Imperial College London

Department of Earth Science and Engineering

MSc in Environmental Data Science and Machine Learning

Independent Research Project
Final Report

Building a route optimization system that takes elevation into consideration

by

Jinsong Dong

Email: jinsong.dong22@imperial.ac.uk
GitHub username: edsml-jd622
Repository: https://github.com/ese-msc-2022/irp-jd622

Supervisors:

Sesinam Dagadu
Dr. Yves Plancherel

August 2023

Abstract

Urban traffic planning and route optimization in real-world road networks are crucial for efficient transportation systems. This study presents a multifaceted approach to address these challenges. It introduces a tool capable of integrating 2D road data with elevation information, effectively creating a 3D representation of road networks. The tool also offers route planning and visualization capabilities.

The integration function can be applied to any user-specified region. When it comes to route planning, the tool achieves a relative error of 7.074% in finding the shortest path and 13.05% in determining the most time-efficient route, compared to Google Maps.

To extend its utility, the study delves into solving the Traveling Salesman Problem (TSP) within real-world street networks. A modified Pointer Network is implemented, which can predict TSP solutions from adjacency matrices. This model exhibits an accuracy rate of 67.0%, with an average route length ratio of 1.02 concerning optimal route length.

The combination of these advancements offers a robust solution for urban traffic planning, incorporating terrain considerations and real-world road network complexities. This research lays the groundwork for more efficient and sustainable transportation systems in urban areas.

1 Introduction

1.1 Route optimization problem

A route optimization problem for real-world road networks can be divided into two parts generally. The first part is finding the shortest path problem, which is finding a path between two intersections on a road map such that the sum of the distance of the constituent road is minimized. For this part, the famous Dijkstra's algorithm[1] based on the graph theory is already a mature method. But Dijkstra can only solve the single-source shortest path problem with non-negative edge weight. The Bellman-Ford algorithm[2] can deal with the shortest path problem with negative edge weight, but it is slower than the Dijkstra algorithm. Some research has been done to improve the computing efficiency of these algorithms[3]. For a road network, distance is a non-negative weight, so the Dijkstra algorithm is a better choice.

The second part is making route optimization. The route optimization problem in this project is a kind of Travel Salesman Problem(TSP) [4], which means given a list of cities and distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city [5]. TSP is a classic NP-hard problem, which means it can not be solved in polynomial time. One example of a solution of a TSP is shown in Fig. 1. In this example, there are 35 locations in total, and the shortest way between these locations is drawn in the line. There are $35!$ possible combinations of the route total, the 'brute-force solution' will take a very long time to get the results.

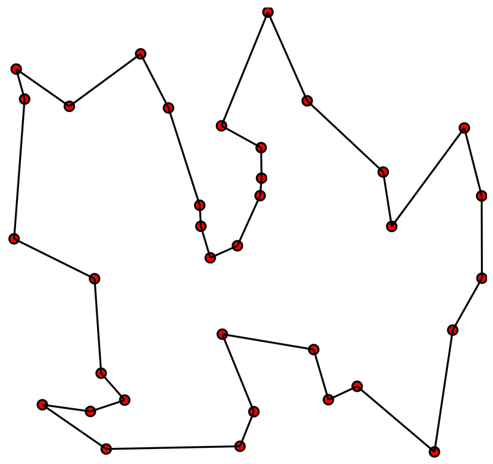

Figure 1: A solution example for TSP. Red points represent locations, lines represent the optimized route.

Algorithms to reduce the calculation time are essential. Many heuristic algorithms have been utilized in solving TSP[6], for example: ant algorithm, greedy method, simulated annealing, tabu search and genetic algorithm [7]. The genetic algorithm randomly samples values of the changing cells between lower and upper bounds to generate a set of combinations of possible routes, and choose the best one. Genetic algorithms can give a near-to-optimal solution in a relatively short time, but we cannot know how near it can be to the best route. Ant Colony Optimization algorithm(ACO) is one of the metaheuristic methods for solving TSP, it works like observation of real ants, and upon finding food return to their colony while laying down pheromone trails[8]. The ACO can also get a near-optimal solution to the traveling salesman problem, it's able to find the global optimum in a finite time. Some improved ACO algorithms have also been proposed like Parallel implementation of ant colony optimization [9]. Christofides algorithm[10] utilizes Minimum spanning tree(MST) to get the approximation solution of traveling salesman problem, which guarantees its solutions will be within a factor of 3/2 of the optimal solution length. Christofides algorithm has the limitation that it can only be applied on the instances where the distances form a metric space (they are symmetric and obey the triangle inequality) [11]. The simulated annealing(SA) algorithm is a kind of greedy algorithm. It uses randomness as part of its search for the best solution, so it is a stochastic global search algorithm. The simulated annealing algorithm is good at jumping out of the local minimum[12].

