Matching Algorithms in Ridesharing Problem

An T. Dao, An G. To, Long NT. Le and Khoi N. Bui
Advanced Program of Computer Science
Falcuty of Information Technology
University of Science, VNU-HCMC
Email: {dtan,tgan,lntlong,bnkhoi}@apcs.vn

***Abstract*—The great number of vehicles causes traffic congestion, wastes of energy and air pollution. Because of differences between the number of vehicles and the real necessary traffic demand in a negative way, people propose solutions of how to manage not only to control the number of vehicles but also to solve daily traveling problems. This fact gives the authors ideas to propose a matching algorithm to match people's daily routes together to significantly decrease the number of available vehicles, which may lead to reduce air pollutants, and slow down the speed of climate change. Within a simulated map of areas in real geographical map of Ho Chi Minh City, the authors conduct 4 experiments which take advantages from Dijkstra's, Floyd-Warshall and Hungarian algorithms. The authors compare results from experiments and conclude the suitable situation for each experiment. Our experiments can be integrated to a mobile device's map application to recommend people give a hand to preserve the environment.**

I. Introduction

Traffic congestion affects both human beings and global environment. Even from 2000, according to Texas Transportation Institute, vehicles in the 75 largest urban areas in the world created the number of over 3 billion hours of delay, which caused damages not only with 5.7 billion gallons fuel wasted but also with $67.5 billion of productivity lost [1]. Traffic congestion occurs whenever there are more vehicles than capable space of roads [1].

The number of vehicles on roads everyday is one of the main reasons for traffic congestion [2]. Therefore, it is necessary to reduce the number of vehicles travel daily. With an idea about ***ridesharing*** [3], the number of cars on the roads can be reduced then the quantities of pollutants from automobiles or motorbikes [4].

The idea of ridesharing has been conducted and became business like Uber [5]. The ridesharing service of Uber called Uberpool. Uberpool matches customers (passengers) with riders who have the same direction. Another ridesharing company is Lyft. According to its website http://lyft.com, Lyft trips are focus on short trips within cities.

Although Uber and other ridesharing services are popular in Europe, they are not well-known in Viet Nam [6]. A potential application named PinBike is developed in the late 2015 in Viet Nam with the similar ridesharing idea. Not only in a developing country like Viet Nam but also all around the world, ridesharing is a judicious approach. To develop fundamental foundations for an application based on the ridesharing idea, the authors build and conduct experiments of route matching algorithms in ridesharing problems.

To create an environment for our matching algorithms, one of approval approaches is a model of a Graph problem and matching pairs in this Graph problem. The authors choose to implement matching algorithms and they take advantages from available algorithms called Dijkstra [7], Floyd-Warshall [8] [9] and Hungarian [11] algorithms. The condition is intersections and roads are represented as vertices and edges respectively in Graph. Dijkstra algorithm is used to determine the shortest paths from one node to other nodes in the Graph [7]. Floyd-Warshall algorithm can calculate not only shortest paths from one node as Dijkstra algorithm but also shortest paths from any node to any node in Graph [8] [9]. **Hungarian** algorithm is used to solve the assignment problem or matching problem in polynomial time. [10].

To evaluate the reality of applying ideas about Graph Theory in ridesharing problems, the authors build and conduct 3 experiments of route matching algorithms based on simultaneously scenarios with real geographical map. The authors prepare 3 kinds of data files:

1) Details of the area (starting points, destination points, distances between two crossroads, .etc).
2) Offer file.
3) Demand files.

The authors proceed 4 experiments on the real maps of Ward 3 and 4 of District 5 in Ho Chi Minh City, Viet Nam and compare results from these experiments. From comparisons, the algorithm of maximize the number of matching pairs achieves the highest efficiency but it takes longer time than others. Meanwhile, the more times the algorithm of random selections in picking order of offers runs, the better it results are.

The rest of this paper is organized as follows. In section II, we review the background and related works in ridesharing. The main processes of our matching algorithm are presented in section III. Section IV is for experiment results and evaluations. The conclusion and future works are presented in section V.

