On Order Dispatch and Route Planning Problem

Yang Luo(518021910747, floating-dream@sjtu.edu.cn),
Futao Wei (518021910750, weifutao@sjtu.edu.cn),
Jinyi Xiang (518030910256, grantsjtu@sjtu.edu.cn)

Department of Computer Science,
Shanghai Jiao Tong University, Shanghai, China

Abstract. In this report, we study the Order Dispatch and Route Planning (ODRP) problem. First, we introduce the background of the problem. Then we give the definition of the ODRP problem and establish the proof of its NP-completeness. Next, we provide a feasible algorithm for the problem and conduct experiments based on datasets from real world.
Keywords: Order Dispatch, Route Planning.

1 Introduction

1.1 Problem Background

There has been a rapidly growing demand for public transportation in mega-cities like Beijing and Shanghai. Consequently the burden on main transportation hubs, like Shanghai Pudong International Airport, is increasing, and traditional transportation means, such as subways, taxis, and buses, fail to fully meet people's demand to go home from the transportation hubs.
To this end, we propose bus carpooling as a way to relieve the transportation pressure. We'll provide a bus-booking app for passengers, on which they can book a bus for leaving the airport ahead of schedule, maybe when they're in the plane. When booking, they need to specify their flight schedules and destinations. With the time and destination, we'll be able to dispatch orders with near destinations to the same bus and plan a customized route for the bus, which will undoubtedly save a lot of time for the passengers. Besides, it will be much cheaper than taxis since the cost is apportioned among all passengers.

1.2 Problem Definition

Input:

- A graph $G = (V, E)$, each vertex denoting a bus stop, the weight of an edge denoting the distance between two bus stops; a depot vertex v_{src}, denoting the airport where passengers get off the plane and get on the bus
- The number of passengers n
- The number of buses m; the capacity of each bus L
- A destination array D, $D \subseteq V$, the element d_i in D denoting the destination of the i^{th} passenger $(i = 1, 2, \ldots, n)$

Output:

- The passengers dispatched to the j^{th} bus and the route of the j^{th} bus $(j = 1, 2, \ldots, m)$, such that the total distance of the m buses' routes is minimized
- The minimum total distance of the m buses' routes

1.3 Related Work

The ODRP (Order Dispatch and Route Planning) problem is made of two main parts, order dispatch and route planning. The order dispatch part is hardly studied in academia. But the route problem, known as capacitied vehicle route planning problem, has been extensively studied since the early sixties and recently many new heuristic approaches have been brought up to solve CVRP problem. The CVRP extends the well-known Traveling Salesman Problem (TSP), calling for the determination of the circuit with associated minimum cost, visiting exactly once a given set of points. Therefore, many exact approaches for the CVRP were inherited from the huge and successful work done for the exact solution of the TSP.

2 Assumptions

we make the following assumptions to simplify the problem and assure the reliability of data. Each assumption is justified by the closely following statements.

- **When measuring the distance between two locations, we use the linear distance to substitute road distance or walking distance.** For the simplicity of our model, we use the linear distance in all cases.

- **Orders within one kilometer radius of the starting point are considered as the orders departing from the starting point.** This assumption is made to assure the reliability of our data set. Also, 1 kilometer walling distance is typically acceptable to most people.

- **Passengers will get off the bus at the nearest station to their destination in linear distance.** It's intuitive to get off the bus where it has the shortest walling distance. But the it's hard to measure the walking distance from one place to another. Therefore, we just take the station and destination as points and measure the distance between two points.

3 Theoretical Analysis

3.1 Decision Version

To perform NP complexity analysis, we have the decision version of the ODRP problem:
Input:

- A graph $G = (V, E)$, each vertex denoting a bus stop, the weight of an edge denoting the distance between two bus stops; a depot vertex v_{src}, denoting the airport where passengers get off the plane and get on the bus
- The number of passengers n
- The number of buses m; the capacity of each bus L
- A destination array D, $D \subseteq V$, the element d_i in D denoting the destination of the i^{th} passenger $(i = 1, 2, \ldots, n)$

Output:

$$ans = \begin{cases} 1, & \text{if } P \text{ is true} \\ 0, & \text{if } P \text{ is false} \end{cases}$$

where P: there exists an ODRP solution that achieves a total distance of bus routes less than k (some specific value).

