A Case Study of Route Optimisation for Phuket Healthy Drink Delivery System

Tanakorn Karode and Warodom Werapun
College of Computing
Prince of Songkla University
Phuket, Thailand
5835512048@psu.ac.th, warodom.w@psu.ac.th

***Abstract*—This research aims to present the drinking water transportation framework which uses a mobile application to calculate the minimum distance of the transportation to reduce consuming resources and improve routing decision time. The Android application is developed to be able to display positions, traveling routes, approximate total distance and total duration spending for drinking water transportation. This framework provides 2 methods to calculate transportation routes 1) a fast calculating method using Nearest Neighbor algorithm and 2) an exact calculating method using Dynamic Programming algorithm. This research shows the result of testing and comparing processing times between these 2 algorithms to select the most appropriate choice for actual use. The results are illustrated that Dynamic programming always gives the best distance result. However, it spends much time to calculate. Eventually, the experimental results are described that searching the shortest route for 16 places takes about 5 minutes which is too slow for using in the real environment. Although Nearest Neighbor is not provided the minimum distance, the experiment route result is still acceptable by comparing with the experimental result of Dynamic Programming. In addition, it also provides less processing time to get a suitable route.**

Keywords— Logistics, Transportation cost, Android Mobile Application, Algorithm, Nearest Neighbor, Dynamic Programming, Distance, Traveling duration, Execution time

I. INTRODUCTION

The transportation is an important part of the business. A good transportation will lead advantages to a business by reducing costs such as fuel, vehicle, human, time and so on. Moreover, the remaining resources can be used in many other fields. Vehicle routing optimization is a good transportation management that can reduce resource consumption.

According to the increasing number of customers, transportation staffs need to deliver more products to many destinations. Without any tools, it is difficult to know the traveling information such as distance, duration and paths. Google Maps mobile application displays approximate distance, duration and paths of traveling between two points. Unfortunately, it has not provided the traveling information for the tour of many destinations. However, traveling information between two places which is obtained from Google Maps can be used to calculate tour routes by applying the Traveling Salesman Problem (TSP) theory. At present, there are many algorithms to solve TSP but there is no one that can get the exact solution with the least computation time. Therefore, an algorithm is chosen by considering the particular usage and scenario environment.

This paper proposed a framework for solving TSP by developing the Android mobile application. Nearest Neighbor and Dynamic Programming algorithms are used to calculate the traveling route since both algorithms take less development time and they are possible to be an answer for problems. We use real customer address coordinates from Healthy Drink company database in the experiment to identify the most proper algorithm and show that this framework can be serviced in the real-world problem.

II. RELATED THEORY AND WORK

A. Phuket Healthy Drink Company

Phuket Healthy Drink Company provides drinking water delivery for the customers in local scale. This company conducts the technology to apply with management system. Accroding to [1], Phuket Healthy Drink company can reduce many cost consumption by using management technologies. By recording customer locations, there are available data for transportation route calculation which can be used in route planning.

B. Logistics and Milk Runs

Logistics is the method to transport goods, information or resources from one place to another such as from sellers to customers, from manufacturers to sellers, from suppliers to manufacturers and so on. There are 3 types of logistics:

(1) Direct shipment is the direct transportation from source to destination, which will not add or reduce products during the transportation. This method is faster. However, it will cost more. A company must have very good transportation staffs and the distances must not be too far away.

(2) Milk runs [2] is the transportation from one or many sources to many destinations. The customers may prepare the packages of the products which can be reused again. It has advantage of being able to exchange the products immediately such as milk delivery, water delivery, etc. which requires the packages.

(3) Transportation with cross docking is the method to transport the products from source to the product distribution point or product warehouse. Then the products are distributed to nearby customers. It causes the faster product delivery but requires frequently checking warehouse.