Some other research has been extended into the domain of logistics, respective extensions are called Vehicle Routing Problem(VRP)[13], VRP not only optimize the distance between two single points but also optimize the distances between a series of points under different criterion and restrictions. Some research extended VRP to green logistics, they optimize the fuel consumption and emissions[14]. Marc Schröder and Pedro Cabral[15] build a model to estimate the CO_2 emissions for road freight transportation and found eco-friendly routes can yield up to 20% emissions reduction. Jiquan Wang[16] developed an energy consumption prediction algorithm to make route optimization for electric vehicles.

Another way to handle the TSP problem is using the neural network method[17]. Hopfield neural network[18] is the first neural network for handling TSP, which converts the objective function into the energy function of a fully connected neural network. Several research focused on how to improve the Hopfiled method on its convergence[19] and performance[20][21]. Graph neural network is also an architecture to solve TSP problem[22] when dealing with graph data, graph data is a complex data structure that a fully connected network hardly deals with. Neural network with reinforcement learning[23] is a kind of method to solve sequential decision-making problems. From the notable Pointer network proposed Vinyals[24] which used an attention mechanism to take attention as a pointer, many reinforcement networks based on it were developed such as graph pointer network with reinforcement[25] and graph point network with negative travel length as the reward signal[26].

The research about finding the shortest path mentioned above usually uses 2D coordinate data format and none of them did the real-world case in Africa area. The research about route optimization with the deep learning part mentioned above only uses coordinates as the input data format, this data format can only reflect the straight-line distance between locations which is not able to reflect the distance of the real-world road network. In this project, the path planning part is considered with 3D coordinates and a small revision is made on the Pointer Network so that it can use the adjacent matrix as input data format to solve the TSP problem on a real-world network. Snoocode Corporation has designed a route optimization system that works offline on a smartphone and provides results 42000 times faster than the conventional method. However, this method only considers a 2D map. The next step for Snoocode is developing a method that can optimize the route considering land elevation. Elevation is a very important factor for electric bikes and bicycles which are the main transportation of delivery companies in Accra. To be more specific about the function of the route optimization system, suppose there are 20 locations to give delivery to, the shortest way need to be found so that every location can be delivered.

For making a system to do 3D route optimization, the problem is divided into four parts:

- Data processing. Existing software for route optimization or path planning now only uses 2D road data. And the coordinates of available open-source road data are in 2D format. So the first important task is integrating 2D road data and raster elevation data into 3D road data.
- Building 3D road network with processed data. For a real-world road system, it is not correct to do route optimization only with coordinates(straight-line distance) of several locations. Because of the complex road conditions, it is impossible to walk in a straight line. So a network graph that can easily calculate the shortest distance of any pair of locations is the key. With the network, the adjacent for any number of locations can be created for doing route optimization.
- Route optimization. This part contains two methods, one is a conventional algorithm(Work of Rutvji Kulkarni: rutvij.kulkarni22@imperial.ac.uk), and the other one is using a deep learning model. Traditional algorithms can achieve a relatively higher efficiency for route optimization than the brute-force method. However, as the number of target locations to optimize grows, the computing time grows as well. When the number of optimized locations reaches a certain level, even mature algorithms take a long time to optimize. A deep learning model to predict the optimized route for a large number of locations can handle this. Although it may take a very long time to train the model, SnooCODE can integrate it into the offline mobile application to do route optimization in a short time once it is trained. The model architecture used in this research is called Pointer Network developed by Google.

1.2 Code structure

The codes in this research are developed by Python and encapsulated into several classes based on their functionality. The structure of the codes is as shown in Fig **??**. The Road_network class is the main class for integrating 2D road data with elevation data and creating road network graphs. There are also some small functions in this class, for example: 'get_data function' to get the data stored in the class, 'get_shortest_path' function to return all nodes on the shortest path between two locations, 'get_shortest_path_length' function to get the length value of the shortest path between two locations, 'weight_matrix' function to get the adjacent matrix given a list of locations.

The Visualisation class is the class for visualizing the path planning between any pair of locations. This class also have some functions to show 3D road map and 2D road map.

The City class is for storing the coordinates of locations. There is a coordinate transformation function in the City class, If the user inputs a coordinate in latitude and longitude, it will automatically transform it into easting and northing format which is easier for calculating distance.

Besides the codes drawn in the code structure figure, there is also a class storing loss functions for calculating distance/travel time while creating a network, a code script for generating a dataset and a Jupyter notebook containing codes to train the deep learning model.

