Minimising Cost for Travel Bus Operator

Bryan Kai Xuan Koh, Hong Liang Sia, Jun Hui Lim, Louis Jia Jun Chun, Rui Feng Chia, Huaqun Guo[✉], and Kar Peo Yar

Infocomm Technology Cluster, Singapore Institute of Technology, Singapore, Singapore
{2201543,2201493,2201476,2201618,
2201695}@sit.singaporetech.edu.sg, {huaqun.guo,
KarPeo.Yar}@singaporetech.edu.sg

Abstract. The tourist travel bus sector is one of the main contributors to the Singapore economy. However, the cost optimization is a challenging factor when it comes to operating a travel bus. This project, BusPathFinder, aims to identify the shortest and most efficient routes between Changi Airport Terminal 3 and various hotels by utilising advanced pathfinding map algorithms. The project uses the Google Map API and three pathfinding algorithms: Dijkstra Algorithm, Bellman-Ford, and the Travelling Salesperson (TSP). As a result, The BusPathFinder application is materialised as an advanced tool capable of mapping out the best paths using three algorithms from Changi Airport Terminal 3 to different hotels. In addition, BusPathFinder can also visualise these routes in HTML format, as well as calculate the cost of each travel route and the total distance travelled. The project's findings will be applied to improve the efficiency and cost-effectiveness of bus routes in the tourism sector.

Keywords: Pathfinding algorithm · BusPathFinder · Cost optimization

1 Introduction

In today's bustling world of tourism, effective transportation services are necessary to provide a seamless travel experience. In this context, our project, BusPathFinder, aims to revolutionise the tourist travel bus industry through the usage of advanced pathfinding map algorithms. Due to the millions of tourists that arrive at Changi Airport Terminal 3 each year, it is paramount to find optimal bus routes to hotels to streamline operations and raise service standards. Our project investigates advanced pathfinding algorithms and seeks to implement them with BusPathFinder, to significantly save costs as well as improve operations in the current landscape.

The approach of this project is founded on a combination of the Google Map API and 3 pathfinding algorithms: Dijkstra Algorithm, Bellman-Ford, and the Travelling Salesperson. These algorithms will be used to calculate the shortest and most efficient routes between Changi Airport Terminal 3 and different hotels. Simultaneously, the Google Map API will be used to extract real-time data from Google to find which path will have the least amount of Electronic Road Pricing (ERP) gantries to determine the most cost-effective path.

J. Lu et al. (Eds.): IRC-SET 2023, *Proceedings of the 9th IRC Conference on Science, Engineering, and Technology*, pp. 21–30, 2024.
https://doi.org/10.1007/978-981-99-8369-8_3

2 Related Works or Literature

Scenic Athens is a context-aware mobile city guide for Athens (Greece) which uses an innovative algorithmic approach in an attempt to solve the Tourist Trip Design Problem (TTDP). The mobile city guide formulates the optimization problem as a Mixed Team Orienteering Problem with Time Windows (MTOPTW) and presents a metaheuristic method, such as the Iterated Local Search method to solve it [1]. The modified adaptive large neighbourhood search (MALNS) method, using the four destructions and the four reconstructions approach, was proposed in 2021 to present a solution to the family tourism route problem in Thailand [2]. To efficiently determine the shortest tourist route while visiting multiple destinations in a confined area, the study proposes a genetic algorithm (GA) based on biological evolution and natural selection [3].

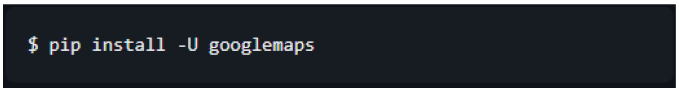
R.R. Donnelley and Sons introduced MapQuest in 1996, used a proprietary bidirectional version of Dijkstra's Shortest Path Algorithm to determine the shortest route for early website users [4]. Maposcope is a route planning app that allows a user to choose stops by typing in addresses, and launches the navigation app as the user navigates from stop to stop. Once arriving at a stop, the user will have to switch back to the app and click done before moving on to the next stop [5]. Waze is a navigation app and collects data from users' devices about traffic conditions to suggest the most optimal routes to other users. However, Waze does not currently support navigating in lanes dedicated to public transportation, bicycles, or trucks. An active internet connection is essential for Waze to function correctly, as it relies on real-time data and map updates [6]. Badger Maps is a route planning and optimization tool designed to streamline and enhance location-based operations for businesses, and allow users to access statistics such as mileage, travel time, and full-day time of their routes [7].

