Column generation methods for vehicle routing problem

Team Name: Go Blue
Team Member: Jiaqi Guo, Bin Yao
Advisor: Viswanath Nagarajan

Abstract

We consider a multi-depot multi-trip vehicle routing problem with identical time windows. We first provide methods that utilize statistics of historical data to facilitate decisions on the number and locations of depots. We present two different formulations based on column generation to solve instances with different sizes. Several experiments are conducted to analyze the performance of our models.

keywords: Vehicle routing; Weighted K-median clustering; Column-generation; Random coloring

Contents

1. Introduction

In the AIMMS-MOPTA competition, we consider the Multi-Trip Vehicle Routing Problem with Time Windows (MTVRPTW), which is a variant of the classical Vehicle Routing Problem with Time Windows (VRPTW). Given an undirected graph (V, E) with the location for each vertex $v \in V$ and the distance between two vertices, we want to decide the number of depots K and their locations meanwhile the fleet size should be optimized. The problem also involves time-window and vehicle capacity constraints: the vehicles are only allowed to work between 6 a.m. and 5 p.m. and the orders can only be delivered between 8 a.m. and 4 p.m. Besides, a vehicle can carry at most 60 units of goods when departing from the depot but reload is allowed. There are several features worth mentioning:

1. The given graph is non-complete, so the shortest distance between node i and j may be unknown, causing inconvenience when developing our models. Therefore, we first apply Dijkstra's algorithm to calculate the distance between all pairs of nodes.

2. The number of depots and their locations is unknown. We can utilize the statistics information of historical demand data plus the distance we calculate to enforce our depot decision to reduce the routing cost.

3. Most of the vehicle routing problem with time windows involves different time windows for all customers. Here we consider a simple variation that all customers have identical time windows. The time constraints in the integer programming formulation can be significantly reduced compared to the general case.

4. The multi-trip route is allowed, say vehicles can return to depots and reload again. This property helps reduce the fleet size, especially when the vehicles can take several trips within the time windows. There are many ways to formulate the integer programming model for reloading cases, but finding a concise and efficient one is not easy. We will present our formulation in section 4.

The model mainly consists of two parts. First, we utilize data statistics information to make decisions on the number of depots and their locations. Afterward, we face daily demand one day ahead and develop routing algorithms for scheduling the vehicles to satisfy the demand. Since the first stage has a fundamental impact on the total cost and only requires one implementation, our goal is to obtain the best solution we could get at the cost of running time. The routing model needs to be implemented every day, so we develop algorithms that can produce good solutions in a reasonable time frame. In this report, we will present our efforts to combine the merits of heuristic and optimization methods to develop the routing model.

The remainder of this report is organized as follows. In section 2, we introduce the location model. In section 3, we provide approaches combining optimization and heuristic methods for solving the routing problem. In section 4, we conduct several experiments and provide our recommendations. In section 5, we summarize the report and propose several directions that worth further exploring.

2. Location model for clustering and allocation

2.1 Introduction

To cut this multi-depot vehicle routing problem into single-depot vehicle routing problems, we consider how to classify all nodes into different clusters with the center of the cluster as the depot. It is a problem that we evaluate the trade-off between the fixed cost of depots, the variable cost of the vehicles and routing. The classical K-median algorithm provides insights into this kind of problem. Since the number of days that a node has demand in a year resulting in the visit times of this node in a year, we should include it into our model to make our clustering closer to the reality.

2.2 Location model without cost of depots

We exclude the building cost of depots here to construct our location model. For each candidate number of depots K, we have

$$\min \sum_{(i,j)\in G} d_{ij}x_{ij}w_j \tag{2.1}$$

$$\text{s.t.} \sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \tag{2.2}$$

$$x_{ij} \le y_i \qquad \forall (i,j) \in G \tag{2.3}$$

$$\sum_{i\in V} y_i \le K \tag{2.4}$$

$$x_{ij}, y_i \in \{0,1\} \tag{2.5}$$

Here we put weight w_j for each vertex j which represent annual number of orders occurs at j. $x_{ij} = 1$ if node j is served by depot i and equals to 0 otherwise. $y_i = 1$ if we node i is selected as the depot location and equals to 0 otherwise.

The objective minimizes the total weighted distance, (2.2) forces all nodes are served by at least a depot. (2.3) means that node i can serve node j only if i maintains a depot. (2.4) constrains the number of depots.

2.3 Decision on the number of depots

Section 2.2 provides an approach to obtain the number of depots and the customer clusters given number K. One problem remains how to decide the optimal number of K. There are two key steps. First is to decide the candidates of K by its lower and upper bound, then we gradually narrow down the gap between the two bounds until we find the K; Another is how to estimate the annual total cost for certain K given calculating all the daily cost is too time-consuming. It is better to utilize the sampling method.