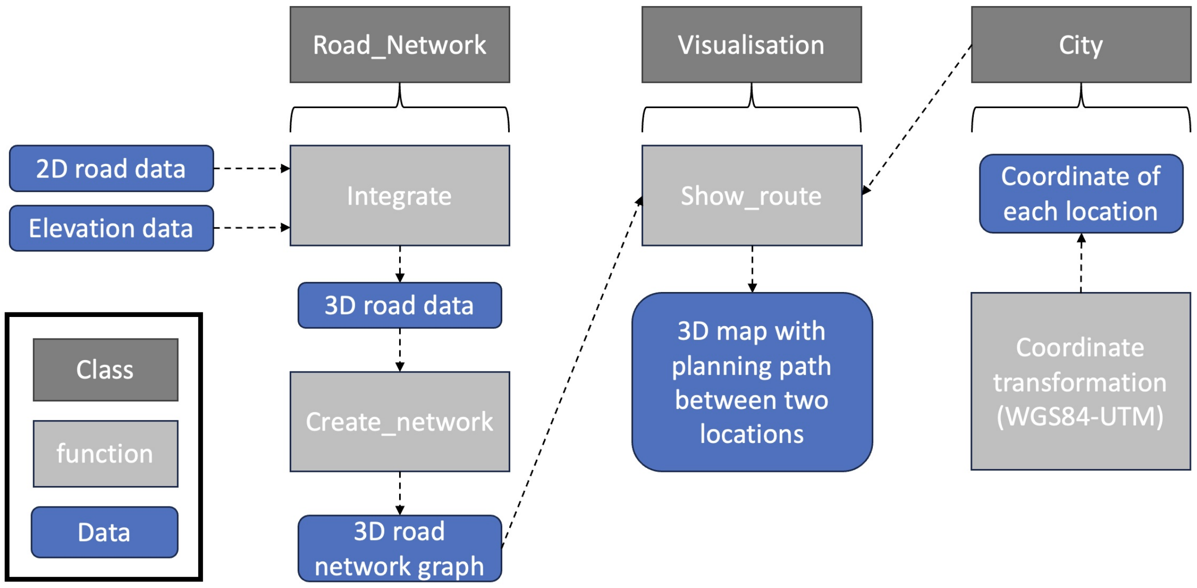

Figure 2: The code structure. The dark grey block is class, the light grey block is a function, the blue block is data.

2 Methods

This project mainly focuses on real-world road networks, but it is hard to consider all the factors of roads because of their complexity. Some assumptions are raised in this research:

- The ratio of traffic to capacity is 0.5 for all road segments.
- There is no traffic light on the road.
- Vehicles travel at the maximum speed specified for each road segment with no traffic congestion in consideration.

2.1 Integrate data for 3D road data

The target city of this research is Accra, Ghana, whose range of coordinates is 5.52N to 5.68N in latitude and 0.31W to 0.01E in longitude. The main essential data needed in this research are the 2D road information data and the elevation data.

The 2D road information data was retrieved from OpenStreetMap by Overpass API, whose last updated time is 2023-06-11T03:07:17Z. The 2D road data is a JSON file, the main part of the data is in key 'elements', and the value of 'elements' is a list containing all road segments in the Accra area. Each element of the list 'elements' is a road segment, containing keys 'nodes', 'geometry', and 'tags'. The value of the key 'nodes' is a list containing several integer numbers that refer to the unique ID of a point, The value of the key 'geometry' is a list containing several dictionaries, each dictionary contains 'latitude' and 'longitude' which refer to the geographical coordinates of a point. In each road segment, the length of 'nodes' is equal to the length of 'geometry', because the elements in 'nodes' correspond one-to-one with the elements in 'geometry' with the same index. Connecting the series of points within each element forms the road segment.

The value of the key 'tags' contains the attribute of each road segment, for example: 'highway' is the type of road segment, and 'oneway' indicates whether the road segment is one-way direction. Fig. 3 shows the road type distribution of Accra, and Fig. 4 shows the access distribution of Accra. The road type of road segments has a significant influence on the travel time calculation in a later chapter as there are different speed limits for different road types. The access type determines whether a road segment can be used for delivery. The attribute 'oneway' is also considered in this research, and road segments that are 'oneway' can only be used in one direction.

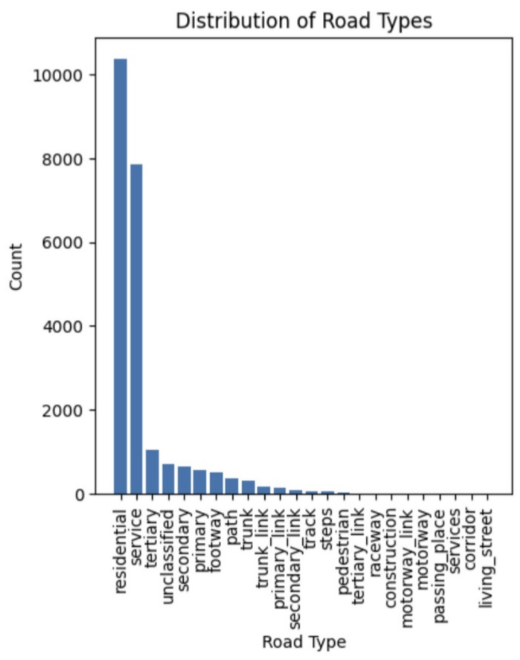

Figure 3: The road types of Accra. X-axis: road types of Accra, Y-axis: the quantity of each type.