3.2 Proof of NP-completeness

First, the ODRP problem is NP, since we can find a poly-time certifier. The certifier simply adds up the distances of m bus routes to determine whether the total distance is less than k.
Second, we'll establish the NP-completeness of the ODRP problem by proving that the Travelling Salesman Problem (TSP) can be polynomially reduced to the ODRP problem, i.e., TSP $\leq_P$ ODRP.
Instance of TSP: A weighted undirected graph $G = (V, E)$; a starting point vertex v_{src}; an integer k
Instance of ODRP: $G = (V, E)$ as the graph; v_{src} as the depot vertex; $|V|$ as the number of passengers; 1 as the number of buses; $|V|$ as the capacity of a bus; the vertex set V as the destination array; k as the value to be compared with the total distance
Then we prove that there exists a route for TSP with total weight $\leq k$ iff there exists an order dispatch & route planning for ODRP with total weight $\leq k$.
$\Longrightarrow$:
If there exists a route for TSP with total weight $\leq k$, then the route in TSP will be the route for the only bus in ODRP. Besides, all the $|V|$ passengers will be dispatched to the only bus. Thus we find a solution with total weight $\leq k$.
$\Longleftarrow$:
If there exists an order dispatch & route planning for ODRP with total weight $\leq k$, then the route in ODRP will be the route for TSP, i.e., a route with total weight $\leq k$.
Hence, we've shown that TSP $\leq_P$ ODRP. The ODRP problem is NP-complete.

3.3 Programming Formulation

The ODRP problem can be formulated as an integer programming. Below are the constraints:

- The number of passengers dispatched to each bus should be no more than the capacity L.

$$\sum_{i=1}^{n} y_{ik} \leq L, \quad \text{for } 1 \leq k \leq m$$

  where

$$y_{ik} = \begin{cases} 1, & \text{if the } i^{\text{th}} \text{ passenger is dispatched to the } k^{\text{th}} \text{ bus} \\ 0, & \text{if the } i^{\text{th}} \text{ passenger is not dispatched to the } k^{\text{th}} \text{ bus} \end{cases}$$

- Each passenger is dispatched to only one bus.

$$\sum_{k=1}^{m} y_{ik} = 1, \quad \text{for } 1 \leq i \leq n$$

- Each bus begins its tour at v_{src} (denote it as number 0).

$$\sum_{j=1}^{v} x_{0jk} = 1, \quad \text{for } 1 \leq k \leq m$$

  where v denotes the number of vertices and

$$x_{ijk} = \begin{cases} 1, & \text{if the route of the } k^{\text{th}} \text{ bus contains the directed edge } (i,j) \\ 0, & \text{if the route of the } k^{\text{th}} \text{ bus does not contain the directed edge } (i,j) \end{cases}$$

- Each bus ends its tour at v_{src} (denote it as number 0).

$$\sum_{i=1}^{v} x_{i0k} = 1, \quad \text{for } 1 \leq k \leq m$$

- The route of each bus should be a cycle.

$$[x_{ijk} = 1] \sum_{t=1}^{n} x_{jtk} = 1, \quad \text{for } 1 \leq k \leq m$$

- The route of each bus should cover all its passengers' destinations.

$$[y_{ik} = 1] \sum_{i=0}^{v} x_{ijk} = 1, \quad \text{for } j = d_i, 1 \leq i \leq n \text{ and } 1 \leq k \leq m$$

  where d_i denotes the destination of the i^{th} passenger.

The target is to minimize the total distance of all the buses:

$$\min \quad \sum_{k=1}^{m} \sum_{i=1}^{v} \sum_{j=1}^{v} w_{ij} x_{ijk}$$

where w_{ij} denotes the distance between vertex i and vertex j.
In summary, we have the ODRP problem transformed to the following integer programming:
Target:

$$\min \quad \sum_{k=1}^{m} \sum_{i=1}^{v} \sum_{j=1}^{v} w_{ij} x_{ijk}$$

Constraints:

$$\sum_{i=1}^{n} y_{ik} \leq L, \quad \text{for } 1 \leq k \leq m \tag{1}$$

$$\sum_{k=1}^{m} y_{ik} = 1, \quad \text{for } 1 \leq i \leq n \tag{2}$$

$$\sum_{j=1}^{v} x_{0jk} = 1, \quad \text{for } 1 \leq k \leq m \tag{3}$$

$$\sum_{i=1}^{v} x_{i0k} = 1, \quad \text{for } 1 \leq k \leq m \tag{4}$$

$$[x_{ijk} = 1] \sum_{t=1}^{n} x_{jtk} = 1, \quad \text{for } 1 \leq k \leq m \tag{5}$$

$$[y_{ik} = 1] \sum_{i=0}^{v} x_{ijk} = 1, \quad \text{for } j = d_i, 1 \leq i \leq n \text{ and } 1 \leq k \leq m \tag{6}$$

4 Destination Selection

4.1 Data Cleaning

We first select the data assigned to our group from the whole data set. But after we drew a sketch of the data, we found that there are some outliers exceeding the area of the given city. We use the Google Map to determine the boundary of Chengdu and Haikou. If the destination or origin of orders goes beyond the boundary, these orders will be left out. The sketch of destinations are consistent with common sense after finishing data cleaning, more orders in the center of city, less order at the suburbs.

