Electric Vehicle Routing Problem

15.093: Optimization Methods

Master of Business Analytics
MIT Sloan School of Management

Authors:

Seth Chatterton
Srikaran Boya

December 8, 2023

1 Problem Statement

Today there are many options for people to have items delivered to them, for example through Amazon, UberEats, and DoorDash. Currently many of the vehicles that perform this delivery are powered with fossil fuels, but as more electric vehicles begin to appear on roads [1] it will be more important to solve the problems unique to these electric vehicles. We plan to take into account not only the traditional supply locations and demand locations applicable to standard last mile delivery, but also electric vehicle charging stations and electric vehicle charging requirements such as range and time to recharge.

We plan to look at a fleet of electric vehicles that make all their deliveries in a single day, similar to Amazon or USPS making all of the required deliveries for that day, with all deliveries to be made known at the beginning of the day. We plan to minimize the maximum time taken to complete all deliveries amongst all vehicles, with some weighted cost of the average energy usage of each vehicle in the fleet. We call this problem the Multiple Vehicle Routing Problem (MVRP) with Battery Constraints.

1.1 Formulation: MVRP with Battery Constraints

We would like to minimize a weighted combination of the length of the longest delivery time by a vehicle and the average energy required by a vehicle to complete all deliveries.

$$\min \max \left\{ \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ijvt} + \tau_{R_{vt}} \ \forall v \in 1...V \right\} + w \cdot \frac{1}{V} \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} e_{ij} x_{ijvt}$$

Please see the appendix for the full formulation, including linearization of the objective function and all constraints. The constraints referenced in the next paragraph are numbered in the full formulation in the appendix.

In plain english, this formulation allows us to ensure that vehicles visit every delivery point at least once (constraint 6), that all vehicles exist and must continue to exist without disappearing or creating new vehicles (constraints 1, 2, 5), and that vehicles must begin and end their journey at the supply depot (constraints 3, 4). These constraints are all shared by the standard MVRP. We add additional constraints to model the battery. Constraints 7, 8, 9, and 10 ensure that the battery level is exactly what it should be at every timestep, taking away battery range when we travel from point to point, and allowing recharging if a vehicle is at a charging station. For each of the battery constraints, the big-M terms remove the constraints if that constraint for that vehicle does not apply, i.e. if we are not at a charging station (modeled by $M \cdot R_{ij} \cdot x_{ijv1}$) or we are not at a specific location ($M \cdot (1 - x_{ijvt})$). The formulation does not require that a vehicle recharge if it is at a recharge station, and also allows for recharge stations to be at delivery locations. Constraint 11 sets the minimum amount of time we have to stay at a charging station to fill our battery up by some amount from one timestep to the next, which is proportional to the amount of battery that is chosen to be filled at that station.

One aspect of this problem that is different to the standard MVRP or the TSP problem in general is that we allow vehicles to visit a location more than once, and allow vehicles to visit locations that other vehicles have visited. This allows for recharging stations to be used multiple times if needed.

Some limitations of this formulation are that since we need to know the battery level and time spent recharging at every timestep, we require $N \cdot N \cdot V \cdot T$ total binary variables, which is T times as many binary variables as the standard MVRP. This makes an exact solution more difficult to find. Additionally, we need to specify the number of timesteps T that we want to model in advance. If we make T too large, the problem will be solved much more slowly, but if we make T too small, we will be constraining our problem more than it should be constrained, potentially missing out on solutions. In our experiments, we set $T = N$, since a single vehicle visiting every point will require N stops, but we use multiple vehicles so we will require fewer than N total timesteps when the delivery work is split amongst all vehicles. This also leaves some space for vehicles to use recharge stations without visiting more than N stops.

2 Data

We make use of a synthetic dataset containing locations of delivery points, locations of charging stations, and locations of supply depots, containing x and y coordinates. We generated this data by uniformly sampling points in a 10km by 10km area. From these coordinates, we create an $N \times N$ distance matrix, which we then convert to an $N \times N$ time cost matrix by dividing by the speed of the vehicles in kilometers per minute. In our case, we choose 1 km per minute, so the time cost matrix is equal to the distance matrix numerically. This is c in our formulation. We also create an energy cost matrix, which represents the energy required to travel from point to point. We set this numerically equal to the distance matrix, with units of kilometers of battery range required to travel that arc. This is e in our formulation. This energy matrix could be modified if we wanted to model elevation changes, or traffic that modifies the energy usage along each arc.