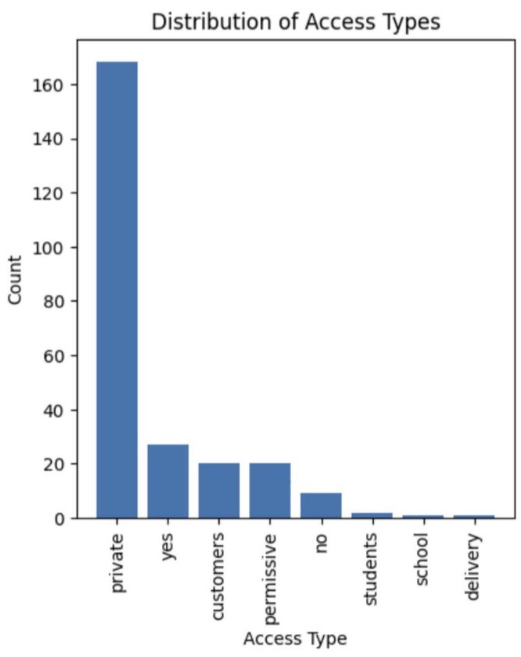

Figure 4: The access types of Accra. X-axis: access types of Accra, Y-axis: the quantity of each type.

The elevation data is SRTM 1 Arc-Second Global retrieved from the website USGS. The elevation data is in raster format, whose resolution is about 30 meters/pixel. The integration of 2D road data and elevation data is to add one key 'ele' in the key 'geometry' for each point in the 2D road JSON file, which represents the elevation of that point. Because the elevation data is a raster data and each pixel of elevation data is in the shape of $30m * 30m$, there might be some different road points in the same grid.

To make the integration of these two data more precise, interpolation is applied so that every road point is assigned its elevation value according to the values of the four pixels surrounding it. The schematic diagram of interpolation is shown in Fig. **??**. The process of interpolation is as follows:

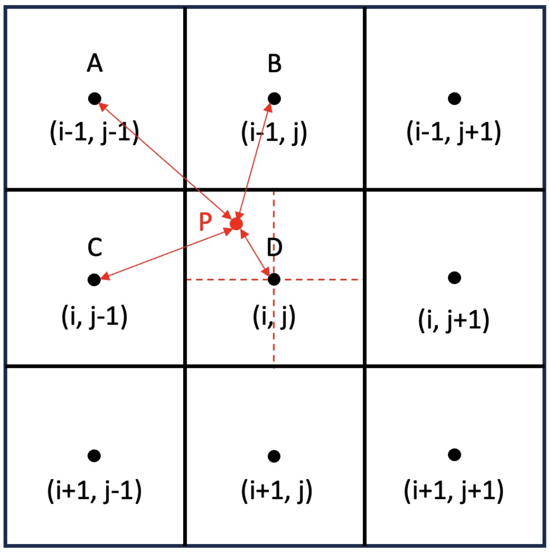

Figure 5: Illustration of interpolation. Grids represent pixels of raster elevation data. Black points represent the center of grids. A, B, C and D represent four grid centers surrounding target red point P. The red dashed line divides grid D into four sub-grids. Red double arrow lines represent the distance between A, B, C, D and P.

1. Get the grid length l.
2. Find the current grid the point p belongs to.
3. Find which subregion of the current grid the current point p belongs to.
4. Use the elevation data of four grids surrounding the point p to interpolate the elevation. In this diagram, $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$, (i, j).
5. Calculate the weight for the point p. For example: the weight of point A is: $w_A = 1/distance(p, A)$
6. Calculate the elevation of point p with the following equation:

$$h_p = \sum_{i=1}^{4} \frac{w_i h_i}{\sum_{i=1}^{4} w_i} \tag{1}$$

The calculated elevation of each point is added in the 2D road JSON file as the key 'ele' in the key 'geometry'. After this process, the 2D road data is turned into 3D road data.

2.2 Building 3D road network

To easily calculate transportation indicators like distance, and travel time between any two pairs of points in Accra, building a 3D road network is essential. In this research, a 3D road network graph is built by the open-source package NetworkX based on the integrated 3D road data. The transportation road network graph consists of nodes and edges, nodes are points in the 3D road data and edges are formed by sequentially connecting the points of each road segment. Every node will be assigned a unique ID while creating, which is the same ID in the 3D road data file.

The one-way road is considered based on the attribute 'oneway' in the 3D road data when creating an edge. If the road segment is a one-way road, the edges of this segment are created only once in the given sequence. However, if the road segment is not a one-way road, the edges of this segment are also created in reverse order. As a consequence, if the road is not one-way, there will be two-directional edges in opposite directions between two nodes in the segment, if the road is one-way, there will only be one-directional edge between two nodes in the segment.