We use a weak result for the lower bound L by ensuring all the customers have the shortest distance less than 220 to the depot because otherwise a vehicle cannot return to the depot on time. This gives us $L = 3$. For upper bound, we estimate it by making full use of the vehicle capacity. We have

$$U = \frac{\text{Average Daily Demand}}{\text{Capacity of vehicle}} \approx 14. \tag{2.6}$$

We pick several representative numbers such as $K = 5, 10, 15$, and get the clustering results for each of them. We then run the routing model on each cluster to obtain an estimated total cost. Given the historical data has seasonality, we should sample the days that can reflect the pattern. After sampling, we calculate the number of vehicles and routing costs. For the number of vehicles, we choose the day with the highest demand and calculate the vehicles needed. For routing cost, we implement our routing model in section 4 to obtain routing cost for each day and utilize it to estimate the aggregated routing cost in a year. The last step is selecting the optimal K by the estimated total cost.

2.4 Location model with cost of depots

We also consider including fixed cost of depots in the objective. We can adapt the objective of K-median algorithm to include the fixed cost of depots and set a parameter for the distance part. We have

$$\min \lambda \sum_{(i,j)\in G} d_{ij} x_{ij} w_j + \sum_{i \in V} c_i y_i \tag{2.7}$$

$$\text{s.t.} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{2.8}$$

$$x_{ij} \leq y_i \qquad \forall (i,j) \in G \tag{2.9}$$

$$\sum_{i \in V} y_i \leq U \tag{2.10}$$

$$x_{ij}, y_i \in \{0, 1\} \tag{2.11}$$

We set λ here to make the routing cost and the vehicle renting cost additive. We estimate the value of λ by utilizing the location model's objective value without costs of depots and the exact routing costs. We only need to solve this model once and obtain the number of depots by summing the value of y.

3. Optimization models for vehicle routing problem

3.1 Introduction

The heuristic method for solving this routing problem does give us insights into this problem. Given the routing problem for each cluster typically face less than 20 orders a day, we consider using optimization methods to solve the routing problem to obtain near-optimal solutions. The routing solution for a cluster can be viewed as a bunch of single-trip or multi-trip routes that cover all the customers (we only consider customers with positive demand), we can apply column generation methods to generate the routes when needed as enumerating all the feasible routes is impossible. The challenge is to solve the pricing problem efficiently. In section 4.2, we first develop an integer programming to solve it. After knowing its downside, we provide a heuristic model for it in section 4.3, which significantly decreases the running time.

3.2 Formulation I: use multi-trip route as columns

The routing solution to a given depot can be viewed as a combination of multi-trip routes where each route corresponds to the schedule for each vehicle. The routing objective is to find a set of valid multi-trip routes that cover all the demands. This problem can be solved using column generation methods. We first provide the master problem as follows:

$$\min \quad \sum_{\tau \in \Omega} (c_\tau + 30) x_\tau \tag{3.1}$$

$$\text{subject to} \quad \sum_{\tau \in \Omega} a_{i\tau} x_\tau \geq 1 \quad \forall i \in V^- \tag{3.2}$$

$$x_\tau \geq 0 \quad \forall \tau \in \Omega. \tag{3.3}$$

Here Ω is the set of all generated multi-trip routes. For each route $\tau \in \Omega$, c_τ is routing cost. a_τ is a column corresponds to a valid multi-trip route where $a_{i\tau} = 1$ if route τ serves customer i and 0 otherwise. Therefore, the first constraint means that all customers must be served. The objective is to select a set of routes that minimize the total cost.

The pricing problem is to find whether there exists a valid multi-trip route τ such that

$$c_\tau + 30 - p^T a_\tau < 0 \tag{3.4}$$

where p is the simplex multiplier. If such a route exists, we add it to the master problem and resolve it, otherwise we stop and resolve the master problem as integer programming and obtain a near-optimal routing solution.

The challenge of the column generation method is to develop an efficient algorithm for the pricing problem. Given the size of daily demand is typically less than 20 customers a day, we first consider formulating multi-trip routes as an integer programming model.

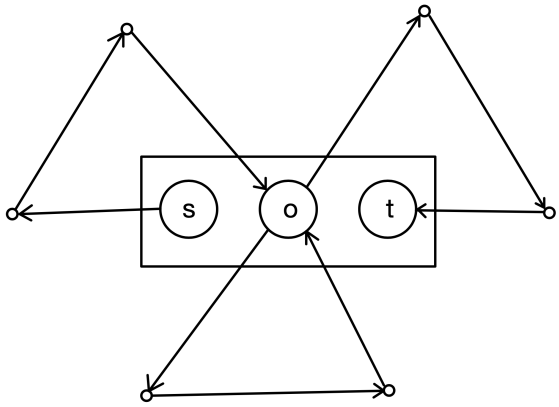

Figure 3.1: multi-trip route with dummy depots

To better capture the reloading feature, we first generate three copies of the depots s, t and o where s denotes the starting point, t denotes the endpoint and o denotes the case when vehicles return to the depot to reload and depart again (see figure 4.1). The integer programming can be formulated as follows:

$$
\begin{aligned}
\min \quad & c_\tau + 30 - p^T a \\
\text{subject to} \quad & c_\tau = 0.7 \sum_{(i,j)\in E} dist(i,j) x_{ij} \\
& \sum_{j\in V^-} a_{sj} = \sum_{j\in V^-} a_{jt} = 1 \\
& \sum_{j\in V^-} x_{oj} = \sum_{j\in V^-} x_{jo} \\
& \sum_{j\in V} x_{ij} = \sum_{j\in v} x_{ji} = a_i \qquad \forall i \in V^- \\
& l_s = l_o = 60 \\
& l_j + d_i - l_i \le (1 - x_{ij})M \qquad i,j \in V \\
& \sum_{(i,j)\in E} t_{ij} x_{ij} \le 11 \\
& \sum_{(i,j)\in E} t_{ij} x_{ij} - \sum_{(s,j)\in E} t_{sj} x_{sj} \le 9 \\
& \sum_{(i,j)\in E} t_{ij} x_{ij} - \sum_{(j,t)\in E} t_{jt} x_{jt} \le 10 \\
& \sum_{(i,j)\in E} t_{ij} x_{ij} - \sum_{(s,j)\in E} t_{sj} x_{sj} - \sum_{(j,t)\in E} t_{jt} x_{jt} \le 8 \\
& \sum_{i,j\in S} x_{ij} \le |S| - 1 \qquad \forall S \in V^-, |S| \ge 2. \\
& x_{ij}, a_i \in \{0,1\} \qquad \forall (i,j) \in E, i \in V^- \\
& l_i \ge 0 \qquad \forall i \in V^-
\end{aligned}
$$

There are three types of variables: a is the column we need to generate for the master problem where $a_i = 1$

if the route serves customer i and 0 otherwise. We preset E as the set of all feasible arcs (i, j), then for any $(i, j) \in E$, $x_{ij} = 1$ if the route travels through arc (i, j) and 0 otherwise. For each node except for endpoint t, l_i is the vehicle's left load after serving node i. (4.10) assumes vehicles depart from the depot with maximum load.

Constraint (4.6) states the expression of the cost of route τ. (4.7) is degree constraint for starting point and endpoint. (4.8) is degree constraint for dummy depot o that the times for departure should equal to the times for arrival in the middle of the route. (4.9) are degree constraints for all customers which enforce the relationship between a and x. (4.11) utilize big-M method to impose the relationship between l_i and l_j if route goes to j from i. (4.12)$\sim$(4.15) are time constraints. Notice that there are only four time constraints since the time windows for all customers are identical. (4.16) are subtour-elimination constraints. Given that the size of subtours is humongous, we generate them on the fly.

If we succeed in finding a solution with negative objective, then we find a column with negative reduced cost and add it to the master problem. We repeat this process until we cannot find such a column. The last step is to solve the master problem with all the columns generated in Ω as an integer programming. The model is given as follows:

Algorithm 1: column generation I for routing problem

while *True* **do**
  Solve master problem (4.1)$\sim$(4.3)
  Obtain the simplex multiplier p
  Solve pricing problem (4.5)$\sim$(4.18)
  if *optimal value is non-negative* **then**
    break
  end
  Add optimal solution a to the constraint matrix of the master problem as column.
end
Resolve master problem as integer programming and set (4.2) as equality constraint.
Let $C = \sum_{\tau \in \Omega}(c_\tau + 30)x_\tau$ and $N = \sum_{\tau \in \Omega} x_\tau$
Return Daily routing and vehicle cost C and number of vehicles N.

We implement this model in Python and Gurobi, and it can output solutions around 10 seconds for instance with around ten customers. However, it is incapable of dealing with medium instances within a reasonable time frame. Therefore, we provide the second formulation in the next section, where the definition of the column is slightly changed, and the heuristic algorithm is applied to speed up implementation.

3.3 Formulation II: use single-trip route as columns

Formulation I fails to deal with medium instances (15$\sim$30 customers) because the pricing problem's running time will increase gradually. Therefore, we consider two modifications. First, instead of generating a multi-trip route that involves enormous complexity, we generate a single-trip route each time. After obtaining a bunch of routes, we assign them to vehicles so that time windows are still satisfied.

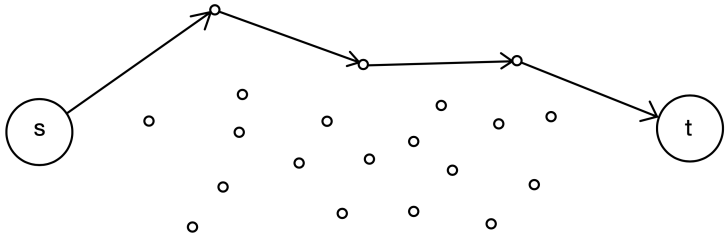

Figure 3.2: Shortest-path problem

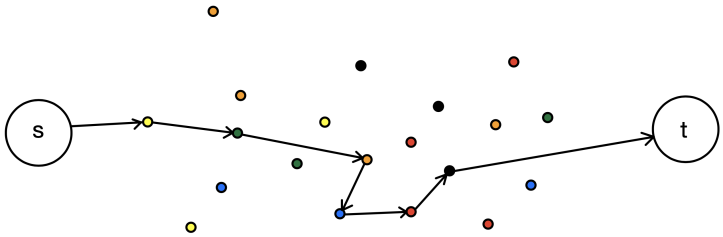

Figure 3.3: Random coloring heuristics

Recap the master problem in Formulation I:

$$\min \quad \sum_{\tau \in \Omega} (c_\tau + 30) x_\tau \tag{3.5}$$

$$\text{subject to} \quad \sum_{\tau \in \Omega} a_{i\tau} x_\tau \geq 1 \quad \forall i \in V^- \tag{3.6}$$

$$x_\tau \geq 0 \quad \forall \tau \in \Omega. \tag{3.7}$$

Now we still use the same formulation as our master problem, the only difference is that each column corresponds to a single-trip route.

The pricing problem is still finding a single-trip route τ such that

$$c_\tau + 30 - p^T a_\tau < 0 \tag{3.8}$$

Ignoring the constant, we need to minimize $c_\tau - p^T a_\tau$, which is equivalent to the shortest path problem: consider s and t still serve as the starting point and endpoint of the depot. Each arc (i, j) has weight associates with it:

$$w_{ij} = c_{ij} - \frac{p_i}{2} - \frac{p_j}{2} \tag{3.9}$$

where c_{ij} is the cost traveling from i to j. p_i and p_j are simplex multipliers associate with customer i and j. The total weight of a path τ from s to t is $c_\tau - p^T a_\tau$, which equals to the object of the pricing problem with only differ from a constant.

Although there are many efficient algorithms for the shortest-path problem, it fails to fit in here because each time we need to examine whether the path forms a cycle, which requires huge memory. Therefore, we utilize some combinatorial optimization methods to develop a heuristic algorithm for this problem.

Consider an instance with 20 customers, and each arc among them has weight as (4.23) (see figure 3.2). We want to find a path from s to t with the smallest weight while still satisfy the capacity and time constraints.

Given that the average order quantity is around 10, a vehicle can serve around six customers between each reloading. Let K be a random number that takes different integer value from four to eight with preset possibilities, this help exploring more possible routes that visit different number of customers. We utilize a method called random coloring [11], which randomly clusters the nodes into K classes. This process can be described as randomly assigning K different colors to all the customer nodes (see figure 3.3). Instead of finding a minimum weight route with six customers, we reduce the size of the problem by considering finding a minimum weight route that touches six colors exactly once. The dynamic programming model to solve this problem is followed:

State: (u, p, cap) where $u \in V$ including s and t; p is a K-dimensional vector with binary numbers that indicates the colors being visited for the shortest path from s to u. For example, [1,1,0,0,0,0] denotes that the shortest path has visited color 1 and 2 only; *cap* is a number between 0 and 60 that denotes the total capacity being used from s to u in the shortest path.
Decision: decide which node comes immediately before u.
Value function: $V(u, p, cap)$ denotes the minimum weight from s to u where the colors being visited corresponds to vector p, plus the capacity being used is bounded by *cap*.
Optimality equation: for any valid state (u, p, cap)

$$V(u, p, cap) = \min_{v \in V, cap - demand(v) \geq 0, p, p' \text{ are valid}} \{V(v, p', cap - demand(v)) + w_{vu}\}$$

Boundary condition:

$$V(u, p, cap) = \begin{cases} 0 & \text{if } u = s, p = \mathbf{0}. \\ \infty & Otherwise. \end{cases}$$

Here $\mathbf{0}$ is a K-dimensional vector with all zeros. After running this dynamic programming model, we only concern with $V(t, \mathbf{1}, 60)$ where $\mathbf{1}$ is a K-dimensional vector with all ones. If the minimal weight route has negative weight and the total time is less than 11 hours, we can add this trip to the constraint matrix of the master problem.

After repeatedly solving the master problem, we obtain a bunch of single-trip routes. The last step is to assign these routes to the preset number of vehicles so that they can satisfy the time windows, which can be done by integer programming formulation. To make the model doable, we will store all the necessary information before we solve the model. For each single-trip route τ, we store the time from depot to the first customer as $ft(\tau)$ and from the last customer to depot as $lt(\tau)$. We only need these two types of information because it is only the first and last trip that matters for time constraints. We also prebuild a set I_i for all customers i indicates the index of routes that cover customer i, which can come in handy when we want to use all the select routes to cover all the customers. The integer programming model is given in the next page.

Let Ω be the set of all generated routes, U be the set of all vehicles, Π be the set of all pair of customers and routes. There are 5 types of binary variables: $x_{u\tau} = 1$ if we assign route τ to vehicle u; $y_\tau = 1$ if route τ is selected; $z_u = 1$ if vehicle u is selected; $f_{u\tau} = 1$ if vehicle u runs route τ as its first trip and $l_{u\tau} = 1$ if vehicle u runs route τ as its last trip.