Drinking water transportation is similar with Milk runs method because the products are distributed to many customers. In addition, the customers can exchange water bottles. This method aims to reduce cost of management and transportation complexity by selecting the nearby route or the most efficient transportation route. Therefore, it can be scheduled the deliver time to reducing traffic jam and less CO2 producing [3].

C. Traveling Salesman Problem

TSP is one of the problems that have received the attention of researchers continuously. There are many methods which are developed for finding the answers better and faster. This problem decides to find a route when traveling to N places. That route needs to pass through all places and returns back to the starting point. For example, a

978-1-7281-2544-2/19/$31.00 ©2019 IEEE

salesperson travels to sell the products to 10 customers in city A to city J where city D is the starting point. The salesperson starts from city D then travels to all of the other cities and back to city D which is the starting point. An example answer of this problem is D-A-J-B-I-C-H-G-E-F-D which is a sequence of cities that the salesperson traveled to. The goals of solving this problem is to find a route that consumes the least resources which might be distances, duration, fuel and so on. Moreover, the problem is solved with the least computation time.

During 1930s, the Viennese Karl Menger developed a method of traveling from the nearest city or Nearest Neighbor Heuristics [4]. During the years 1950 - 1960, TSP received a lot of attention. The mathematical models and the exact methods were designed for solving the problem begins with [5] which was able to solve the problem that includes 49 cities and got the optimum solution. From [6] proved that the Hamitonian cycle game is the NP-Complete problem which causes the TSP to be the same problem as NP-Hardness. It showed that TSP is a difficult problem and there is no method that takes polynomial time to solve the problem. Therefore, many researchers try to find solutions that can solve the problem quickly with the best answers at the same time.

D. Algorithms for solving problem

Algorithm 1: Nearest Neighbor

Input: An n x n distance matrix D[1…n, 1…n].
Output: A list of the vertices which represents to a route.

```
1: Procedure Nearest Neighbor Algorithm
2:    for i <- 1 to n do
3:        Visited [i] <- false
4:    Initialize the list Path with 1
5:    Visited[1] <- true
6:    Current <- 1
7:    for I <- 2 to n do
8:        Find the lowest element in current row and
          unvisited   column j containing the element.
9:        Current <- j
10:       Visited[j] <- true
11:       Add j to the end of list Path
12:   Add 1 to the end of list Path
13:   return Path
```

Nearest Neighbor algorithm is the method to choose the least distance city from current city which consists of several steps that are shown in Algorithm 1.

From the iteration pattern which finds the minimum distance in 2D matrix causes the time complexity to be $O(n^2)$.

Algorithm2: Dynamic Programming

Input: An n x n distance matrix D[1…n, 1…n]
Output: A list of the vertices which represents to a route.

```
1: Procedure Dynamic Programming Algorithm
2:    C ({1},  0 = (1
3:    for s = 2 to n do
4:       for all subsets S Є {1, 2, 3, …, n} of size s and
           containing  1
5:          C (S,  ∞ = (1
6:          for all j Є S and j ≠  1
7:                   C (S, j) = min {C (S – {j}, i) + d(i, j) for
                     i Є S and i ≠ j}
8:    return min j C ({1, 2, 3, …, n}, j) + d(j, i)
```

Dynamic Programming algorithm [8] has steps according to the Algorithm 2 as followings:

(1) Define 1 to be the starting point. Generate all possible subsets from the cities exclude the starting point. The subsets refer to a path that has already passed, for example {}, {2}, {3}, {4}, {2,3}, {2,4}, {3,4}, {2,3,4} (when there are 3 destinations). The set {2,3,4} means the path that has already passed city 2, 3 and 4.

(2) Find the minimum distance route from every city in each subset (except city 1) without repeating the same route. The minimum distance of a subset is represented with C(S, i) , where S is the subset of passed cities and i is current city. For example, C({2,3}, 4) means the distance from a route of city 2 and 3 comes to city 4 which can be written as following equations.

$$C(S, i) = d(\{1\}, i) \quad (1)$$

When there are less than 2 members in subset and d({1}, i) is the distance from starting point to city i.

$$C(S, i) = min\{ C(S-\{i\}, j) + d(j,i) \} \quad (2)$$

When j is in subset S, j is not equal to i and j is not 1.