3 Implementation

3.1 Gurobi Implementation

We implemeted the model in Julia's JuMP package using Gurobi as the solver. The code can be seen in the appendix. The code has trouble converging for large values of delivery points because of the number of binary variables we have to consider. By setting time limits on the execution time we can still find very good solutions, or even the optimal solution that has not been proven to be optimal yet due to the running time constraints. Results can be seen in the next section.

3.2 Heuristic Algorithms

To handle increasing the scale of the problem, we implemented several heuristic solutions to find reasonable, but probably not optimal solutions. The algorithms we applied include

1. **2-Opt algorithm:** It iteratively reverses the order of a subset of tour's edges to decrease the total distance travelled, aiming to find a shorter overall route.

2. **Genetic Algorithm:** An adaptive heuristic algorithm, inspired by principles of natural evolution and genetics. They simulate the process of natural selection, where the fittest individuals are selected for reproduction in order to produce offspring for the next generation.

3. **GA with 2-opt:** A variant of GA where the 2-opt feature is added as one of the parameters in the mutation process of GA.

Similar to the Gurobi implementation, we tested the above algorithms for different datasets mentioned in the next section.

4 Computational Experiments

We experimented with 4 different datasets. A small and large dataset for a regular routing problem and battery constrained routing problem. We compared Gurobi and heuristic approaches for the regular cases, and compared Gurobi with a greedy solution for the battery constrained small dataset case. Given the results for the above datasets we also evaluated our objective for a large battery constrained dataset.

- **small** : 9 demand points, 3 vehicles and a supply point (also a depot)
- **large**: 49 demand points, 5 vehicles and a supply point (also a depot)
- **B-small**: 9 demand points, 3 vehicles, 5 recharge stations and a supply point (also a depot) with a battery capacity of 14
- **B-large**: 31 demand points, 5 vehicles, 16 recharge points and a supply point (also a depot) with a battery capacity of 25

Table 1: Comparison of objective values for different methods

Dataset	Greedy	Gurobi	GA	2opt	GA-2opt
Small	35.35	17.52	17.52	17.83	17.54
Large	70.04	29.10	41.76	27.99	31.47
B-Small	32.14	28.63	-	-	-
B-Large	-	18.85	-	-	-

Table 2: Comparison of time taken (iterations) for different methods

Dataset	Greedy	Gurobi	GA	2opt	GA-2opt
Small	1ms	30s	7.6s (10^3 it)	854ms (10^5 it)	8s (10^3 it)
Large	1ms	1800s	24s (10^3 it)	22.8 s (10^6 it)	56s (10^3 it)
B-Small	1ms	120s	-	-	-
B-Large	-	3600s	-	-	-

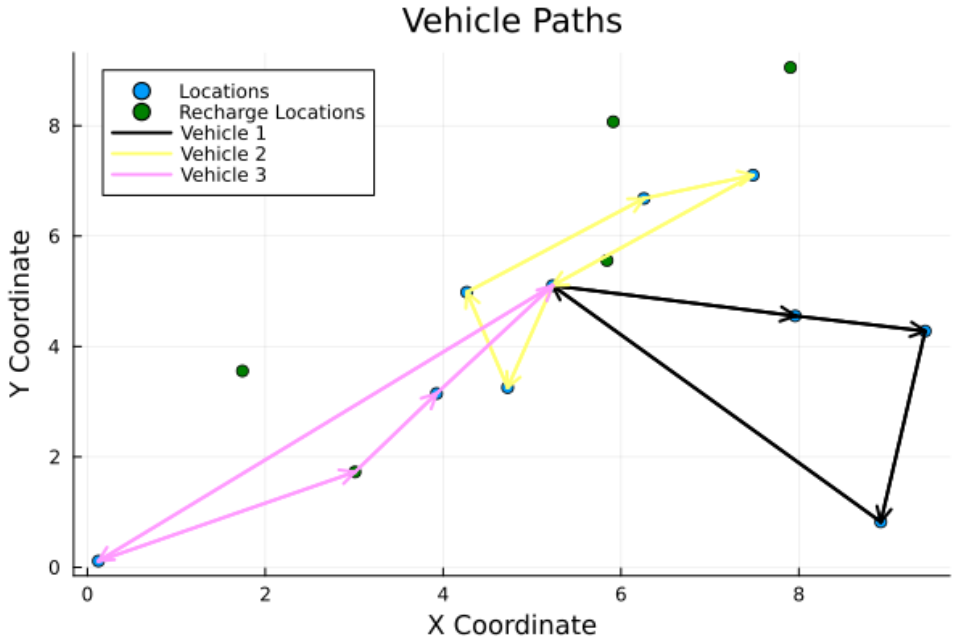

Figure 1: The solution found by Gurobi for the small problem with battery constraints (9 delivery points, 1 supply depot, 5 charging stations, 3 vehicles, 14 km battery range per vehicle)