When creating edges for each road segment, each edge would be assigned with some attributes, like distance and travel time. Users can also define their own attributes like battery consumption in the class 'CostFunctions'. The distance attribute is calculated based on the following equation:

$$distance(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{2}$$

where i and j indicate two different nodes, and x, y, z indicate the coordinates of a node. The travel time attribute is calculated based on the following equation:

$$T_i = T_i^0(1 + \alpha(v/c)^\beta) + Gradient_i \tag{3}$$

$$Gradient_i = \frac{G.D.}{S_i} \times P_i(P_i > 0) \tag{4}$$

where T_i^0 is the free travel time of the road segment; v/c is the ratio of traffic to capacity, based on the assumption mentioned before, this value is 0.5 for all road segments; α and β are two parameters ($\alpha = 0.15, \beta = 4$); $G.D.$ is the road segment length with a slope; S_i is the free flow speed of the road segment; P_i is the slope weight of the road segment.

The free flow speed of different road segments is on the road type, as shown in Table 1.

Road type	Speed limit (km/h)
Primary	50
Secondary	50
Tertiary	50
Trunk	50
Residential	20
Motorway	80
Service	20
Unclassified	50

Table 1: Speed limit of different road types

This network can be used to calculate some information between any pairs of points in the graph. Information including distance, and traveling time. This information can be added according to the purpose of the user. After creating the network graph, users can use it to get the best path between any pairs of locations in Accra based on the cost function they choose. The algorithm to achieve this functionality is the Dijkstra algorithm which is built-in in the NetworkX package.

In this research, I implement a function that can output the adjacent matrix based on the input of a list of locations. Each entry of the adjacent matrix represents the cost between the corresponding pair of locations. This function is the most important part of generating data to train the deep learning model for the TSP problem of a real road network.

2.3 Pointer Network for TSP problem of 3D road network

The deep learning model architecture used in this research is Pointer Network[24]. The Pointer Network's structure as shown in Fig. 6 uses one encoder to receive and transcode the adjacent matrix data into the hidden layer. Then it uses a decoder and an attention mechanism to calculate the mechanism score with the hidden layer of the encoder. The mechanism score is used to produce the pointer or index of the next location to go.

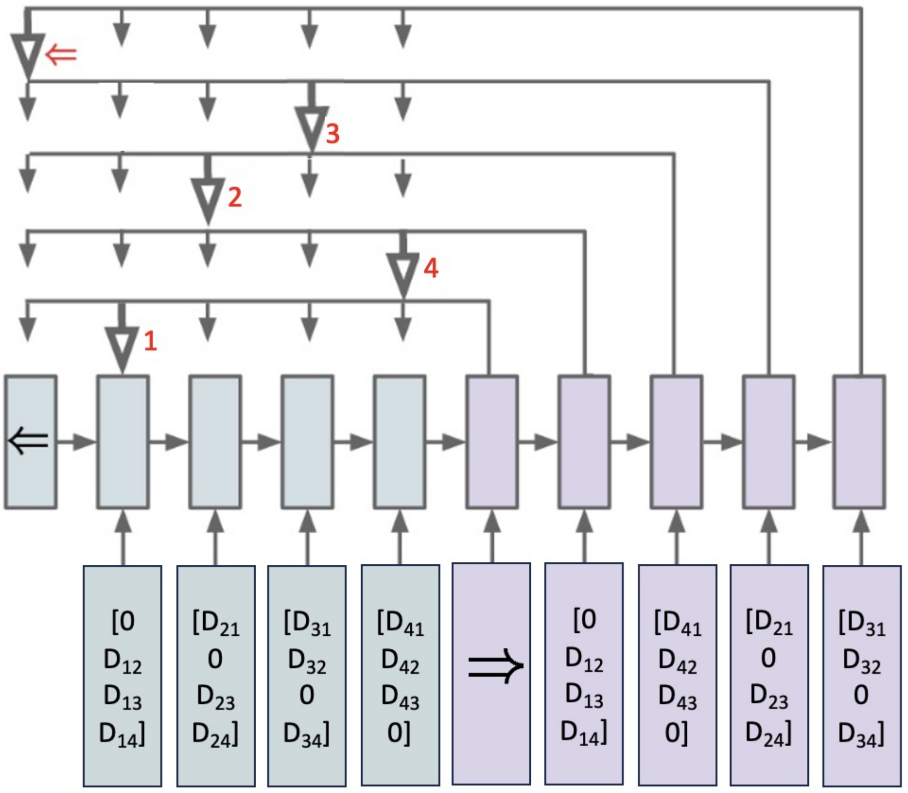
⇐
3
2
4
1
⇐
[0 D_{12} D_{13} D_{14}]
[D_{21} 0 D_{23} D_{24}]
[D_{31} D_{32} 0 D_{34}]
[D_{41} D_{42} D_{43} 0]
⇒
[0 D_{12} D_{13} D_{14}]
[D_{41} D_{42} D_{43} 0]
[D_{21} 0 D_{23} D_{24}]
[D_{31} D_{32} 0 D_{34}]

Figure 6: Illustration of pointer network. Ptr-Net - An encoding RNN converts the input sequence to a code (blue) that is fed to the generating network (purple). At each step, the generating network produces a vector that modulates a content-based attention mechanism over inputs ([5, 5]). The output of the attention mechanism is a softmax distribution with a dictionary size equal to the length of the input. The input at each step is an entry of the adjacent matrix.