(3) In every calculation of C(S, i), remember the last city into set P. For example, from C({2,3},4), assume that city 2 is the last city from the subset S, city 2 is remembered in set P.

(4) The calculation will finish when there is a subset that has equal member as number of destinations. That subset will be the minimum distance route. For example, when there are 3 destinations, C({2,3,4},1) will be the minimum distance of full round traveling.

(5) Find the sequence of traveling by using set P.

From the above steps, there is at least $O(2n^2)$ sub-problems. Each sub-problem has a linear time to compute. As a result, time complexity of Dynamic Programming algorithm is $O(n^2 2^n)$.

III. Propose the Framework

This section presents the scope of research, system architecture, framework usage and testing methodology respectively.

A. Scope of Research

This research aims to present the framework for cost reduction from transportation by calculating the minimum distance route. Every destination is the real customer addresses which are obtained from Phuket Healthy Drink Company database. The driver delivers drinking water units to approximately 30 customers in each round.

B. System Architecture

The system is implemented by using Android mobile application. The system architecture has been illustrated as follows:

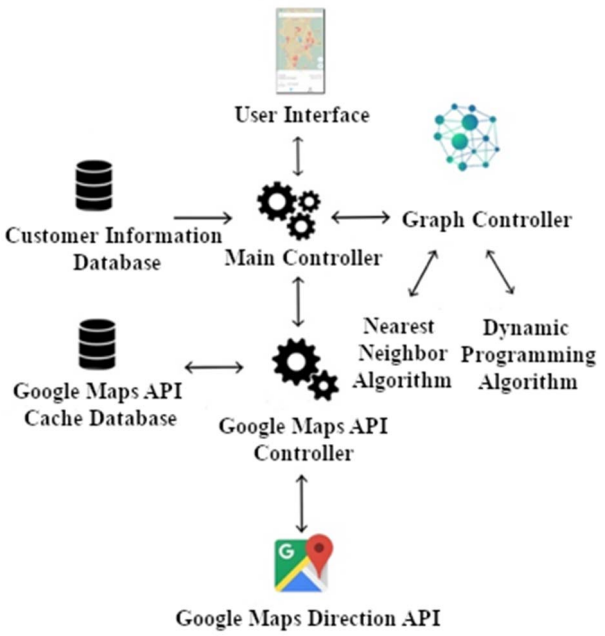

Fig. 1. System architecture of mobile application

Fig. 1 represents to the structure of system that is used for testing. The components are described as follows:

(1) User Interface is responsible for interaction with users. User command will be forwarded to the main controller and sent to the other sections in order to perform processing procedure. The results of those processing will be sent back from the controller and displayed in User Interface.

(2) Main controller is the center system that connects all sections together. It manages to send tasks to the other sections. The results of processing will be sent to the main controller before displaying in User Interface.

(3) Customer information database stores customer name, customer address and set of coordinates (latitude, longitude). Within the system, the customer data is collected in JSON (JavaScript Object Notation) format that the main controller can fetch and transform the data to be able to use in application. The data will be shown to users for selecting the places to travel in each round.

(4) Graph controller handles graph calculation from collected data. The connected cities will be stored and represented in a form of graph which can get the distance for every pair of cities easily. When users select any destinations, graph controller will store the relations between each of nodes. There are 3 types of the relations between nodes: traveling distance, traveling duration and polyline codes which can be used to draw the path between nodes on User Interface. An important task of graph controller is to send the graph to calculate the minimum distance routes. This framework provides 2 well known algorithms to calculate the minimum distance routes:

a. Nearest Neighbor algorithm is represented in form of fast calculation since the time complexity is $O(n^2)$ which is considered to be the rapid calculation for local transportation with approximately 30 destinations. This algorithm is classified in heuristics type of algorithms because the result of this algorithm is not always the optimum solution. However, it can be used in the aspect of practical using.

b. Dynamic Programming algorithm is represented in form of exact calculation because the result of calculation will always be the optimum solution. In contrast, a main problem of this algorithm is slow execution time with $O(n^2 2^n)$ time complexity which is not suit for many destinations.

