CO 370: Deterministic OR Models

Fall 2023 Final Project

# The CVRP Problem

Prepared by:

Banat Khural

Dhruv Trivedi

*December 15, 2023*

## Introduction

In the realm of operational research, the pursuit of enhancing efficiency and cost-effectiveness is a challenge that finds its roots in real-world applications. This report delves into a critical operational challenge faced by a local restaurant in Waterloo, aiming to optimize its delivery logistics through the application of the Capacitated Vehicle Routing Problem (CVRP) principles. The significance of this endeavor lies not only in its academic appeal but also in its potential to drive tangible improvements in operational efficiency and cost reduction for the business.

The focal point of this project is the optimization of delivery logistics for a local restaurant that extends its services across diverse locations within Waterloo and Kitchener. The cruciality of this challenge lies in the management of the delivery fleet to ensure timely and cost-effective deliveries. Each delivery vehicle is constrained by a limited capacity, necessitating strategic planning to meet the demands of all customers without surpassing these predefined limits. The complexity lies in the variability of customer locations and demands, making it imperative to develop an optimized routing plan that minimizes the total travel distance and time, while concurrently ensuring efficient service to every customer.

This presents an interesting graph-esque problem with its roots in linear programming. However, the optimization of delivery logistics is not merely an academic exercise; it has profound real-world implications. A successful application of CVRP principles in this context holds the promise of streamlining delivery operations, reducing costs, and ultimately enhancing customer satisfaction. As the restaurant navigates through the intricate web of urban locations and varying demands, the optimized routing plan becomes a strategic tool for operational success.

The following report will dive into four main sections: defining the challenge to Restaurant as a business, gathering and cleaning the data, framing the problem as a CVRP problem, and finally a dive into the results of analyzing and formulation as an optimization problem that maximizes routes and drivers for maximum efficiency. The ultimate goal is to offer practical insights that can be implemented to revolutionize the restaurant's delivery service, exemplifying the transition between academic theory and real-world applicability.

**Challenge to Restaurant**

This report will first address the challenge of a local food concept restaurant for students, families, young professionals and anyone who desires convenient, home-style fresh comfort food (*Local Restaurant*, n.d.). The restaurant operates a delivery service encompassing various locations within Waterloo and Kitchener. The primary challenge lies in managing the delivery fleet to ensure timely and cost-effective deliveries. Each delivery vehicle has a limited capacity, necessitating careful planning to meet all customer demands without exceeding these limits. Inefficient routing not only results in increased operational costs, driven by higher fuel consumption and extended delivery times but also risks compromising the quality of customer service. In a competitive market where customer satisfaction is important, delays or inaccuracies in deliveries can lead to customer dissatisfaction and, subsequently, a potential loss of consumers. Competitors like Uber Eats and Doordash pose significant threats to a restaurant deciding to operate deliveries in-house. Moreover, the delivery fleet's mismanagement may strain the restaurant's overall resource allocation, affecting its ability to balance operational costs with revenue generation. Therefore, addressing the intricacies of the restaurant is not merely an

exercise in improving efficiency; it is a strategic imperative for sustaining and enhancing the business's competitiveness, customer loyalty, and bottom line.

**The Dataset**

For this report, the dataset has been extracted with the aid of an inside source, as one of the collaborators of this report is an existing employee of the restaurant. The dataset comprises 32,978 entries, each representing a delivery job, and is structured into 47 fields and it spans from January 1, 2022, to January 1, 2023. The one-year time frame provides a comprehensive view of the delivery operations for the entire year, allowing for a thorough analysis of seasonal variations, peak periods, and overall delivery efficiency. For the sake of maintaining confidentiality and avoiding privacy concerns, the first step in cleaning this dataset involved masking personal information of delivery orders such as names of customers, as well as removing information of the drivers themselves. The full dataset variables/columns of the dataset are listed in **Exhibit 1.**

**Data Cleaning**

After careful analysis of the dataset, it was found that some columns have mixed types or null values, indicating variability in data entry or optional fields, as well as 47 variables presented complexity in the amount of data to track. Therefore, the dataset was cleaned to remove irrelevant columns such as “Order Recipient Email” or “Order Status,” given the problem to solve was based around driver routes and times. A full list of variables/columns after cleaning can be found in **Exhibit 2.** With the remaining 28 columns, the data cleaning focused on a one-one correspondence procedure.