Constraint (4.25) means route τ is selected if and only if it is assigned to one of the vehicles. (4.26) means route τ can assign to the vehicle u if u is on duty. (4.27) and (4.28) ensure that if vehicle u is selected, it should have a route as its first and last route. (4.29) and (4.30) guarantee the validity between variable x and f and l. (4.31)$\sim$(4.34) are time constraints. (4.35) requires all customers are covered by at least one route.

The model is given in the next page. Experiment results show that this formulation can output routes within a reasonable time frame for medium instance, although it runs slower than formulation I for small instance. Therefore, we can utilize these two formulations to provide solutions for different sizes of instances to speed up implementation.

$$\min \quad \sum_{(u,\tau)\in\Pi} c_\tau x_{u\tau} + 30\sum_{u\in U} z_u \tag{3.10}$$

$$\text{subject to} \quad \sum_{u\in U} x_{u\tau} = y_\tau \qquad \forall \tau \in \Omega \tag{3.11}$$

$$x_{u\tau} \le z_u \qquad \forall (u,\tau) \in \Pi \tag{3.12}$$

$$\sum_{\tau\in\Omega} f_{u\tau} = z_u \qquad \forall u \in U \tag{3.13}$$

$$\sum_{\tau\in\Omega} l_{u\tau} = z_u \qquad \forall u \in U \tag{3.14}$$

$$f_{u\tau} \le x_{u\tau} \qquad \forall (u,\tau) \in \Pi \tag{3.15}$$

$$l_{u\tau} \le x_{u\tau} \qquad \forall (u,\tau) \in \Pi \tag{3.16}$$

$$\sum_{\tau\in\Omega} c_\tau x_{u\tau} \le 11 \qquad \forall u \in U \tag{3.17}$$

$$\sum_{\tau\in\Omega} c_\tau x_{u\tau} - \sum_{\tau\in\Omega} ft(\tau) f_{u\tau} \le 9 \qquad \forall u \in U \tag{3.18}$$

$$\sum_{\tau\in\Omega} c_\tau x_{u\tau} - \sum_{\tau\in\Omega} lt(\tau) l_{u\tau} \le 10 \qquad \forall u \in U \tag{3.19}$$

$$\sum_{\tau\in\Omega} c_\tau x_{u\tau} - \sum_{\tau\in\Omega} ft(\tau) f_{u\tau} - \sum_{\tau\in\Omega} lt(\tau) l_{u\tau} \le 8 \qquad \forall u \in U \tag{3.20}$$

$$\sum_{\tau\in I_i} y_\tau \ge 1 \qquad \forall i \in V^- \tag{3.21}$$

$$x_{u\tau}, y_\tau, z_u, f_{u\tau}, l_{u\tau} \in \{0,1\} \qquad \forall (u,\tau) \in \Pi, u \in U, \tau \in \Omega \tag{3.22}$$

Algorithm 2: column generation II for routing problem

Set T_1, T_2 as the stopping criteria.
while *True* **do**
  Solve master problem (4.19)∼(4.21)
  Obtain the simplex multiplier p
  Set K be a random number that takes value 4, 5, 6, 7 and 8 with preset possibilities.
  Solve the dynamic programming mentioned in this section with K 10 times
  if *No valid route with negative cost found in consecutive T_1 times or more than T_2 routes has been generated* **then**
    break
  end
  Add all the valid paths to the constraint matrix of the master problem as columns.
end
Solve integer programming (4.24)∼(4.36)
Let $C = \sum_{(u,\tau)\in\Pi} c_\tau x_{u\tau} + 30\sum_{u\in U} z_u$ and $N = \sum_{u\in U} z_u$
Return Daily routing and vehicle cost C and number of vehicles N.

Remember that we want to minimize the fleet size because the cost of a vehicle is fixed no matter what fraction of days it is used. Our formulation in this section minimizes the daily cost rather than the fleet size. Therefore, the output result sometimes suggests two vehicles with minimum daily cost while it could be operated with just one vehicle. To obtain a solution with minimum fleet size, a straightforward adaption is to set a high number for vehicle renting costs, which forces the model to search for solutions with fewer vehicles.

4. Results

4.1 The performance of our algorithms

In this section, we present our algorithm's performance by showing the iteration times versus solution improvement. The following figures suggest the relationship between iteration times and solution improvement.

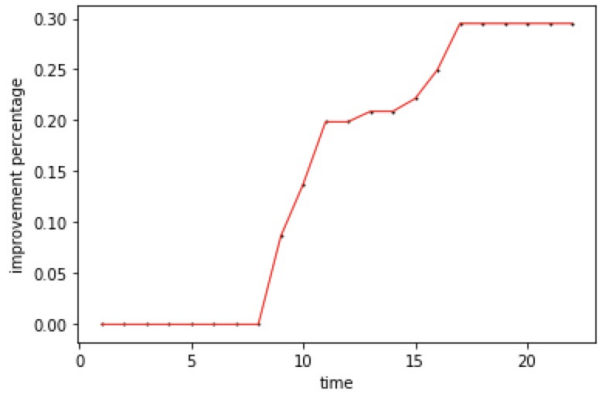

Figure 4.1: Improvement percentage comparing to initial solution versus iteration time