3 Solution Designed

3.1 API Used

The Python client for Google Maps Services [8] empowers developers to perform various tasks, including geocoding, finding directions, obtaining direction matrices, and integrating other features of the Google Maps Platform Web Services into a Python application. To utilise the Python client for Google Maps Services, developers are required to have Python 3.5 or a later version installed on their system, in addition to possessing a valid Google Maps API key. For every Google Maps Web Service request, an API key or client ID is necessary. API keys are generated through the 'Credentials' page within the 'APIs and Services' section of the Google Cloud console.

The Python client for Google Maps Services can be installed by running the following command:



```
$ pip install -U googlemaps
```

This command will ensure that the latest version of the Google Maps package is installed or update it to the latest version if it is already installed. Once the installation is complete, developers can begin integrating and using the Google Maps services within their Python applications.

The Python Client for Google Maps Services allows two primary Google Maps APIs namely Directions API and Places API utilise in the project. The Directions API [9] is a service that processes HTTP requests and provides directions between locations in either JSON or XML format. This API enables users to calculate directions for various modes of transportation. Additionally, it supports multipart directions using a series of waypoints along the route. The Directions API prioritises efficiency when calculating routes, taking into account travel elements like travel time, distance, and the number of turns to provide the most optimal and accurate directions to users. In our project, we leverage the Directions API to create routes from Changi Airport Terminal 3 to different hotels with the use of various algorithms.

The Places API [10] is a versatile service that accepts HTTP requests to retrieve location data through various methods. It provides formatted information and imagery related to establishments, geographic locations, and notable points of interest. It offers location-aware features that grant easy access to detailed location data for users. To interact with the Places API, developers can send requests in standard URL format, specifying the service endpoint, such as /place or /photo. The API accepts requests in either JSON or XML format. Specific parameters relevant to the chosen endpoint can be included in the service request to retrieve refined and relevant data for application use. In BusPathFinder, the Places API is used to extract data on ERPs. These ERPs data are then integrated into the driving routes to consider cost-effectiveness during navigation. By incorporating ERP information, our system can provide users with route options that optimise both travel time and cost, taking into account ERP charges along the way.

3.2 Dijkstra's Algorithm

Dijkstra's algorithm is a well-known approach for finding the shortest path between two nodes in a graph. Dijkstra's algorithm can be utilised to determine the best path between the hotels in our bus travel operation. This can be helpful for a number of reasons such as planning bus routes, optimising travel distance, and minimising Electronic Road Pricing (ERP) gantries. The time complexity of Dijkstra's algorithm is O(V^2) where V is the number of hotels on the map. Therefore, it is an effective technique to calculate the most cost-effective and quickest routes in general.

3.3 Bellman-Ford's Algorithm

The Bellman-Ford algorithm is another algorithm for finding the shortest path between a source node and all other nodes. Unlike Dijkstra's algorithm which is only able to account for positive weights between nodes, the Bellman-Ford algorithm takes into account negative weights, but at the cost of added time complexity of O(V * E), where E is the number of edges in the map that link to the hotels. The Bellman-Ford algorithm also potentially risks a state called Negative Cycle where the cost is forever changing when performing edge relaxation causing a scenario similar to feedback loop, where the algorithm would run forever. Bellman-Ford's algorithm can be utilised to determine the shortest path between hotels for our scenario of bus travel operation. While slower than Dijkstra's algorithm in terms of time complexity, the ability to find shorter paths taking into account negative weights would result in a faster travel time for our bus travel operation, in turn leading to lower operating costs.

3.4 Travelling Salesperson Problem (Using the Nearest Neighbour Algorithm as the Solution)

The travelling salesperson problem (TSP) is determining the shortest possible route to visit each city once and returning to the beginning of the tour, given a list of cities and the distances between them. One solution that resolves the travelling salesperson problem is known as the travelling salesperson algorithm.