Given the driver information and unique ids, we identified the repetitive nature of some orders, as one driver was capable of multiple orders and hence the need for optimizing the drivers' routes. Therefore, a new column "Assigned Driver Name" as well as the "Job Assigned Driver UUID" columns became the main trackers for each driver. The first step in cleaning any dataset is to remove missing values, and this step was crucial in order to resolve issues with establishing a one-to-one correspondence. We initially checked for a one-to-one correspondence between 'Assigned Driver Name' and 'Job Assigned Driver UUID'. After removing the problematic entries (associated with 'PP'), a one-to-one correspondence was established (See **Exhibit 3**). We then created a unique identifier for each driver by concatenating 'Assigned Driver Name', 'Job Assigned Driver User UUID', and 'Job Assigned Driver UUID'. This identifier was converted into a numeric code, prefixed with 'Driver_', to form the 'Drivers Unique ID' and hence removing the name and UUID columns for better consolidation of the data. Throughout this process, the goal was to ensure that each driver had a unique and identifiable code in the dataset while maintaining the integrity and correspondence of the data. Our final dataset resulted in a "Drivers Unique ID" column, for which we verified one-to-one correspondence.

**Data Manipulation and Analysis**

Given the constraints of our software capabilities as well as the extensive routes that could be formed from our 32,978 entries, we proceeded with subsampling, a method that reduces data size by selecting a subset of the original data. Our subsampling focused on finding the number of deliveries in a given week to use the best sample possible, and thus the following steps were taken:

1. Convert the 'Job Promised Delivery At' column to the DateTime format to facilitate date-based operations.
2. Extract the week number and day of the week from the delivery date.
3. Group the data by the week number and day of the week, and count the number of deliveries for each day.
4. For the first week in the dataset, identify the day with the maximum, minimum, and mean number of deliveries.

Results of the analysis for each week with the minimum, maximum, and average were calculated and an example snippet for Week 1 can be found in **Exhibit 4.**

With the subsampling done, the focus was to calculate distance as within the columns, one of the most significant variables is time it took for an order to be delivered which would aid in our understanding of route optimization. Within our data, we grouped “Week Number”, “Day of Week”" and “Job Group UUID” to calculate total kilometers traveled for each round of deliveries. It should be noted here that the distance used was the Crowfly Distance in km, as that was the distance provided in our dataset. After analysis of these distances, we compiled the total kilometers for each round of delivered orders**.** Given that an order route would consist of going to the delivery address and returning back to the restaurant (if there is only 1 address in the route), the code accounts for doubling the distance and sorting the data such that a Round number is assigned to reflect the number of rounds taken to and fro the delivery locations. For further cleaning, the final file accounts for missing and duplicate values within the relevant columns.

**Geocoding**

After the lengthy analysis was conducted and the final, cleaned dataset was obtained, we returned to address the original issue the restaurant faces: optimization of routes. For this focus on the routes, it was increasingly important to code and augment the dataset given the geographical coordinates of delivery addresses. With Google Maps API and Python, our team was able to manipulate the latitudes and longitudes of each delivery address such that each address input could be manipulated into columns of coordinates for the sake of mapping the routes using numerical values, with an update reflecting the geomapping of the delivery addresses.

Our biggest hurdle with handling this data was the complexity and size of the dataset, leading to a final decision to subsample 20 entries to work with and demonstrate optimizing route abilities. This restriction is further discussed in our Errors and Risks section of the report.

**Formation of the CVRP**

With the final set of 20 entries filtered to January 8, 2022 and reflecting valid geographic information and non-empty values for crucial variable columns, our report will proceed to describe the formulation of the CVRP. Given that each vehicle can carry up to 6 orders and the restaurant has a fleet of 5 vehicles, we establish the starting point to be the restaurant location at 130 Columbia St W, Waterloo, ON (43.479584307828894, -80.53734239887197). The primary goals are to minimize the total distance traveled and ensure that each vehicle's capacity is not exceeded.