(5) Google Maps API controller is main section that manages data which contains traveling distance, traveling duration and polyline codes for pair of cities. At first, Google Maps API controller looks up the data in cache database. If data exist, Google Maps API will send them to the main controller. Otherwise, Google maps API fetches data from Google Maps Directions API. After that, data will be saved in Google Maps API cache database and sent to the main controller.

(6) Google Map API cache database is a storage for saving Google Maps Directions API fetching result. This section is used to reduce Google Maps Direction API connection rate. Reducing the connection rate improves operation speed and reduces Google Maps API service fee.

(7) Google Maps Direction API is a service provided by Google. This service is used to find the distance and duration for traveling between two points. In addition, it provides a polyline code that can displays traveling routes.

C. Using Framework

Android mobile phone, Internet and GPS is required for using this system. The following steps is guideline for using the framework:

(1) Open the application and allow location permissions.

(2) Select and add customer coordinates to list of destinations. The application will calculate a route for traveling from a starting point to all of the other places and back to a starting point.

(3) There are two calculation methods. The fast execution by using Nearest Neighbor algorithm and the exact calculation by using Dynamic Programming algorithm. A user can select a method by tapping the algorithm icon at the bottom of the screen.

(4) Total distance, duration and traveling order will be shown at the bottom of the screen. There will be numbers displayed on the map pins as shown in Fig.

2. A user can travel by following these pins in order to follow the calculated routes.

(5) A user can change the order of traveling by drag and drop the places at the bottom of the screen. The distance, duration and routes will be re-calculated respect to the new place order.

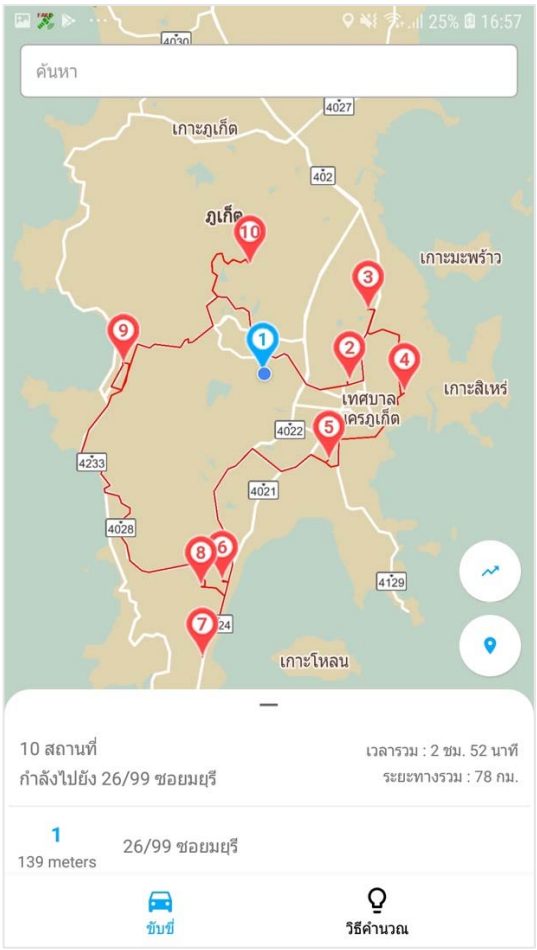

Fig. 2. Application User Interface

D. Testing Methodology

System testing procedure focuses on measuring algorithm performance in the aspect of practical using. The detail of testing will be described below.