The Nearest Neighbour Algorithm is an efficient heuristic algorithm that finds the shortest tour in a bus travel operator. It operates by developing a route iteratively, starting with one city and then appending the next city to the tour in order to minimise overall travelling distance. The algorithm terminates after all the cities have been appended to the tour. The Nearest Neighbour algorithm can be utilised to plan the shortest and most precise tour in a graph. The time complexity of the Nearest Neighbour algorithm is O(V^2), where V is the number of routes in the map. The Nearest Neighbour Algorithm excels at efficiently finding a relatively short tour in the Travelling Salesperson Problem, making it our choice for bus travel operators looking for practical and time-efficient route planning solutions that do not require exhaustive calculations or negative weight considerations.

3.5 Algorithm Comparison

Table 1 shows the comparison in time complexity of the 3 different algorithms implemented in the project. Because the number of edge E is bigger than the number of hotels V, the time complexity of Dijkstra's algorithm and Travelling Salesperson algorithm is the same, while the time complexity of Bellman-Ford's algorithm is worse. Hence Dijkstra's algorithm and Travelling Salesperson algorithm are preferred in our project.

Table 1. Comparison table

Algorithm	Time complexity
Dijkstra's algorithm	O(V^2)
Bellman-Ford's algorithm	O(V * E)
Travelling salesperson algorithm	O(V^2)

3.6 System Diagram

Figure 1 compiles the overall flowchart of the program. The system starts with:

1. Loading an Excel sheet and reading the data
2. Creating a map object by adding hotel parkers
3. Calculating the distance between hotels by looping all pairs of hotels
4. Finding the shortest path using an algorithm
5. Detecting and counting ERP gantries passed
6. Generating data output in the following items:

- Map.html
- Shortest_path_table.html
- Total_cost.html.

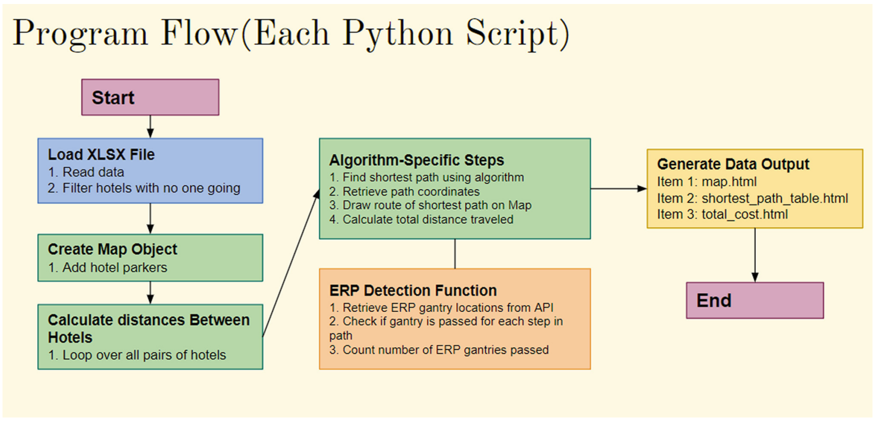

Program Flow(Each Python Script)
Start
Load XLSX File
1. Read data
2. Filter hotels with no one going
Create Map Object
1. Add hotel parkers
Calculate distances Between Hotels
1. Loop over all pairs of hotels
Algorithm-Specific Steps
1. Find shortest path using algorithm
2. Retrieve path coordinates
3. Draw route of shortest path on Map
4. Calculate total distance traveled
ERP Detection Function
1. Retrieve ERP gantry locations from API
2. Check if gantry is passed for each step in path
3. Count number of ERP gantries passed
Generate Data Output
Item 1: map.html
Item 2: shortest_path_table.html
Item 3: total_cost.html
End