II. Background & Related Works

A. Background

Ridesharing has another name as carpooling, is used to define the connections between drivers and riders to share vehicles in traveling, to reduce costs, fuel, and roads capacity [4]. This type of traveling take advantages of some technological advances [12]:

- GPS navigation devices to specify locations precisely.
- Smartphones for a user to provide their offers or demands.

The early ridesharing projects begin in the 1990s and faces lots of obstacles which cannot be solved since the lack of user network, suitable communication way, and system [13]. But those shortcomings are fixed from time to time and there are some remarkable work, i.e, Real-Time City-Scale Taxi Ridesharing from University of Illinois at Chicago, USA [14], an application named The Ride is released in the early 2015 in Canada [15].

The ridesharing service of Uber, which is mentioned in section I, called Uberpool. Uberpool matches customers (passengers) with the riders who have the same direction. Lyft — Uber competitor — launches as an on-demand ridesharing network. Lyft trips is focus on short trips within cities.

However, these applications: Yatrashare, FillCar, Uber, Lyft and Haxi are considered as another form of Taxi services because drivers do not share a destination with their passengers. Simply, these applications outsources rides to commercial drivers. These applications just help transfer the process of calling taxi into mobile application. Therefore, these services are contradict the original idea of Ride-Sharing. Transportation experts have called these services "ridesourcing" [16].

As mentioned in Section I, in Viet Nam, Uber and other ridesharing services is not as popular as in Europe. We are not as open with strangers as Europeans are. Says Nam Nguyen, founder and chief executive of Di Chung, a Hanoi-based ride-sharing website [6]. Many Vietnamese people consider private vehicles like cars and motorbikes are important asset. Many Vietnamese also consider car as a source of pride [6]. The amount of car or motorbike robbery is increasing as well as their advancing technique [17]. Therefore, in order to encourage Vietnamese in sharing their private vehicles is quite difficult. The most important thing in this problem is constructing the customer trust to share their car or motorbike.

B. Related Work

Inside the ridesharing problem, matching algorithm is one of the primary factors. According to NRMP (National Resident Matching Program), which has the 2012 Nobel prize of Economic Sciences for their research on the algorithm [23], matching algorithm is to place applicants and proper fellowship into groups, and give the best outcome in expected order based on the relativity to the applicants [24] [25].

The ridesharing problem of creating schedules for vehicles to deliver passengers can be solved by auction approach. There are several rounds in this auction. In these rounds, each passenger places their bids for the vehicle at the lowest additional cost instead of the highest one. For each vehicle, the passenger that bids the lowest cost and accept to be transported is the winner. This passenger is added to the schedule of the vehicle. In the case that multiple passengers together place bids on a single vehicle, the vehicle with the lowest cost wins. Rounds of the auction continue until all passengers get their vehicles.

However, creating a large scale real-time ridesharing service is a hard problem. With a large scale like city scale, the problem becomes non-trivial problem. The main difficulty is inventing a real-time matching algorithm that can determine the best driver for the incoming passengers requests.

One of the old approaches is the Dial-a-Ride Transit (DART) or Demand Responsive Transport which is defined as "an advanced, user-oriented form of public transport characterised by flexible routing and scheduling of small/medium vehicles operating in shared-ride mode between pick-up and drop-off locations according to passengers needs" [26]. DRT systems are appropriate for the areas of low passenger demand or low travel demand periods [27]. Although, DRT is used in low passenger demand, DRT can provide coverage effectively [28]. This approach using Branch and bound or mixed integer programing are not efficient in processing large scale situations. Branch and bound (BB or B&B) is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as general real valued problems. In 1960, A. H. Land and A. G. Doig first proposed BB method for discrete programming [29]. Branch and bound becomes the most commonly used tool for solving NP-hard optimization problems [30].

Furthermore, almost recent solutions concentrate in scenarios where requests are known ahead of time. In this paper, we also do the same, focus on the scenarios, which are routines in our situation. Because the authors build a solution to matching routine routes in daily life, our algorithm is no need to be dynamic, which means can process requests in real-time or near-time.