The essential terms to the CVRP are defined in **Exhibit 5**. Mathematically, CVRP can be modeled using graph theory, where the depot and customers are nodes in a graph, and the routes are edges. The objective is to find the shortest paths that connect these nodes, considering the capacity constraints. This directly correlates to the linear programming models we have explored in CO 370 this semester, and thus we have utilized the aid of Google OR-Tools to solve the CVRP problem presented here. At its core, CVRP is solved using combinatorial optimization techniques. The solver (like the one in Google OR-Tools) typically uses heuristics to efficiently explore the possible routes and find an optimal or near-optimal solution. In summary, solving VRP involves a blend of graph theory, combinatorial optimization, and practical application of geographic and routing data. The use of tools like OR-Tools and visualization libraries bridges the gap between theoretical optimization and practical application, providing tangible solutions for complex logistical challenges.

Our code addresses the Vehicle Routing problem and presents an issue/threat to the supply chain management of the restaurant. The VRP involves determining the most efficient routes for a fleet of vehicles to deliver goods to a set of locations. Using all of this information, we formulate an optimization problem, with details below:

*Objective Function:* Minimize the total distance traveled by all vehicles in the fleet.

*Variables:*

- A decision variable to represent if the vehicle i travels to address j.
- A variable to represent the load of vehicle i after visiting address j.

*Constraints:*

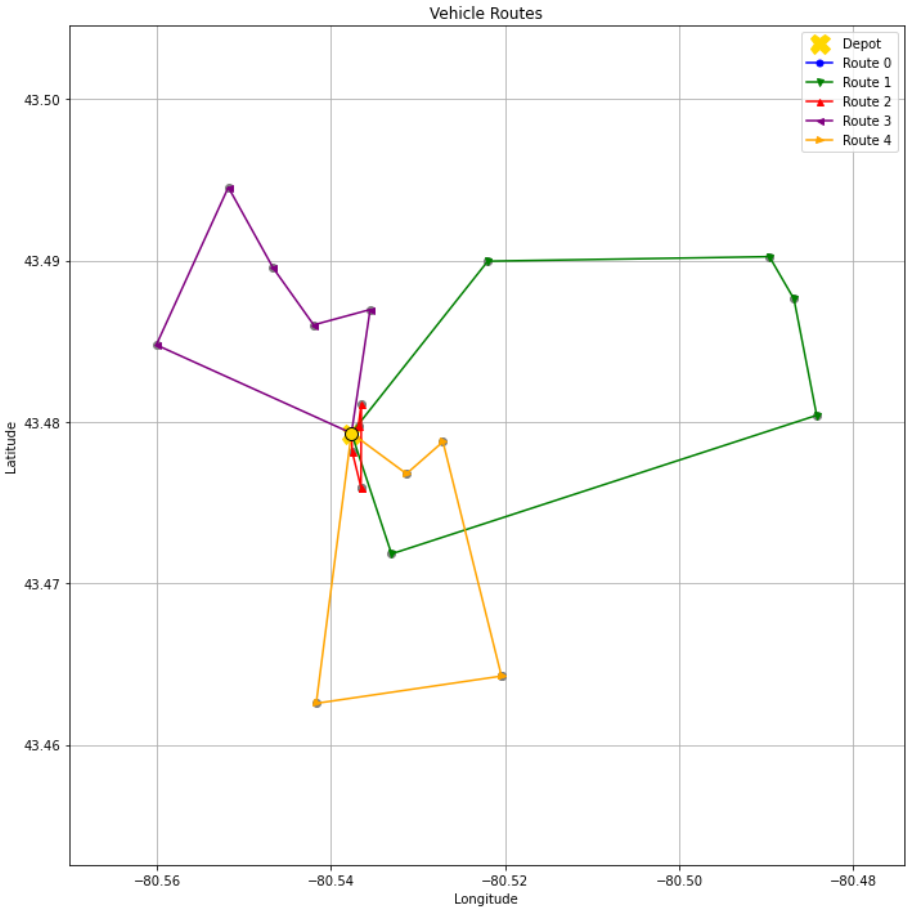
- Each vehicle must start and end at the restaurant.

- Each address must be visited exactly once, by exactly one vehicle.
- Total load (orders carried) of each vehicle must not exceed capacity.

The mathematical representation of this formulation can be found in **Exhibit 6**.