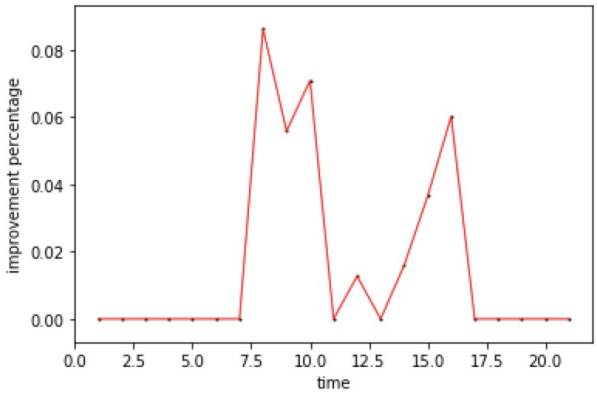

Figure 4.2: Improvement percentage comparing to the previous solution versus iteration time

From the figures we know that the objective value decreases rapidly in the first several times of iterations, and then it shows a convergence to the optimum.

Next, we plot the running time of the location model for different K. Figure 4.3 suggests the relationship between running time versus the variation of K from 13 to 17.

Figure 4.4 shows the running time of our routing model for each cluster under $K = 15$. For depot 15222, 17017, 18519, and 19119, each of them has a larger scale than the others, indicating higher daily demand. Thus, we can conclude that the cluster scale has an impact on the running time of our routing algorithm, a large scale of the cluster often suggests a long running time.

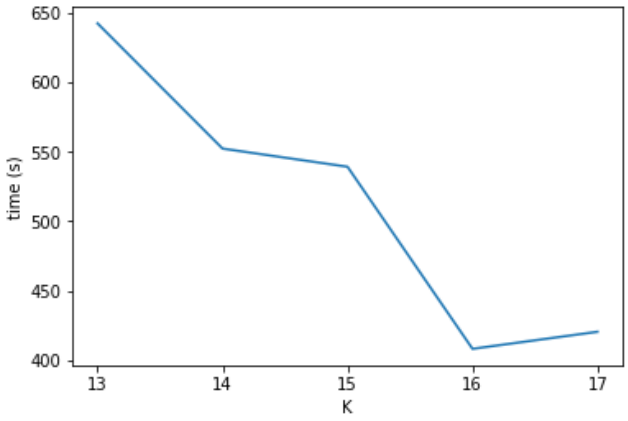

Figure 4.3: The running time versus the number of depots

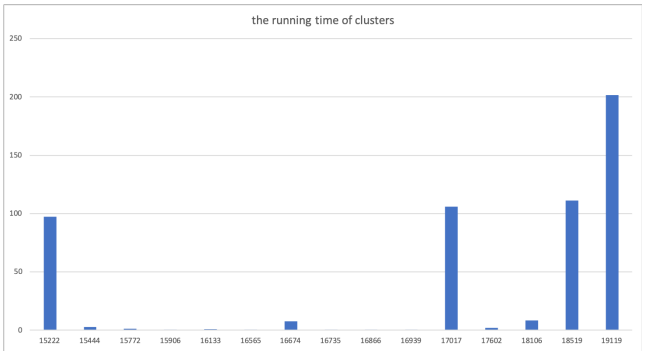

Figure 4.4: The running time of our algorithm ($K = 15$, center zipcode = 15222)

4.2 Decision on the optimal number of depots

In this section, we first pick K=5,10, and 15, and calculate the total cost under each number of depots. We find a unimodal pattern in the range $[10, 15]$, thus we further pick $N = 13, 14, 15, 16, 17$ to locate the exact optimum K, which is 15 in this setting. Figure 4.5 below shows the unimodal pattern.

We pick the optimum $K = 15$ as the solutions to the demand 2018 data, the total cost with its components are shown in the table 4.1 below.

Table 4.1: the total cost and its components

Routing cost	Vehicle number	Fixed cost of Vehicle	Total cost
643,425.65	15	164,250	807,675.65

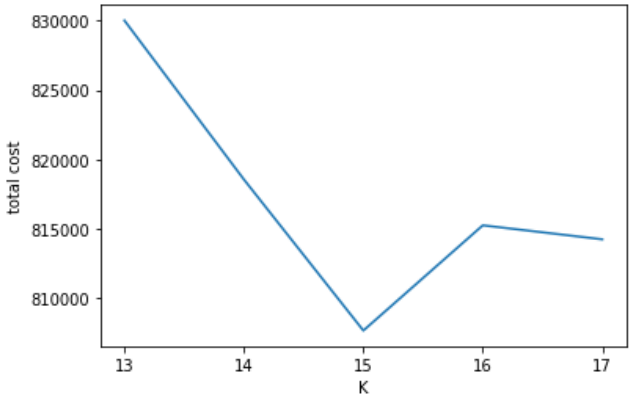

Figure 4.5: The number of depots versus total cost

4.3 The impact of considering the fixed cost of depots

In this section, we include the fixed cost of depots in our model. Since we have determined the optimal $K = 15$ in the previous part, we set 15 as the upper bound of our fixed-cost-included location model, the following table 4.2 compared the results under two different objectives.