The first appearance of the algorithm was discovered from 1600 BC by a Babylonians [18]. Until now, there are a lot of algorithms that exist in the world. In 1959, Dijkstra's algorithm was first published by E. Dijkstra [7] and became one of the fastest single source shortest path algorithms [19]. This algorithm plays an important role in the efficiency of the Internet as well as finding the shortest path in a graph [18]. Today, although there have been more advanced algorithms but Dijkstra's algorithm remains and exists by its stability [18].

Another algorithm for the same purpose is Floyd-Warshall's algorithm. If Dijkstra's algorithm finds the shortest path from one vertex to all other vertices given in the graph, the Floyd-Warshall will find the shortest path between any vertices after one execution Floyd-Warshall's algorithm excels by the simplicity and highly effective in a network or graph [8] [9].

The Hungarian algorithm is developed and published by Harold Kuhn in 1955 [11]. This receptive algorithm solves the assignment problem which is one of the fundamental combinatorial optimization problems like minimizing the total cost or total weight of a graph, etc [11]. The time complexity of the original algorithm is $O(n^4)$ but with some improvements and difficulties in implementation, it could reach $O(n^3)$ [22].

III. Our Proposed Method

A ridesharing problem can be described as a set of offers O, another set of passengers P, and map (V, E) with V is the

set of vertices, $|V|$ is the number of vertices, E is the set of edges and $|E|$ is the number of edges.

Assume that the set of offers has N offers. Each offer has its starting location denoted by $so \in V$ and destination location denoted by $so \in V$.

There are M demands in set P. Each demand has its starting location, denoted by $sd \in V$ and destination denoted by $ed \in V$.

Suppose there is an offer with with the starting point A and the ending point B and a demand wants to go from C to D. Their routes are briefly described on Figure 1.

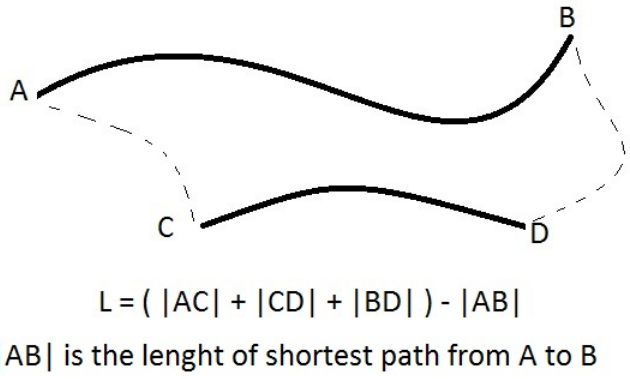

Fig. 1. The extra distance L for an offer when he or she shares his or her vehicles

The authors use the parameter L to measure the extra distance the offer travels when he or she picks up the demand instead of traveling alone.

Brief description of a term Matched for the assigned offer and demand is when the total extra distance to travel of that offer for that demand is less than or equal to the addition of dR (total length of offer routes) and maximum of L. Following is the equation of that:

$$L + dR \leq L_{max} + dR$$
$$\Leftrightarrow L \leq L_{max}$$

In this proposal, the authors set L_{max} is 2km since it is the acceptable constraint for extra distance.

The author proceed two type of experiments:

1) Type 1: $N = 1$, $M = 30$
   The authors build with only one offer and figure out which demand is matched.
2) Type 2: $N >= 30$, $M >= 30$ The authors build with a list of offers and another list of demands.

A. Method Type 1

Because the number of offers is 1, we use the Dijkstra algorithm to find results of function L. We arrange the results of matching pairs in ascending order of function L.

B. Method Type 2

- Because the number of offers is greater than one in experiment 1 ($N >= 30$), we use the Floyd-Warshall algorithm to find $M * N$ results of function L.
- The authors build a bipartite graph [31] contains two set of vertices with a set O of offers and a set P of demands. The edge IJ has its weight
  $L_{ij} = \text{Min}V(sp_j, R_i) + \text{Min}V(ep_j; R_i)$
  $(i : 1 \rightarrow N; j : 1 \rightarrow M)$
- The result of a test is based on the sum of L in that test, the smaller L is, the better the result is. Then, the offers do not need to travel a long distance to pick up the demands.