4.2 Methodology

The basic rule for destination selection is to set stops for passengers' convenience and efficiency. So we take the k-means clustering as our approach to select the destinations.

Given a set of historial orders, we regard the destination of for i_{th} order as an observation g_i, where g_i is a two-dimensional vector consisting of the geographical information of destination. All the observations form a set $\{g_1, g_2, \ldots, g_n\}$. Our k-means clustering algorithm aims to partition the set into k sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares.[1] We can formulate the objective as follows:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\| = \arg\min_{S} \sum_{i=1}^{k} |S_i| \mathrm{Var} S_i$$

Where μ_i is the mean points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{S} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} \|x - y\|$$

After clustering, we use center of each cluster as the bus stop, but this may some problems like: the bus stop is situated at a non-stop area or is not convenient for drivers to stop. In the world of transit, The following factors should also be taken into consideration at the planning stage:

- Proximity to adjacent junctions
- Proximity to pedestrian crossings;
- Bends or crests in the road;
- On-street parking

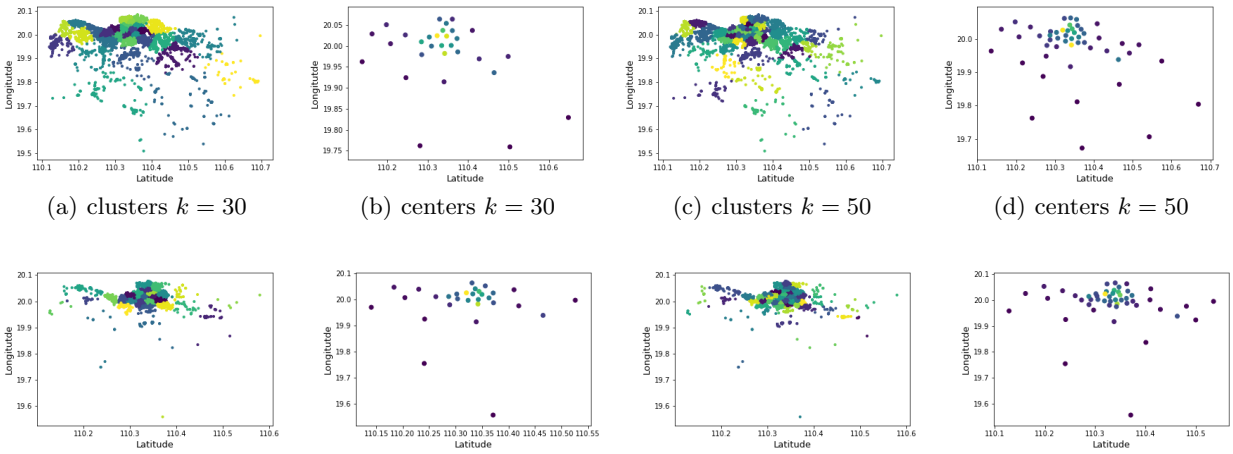
To meet the requirements mentioned above, we manually check the points we chose and slightly move the stops to better locations if possible. The following section will display our clustering results ans the center we stations we pick.

4.3 The Determination of Virtual Airport

We use a simple method to determine our virtual airport. We take the mean value of all the order's starting longitude and latitude. The values computed this way are set to be the longitude and latitude of starting point , respectively. Simple as it may seem, it's also quite effective since the orders starting within 1 km from the starting point amount to 10,500.

4.4 Visualization of Results

We first compare the clustering results before order selection and after order selection, taking Haikou as an example. Then we show the final chosen stations of Chengdu.



(a) clusters $k = 30$ (b) centers $k = 30$ (c) clusters $k = 50$ (d) centers $k = 50$

(e) clusters $k = 30$ (f) centers $k = 30$ (g) clusters $k = 50$ (h) centers $k = 50$

Fig. 1. Comparison of clusters and stops before order selection and after

The first row in Fig.2 are before order selection and the seconds are after order selection. As is shown in the picture above, the original order set and the chosen orders are in the same distribution, which proves the rationality of our choosing algorithm. From this picture, we can tell that the northern part of the Haikou city is more prosperous and densely populated. Thus, we set more stops at the northern part.