Table 4.2: the impact of depots cost on objectives

Objective	Without depots cost	With depots cost
Depot number	15	10
Vehicle number	15	17
Center zipcode	15222 15444 15772 15906 16133 16565 16674 16735 16866 16939 17017 17602 18106 18519 19119	15332 15851 15901 16314 16917 17264 17582 17860 18503 19477
Depots cost	1619140	627800
Routing cost	643,425.65	747591.08
CO2 emissions (tons)	827.26	961.19

The two illustrations show the exact clustering results and the center locations. We use red squares to denote the center locations.

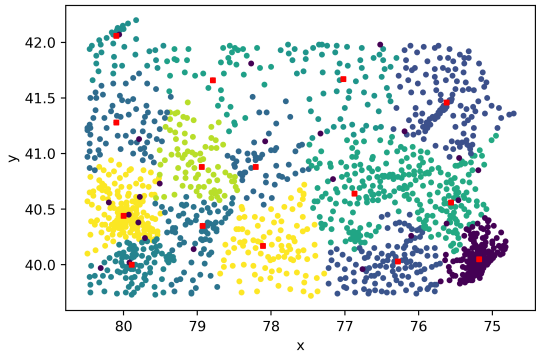

Figure 4.6: Depot locations and customer clustering without depot costs

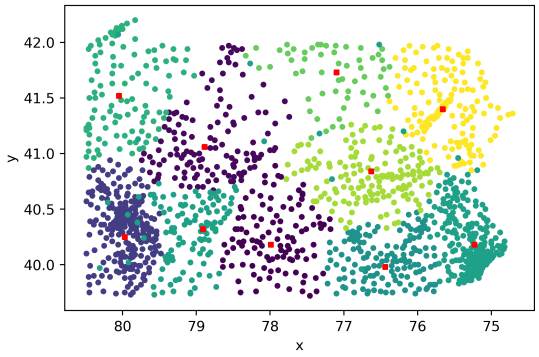

Figure 4.7: Depot locations and customer clustering with depot costs

These results suggest that when we consider the cost of depots, we are more likely to avoid picking locations with unrealistic expensive costs. Besides, the number of depots decreases to lower the costs of the depot further. Routing cost and CO2 emissions increase but can be offset by the savings from the cost of depots. In a nutshell, it is more profitable to include the cost of depots in our model.

4.4 The minimum number of depots

We start from $K = 1$ and keep increasing the number of depots. We consider the maximum distance in each cluster as the judgment towards whether we can serve every demand. In the setting of our problem, the maximum distance for the farthest demand point should be less than 220 miles (because the round trip should be less than 440 miles, which is constrained by the maximal total driving hours).

The following table 4.3 shows our process.

Table 4.3: process of determining the minimum number of depots

the number of depot	computational time (s)	maximal distance (mile)
1	160.10	261.17
2	170.466	230.27
3	259.061	153.56

Hence, we need at least three depots to serve all customers without considering the number of vehicles.

4.5 Robustness analysis

In this section, we compare the results of our model under 2018 demand data versus 2019 demand data. The results suggest the center locations are identical under two data sets with a small difference in customer clustering. It turns out that the difference is negligible because those customers have zero annual demand. Thus it can be assigned to any clusters with zero contribution to the total cost.

Table 4.4: the comparison of demand 2018 and 2019

	15332	15851	15901	16314	16917	17264	17682	17860	18503	19477	total
2018	2	2	2	1	1	2	2	2	1	2	17
2019	2	2	1	1	1	1	1	1	1	2	13

The results above suggest that our model has strong robustness in comparing the demand data 2018 with 2019. Since the exact demand and demand distribution vary slightly between these two years, this result proves the robustness and accuracy of our model to some extent.

4.6 The relationship between daily routing costs and aggregated daily demand

In this section, we consider the relationship between daily routing costs and aggregated daily demand. We use the result of 2018 demand data and select a cluster with center 15332. We pick 16 days of 2018 to observe the routing cost of these days versus the demand of these days, the following figure 4.8 shows the functional relationship between these two factors. We may utilize the regression method to obtain the functional relationship between the routing cost and the total demand.

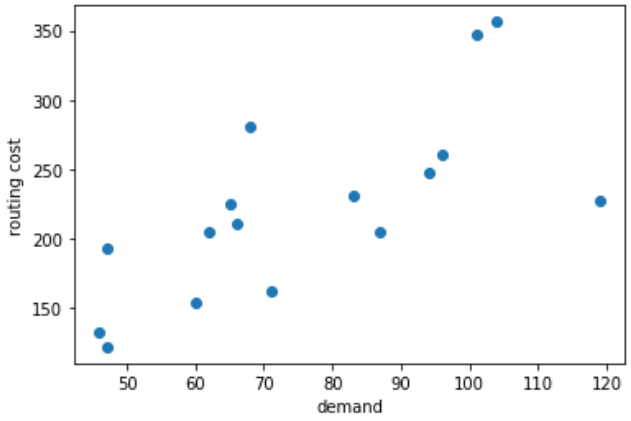

Figure 4.8: the aggregated daily demand versus the routing costs (demand data = 2018, center zipcode = 15332)

4.7 The average duration of a delivery trip

In this section, We consider the relationship between the number of depots and the average duration of a delivery trip. We observe the pattern by picking the number of depots K around the optimal neighbor (from 11 to 17). We can find the impact of the number of depots on the average duration of a delivery trip.