5 Discussion and Impact

The computational experiments conducted across four different datasets offer a comprehensive view of the performance landscape of various optimization methods in routing problems. The datasets enable evaluation of algorithm efficiency and effectiveness across varying problem complexities.

Gurobi demonstrated robust performance across both small and large datasets, maintaining optimal objective values with reasonable computation times. This underscores Gurobi's strengths as a solver in dealing with both the scale and the complexity of routing problems. In particular, its ability to produce a reasonable solution for the large dataset within 1800s seconds while maintaining a competitive objective function value indicates its practicality for real-world scenarios where time and accuracy are crucial.

The Genetic Algorithm (GA) showcased equivalent performance to Gurobi in the small dataset with a significant reduction in time, only 7.6 seconds. However, its limitation became apparent in the large dataset scenario where the time increased substantially to 245 seconds, and the objective function value deteriorated. This suggests that while GA is highly efficient for smaller-scale problems, its efficacy diminishes with increasing problem size.

The 2-opt heuristic, traditionally celebrated for its simplicity and speed, displayed a notable increase in computation time from the small to the large dataset, which may be indicative of its less-than-ideal scalability. Nevertheless, its swift execution time of 854ms for the small dataset and ability to deliver a competitive objective function value in the large dataset within 22.8 seconds reveal its utility as a quick, preliminary approach to routing problems.

The hybrid GA-2opt method did not demonstrate a clear advantage over the individual methods. While it performed slightly better than GA in the small dataset, it did not outperform Gurobi and

even took longer (56 seconds) in the large dataset. This may indicate that the additional complexity of the hybrid method does not translate into proportional performance gains.

For the specialized B-small dataset, Gurobi's versatility is again highlighted, as it provided solutions within a reasonable time frame of 120 seconds.

The Greedy algorithm, while delivering instantaneous results (1ms), suffered in terms of solution quality, as seen in its higher objective function values across all datasets. This trade-off between speed and accuracy is characteristic of heuristic methods, which may be appropriate for very time-sensitive situations where immediate, though not optimal, solutions are acceptable.

We ran one additional experiment of the battery constrained MVRP using our Gurobi formulation to see how it would handle a larger problem with more deliveries and more charging stations. Visually, the solution looks reasonable, even though we only ran it under a time constraint of 3600 seconds and it did not converge in time. See Figure 8 in the appendix for a visualization of this run. One thing to note when comparing larger problems to smaller problems is that since our formulation uses the average energy taken per vehicle as opposed to the total energy used among all vehicles, larger problems do not necessarily have larger objective values if the vehicle fleet size is also larger. This is why the larger problem can have a smaller objective value.

The real-world impact of the optimization methods analyzed in our study is significant, particularly in the last mile delivery [2] for the logistics and transportation sectors. For instance, the efficiency of Gurobi in handling complex routing problems can translate into substantial cost savings and improved service levels for logistics companies. The ability of the Genetic Algorithm to quickly solve smaller-scale problems could be leveraged for on-demand delivery services, where rapid planning is essential. Meanwhile, the simplicity and speed of the 2-opt heuristic may benefit applications in dynamic routing, where routes need to be recalculated on-the-fly in response to changing conditions. We recommend using the exact Gurobi formulation solver for smaller problems with fewer deliveries, where tiny differences in delivery times and energy uses could make a large difference, and one of the heuristic approaches for problems where Gurobi can not find a good solution in a reasonable time.

These algorithmic insights could guide businesses in choosing the right approach for their operational scale and customer service objectives, ultimately leading to enhanced resource utilization, reduced environmental impact through optimized routes, and heightened customer satisfaction by ensuring timely deliveries.

6 References

References

[1] IEA (2023). Global EV Outlook 2023: Trends in Electric Light-Duty Vehicles.

[2] Capgemini Research Institute (2019). The Last-Mile Delivery Challenge.