The following are the results of orders from Chengdu, see Fig.3 . From the picture, we can see that while Haikou has a fan-shaped distribution, Chengdu is more like a circular distribution.

In the following experiments, the stations are set at the center of these clusters unless otherwise stated.

5 Alogrithm Design

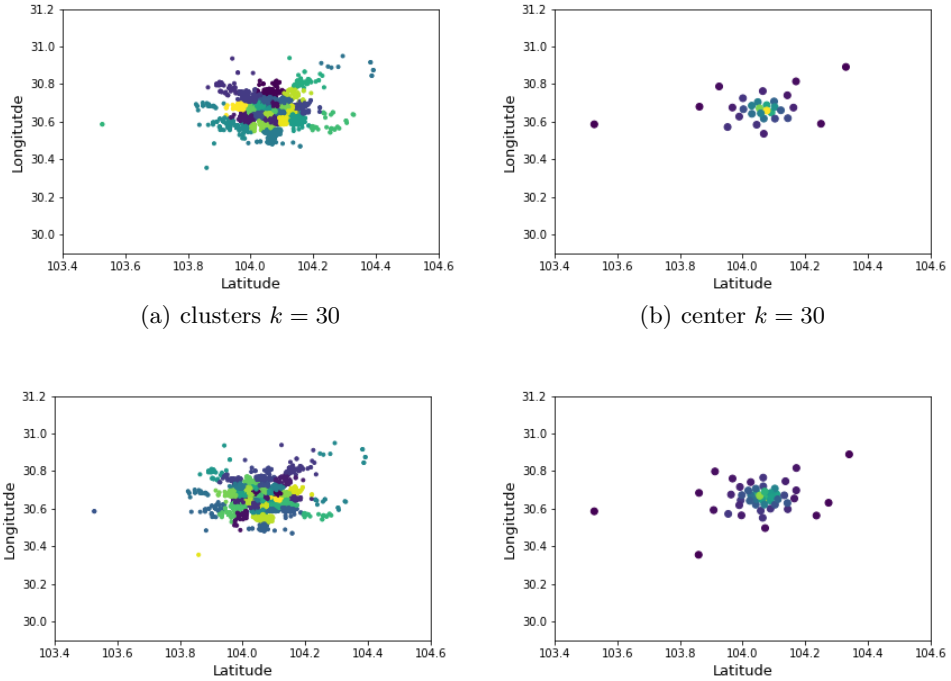
5.1 The Greedy Algorithm

We need to select all orders received at a certain time. We use this algorithm to design routes. For each order, if its destination belongs to a route, it will be directly assigned to that route. If the order proportion of a station is less than a certain value, it will be discarded directly.

Due to the large amount of receiving data, we directly consider the capacity of the route. Because the demand of each station in the data is too large, we have to consider using the path capacity for calculation, otherwise the vehicle capacity value is too small, we can not deal with the problem of high demand. If the demand is small, the problem of path capacity will degenerate into the problem of vehicle capacity (that is to say, the path capacity is regarded as a large vehicle, which can have a large capacity).

This algorithm hopes to get as few routes as possible to make the capacity of the route approach saturation as much as possible.

Then allocate enough vehicles for the needs of the route. So we do the following operations:



(a) clusters $k = 30$ (b) center $k = 30$

(c) clusters $k = 50$ (d) centers $k = 50$

Fig. 2. Clustering Results of Chengdu

– Regard all stations as nodes, calculate the distance between nodes as the weight of edge, and then sort the weight of edge from small to large.

– Check the edges in order of sorting in the graph. If the edges meet the following rules, they will be added to the graph. Otherwise, they will not be considered.

  Rule 1: after adding the edge, the number of connecting edges at any point will not be greater than 2;

  Rule 2: adding this edge will not generate a ring in the graph;

  Rule 3: the demand weight of a single path generated by adding this edge will not exceed the route capacity

– Until all edges are traversed. At this time, the path endpoint obtained in the search graph will be connected with the starting point to form a cycle.

– Then according to the pricing strategy, the results are obtained.