The decoder is a masked LSTM model, after a location is predicted in a time step, the pointer for this location will be masked.

To train this model, a 70000 dataset is generated. Each data point contains one adjacent matrix and one label. The adjacent matrix is of five different random locations in Accra, the entry of the adjacent matrix is calculated by the distance cost function, and the label is the optimized route for these five locations in a format of the list of indices. The optimized route for each data point is calculated by brute-force algorithm, so it can be sure that the route is the optimal solution.

The model is trained for 5000 epochs, with the 2048 batch size.

3 Results

3.1 Performance of Integration

The tool developed in this research shows that route optimization performance for any two locations in Accra is close to Google map's performance. The visualization of the path done by this tool is shown in Fig. 7.

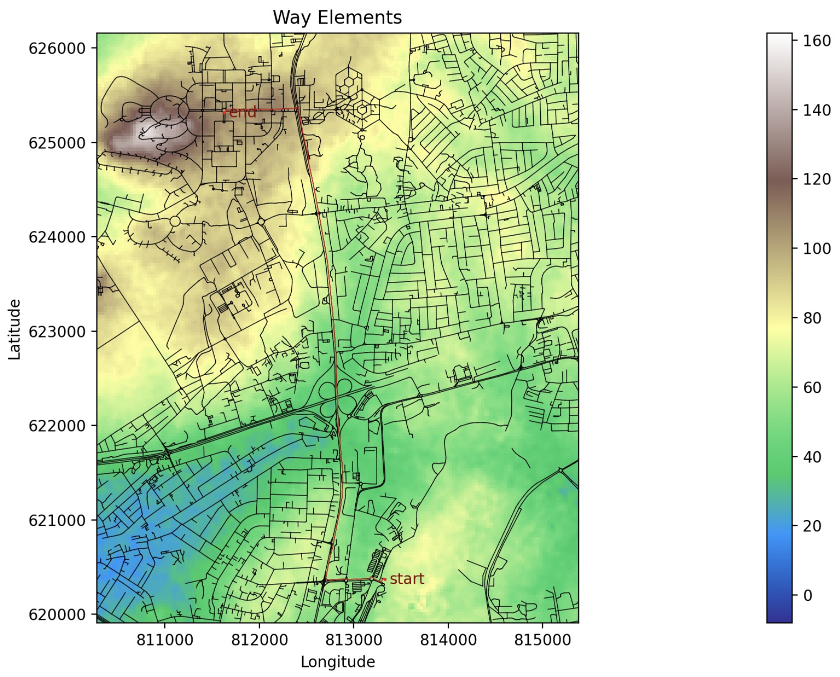

Figure 7: Path from Kotoka airport to University of Ghana by this research. Black lines are streets. Red lines are the shortest path. Colors represent elevation.

The corresponding path planning done by Google is shown in Fig. 8. It can be seen that the travel paths are almost the same in this case.

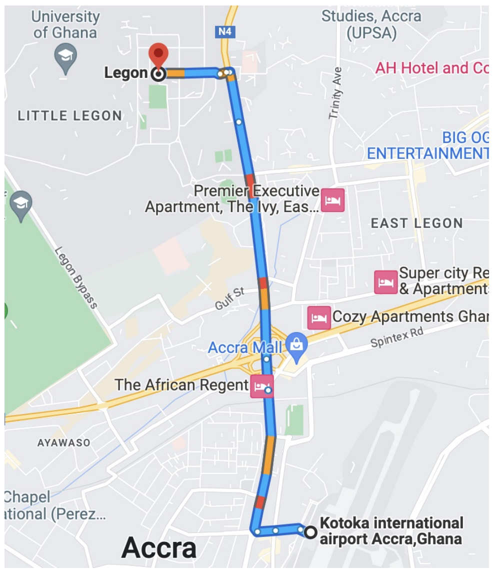

Figure 8: Path from Kotoka airport to University of Ghana by Google.

To verify the performance of the tool developed in this research, the optimized paths of 5 pairs of locations are compared with the path from Google Maps for the same 5 pairs of locations. To verify the distance cost function, walking is chosen for transportation in Google Maps as this is the way to get the shortest way. To verify the time travel cost function, driving with a clear road is chosen for transportation in Google Maps as this is the way to get the least time-consuming path.

The five pairs of locations are as follows:

1. Kotoka International Airport - University of Ghana
2. Kotoka International Airport - Accra zoo
3. Kotoka International Airport - National Museum of Ghana
4. Accra Zoo - University of Ghana
5. Accra Zoo - National Museum of Ghana

case	distance	distance(Google)	Relative Error	time	time(Google)	Relative Error
unit	km	km		minute	minute	
1	6.55	6.5	0.77%	12.85	13	1.15%
2	6.94	7.7	9.9%	14.32	15	4.5%
3	7.5	7.4	1.4%	13.62	13	4.8%
4	5.4	6.8	20.6%	12.18	22	44.6%
5	9.97	9.7	2.7%	18.89	21	10.04%

Table 2: Route results compare with Google.

