path toward the destination will be altered. Dijsktra's algorithm is ideal for single-objective optimisation modelling but falls short when considering multi-faceted preferences.

(c) Despite the original Dijkstra algorithm being ineffective, there are a few enhancements and processes that can be made to incorporate real-world factors into the algorithm. Google Maps uses enhancements such as image recognition, machine learning and geospatial data analysis to generate useful and timely data based on real-world occurrences.
Image recognition helps identify satellite images of the roads and streets and creates a detailed area map.
Machine learning feeds new data into the model and is constantly updated to recognise patterns such as common peak traffic hours.
The **geospatial data analysis** allows for interpreting several features, a common one is the density of Google Maps users in an area using a mobile's location services, which helps identify red areas as points of congestion, orange as points of slow movement and green as traffic-free areas.

Google Maps have also been shown to utilise an adaptation of Dijkstra's algorithm known as the **A* search algorithm** which uses a similar approach to Dijkstra's but uses a more heuristic approach that helps guide it towards the destination node[4].

Each step chosen is picked to a value "f" which is equal to the sum of "g" and "h"; at each step, it picks the node that has the lowest "f" and processes it.

f(n) = g(n) + h(n)
Where **n** is the previous node on the path,
g(n) is the cost of the path from the start node to n,
h(n) is a heuristic that estimates the cost of the cheapest path from n to the target node[5].

Navigational routes can be optimised through a combination of the processes above and many more data collection features such as satellites, live crowd data, data and image processing, and vehicular preference mapping.

References

1. Oceana Europe. (2022). *Shipping Pollution*. [online] Available at: https://europe.oceana.org/shipping-pollution-1/#:~:text=Over%2090%20percent%20of%20world,by%20some%2090%2C000%20marine%20vessels [Accessed 13 Aug. 2023].

2. Csacademy.com. (2023). *CS Academy*. [online] Available at: https://csacademy.com/app/graph_editor/ [Accessed 22 Aug. 2023].

3. Does A. Which Algorithm Does Google Maps Use to Compute the Direction between 2 Points? | Saturn Cloud Blog [Internet]. Saturncloud.io. 2023. Available from: https://saturncloud.io/blog/which-algorithm-does-google-maps-use-to-compute-the-direction-between-2-points/#:~:text=The%20Algorithm%3A%20Dijkstra's%20Shortest%20Path,between%20nodes%20in%20a%20graph. [Accessed 2023 Aug 16].

4. How Does Google Map Works [Internet]. GeeksforGeeks. GeeksforGeeks; 2022. Available from: https://www.geeksforgeeks.org/how-does-google-map-works/ [Accessed 2023 Aug 23].

5. GeeksforGeeks. A* Search Algorithm | GeeksforGeeks [Internet]. YouTube. 2017 [cited 2023 Aug 18]. Available from: https://www.youtube.com/watch?v=vP5TkF0xJgI