**Results**

With our dataset and OR tools, we are able to produce a "shortest possible route" for each vehicle in the fleet given constraints such as not returning to the restaurant until all deliveries are complete, as well as optimizing the routes within the Waterloo region it serves. See below for a graphical representation of the solution found, as well as a geographic representation that can be found in **Exhibit 7**. The results for each car from the code can also be found in **Exhibit 8.**



**Errors and Risks**

It is important to note that this entire formulation is dependent on the dataset we cleaned and formulated, which was obtained from the restaurant itself. Given that we are not working directly with the restaurant while formulating this report, there is a large possibility of reporting error, as we cannot validate all of the data points. Another error to be mindful of lies in our manipulation of data: while removing missing or NA values is a common practice while cleaning data, we have no way to track any patterns or reasoning for the missing value and thus removing those entries may have skewed our dataset. Lastly, we faced issues in our computational power and commercial licensing access to be able to manipulate the entire dataset, and thus our subsampling of 20 entries could pose an issue as we are unable to analyze all the data and find optimized routes for larger sets of data.

In terms of risks to the company, while optimization of fleets and routes is going to solve crucial supply chain and logistics issues, it would be equally significant to understand whether this route optimization validates the number of deliveries and cars in their arsenal: for example, if the route optimization we have presented reflects larger delays in orders delivered or a restriction on number of deliveries made, restaurant may use this information to expand their delivery fleet or consider completely outsourcing deliveries with Uber Eats/Doordash as these services already pose a competitive threat.

Overall, operations research is important to companies like the restaurant looking to optimize their logistics and our analysis and determinations of the CVRP problem here present a real-life

solution that is applicable far beyond the classrooms of CO 370, and into the business considerations of many companies across the world.

**Exhibits**

**Exhibit 1: List of All Variables/Columns**

1. Store UUID (object): A unique identifier for the store.
2. Store Name (object): Name of the store.
3. Store Partners External ID (float64): External ID associated with store partners.
4. Order UUID (object): Unique identifier for the order.
5. Order Status (object): Status of the order (e.g., completed, cancelled).
6. Order Partners Unique Internal Order ID (object): A unique internal ID for the order, specific to partners.
7. Order Recipient Name (object): Name of the person receiving the order. Order
8. Recipient Email (object): Email of the order recipient.
9. Order Recipient Phone Number (object): Phone number of the order recipient.
10. Order Recipient First Line Address (object): First line of the address for the order delivery.
11. Order Recipient City (object): City of the order recipient.
12. Order Recipient State (object): State of the order recipient.
13. Order Recipient Postal Code (object): Postal code of the order recipient.
14. Order Recipient Country (object): Country of the order recipient.
15. Delivery Instructions (object): Instructions for delivery.
16. Order Total (float64): Total amount of the order.
17. Subtotal (float64): Subtotal amount of the order.
18. Tip Amount (float64): Tip amount for the order.
19. Delivery Fee (float64): Fee for delivering the order.
20. Tax (float64): Tax amount for the order.
21. Created At (object): Creation date and time of the order.
22. Requested For (object): Requested date and time for the order.
23. Completed At (object): Completion date and time of the order.
24. Cancelled At (object): Cancellation date and time of the order, if applicable.
25. Job UUID (object): Unique identifier for the job.
26. Job Status (object): Status of the job.
27. Job Partners External ID (float64): External ID associated with job partners.
28. Job Type (object): Type of the job.
29. Job Created At (object): Creation date and time of the job.
30. Job Requested For (object): Requested date and time for the job.
31. Job Completed At (object): Completion date and time of the job.
32. Job Cancelled At (object): Cancellation date and time of the job, if applicable.
33. Collector UUID (object): Unique identifier for the collector.
34. Collector Name (object): Name of the collector.
35. Collector External ID (float64): External ID for the collector.

36. Vehicle UUID (object): Unique identifier for the vehicle.
37. Vehicle Type (object): Type of the vehicle used.
38. Job Group UUID (object): Unique identifier for the job group.
39. Dropoff Sequence Number (float64): Sequence number for the dropoff.
40. Special Instructions (object): Any special instructions for the job.
41. Crowfly Distance (KM) (float64): Distance in kilometers (as the crow flies) for the delivery.
42. Crowfly Distance (MI) (float64): Distance in miles (as the crow flies) for the delivery.
43. Bearing (float64): Bearing for the delivery.
44. Odometer At Collection (float64): Odometer reading at the time of order collection.
45. Odometer At Delivery (float64): Odometer reading at the time of order delivery.
46. Odometer Upon Return To Store (float64): Odometer reading upon return to the store.
47. Fleetshare (object): Indicates if the delivery was part of a fleetshare system.