Fig. 1. Overall system flow

4 Solution Implementation

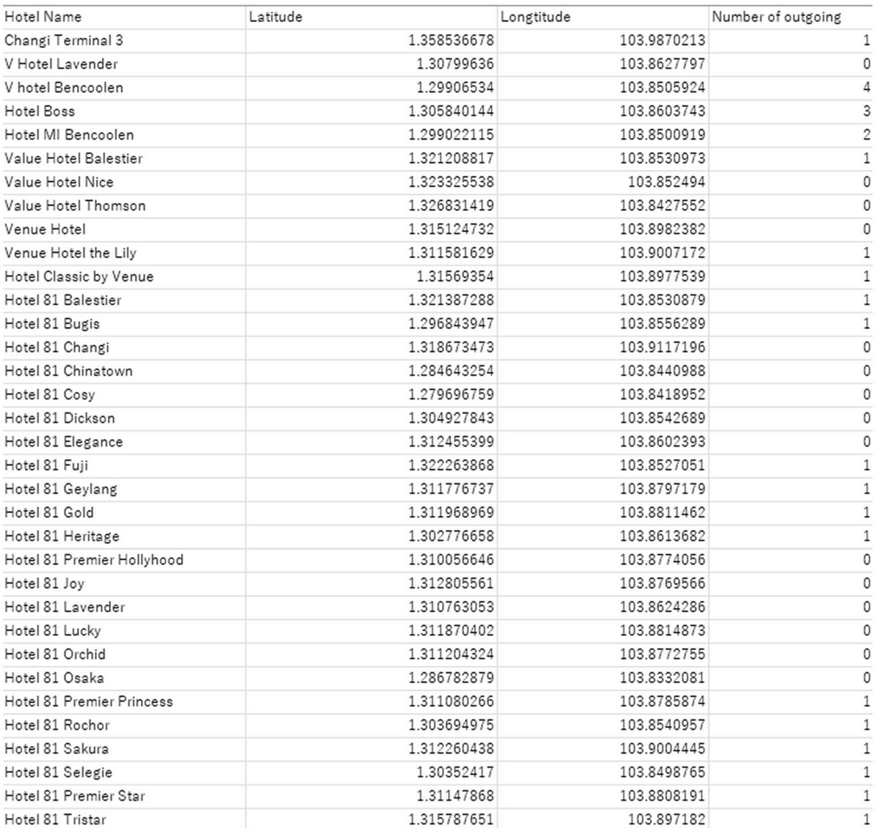
4.1 Dataset Used

In this project, we will be demonstrating a use case scenario where we have a number of customers using our travel bus operating service. The figure shows an Excel file of hotel names, their coordinates, as well as the number of customers going to those hotels (Fig. 2). Those with 0 as the number of outgoing will be excluded from the calculation.

5 Results and Insight

5.1 Execution of the Test Case

Once the number of customers has chosen their respective hotels, the sheet will be updated and run by Google API as well as all the 3 algorithms executed in the main.py execution file. Figure 3 shows the output of the basic Google map and Dijkstra's map which enables the user to see the files being executed properly. It displays the process



Hotel Name	Latitude	Longtitude	Number of outgoing
Changi Terminal 3	1.358536678	103.9870213	1
V Hotel Lavender	1.30799636	103.8627797	0
V hotel Bencoolen	1.29906534	103.8505924	4
Hotel Boss	1.305840144	103.8603743	3
Hotel MI Bencoolen	1.299022115	103.8500919	2
Value Hotel Balestier	1.321208817	103.8530973	1
Value Hotel Nice	1.323325538	103.852494	0
Value Hotel Thomson	1.326831419	103.8427552	0
Venue Hotel	1.315124732	103.8982382	0
Venue Hotel the Lily	1.311581629	103.9007172	1
Hotel Classic by Venue	1.31569354	103.8977539	1
Hotel 81 Balestier	1.321387288	103.8530879	1
Hotel 81 Bugis	1.296843947	103.8556289	1
Hotel 81 Changi	1.318673473	103.9117196	0
Hotel 81 Chinatown	1.284643254	103.8440988	0
Hotel 81 Cosy	1.279696759	103.8418952	0
Hotel 81 Dickson	1.304927843	103.8542689	0
Hotel 81 Elegance	1.312455399	103.8602393	0
Hotel 81 Fuji	1.322263868	103.8527051	1
Hotel 81 Geylang	1.311776737	103.8797179	1
Hotel 81 Gold	1.311968969	103.8811462	1
Hotel 81 Heritage	1.302776658	103.8613682	1
Hotel 81 Premier Hollyhood	1.310056646	103.8774056	0
Hotel 81 Joy	1.312805561	103.8769566	0
Hotel 81 Lavender	1.310763053	103.8624286	0
Hotel 81 Lucky	1.311870402	103.8814873	0
Hotel 81 Orchid	1.311204324	103.8772755	0
Hotel 81 Osaka	1.286782879	103.8332081	0
Hotel 81 Premier Princess	1.311080266	103.8785874	1
Hotel 81 Rochor	1.303694975	103.8540957	1
Hotel 81 Sakura	1.312260438	103.9004445	1
Hotel 81 Selegie	1.30352417	103.8498765	1
Hotel 81 Premier Star	1.31147868	103.8808191	1
Hotel 81 Tristar	1.315787651	103.897182	1