The authors conduct 3 experiments based on 3 methods to minimize sum of L:

1) From the list of offers, each offer is considered in the order from top to bottom. After an offer matches with a demand from the list of demand - the demand has been chosen is the demand which have L(offer,demand) is minimum in the list of demand of that offer - this demand is removed from the list. Thus, the rest of the list of offers only has chance to match with the rest of one of demands.
2) An offer is randomly picked from the list of offers and the result is released corresponding to a demand from the list of demands - how to choose that demand is the same as the previous method. After that, a random selection is made again and a random offer is considered. This process repeats till there is no offer or demand remaining in the lists. A time random generate a result. The authors do several randoms to get the better result by take the minimum sum of L. The more times of random, the better the result is. If the times of random is big enough, we can find the best result - like the third method.
3) The authors use *Hungarian algorithm* to find the best result - the result have the minimum sum of L.

IV. Experiment & Results

In this section, the authors conduct four experiments to compare their results and figure out which method is the most suitable for which ridesharing situations. There are three steps the author follow in the whole process of experiments. These steps is shown in Figure 2.

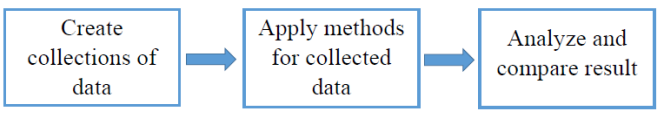

Fig. 2. Steps in experiments

A. Collect Data

To conduct experiments and evaluate the efficiency between 4 experiment approaches, the authors prepare information about maps, collection of offers and collection of demands.

- First: Map
  The authors want to apply the ridesharing idea on a crowded area in real map and District 5 in Ho Chi Minh City is a good option. In 2011, District 5 is in 3rd rank of population density in Ho Chi Minh City [32]. In addition, there are 4 million motorbikes in 2012 in Ho Chi Minh City and this number do not stop growing [33]. Then, the motorbikes density in District 5 must be great and considered as crowded.
  Thus, the authors take advantages of accuracy from Google Maps and reuse information from that to build the map of District 5, Ho Chi Minh City, Viet Nam in graph. Intersects on streets are considered as vertices. Streets between every two-intersect are edges and weight of each edge is length of this street. That graph is saved in text file with the format: the number of vertices, the number of edges and each edge is format to 2 vertices and distance between these vertices.
- Second: Collection of Offers
  The authors randomize routes of offers. We create two files: one has only one offer and another has 30 or more offer routes corresponding to Method Type 1 and Type 2 respectively. Offers have to set their starting and ending point in their routes and the authors find the shortest paths for them. Each route is a path on the graph and formatted series of vertices on that route.
- Third: Collection of Demands
  The authors randomize routes of demands. There is only one file shared by both two experiments. Our file has 30 or more demand routes. Demands only have to set their starting and destination points.

B. Process

The authors conduct four experiments in C++ environment on a Toshiba laptop with the processor of 2.3 GHz Intel Core i5 dual core and 6GB RAM: experiment 1 works with only one offer, whereas others use a list of 30 offers in different ways. The common thing of these experiment is that they share the same list of demands and the map file.

To speed up matching algorithms in experiment 2, 3 and 4, the authors apply Floyd-Warshall algorithm and achieves the shortest paths between every pairs in the whole graph.

1) Experiment 1: The goal of this experiment is to find which demand could be best assigned to the offer. This experiment is important because it is applied in later experiments. The authors use Dijkstra's algorithm to calculate outcomes which are sorted with the parameter L in ascending order. The best assigned demand is the one has the minimum distance L to the offer.

2) Experiment 2: The goal of this experiment is to maximize the number of matching pairs. Which means to find more and more matching offer-demand pairs. The ideal result is minimum total function L. The less total L is , the more efficient the ridesharing algorithm is. All matching pairs satisfied the definition of matched in experiment 1. Thus, all matching pairs have $L \leq L_{max} = 3$km.