The compared results are shown as Table 2. As we can see, the mean relative error of distance is 7.074%. This is a relatively good result, which means that this tool can find the shortest path in most cases. But there still are some cases that are not the shortest way like case 5. The reason may be that this tool only finds the effective route of road segments with the 'highway' tag, so some road segments without the 'highway' tag may be ignored, the tool has to skip these road segments and use some longer road segments when finding the best route. What's interesting is that there are some paths found by this tool shorter than the results from Google. This may be because some road segments can not be accessed but have no 'access' tag in the JSON file, so the tool makes it able to be accessed by default. These road segments can be a shortcut which can make the path shorter.

For the travel time part, the tool's result seems to not be good enough. The average relative error is 13.02%, which is relatively high. It can be seen from the table that most travel time calculated by the tool is smaller than travel time from Google Maps. There might be two reasons for that. The first reason is the vehicle speed of each road segment in this tool is the max speed limit. Although the travel time from Google Maps is also calculated by the condition of clear roads, the vehicle speed for calculating travel time does not reach the max speed limit in Google Maps. The second reason is that Google map considers how long it takes to wait at a traffic light, even though the condition of the roads is clear. Considering these two factors together, the time calculated by the tool can be much smaller than the travel time from Google Maps. The travel time from Kotoka airport to the University of Ghana seems very close to the travel time from Google Maps. The reason for this might be the route from Kotoka airport to the University of Ghana is mainly composed of trunk road segments, and there are few traffic lights on the trunk roads. Although the travel time calculation might have deviation from the real-world travel time, the path planning by the tool is with reference significance.

3.2 Performance of Pointer Network

To evaluate the performance of the trained Pointer Network, two criteria are chosen. The first criterion is accuracy. The correct answer should make the values and positions of predicted indices all correct. The second criterion is the route length ratio, which is the ratio between the total length of the predicted route and of the true optimized route. The smaller the route length ratio is, the higher performance the model has, and the best route length ratio should be 1.

A test dataset with the size of 70000 is created to verify the performance of the model. The result shows that the accuracy of the model is 67.0%, and the mean route length ratio is 1.02. The performance of the pointer network with an adjacent matrix as input shows that The model can predict almost the best-optimized route. Although the accuracy of the model is less than 70%, the average route length ratio is very close to 1. The main reason for the error is that there might be two points that are very close to each other or the distance of the two points is close to 0 because the dataset is generated randomly. The order of two locations that are very close to each other has little effect on the route distance calculation, so the model may get confused with this kind of pair of locations and yield a wrong order for the two locations.

4 Discussion

In this research, a tool for integrating 2D road data and elevation data into 3D road data is developed. The main difficult task for this process is how to merge two different types of data. The 2D road data consists of several road segments, and each road segment consists of several discrete points, while the elevation data is raster data with a 30-meter resolution. To address this problem, two methods are considered. The first one is using the value of the block to which the road segment point belongs. This method is simple but cannot handle the case that two points belong to the same grid of elevation data. The second one is interpolation for each point in the road data. In the tool developed by this research, which of the four subregions of the elevation grid each point belongs to should be found first, and then values of this grid and the other three grids that are adjacent to the subregion are used to do interpolation. The weight for doing interpolation is inversely proportional to the distance between the target point and the grid center. This method is more time-consuming than the first one, but it can achieve a relatively higher precision.

After the integration process, the network for a city is created. The main difficult task to resolve in this process is how to make an easily used network to do route planning. The mature method to resolve this task is creating a graph for the road network, which consists of nodes and edges. Nodes are created from the points in integrated road data, and edges are created by connecting the nodes sequentially. When creating edges, two cost functions are integrated distance and travel time. Users can do route planning based on the cost function they choose. The distance cost function can make route planning in the real-world street map which is close to Google Maps. The limitation in this part is that the travel time cost function is based on several hypotheses, such as the road is clear, there is no traffic light, and vehicles go at the maximum speed. These hypotheses may lead to the calculated travel time deviated from real travel time, but it can still be used to give the least time-consuming route. Another main reason for the deviation is the road data itself. The tags of road data collected from OSM may be not enough to reflect the real-world case. For example, some roads may be only available for pedestrians, and some roads may not be able to be accessed. This may cause some little error on a small amount of road segment, because some private-access way may be a shortcut. To improve the performance of this process, data should be refined gradually, which can make the network closer to the real-world case.