The Pseudocode is as following:

Algorithm 1: Greedy Algorithm

Input: K stations $n[k+1]$, including order number, longitude and latitude;
//$n[0]$ is used for the source station.;
Output: paths contain several stations;
$e[(k \times k-1)/2] \longleftarrow 0$ $count1 \longleftarrow 0$ **for** $i = 1$ *to* n **do**
  for $j = i$ *to* n **do**
    $e[count1] \longleftarrow$ the distance between $n[i]$ and $n[j]$;
    $count1++$;
  end
end
Sorte //make e rank from small to large;
$Graph[k+1][k+1] \longleftarrow 0$;
for $i = 1$ *to* $count1$ **do**
  Add two ends of the $e[i]$ to *Graph*;
  if *one or two of the ends' degree* > 2 **then**
    Delete two ends of the $e[i]$ to *Graph*;
    continue;
  end
  if *there exist a cycle in Graph* **then**
    Delete two ends of the $e[i]$ to *Graph*;
    continue;
  end
  Find the path which contains $e[i]$ and sum the demands of all nodes in the path, denoted as c;
  if $c >$ *the CAPACITY of the path* **then**
    Delete two ends of the $e[i]$ to *Graph*;
    continue;
  end
end
$Count2 \longleftarrow 0$;
for $i = 1$ *to the number of paths in the Graph* **do**
  Connect the two endpoints of each path to the starting node, then we get $cycle[count2]$;
  $count2++$;
end
Calculate the profit by the pricing Strategy;
return *cycle*

5.2 Improvement of the Greedy Algorithm

Using the greedy algorithm with the shortest edge can reduce the length of the path, but we found that when generating the path, the end point of the path may be far away from the starting point, which will increase the length. After consulting the related works[2], we learned the solution to this problem:

We can improve the quality strategy based on the $hold - Karp$ model by adding a vector

$$\Pi = (0, -\alpha D_1, -\alpha D_2, ... - \alpha D_k)$$

whose absolute value α represents the priority of adding the far edge from the distribution center; D_i represents the distance from station I to the starting node. When sorting, we use $D_{ij} + \Pi_i + \Pi_j$ instead of the original distance. In this way, we can preferentially connect the points that are far away from the starting point, and the end of the path formed finally will be closer to the starting point. The test results on several data sets show that it can effectively reduce the total length of the path.

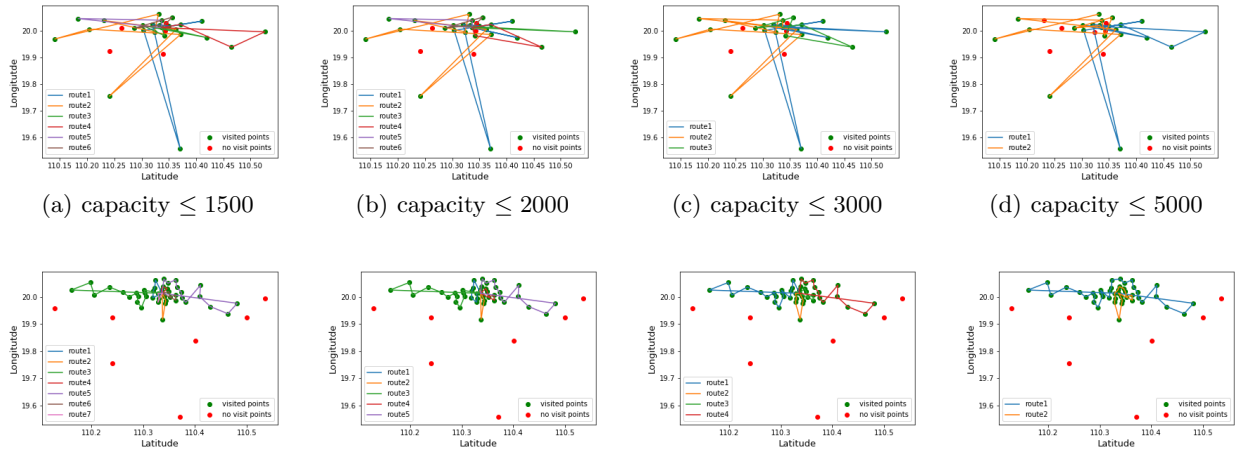
5.3 Time complexity

We input n stations and create $\frac{n(n-1)}{2}$ edges, The most time-consuming stage is the sorting phase whose time complexity is $O(n^2 \log n)$. The operations on the graph use DFS algorithm whose time complexity won't exceed $O(n^2)$. Thus the time complexity of the algorithm is $O(n^2 \log n)$.