3) Experiment 3: In this experiment, the authors use a C++ program to get the result list of matched pairs. The program uses data of routes of offers and demands in the given text files.

- The program is based on a old style of thought. It takes the first offer from the top of the list, then apply experiment 1 to find the corresponding demand of this offer - the demand has result of function L to the offer is minimum. After that, both of them are removed out of their list. This progress is repeated till there is no offer or demand in lists.
- After complete the whole execution, the program releases a text file and a number of total L.
- The text file contains the information of offer-demand pairs and the number of matched pairs. The authors use these information to compare to the results of experiment 2 and experiment 4.
- The total distance L_3 is the sum of the distance L shows the difference in routes of all offer-demand pairs.
- The execution time of the program is recorded for comparisons. A good program need to optimize its both execution time and results.

Experiment 3 runs 30 times with different lists of offers. It is enough to run 30 times to show the importance of the ordering in the inputed list of offers. Because the program in this experiment execute from the top of the list to the bottom, then results depend on how the list of offers arranges. Thus, an offer does not always matches with its best demands.

4) Experiment 4: The authors run a C++ program as experiment 3 with the same lists of offers and demands in experiment 2. However, the experiment 4 runs with a random list of offers every time it runs, whereas the experiment 3 runs with an unchanged list of offers if the authors change nothing with the input data.

- The program chooses an offer randomly from the list of offer. Then, the program applies Method Type 1 to find out the best demand for this offer. After that, both of them are recorded like in the experiment 3. The program still picks randomly till the end of the list of offers or demands.
- From the distance L of every offer-demand pair, the program calculates the sum of them called the total distance L_4.
- The program releases a text file with the total number of matched pairs, the information of all offer-demand pairs and the distance L_4.
- The authors record execution time in every time the experiment 4 runs. Then, we calculate the average time of the whole process and use it to compare to other experiments.

The authors run 30 times the experiment 4. There are 30 or more offers in our list and with the number of repeated times which is 30 all offers have a chance to be the first pick of the program. Then, they share the same probability to match with their best demands.

C. Result

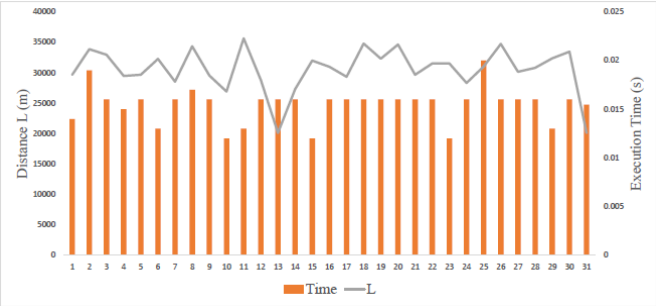

Fig. 3. The Result of Experiment 4 with 30 offers and 30 demands

The Result of Experiment 4 with 30 offers and 30 demands is shown in Figure 3 The Orange column shows the running time. The gray line shows the result of total L in each random time. The average running time is 0.015467 second. The best result can be found after 30 times random is 20108.

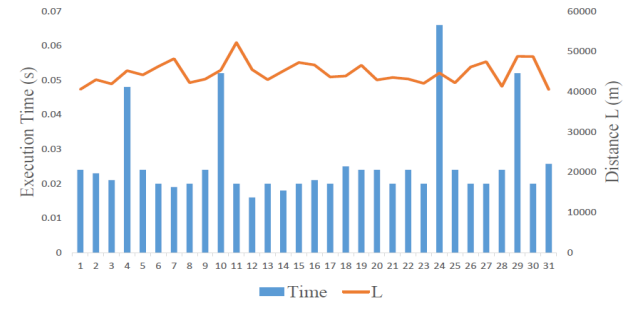

Fig. 4. The Result of Experiment 4 with 50 offers and 50 demands

The Result of Experiment 4 with 50 offers and 50 demands is shown in Figure 4 The Orange column shows the running time. The gray line shows the result of total L in each random time. The average running time is 0.025767 second. The best result can be found after 30 times random is 40579.