**Exhibit 2: List of Variables After Data Cleaning**

1. Order UUID
2. Order Status
3. Order Recipient Name
4. Order Recipient Phone Number
5. Order Recipient First Line Address
6. Order Recipient Zipcode
7. Order Recipient City
8. Order Description
9. Order Placed At Time
10. Job Created At
11. Job UUID
12. Job Promised Delivery At
13. Job Picked Up At
14. Job Delivered At
15. Job Status
16. Job Delivery Status
17. Job Assigned Driver First Name
18. Job Assigned Driver Last Name
19. Job Assigned Driver User UUID
20. Job Assigned Driver UUID
21. Geoverified
22. Job Group UUID
23. Dropoff Sequence
24. Number Crowfly Distance (KM)

25. Bearing
26. Odometer At Collection
27. Odometer At Delivery
28. Odometer Upon Return To Store

**Exhibit 3: One-One Correspondence Code**

```
# Filter out entries where either 'Assigned Driver Name' or 'Job Assigned Driver UUID' is NaN (missing)
filtered_data = data.dropna(subset=['Assigned Driver Name', 'Job Assigned Driver UUID'])

# Remove rows where 'Assigned Driver Name' is 'Prashant Patel' as causing issues to build one-one correspondence
data_cleaned = filtered_data[filtered_data['Assigned Driver Name'] != 'Prashant Patel']

# Function to check one-to-one correspondence
def check_one_to_one(df, col1, col2):
    return df.groupby(col1)[col2].nunique() == 1

# Check for one-to-one correspondence in the cleaned data
driver_uuid_correspondence_cleaned = check_one_to_one(data_cleaned, 'Assigned Driver Name', 'Job Assigned Driver UUI
print("One-to-one correspondence:", driver_uuid_correspondence_cleaned.all())

# Display the head of the cleaned dataframe
print(data_cleaned.head())
```

```
One-to-one correspondence: True
```

**Exhibit 4: Week 1 Max/Min/Avg Deliveries**

```
  "(   Week Number Day of Week  Delivery Count\n",
  " 2           1    Saturday             132,\n",
  "    Week Number Day of Week  Delivery Count\n",
  " 5           1     Tuesday              76,\n",
  "    Week Number Day of Week  Delivery Count  Distance from Mean\n",
  " 4           1    Thursday             100            0.428571)"
]
```

**Exhibit 5: CVRP Essential Terms**

1. Depot: The starting and ending point for all routes.
2. Customers: Locations where deliveries need to be made.
3. Vehicles: A fleet with a fixed number of vehicles.
4. Capacity Constraint: Each vehicle has a maximum capacity (number of deliveries it can carry).
5. Route Optimization: Minimizing the total distance traveled by all vehicles.

**Exhibit 6: CVRP Formulation**

Decision Variables:

1. $x_{ijk}$: Binary variable, equal to 1 if vehicle *i* travels directly from location *j* to location *k*, and 0 otherwise.
2. $Load_{ij}$: Continuous variable representing the load of vehicle *i* after visiting location *j*.

Objective Function: min $\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k\neq j,k=1}^{m} x_{ijk} \times Distance_{jk}$

1. n is the number of vehicles
2. m is the number of addresses to deliver to
3. $Distance_{jk}$ is the distance between address j and address k

Constraints:

1. Visit each location exactly once: $\sum_{i=1}^{n}\sum_{k\neq j,k=1}^{m} x_{ijk} = 1\ \forall j \neq Depot$
    a. Depot = starting point = Local Restaurant
2. Start and end at Depot: $\sum_{j=1,\, j\neq Depot}^{m} x_{i,Depot,j} = 1\ \forall i,\ \sum_{k=1,\, k\neq Depot}^{m} x_{i,k,Depot} = 1\ \forall i,$
3. Load of each vehicle should not exceed capacity: $Load_{ij} \leq Capacity_i\ \forall i, j$
4. Load changes after each delivery:
   $Load_{ik} = Load_{ij} + Order_k \times x_{ijk}\ \forall i, j, k,\ Load_{i,Depot} = 0\ \forall i$
5. The optimized route should not have any subtours/pass through the depot:
   $Load_{ij} + Order_k \times x_{ijk} - Load_{ik} \leq Capacity_i \times (1 - x_{ijk})\ \forall i, j, k \neq Depot,$
   $Load_{ij} \geq Order_j\ \forall i, j \neq Depot$
   $Load_{i,Depot} = 0\ \forall i$