Fig. 2. Excel sheet for input

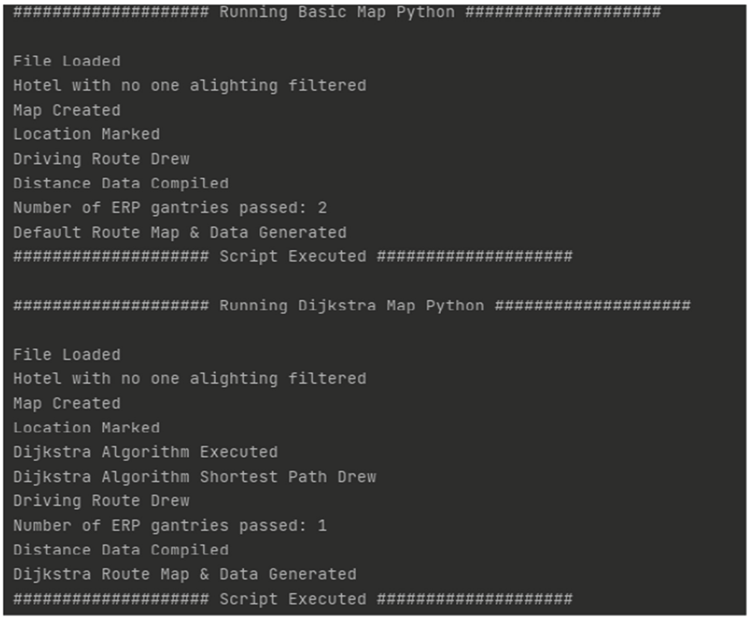
of map creation, marked location, drawing of driving route, the distance data and the number of ERP gantries passed. It is apparent that the difference between both maps is the number of ERP gantries passed during the trip. Figure 4 shows the output of the Bellman-Ford map, Travelling Salesperson map and all the necessary steps required that allow the user to track the progression of the file execution.

5.2 Results

Figure 5 shows the plotted location of various hotels which are our nodes in the map. The red line represents the shortest path route calculated based on the different algorithms while the blue line represents the actual path that the bus operator should be taking that is based on the algorithm.

5.3 Cost Comparison

Table 2 shows the comparison among the 4 algorithms, and it is apparent that both Dijkstra's and Bellman-Ford's algorithm has the least number of ERP gantries passed,