It is obvious to tell that the average duration of a trip will decrease as the number of depots increases, but the trend of decreasing will slow down after the number of depot reaches a cut-off value. This result sounds logical. Firstly when we set more depots, the demand nodes will be more likely covered by its neighbor, which explains the decrease in travel time. Since the travel distance between a node and its neighborhood nodes is

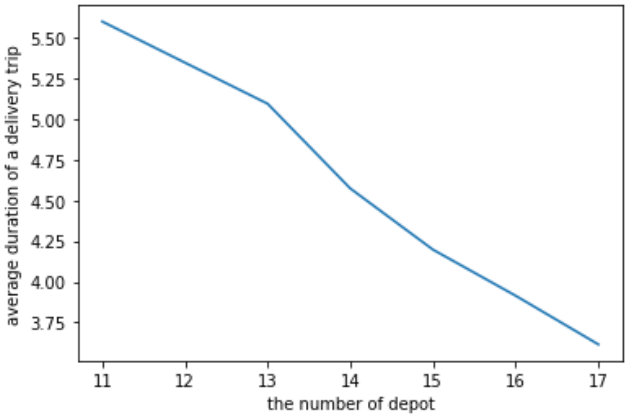

Figure 4.9: the average duration of a delivery trip versus the number of depot

approximately equal in value, when the number of depots reaches some threshold, the average travel time will not vary significantly.

5. Conclusions

In this report, we explore the methodologies for solving multi-depot multi-trip vehicle routing problem with identical time windows. We utilize historical information to provide an accelerated method to decide the number of depots and their locations. We also combine the merits of optimization and heuristic approaches to develop several routing models. Our experiment results suggest that it would be better to include the cost of depots as part of the objective, which yields more profitable solutions with an acceptable cost of the environment. The location and routing model can be implemented within a reasonable time frame and output relatively good solutions. Robustness is also one merit of our models.

Through intensive discussion, we think the following points worth further exploration.

1. The location model is fundamental to the final solutions. We utilize the statistics information to help develop our decisions on the depots. We believe there are still many better tools to exploit the data and provide more precise results.

2. Traditional optimization methods like integer programming, column generation are still less capable of dealing with large instances (greater than 30 customers) within a reasonable time frame. Many papers combine column generation and other combinatorial optimization ideas like set covering problems. We think a similar strategy can apply here to speed up the implementation.

3. To simplify the model, we cluster the customers into different groups with a depot serving all the demands. This formulation assumes that any two clusters cannot have any connection, which makes our model less flexible in dealing with some practical scenarios. For example, in certain days, the group depot A serves has high demand, but the group depot B serves has low demand, while a converse situation happens in other days. In our setting, we have to assign enough vehicles to both depots to satisfy the need for high demand delivery, but this can be done in a smart way that vehicles can travel among different depots and serve different groups. Another scenario is that allowing customers to be served by vehicles from another group. In a nutshell, it is preferable not to constrain the schedules within preset clusters, though this involves more complex models with longer running time.

4. Our routing models highly rely on the assumption that all vehicles have the same time windows, thus making it less applicable to general cases. Another topic is to develop models for general time windows.

Bibliography

[1] L. V. Snyder and Z.-J. M. Shen. *Fundamentals of Supply Chain Theory*. WILEY, Hoboken, 2nd edition, 2019.

[2] D. Simchi-Levi, X. Chen, and J. Bramel. *The Logic of Logistics*. Springer, New York, 3rd edition, 2013.

[3] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104-118, 2005.

[4] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119-139, 2005.

[5] D. Cattaruzza, N. Absi, and D. Feillet. The Multi-Trip Vehicle Routing Problem with Time Windows and Release Dates. *Transportation Science*, 50(2):1-18, 2016.

[6] Q. Hu and A. Lim. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276-286, 2014.

[7] H. Ling, A. Lim, and B. Rodrigues. Solving The Pickup And Delivery Problem With Time Windows Using "Squeaky Wheel" Optimization With Local Search. *AMCIS 2002 Proceedings*, Paper 319, 2002.

[8] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. A new exact algorithm to solve the multi-trip vehicle routing problem With time windows and limited duration. *4OR*, 12, 235-259, 2014.

[9] N. Azi, M. Gendreau, and J. Potvin. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178, 755-766, 2007.

[10] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[11] M. Yu, V. Nagarajan, and S. Shen. Improving column generation via random coloring and parallelization for vehicle routing problems. 2019.