**Exhibit 7: Geographic Representation of Shortest Route**

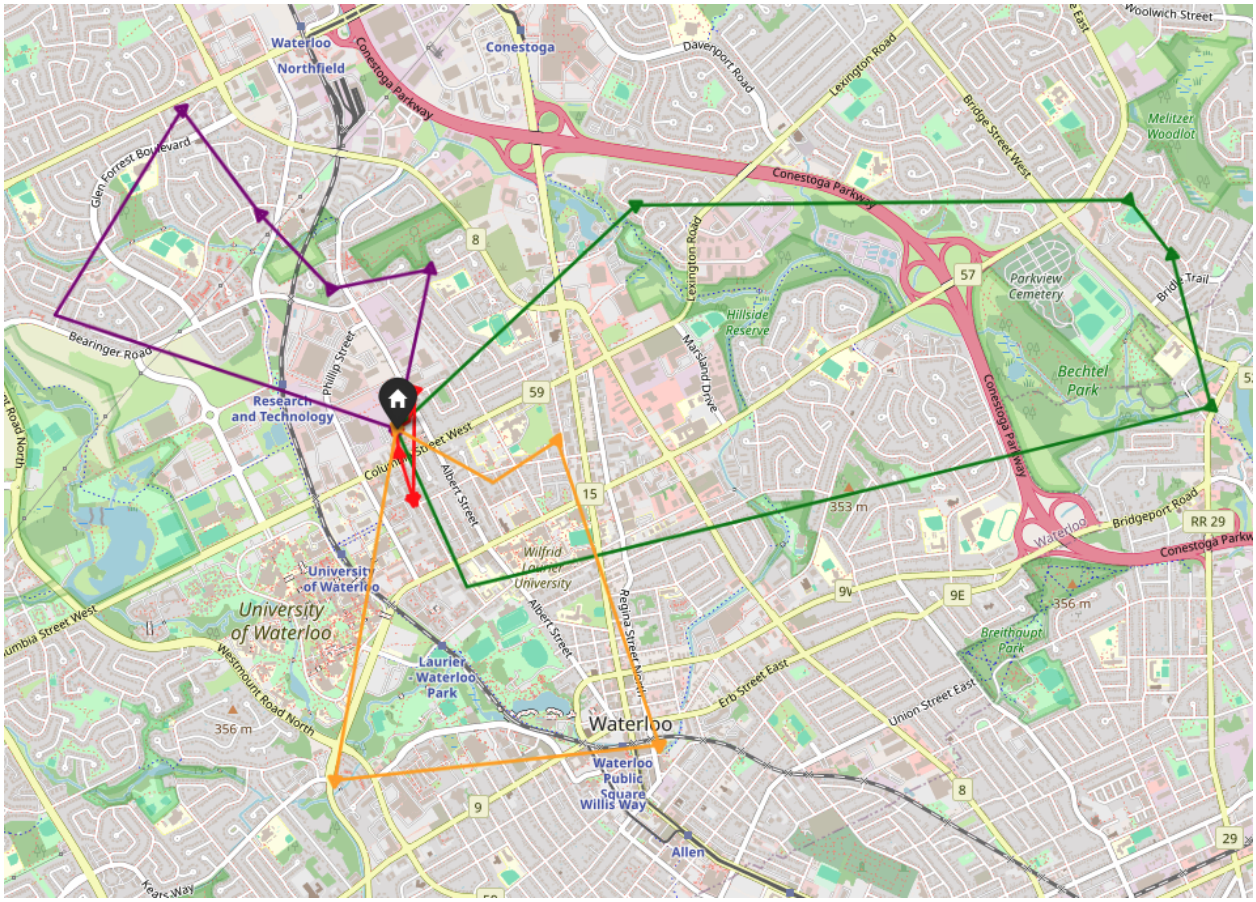



**Exhibit 8: Code Results**

```
Route for vehicle 0:
 0 -> 0
Distance of the route: 0m
Load of the route: 1

Route for vehicle 1:
 0 -> 5 -> 6 -> 9 -> 11 -> 2 -> 0
Distance of the route: 14069m
Load of the route: 6

Route for vehicle 2:
 0 -> 17 -> 10 -> 14 -> 19 -> 20 -> 0
Distance of the route: 1707m
Load of the route: 6

Route for vehicle 3:
 0 -> 15 -> 18 -> 16 -> 12 -> 3 -> 0
Distance of the route: 9226m
Load of the route: 6

Route for vehicle 4:
 0 -> 7 -> 8 -> 1 -> 4 -> 13 -> 0
Distance of the route: 8243m
Load of the route: 6

Total distance of all routes: 33245m
Total load of all routes: 25
```