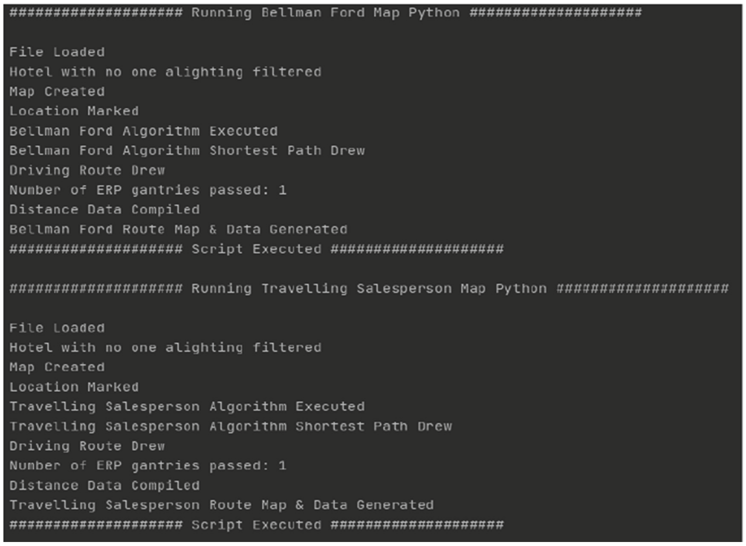
```
#################### Running Basic Map Python ####################

File Loaded
Hotel with no one alighting filtered
Map Created
Location Marked
Driving Route Drew
Distance Data Compiled
Number of ERP gantries passed: 2
Default Route Map & Data Generated
#################### Script Executed ####################

#################### Running Dijkstra Map Python ####################

File Loaded
Hotel with no one alighting filtered
Map Created
Location Marked
Dijkstra Algorithm Executed
Dijkstra Algorithm Shortest Path Drew
Driving Route Drew
Number of ERP gantries passed: 1
Distance Data Compiled
Dijkstra Route Map & Data Generated
#################### Script Executed ####################
```

Fig. 3. Output of basic map and Dijkstra's map



```
#################### Running Bellman Ford Map Python ####################

File Loaded
Hotel with no one alighting filtered
Map Created
Location Marked
Bellman Ford Algorithm Executed
Bellman Ford Algorithm Shortest Path Drew
Driving Route Drew
Number of ERP gantries passed: 1
Distance Data Compiled
Bellman Ford Route Map & Data Generated
#################### Script Executed ####################

#################### Running Travelling Salesperson Map Python ####################

File Loaded
Hotel with no one alighting filtered
Map Created
Location Marked
Travelling Salesperson Algorithm Executed
Travelling Salesperson Algorithm Shortest Path Drew
Driving Route Drew
Number of ERP gantries passed: 1
Distance Data Compiled
Travelling Salesperson Route Map & Data Generated
#################### Script Executed ####################
```

Fig. 4. Output of Bellman-Ford and TSP's algorithm map

the least distance travelled and the lowest cost as compared to Basic Google API and TSP. They are equal because there is no negative weightage in the graph. However, it is highlighted before that the Travelling Sales algorithm has a fully planned route back to

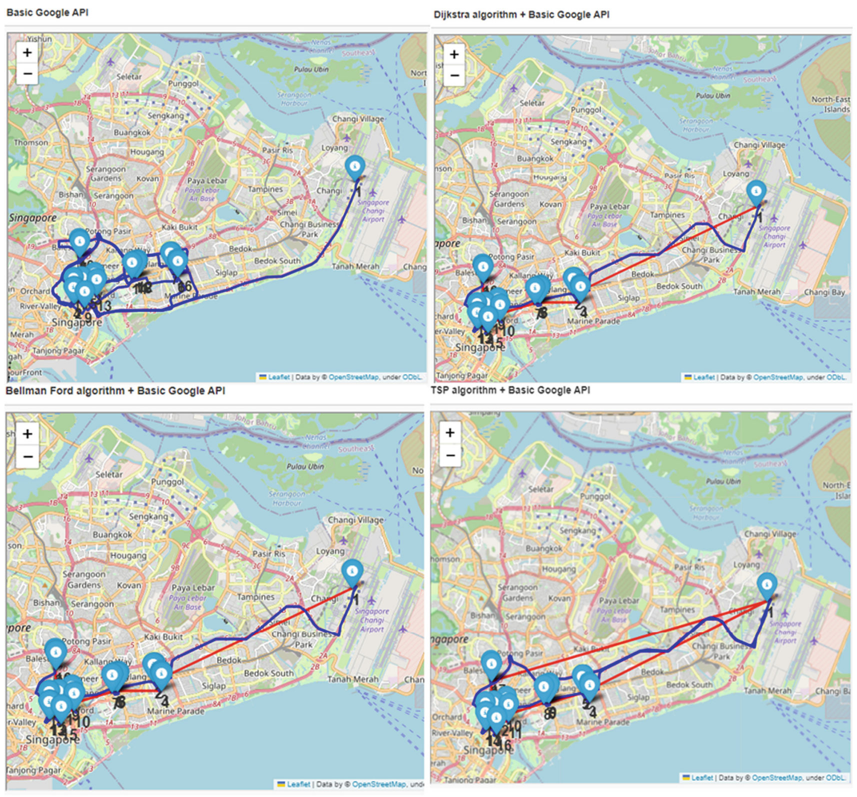

Basic Google API
Dijkstra algorithm + Basic Google API
Bellman Ford algorithm + Basic Google API
TSP algorithm + Basic Google API