(1) Testing equipment is Samsung Galaxy C9 Pro with Qualcomm Snapdragon 653 Octa Core 1.95 GHz processing unit and 6 GB RAM.

(2) The testing is performed by selecting destinations and recording calculation results which consist of traveling distance, traveling duration and execution time.

(3) The execution time is collected by recording the beginning and finishing time of calculation. Then find the difference between them.

(4) As the execution time is vary, we performed 10 times for the same destination routes and calculated their average values.

(5) To clearly identify the worst case of Nearest Neighbor algorithm (respect for real customer coordinates), we performed 10 different routes in the experiment.

IV. EXPERIMENT RESULTS

As mentioned in testing methodology, we performed 10 different routes in this experiment. We knew that Dynamic Programming always provides the minimum distance routes. Hence, we select the most different distance result between both algorithms to show in the following charts.

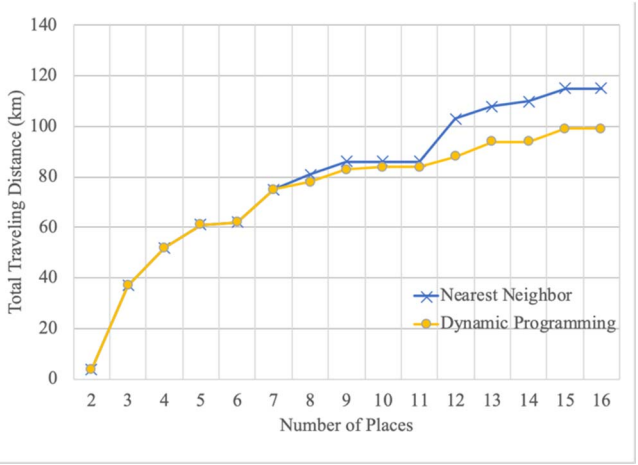

Fig. 3. Total distance chart

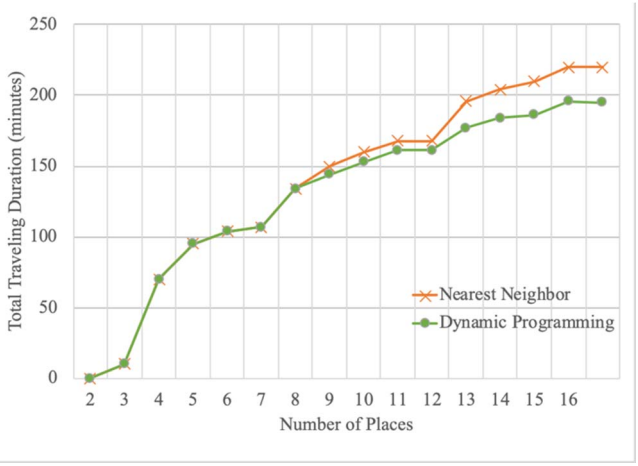

Fig. 4. Total traveling duration chart

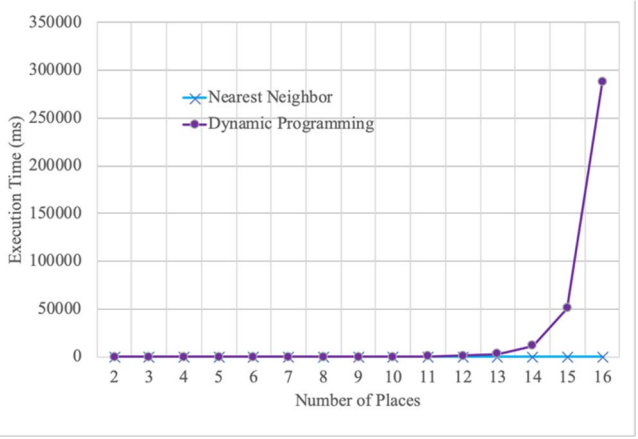

Fig. 5. Average execution time chart

From the experiment results show that both algorithms provide the same total distance in 1 – 7 destinations.