7 Appendix

7.1 Full MVRP with Battery Constraints Formulation

We would like to minimize a weighted combination of the length of the longest delivery time by a vehicle and the average energy required by a vehicle to complete all deliveries.

$$\min \max \left\{ \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ijvt} + \tau_{R_{vt}} \ \forall v \in 1...V \right\} + w \cdot \frac{1}{V} \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} e_{ij} x_{ijvt}$$

$v \in 1...V$ represents a vehicle in the set of vehicles
$i, j \in 1...N$, where N denotes the total number of locations including supply, demand, transhipment, and charging nodes
$t \in 1...T$, where T denotes the maximum number of timesteps to consider
$x_{ijvt} \in \{0, 1\}$ represents whether the path from location i to location j is traveled by vehicle v at timestep t, $\forall i, j \in 1...N,\ \forall v \in 1...V,\ \forall t \in 1...T$
$\tau_{R_{vt}} \in \mathbb{R}\ \forall v \in 1...V,\ \forall t \in 1...T$ is a variable indicating the time spent recharging by vehicle v at timestep t
$c_{ij} \in \mathbb{R}\ \forall i, j \in 1...N$ denotes the time cost in minutes to travel from node i to node j
$e_{ij} \in \mathbb{R}\ \forall i, j \in 1...N$ denotes the energy usage to travel from node i to node j
$w \in \mathbb{R}$ represents the tradeoff we are willing to make between delivery times and energy usage. A higher w means we will value energy efficiency more, and a lower w means we will value the maximum time cost more.
S is the set of supply depots, where vehicles begin and end their day
$B_{vt} \in \mathbb{R}\ \forall v \in 1...V,\ \forall t \in 1...T$ is a variable indicating the battery level of vehicle v at timestep t
$R_{ij} \in \{0, 1\},\ \forall i, j \in 1 \dots N$ represents the arcs that can be recharged on. R is a diagonal binary matrix, indicating that if we stay at a charging station for a timestep (travel from point i to point i, where i is a recharge), we can recharge
$G_{0v} \in \mathbb{R}$ is the battery capacity and initial battery level of every vehicle
$d \in \mathbb{R}$ is the recharge rate, i.e. how many kilometers of range does a vehicle get per minute of charging

This objective function requires auxiliary variables to linearize the terms inside the max function.

$$\min\ y + w \cdot \frac{1}{V} \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} e_{ij} x_{ijvt}$$

$$y \geq \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ijvt} + \tau_{R_{vt}}\ \forall v = 1...V$$

Next, we will define the constraints.

1. $\sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ijvt} = 0$ (node flow balance)
2. $\sum_{k=1}^{N} x_{kivt} = \sum_{k=1}^{N} x_{ikv(t+1)},\ \forall i \in 1, \dots, N,\ v \in 1, \dots, V,\ t \in 1, \dots, T-1$

(individual truck flow balance)

3. $\sum_{v=1}^{V}\sum_{i\in S}^{S}\sum_{j=1}^{N} x_{ijv1} = V$ (depot vehicle supply constraint)
4. $\sum_{v=1}^{V}\sum_{i\in S}^{S}\sum_{j=1}^{N} x_{ijvT} = V$ (depot vehicle demand constraint)
5. $\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ijvt} = 1,\ \forall v \in 1,\ldots,V,\ t \in 1,\ldots,T$ (each vehicle must be in a single location at each timestep)
6. $\sum_{i\in\text{deliveries}}\sum_{v=1}^{V}\sum_{t=1}^{T} x_{ijvt} \geq 1,\ \forall j \in 1,\ldots,N$ (all deliveries must be made)
7. $B_{vt} \leq B_{v(t-1)} - e_{ij}x_{ijvt} + M \cdot R_{ij} \cdot x_{ijvt} + M \cdot (1 - x_{ijvt}),\ \forall i,j \in 1,\ldots,N,\ v \in 1,\ldots,V$ (battery constraints, taking into account draining battery when we travel and ability to charge at charging stations)
8. $B_{v1} \leq G_{0v} - e_{ij}x_{ijv1} + M \cdot R_{ij} \cdot x_{ijv1} + M \cdot (1 - x_{ijv1}),\ \forall i,j \in 1,\ldots,N,\ v \in 1,\ldots,V$ (battery constraints for t=1)
9. $B_{vt} \geq B_{v(t-1)} - e_{ij}x_{ijvt} - M \cdot (1 - x_{ijvt}),\ \forall i,j \in 1,\ldots,N,\ v \in 1,\ldots,V$ (bound the battery from below, otherwise it can pick any value from the upper bound to 0)
10. $B_{v1} \geq G_{0v} - e_{ij}x_{ijv1} - M \cdot (1 - x_{ijv1}),\ \forall i,j \in 1,\ldots,N,\ v \in 1,\ldots,V$ (bound the battery from below for timestep 1)
11. $t_{R_{vt}} \geq \frac{1}{\mathrm{d}} \cdot (B_{vt} - B_{v(t-1)}) \cdot R_{ij} - M \cdot (1 - x_{ijvt}),\ \forall i,j \in 1,\ldots,N,\ v \in 1,\ldots V$ (if we use a recharge station, the time spent charging is proportional to the amount of battery we refill)