The Result of Experiment 4 with 100 offers and 100 demands is shown in Figure 5 The Orange column shows the running time. The grey line shows the result of total L in each random time. The average running time is 0.0691 second. The best result can be found after 30 times random is 65251.

The Result of Experiment 4 with 300 offers and 300 demands is shown in Figure 6 The Orange column shows the running time. The gray line shows the result of total L in each random time. The average running time is 0.304433 second. The best result can be found after 30 times random is 128533.

The Result of Experiment 4 with 500 offers and 500 demands is shown in Figure 7 The orange column shows the

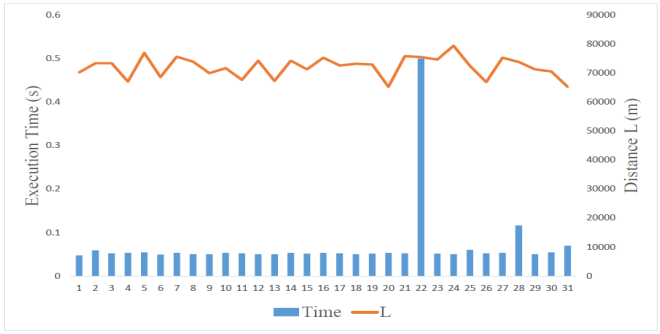

Fig. 5. The Result of Experiment 4 with 100 offers and 100 demands

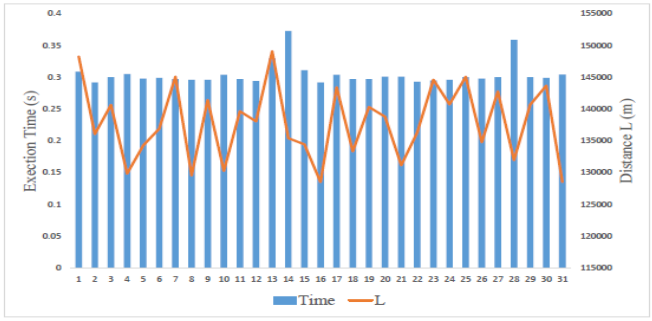

Fig. 6. The Result of Experiment 4 with 300 offers and 300 demands

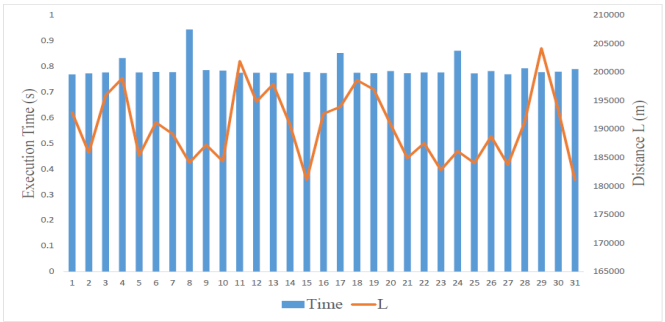

Fig. 7. The Result of Experiment 4 with 500 offers and 500 demands

running time. The gray line shows the result of total L in each random time. The average running time is 0.7892 second. The best result can be found after 30 times random is 181069.

The Result of Experiment 3 with several number of offer and demand is shown in Figure 8 . The more offer is, the more total L is. The running time is also increasing.