Increasing the number of destinations causes the difference between total distance results to gradually increase as shown in Fig. 3. In addition, the total traveling duration is likely the same characteristics as shown in Fig. 4.

From Fig. 5, the total execution time is the same for calculating for 1 – 13 destinations. Increasing to 13 destinations, the execution time of Dynamic Programming increases gradually. After increasing more destinations, it will increase dramatically. We cannot perform more than 16 destinations testing because the execution time of Dynamic Programming is too long.

According to the experiment results, we can identify the advantage and disadvantage of both algorithms. Dynamic Programming always provides the same and better results. In contrast, the execution time increases quickly. For example, the execution time of calculation for 16 destinations is around 5 minutes and more than 20 minutes for 17 destinations. With too long execution time, Dynamic Programming is considered to be an unsuitable method for this kind of usage. On the other hands, Nearest Neighbor algorithm can solve the problems much quicker. Even it does not always provide the optimum solutions, but its results are not much different from the best results. Therefore, Nearest Neighbor is the better method for this scenario.

V. CONCLUSION AND FUTURE WORK

The results of this study indicated that an exact algorithm with $O(n^2 2^n)$ time complexity is not suitable for using with routing optimization in a mobile application. In contrast, Nearest Neighbor algorithm which finds the answer in greedy way can provide the feasible results since the service area is not much diffuse. There are many other new algorithms which are able to solve TSP well such as Genetic Algorithm (GA) [9], Particle Swarm Optimization (PSO) [10], Swallow Swarm Optimization (SSO) [11], Firefly and K-means Clustering [12]. In future work, we will test those new algorithms for the purpose of best using in real environment.

The one completeness of this research is this system can display the overall traveling information for each round of transportation. A driver or a manager can use the application to plan the transportation route easier. Although the minimum distance routes are required for least cost consumption, but sometimes there are many factors that a driver must re-order the routes manually such as more priority delivery, traffic jam, road accident, etc. The developed system is available to re-order destinations manually with the ability to show new calculated results. With the proposed system, Healthy drink company can improve their transportation management which reduces 20% transportation cost for each month [1].

ACKNOWLEDGMENT

This work is a part of Thailand Research Fund (Industry) research (No: RDG6050023). The authors would like to thank you for the financial support. Thanks to College of Computing, Prince of Songkla University, Phuket Campus.

REFERENCES

[1] W. Warodom, S. Wisit, C. Kullawat, K. Tanakorn, “Smart Logistics Framework: A Case Study of Phuket RO Water System”, 2018.

[2] M. L. M. Theeratham, “Vehical routing in milk run operations: A column generation based approach”, .2010

[3] G. S. B. a. G. Saini, “Milk run logisitcs: Literature review and directions”, .2014

[4] David S. Johnson, “Local optimization and the Traveling Salesman Problem”, 2005.

[5] G. , D. F. a. S. J. Dantzig, “Solution of a Large Scale Travelling-Salesman Problem”, .1954

[6] M. H. M. Karp, “The traveling-salesman problem and minimum spanning trees: Part II,” 1971.

[7] C.A.J. Hurkens, G.J. Woeginger, OperationsResearch Letters, 32, 1-4, 2004

[8] M. Held, R. M. Karp, “A Dynamic Programming Approach to Sequencing Problems”, 1962.

[9] J. Liu, W. Li, “Greedy Permuting Method for Genetic Algorithm on Traveling Salesman Problem”, 2018

[10] S. Yuji, K. Takaaki, M. Tatsuya, “An application of particle swarm optimization to traveling salesman problem”, 2010.

[11] B. Safaa, E. R. Mohammed, “Discrete swallow swarm optimization algorithm for travelling salesman problem”, 2017.

[12] A. Jaradat, B. Matalkeh, “Solving Traveling Salesman Problem using Firefly and K-means Clustering”, 2019.