In plain english, this formulation allows us to ensure that vehicles visit every delivery point at least once (constraint 6), that all vehicles exist and must continue to exist without disappearing or creating new vehicles (constraints 1, 2, 5), and that vehicles must begin and end their journey at the supply depot (constraints 3, 4). These constraints are all shared by the standard MVRP. We add additional constraints to model the battery. Constraints 7, 8, 9, and 10 ensure that the battery level is exactly what it should be at every timestep, taking away battery range when we travel from point to point, and allowing recharging if a vehicle is at a charging station. For each of the battery constraints, the big-M terms remove the constraints if that constraint for that vehicle does not apply, i.e. if we are not at a charging station (modeled by $M \cdot R_{ij} \cdot x_{ijv1}$) or we are not at a specific location ($M \cdot (1 - x_{ijvt})$). The formulation does not require that a vehicle recharge if it is at a recharge station, and also allows for recharge stations to be at delivery locations. Constraint 11 sets the minimum amount of time we have to stay at a charging station to fill our battery up by some amount from one timestep to the next, which is proportional to the amount of battery that is chosen to be filled at that station.

7.2 Visualizations

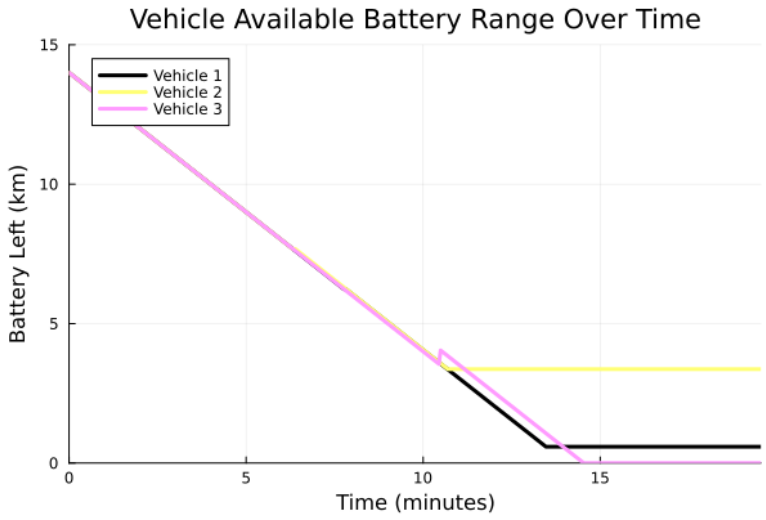

Figure 2: Battery levels over time for the solution found by Gurobi for the small problem with battery constraints. Note that vehicle 3 requires a small recharge in order to make it back to the supply depot, and ends with exactly 0 battery left. The other vehicles make it back to the supply depot with battery to spare.