The Result of total L in 3 experiments 2,3 and 4 is shown in Figure 9 The Result of Experiment 3 and Experiment 4 is increasing with the increasing of the number of offer. Moreover, the result of Experiment 3 and 4 is not much different from each other. The result of Experiment 4 depends on how many time random we conduct in Experiment 4. The more random times, the better result is. As we can see, 30 random times is still good with $N = 50$ and $N = 100$, the result of Experiment 4 is not much different from Experiment

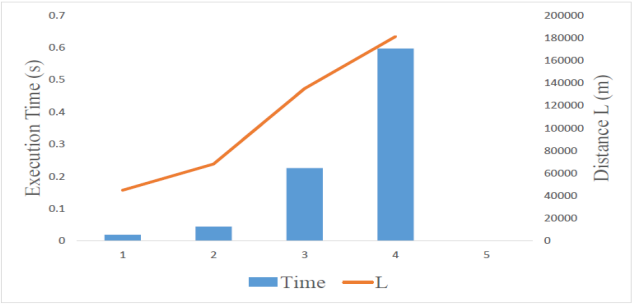

Fig. 8. The Result of Experiment 3 with several numbers of offers and demands

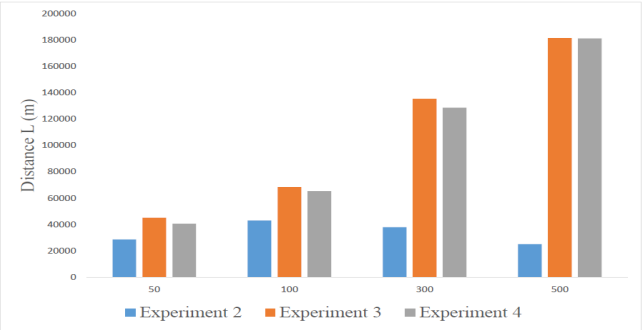

Fig. 9. The Comparison of total distances L in Experiment 2, Experiment 3 and Experiment 4

2. However, when $N = 300$ and $N = 500$, the result is much larger than result in Experiment 2.

We know that result of Experiment 2 is the best result with the minimum total L. The behavior of result of Experiment 2 is interesting. When N increases from 50 to 100, the result also increases, but when $N = 300$ and $N = 500$, the result decreases. This can be explained that with the small area like the map in Experiment- there are 42 vertices, the more offer and demand, the probability of an offer has to travel more to pick up demand is lower. Thus, Method of Experiment 2 is good for the small scale area. However, with the large scale area, method of Experiment 2 can need too much running time.

When we deal with large scale area, the method of Experiment 4 - the random approach is better and we can improve the result by increasing the number of random time.

D. Analyze results

The authors compare results gained from experiment 2, 3 and 4. Experiment 2 has a significant number of offer-demand pairs with the lowest L among other experiments. We know that the result of Experiment 2 is the best result.

With the same input file of offers, experiment 3 shows the less efficiency than experiment 4. Results of experiment 3 and 4 are not as good as ones of experiment 2. However, the execution time of experiment 3 and 4 are better than one experiment 2.

Therefore, Method of Experiment 2 is good for the small scale area. However, with the large scale area, method of Experiment 2 can need too much running time.

When we deal with large scale area, the method of Experiment 4 — the random approach is better and we can improve the result by increasing the number of random time.

V. Conclusion

Base on the Dijkstras, Floyd-Warshall and Hungarian algorithms, the authors build matching methods dealing with optimizing the daily routes between offers and demands in ridesharing problems. Furthermore, the authors take advantages from Google Maps and reuse the information of the real maps of Ward 4 and Ward 5 of District 5, Ho Chi Minh City, Viet Nam to construct a Graph for our methods.

To verify the effectiveness of methods, the authors conduct four experiments and compare their results. The objective of experiment 1 is to find which demand could be best assigned to the offer. Then, the method of experiment 1 is applied in later experiments. After implementing the experiment 2, 3 and 4 and collecting outcomes, the authors realize that the efficiency of the experiment 2 is the best and then the experiment 4 and 3 follow accordingly. About the execution time of each experiment, the experiment 3 is the fastest one while the experiment 2 takes the longest time.

On the other hand, the method of using randomization is better than the one of the approach from top to bottom. In addition, their running time are better than the method based on Floyd-Warshalls algorithm.

These methods can be applied on a large map full of details like the whole Ho Chi Minh City. Moreover, an application on mobile devices can become a considerable solution for ridesharing problems. Then solved problems come along with traffic congestion are probably solved.