6 Experiments

6.1 Visualization of the Routes

Here we only show the routes for Haikou under various conditions. We set different capacity constraints for the routes. The capacity is for the route not the bus. Then we set $\alpha = 0.5$ for the hold-Karp model. If the order quantity of a station accounts for less than 0.0015 of the total order quantity, we won't go to these stations. In this case, the capacity for each bus is fixed, that is 40 passengers per bus. There might be multiple buses shuffling on this route. the detailed routes can be seen as follows:



(a) capacity ≤ 1500 (b) capacity ≤ 2000 (c) capacity ≤ 3000 (d) capacity ≤ 5000

(e) capacity ≤ 1500 (f) capacity ≤ 2000 (g) capacity ≤ 3000 (h) capacity ≤ 5000

Fig. 3. Comparison of clusters and stops before order selection and after

The upper row of Fig.4 is computed when $k = 30$ while the lower row is when $k = 50$. As is shown in Fig.4, typically, if we set the capacity lower, we have more diverse routes and passengers can arrive in shorter time. Once we consider more stations, say $k = 50$, we will mainly running our buses in urban area.

6.2 Pricing Strategy

We decide to place the pricing strategy section in experiments since we need to figure out our actual cost for the operation and then determine our price.

We set the cost for a car as 100, the cost per km as 1.5. And the price is calculated according to the distance between the destination and the starting point. The price of one kilometer from the starting point is 2. The starting price is set as 5. The experiment results are illustrated in Table.1

The fixed cost stands for the time-independent cost, like rent, while the cost per kilometer is a time-dependent. Here time-dependent cost means the total cost is a linear function of time, like the gasoline, drivers' pay and some other operational cost.

Since we only consider one shuffle for each bus, we have to set a high fixed cost to make ends meet. Once we use to buses continuously, we have reason to believe to that we can slash our prices by half and also make a profit.

	Haikou				Chengdu			
	capacity = 30		capacity = 40		capacity = 30		capacity = 40	
	k=30	k = 50	k=30	k = 50	k=30	k = 50	k=30	k = 50
Cost	33607	41767.9	25380.4	31461.9	43955.5	52169.8	33195.7	39286.4
Profit	41701.5	33774.5	49928	44080.5	58831.8	58622.9	69591.5	71506.3

Table 1. The experiment results on cost and profit

7 Strengths and Weaknesses

Advantages:

1. We cover the densely populated area using our station selection approach. And we set more stations at places
2. We give the programming for this problem and we also get the optimal solution at a small scale.
3. $hold - Karp$ model is used as an optimization, which can reduce the total distance of the route as much as possible and also reduce the cost.
4. Our feasible solution can be computed by greedy heuristic algorithm in a short time. It means our algorithm is applicable in scenarios with high real time requirements. The feasibility of the algorithm has been proved by the test of the actual data set, and the route can be designed based on given order information.

Disadvantage:

1. The feasible solution involved a large amount of calculation. We may not be able to calculate so many edges by other methods due to the limitation of our time and energy. We hope to further study this problem in the future
2. We barely cover the outlying areas. Since few orders' destination is too far from the urban area, it takes huge cost to set a station there.
3. Each bus is only used once in this problem. In real world transit, the buses will be used repeatedly.
4. We only consider the case for one departure station. We can have more departure stations. Further interaction between departure stations should be taken into account.

8 Conclusion

In this paper we implement both optimal and feasible algorithm to solve this problem. while the optimal solution can utilize the resources to the best, it takes lots of time to get such a solution.The feasible version, though not the best, does give a solution good enough. In real world application, our algorithm can effectively utilize vehicle resources, reduce energy consumption, and save road resources. Also, our pricing strategy charges a reasonable amount, covers all cost and yields a certain amount of money. As the sharing economy is rising, our algorithm, or our platform can surely make a difference.

Acknowledgements

Through this project, we gain a better understanding of the algorithms learned in this class. We also come to realize that the computing a feasible solution in a short time is really important in real time scenario even though you may have an optimal algorithm.

We would like to pay our special regards to Prof. Gao for her guidance through the whole class. We would like to recognize the invaluable assistance that TAs all provided during our study. We wish to thank all the people whose assistance was a milestone in the completion of this project.

References

1. "K-means clustering." Wikipedia, The Free Encyclopedia. Wikipedia, https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=960431422. Last accessed 5 Jun 2020
2. Author, WenZhen Rao, Chun Jin.: A fast greedy algorithm for large scale CVRP problems DOI:10.13587/j.cnki.jieem.2014.02.026