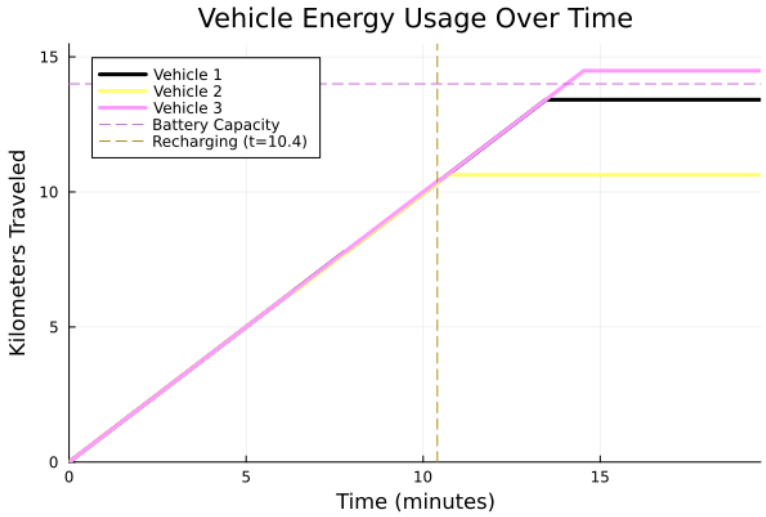

Figure 3: Energy used in units of kilometers traveled by a vehicle over time for the solution found by Gurobi for the small problem with battery constraints. Note that vehicle 3 requires a small recharge in order to make it back to the supply depot, and very briefly waits at 10.4 minutes into deliveries. The other vehicles make it back to the supply depot with battery to spare. Note the maximum battery capacity is drawn with a horizontal line, and that vehicle 3 travels further than the maximum battery range because of its recharge.

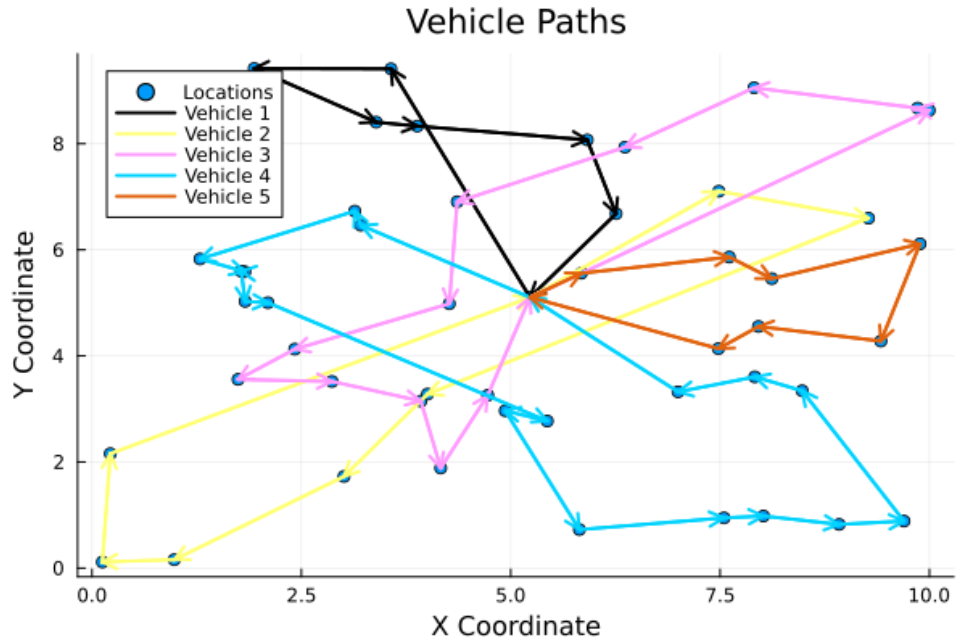

Figure 4: Solution found by Gurobi for the large standard MVRP problem without battery constraints (49 delivery points, 1 supply depot, 5 vehicles). This solution was run with a time constraint of 1800 seconds, or half an hour, and Gurobi had not converged to an optimal solution in that time as can be seen by the crossing paths. Running Gurobi for longer would allow for better solutions.