After the two processes mentioned before, the dataset can be generated by the tool for training the revised Pointer Network. The model can not distinguish locations that are close to each other very well, which affects the accuracy of the model. Despite the relatively low accuracy, the model can still achieve a route optimization that is close to the optimal answer on average. To improve the performance of the model, more data with more complex modes can be fed to the model during training so that the model can learn how to handle these modes. The original Pointer Network can be extended for predicting optimized routes for more locations than the number of training locations. But in my revised Pointer Network, the input of the model is changed to the adjacent matrix. The strength of this revision is that it can be used for route optimization of the real-world road network. The limitation of this revision is that it can not be extended to predict the optimized route with more locations than the number of locations during training. Because the input layer of the model is fixed, the size of the adjacent matrix is according to the number of locations to do route optimization. To improve this, users can train different models for different numbers of cities, or the structure of the model could be improved in the future so that it can adapt to the variable size adjacent matrix.

5 Conclusion

In this research, a tool that can integrate 2D road data and elevation data into 3D road data as well as route planning and visualization is developed. The integration function of the tool can be used on any other area in addition to Accra specified by users. The route planning function of the tool can achieve 7.074% relative error for finding the shortest path, and 13.05% relative error for finding the most time-saving path compared to Google Maps. Using the data generated from the tool mentioned above, a revised Pointer Network is implemented to predict the answer of TSP given adjacent matrix, which can be used for real-world street route optimization. The accuracy of the model is 67.0%, the average route length ratio compared to the optimal route length is 1.02.

References

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

[2] R. Bellman. On a routing problem. *Quart. Appl. Math.*, 16(1):87–90, 1958.

[3] Federico Busato and Nicola Bombieri. An efficient implementation of the bellman-ford algorithm for kepler gpu architectures. *IEEE Trans. Parallel Distributed Syst.*, 27(8):2222–2233, 2016.

[4] E.L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, 1985.

[5] Wikipedia. Travelling salesman problem.

[6] Shabnam Sangwan Chetna Dahiya. Literature review on travelling salesman problem. *International Journal of Research*, 05(16):1152–1155, June 2018.

[7] Kylie Bryant. Genetic algorithms and the travelling salesman problem. *Department of Mathematics*, December 2000.

[8] Miguel A. Marino Mohammad Reza Jalali, Abbas Afshar. Ant colony optimization algorithm (aco): a new heuristic approach for engineering optimization. *Proceedings of the 6th WSEAS Int. Conf. on Evolutionary Computing*, 16-18:188–192, June 2005.

[9] Mitica Craus Laurentiu Rudeanu. Parallel implementation of ant colony optimization for travelling salesman problem. *Proceedings of the 4th WSEAS Int. Conf. on Soft Computing, Optimization, Simulation and Manufacturing Systems (SOSM 2004),*, Miami(Florida), April 2004.

[10] René van Bevern and Viktoriia A. Slugina. A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem. *Historia Mathematica*, 53:118–127, 2020.

[11] Roberto Goodrich, Michael T.; Tamassia. *Algorithm Design and Applications*. Number 513-514. Wiley, 2015.

[12] Gao Ye and Xue Rui. An improved simulated annealing andgenetic algorithm for tsp. In *2013 5th IEEE International Conference on Broadband Network and Multimedia Technology*, pages 6–9, 2013.

[13] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers and Industrial Engineering*, 99:300–313, 2016.

[14] Eliana M. Toro O., Antonio H. Escobar Z., and Mauricio Granada E. Literature review on the vehicle routing problem in the green transportation context. *Luna Azul*, 1909-2474:362 – 387, June 2016.

[15] Marc Schröder and Pedro Cabral. Eco-friendly 3d-routing: A gis based 3d-routing-model to estimate and reduce co2-emissions of distribution transports. *Computers, Environment and Urban Systems*, 73, 08 2018.

[16] Jiquan Wang, Igo Besselink, and Henk Nijmeijer. Battery electric vehicle energy consumption prediction for a trip based on route information. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 232(11):1528–1542, 2018.

[17] Yong Shi and Yuanying Zhang. The neural network methods for solving traveling salesman problem. *Procedia Computer Science*, 199:681–686, 2022.

[18] J. J. Hopfield and D. W. Tank. “neural”computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985.

[19] Yanfen Luo. Design and improvement of hopfield network for tsp. *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, 2019.

[20] Junfei Qiao, Zhiqiang Hu, and Wenjing Li. Hysteretic noisy frequency conversion sinusoidal chaotic neural network for traveling salesman problem. *Neural Comput. Appl.*, 31(11):7055–7069, nov 2019.

[21] Lucas García, Pedro Talaván, and Javier Yáñez. Improving the hopfield model performance when applied to the traveling salesman problem. *Soft Computing*, 21, 07 2017.

[22] Marcelo O. R. Prates, Pedro H. C. Avelar, Henrique Lemos, Luis Lamb, and Moshe Vardi. Learning to solve np-complete problems - a graph neural network for decision tsp, 2018.

[23] Martijn van Otterlo and Marco Wiering. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[24] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2017.

[25] Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning, 2019.

[26] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017.