## References

(n.d.). Capacitated Problem. Retrieved December 14, 2023, from https://cse.ucdenver.edu/~cscialtman/complexity/Presentation.pdf

*[1811.07403] A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer*. (2018, November 18). arXiv. Retrieved December 14, 2023, from https://arxiv.org/abs/1811.07403

*[2304.09629] Quantum-Assisted Solution Paths for the Capacitated Vehicle Routing Problem*. (2023, April 19). arXiv. Retrieved December 14, 2023, from https://arxiv.org/abs/2304.09629

*Capacitated Vehicle Routing Problem formulation — AIMMS How-To*. (2020, August 31). AIMMS How-To. Retrieved December 14, 2023, from https://how-to.aimms.com/Articles/332/332-Formulation-CVRP.html

*Capacity Constraints | OR-Tools*. (n.d.). Google for Developers. Retrieved December 14, 2023, from https://developers.google.com/optimization/routing/cvrp

Chen, Y. (2023, June 2). *[2306.01826] Approximation schemes for capacity vehicle routing problems: A survey*. arXiv. Retrieved December 14, 2023, from https://arxiv.org/abs/2306.01826

Chidarala, N. (2023, October 20). *Capacitated Vehicle Routing Problem (CVRP) Optimization using Google-OR Tools and Python*. Medium. Retrieved December 14, 2023, from https://medium.com/@nag96.chidara/capacitated-vehicle-routing-problem-cvrp-optimization-using-google-or-tools-and-python-7848fb5ffd16

*Get started with the Distance Matrix API*. (n.d.). Google for Developers. Retrieved December 14, 2023, from https://developers.google.com/maps/documentation/distance-matrix/start

*Google OR-Tools*. (n.d.). Google for Developers. Retrieved December 14, 2023, from https://developers.google.com/optimization

*How to use the ortools.constraint_solver.routing_enums_pb2 function in ortools*. (n.d.). Snyk. Retrieved December 14, 2023, from https://snyk.io/advisor/python/ortools/functions/ortools.constraint_solver.routing_enums_pb2

*How to use the ortools.constraint_solver.routing_enums_pb2 function in ortools*. (n.d.). Snyk. Retrieved December 14, 2023, from https://snyk.io/advisor/python/ortools/functions/ortools.constraint_solver.routing_enums_pb2

Kim, K. (2020, April 29). *Capacitated Vehicle Routing Problem (CVRP) with Python+Pulp and Google Maps API*. Medium. Retrieved December 14, 2023, from https://medium.com/jdsc-tech-blog/capacitated-vehicle-routing-problem-cvrp-with-python-pulp-and-google-maps-api-5a42dbb594c0

Mortenson, N. (2022, May 3). *How to Solve the Capacitated Vehicle Routing Problem*. OptimoRoute. Retrieved December 14, 2023, from https://optimoroute.com/capacitated-vehicle-routing-problem/

*sakshisingh1809/CVRP-with-GA: Capacitated Vehicle Routing Problem with Machine Learning*. (n.d.). GitHub. Retrieved December 14, 2023, from https://github.com/sakshisingh1809/CVRP-with-GA

Scalia, B. (n.d.). *The Vehicle Routing Problem: Exact and Heuristic Solutions*. Towards Data Science. Retrieved December 14, 2023, from

https://towardsdatascience.com/the-vehicle-routing-problem-exact-and-heuristic-solutions-c411c0f4d734

*Vehicle Routing Problem | OR-Tools*. (2023, January 16). Google for Developers. Retrieved December 14, 2023, from https://developers.google.com/optimization/routing/vrp#distance_matrix_api

*What is Capacitated Vehicle Routing Problem (CVRP) | An Overview*. (2023, May 12). Upper Route Planner. Retrieved December 14, 2023, from https://www.upperinc.com/glossary/route-optimization/capacitated-vehicle-routing-problem-cvrp/