References

[1] "Congestion: A national issue," *U.S. Department of Transportation, Federal Highway Administrator's Office of Operations*, August 2008.
[2] A. Rosen, "What really causes traffic congestion?" *Sheepshead Bites*, July 2013.
[3] "What is Real-Time Ridesharing?" *MIT Real-Time Rideshare Research*.
[4] "Reduction Pollution through Ridesharing," *Sustainable Cities Institute*.
[5] M. Harris, "Uber: why the worlds biggest ride-sharing company has no drivers," *theguardian.com*, November 2015.
[6] K. Buchan, "Vietnam cities told that driving down pollution is a matter of ride-sharing," *theguardian.com*, May 2015.
[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1959.
[8] R. W. Floyd, "Algorithm 97: Shortest Path," *Communications of the ACM*, June 1962.
[9] S. Warshall, "A theorem on Boolean matrices," *Journal of the ACM*, January 1962.
[10] D. Gusfield, *The Stable Marriage Problem : Structure and Algorithms*. Cambridge, Mass: MIT Press, 1989.
[11] H. W. Kuhn, "The Hungarian Method for the assignment problem," *Naval Research Logistics Quarterly*, 1955.
[12] J. M. Hess, "ECO11 Young Future Mobility Leaders," *Ecosummit TV*, May 2011.
[13] "Dynamic Ridesharing Projects, Current, Past, and Proposed," http://dynamicridesharing.org/projects.php.
[14] S. Ma, Y. Zheng, and O. Wofson, "Real-Time City-Scale Taxi Ridesharing," Computer Science Department, University of Illinois at Chicago, Chicago, IL, USA, Tech. Rep., July 2014.

[15] S. Oman, "A new app is helping the taxi industry compete with Uber," *Business Review Canada*, December 2015.
[16] L. Rayle, S. Shaheen, N. Chan, D. Dai, and R. Cervero, "App-Based, On-Demand Ride Services: Comparing Taxi and Ridesourcing Trips and User Characteristics in San Francisco," University of California Transportation Center, Tech. Rep., August 2014.
[17] M. T. Vu, "Nhung kinh nghiem phong, chong cuop danh cho lai xe taxi," *An ninh Thu do*, June 2015.
[18] M. Otero, "The real 10 algorithms that dominate our world," *About Medium*, May 2014.
[19] V. Sangam, "Dijkstras algorithm," *Theory of Programming*, January 2015.
[20] H. Mairson, *The Brandeis Review*. Waltham, M. : Brandeis University, 1992, vol. 12, no. 1, ch. 7, p. 38.
[21] E. Tardos and J. Kleinberg, *Algorithm Design*. Addison-Wesley Longman, Inc., 2006.
[22] M. J. Golin, "Bipartite Matching and the Hungarian Method," *Hong Kong University of Science and Technology*, August 2006.
[23] N. M. 2014, "The Prize in Economic Sciences 2012," *Nobelprize.org*, May 2016.
[24] T. N. R. M. Program, "How the Matching Algorithm Works," http://www.nrmp.org/match-process/match-algorithm/.
[25] N. M. S. Inc., "The Matching Algorithm," https://natmatch.com/ormatch/aboutalg.html.
[26] "Synopsis of DRT European Commission Directorate-General for Energy and Transport," http://www.managenergy.net/resources/461.
[27] E. P. Penelope, "Demand Responsive Transit service (DRTs): Personal-Bus - Tuscany - Florence - Italy," September 2002.
[28] Transit Cooperative Research Program, "A guide for planning and operating flexible public transportation services," The National Academies of Sciences, Engineering, and Medicine, Tech. Rep., February 2015.
[29] A. H. Land and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
[30] J. Clausen, "Branch and Bound Algorithms - Principles and Examples," Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark., Tech. Rep., March 1999.
[31] A. Asratian, *Bipartite graphs and their applications*. Cambridge, U.K. New York: Cambridge University Press, 1998.
[32] "Population and labour," *Statistical Office in Ho Chi Minh City*, 2011.
[33] "Saigon: Motorbike capital of the world," *StrippedPixel.com*, February 2012.