Fig. 5. Map for routing path

Changi Terminal 3, whereas both Dijkstra's and Bellman-Ford's algorithms only take the shortest possible route to its last hotel destination.

At first glance, both Dijkstra and Bellman-Ford's algorithms seem to be the most cost-efficient at a low cost of $34.454 but when taking into account of Travelling Salesperson's return trip and removing its return cost of 19833 m resulting in 32448 m overall which is lower than Dijkstra and Bellman-Ford. It is evident that TSP is efficient in addition to taking into account returning to Changi for another round of ferrying customers which will, in turn, result in higher returns and profit for the bus travelling operation, particularly the real profit increases as TSP is designed to bring the cost to the minimum for a round trip and the Bus Operator would be making multiple round trips in a day.

Table 2. Cost calculation table

Algorithms	Google API	Dijkstra's algorithm	Bellman-Ford's algorithm	Travelling salesperson
No. of ERP passed	2	1	1	1
Total distance travelled (m)	104553	32454.0	32454.0	52281
Total cost/km 1$ = 1 km + ERP	$108.553	$34.454	$34.454	$54.28

6 Conclusion

Given the context of this project, it is overall better to use the travelling salesperson algorithm as its advantages surpass those of Dijkstra's algorithm and Bellman-Ford's algorithm. It is a more effective approach to finding the shortest route between various hotel points then back to the Airport. We are able to create our bus routes more efficiently, which can also effectively save our business operating time and money. For example, we can use the Travelling Salesperson algorithm to find the shortest route back and forth from Changi Terminal 3 while evading ERP. This can help to ensure a minimum travelling cost for the bus route.

In conclusion, the Travelling Salesperson algorithm is a more viable algorithm that can be used for our travel bus operating project. We believe that in the future, the potential of this work can not only help in business operations but also provide better services to customers.

References

1. Gavalas, D., Kasapakis, V., Pantziou, G., Konstantopoulos, C., Zaroliagis, C.: Scenic Athens: a personalized scenic route planner for tourists. In: IEEE Symposium on Computers and Communication, Messina, Italy, pp. 1151–1156 (2016)
2. Khamsing, N., Chindaprasert, K., Pitakaso, R., Sirirak, W., Theeraviriya, C.: Modified ALNS algorithm for a processing application of family tourist route planning: a case study of Buriram in Thailand. Computation **9**(2), 23 (2021)
3. CorCystems: Mapping Technology Before Google Maps. https://www.corcystems.com/insights/20-year-tech-evolution-of-maps/#:~:text=MapQuest's%20system%20utilized%20a%20proprietary,into%20a%20%E2%80%9Cweight%E2%80%9D%20value. Accessed 22 July 2023
4. Cao, S.: An optimal round-trip route planning method for tourism based on improved genetic algorithm. Comput. Intell. Neurosci. **2022** (2022)
5. Maposcope: Getting Started. https://maposcope.com/help/. Accessed 22 July 2023
6. Google: How does Waze work? https://support.google.com/waze/answer/6078702?hl=en&ref_topic=9022747&sjid=2281966397362315587-AP. Accessed 23 July 2023
7. Badger Maps: Features that sell more. https://www.badgermapping.com/features/. Accessed 23 July 2023
8. Google Maps: Python Client for Google Maps Services. https://github.com/googlemaps/google-maps-services-python. Accessed 23 July 2023

9. Google Maps Platform: Directions API overview. https://developers.google.com/maps/documentation/directions/overview. Accessed 23 July 2023
10. Google Maps Platform: Overview. https://developers.google.com/maps/documentation/places/web-service/overview. Accessed 23 July 2023