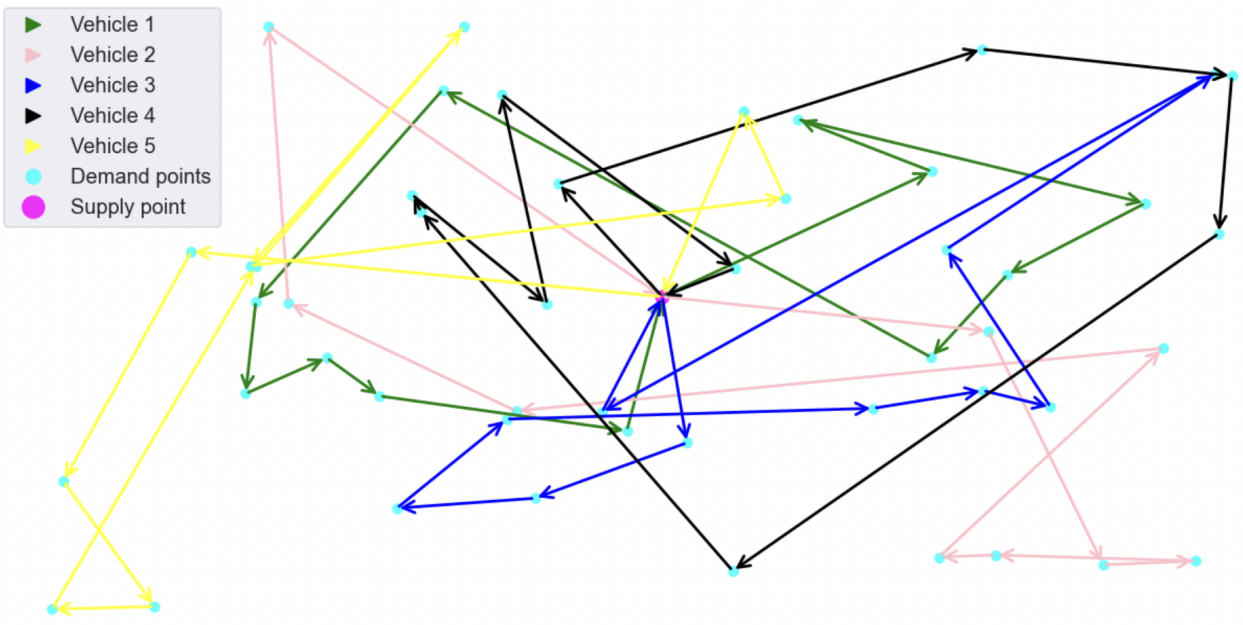

Figure 5: Solution using Genetic Algorithm for the large standard MVRP without battery constraints

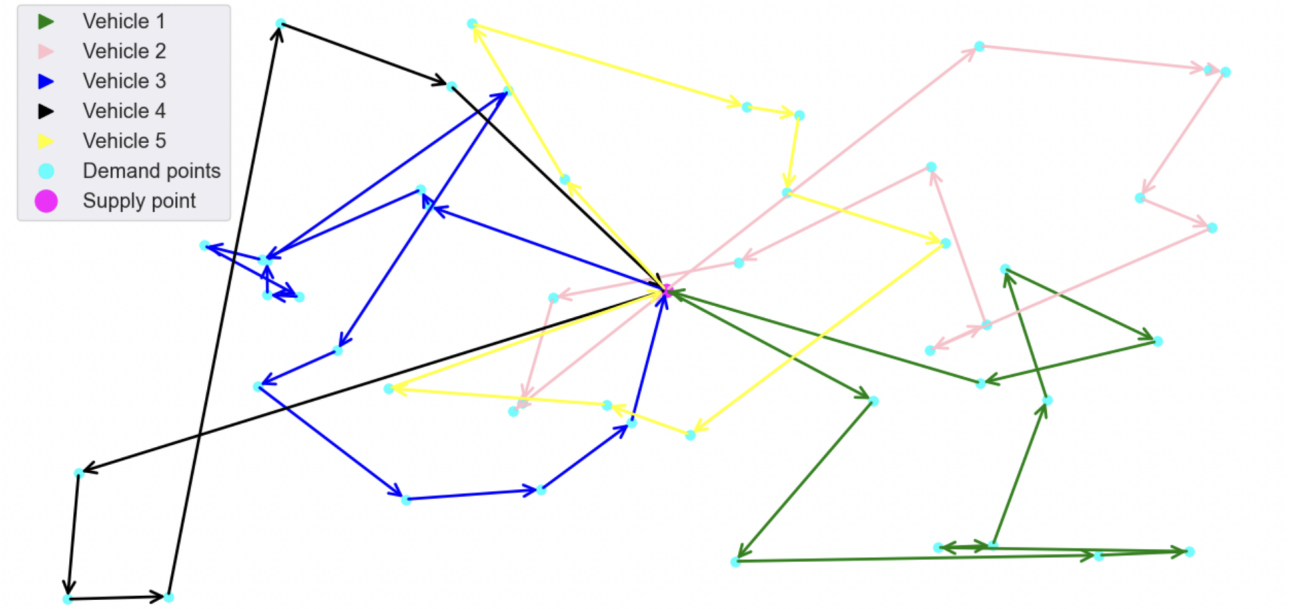

Figure 6: Solution using hybrid Genetic and 2-OPT Algorithm for the large standard MVRP without battery constraints

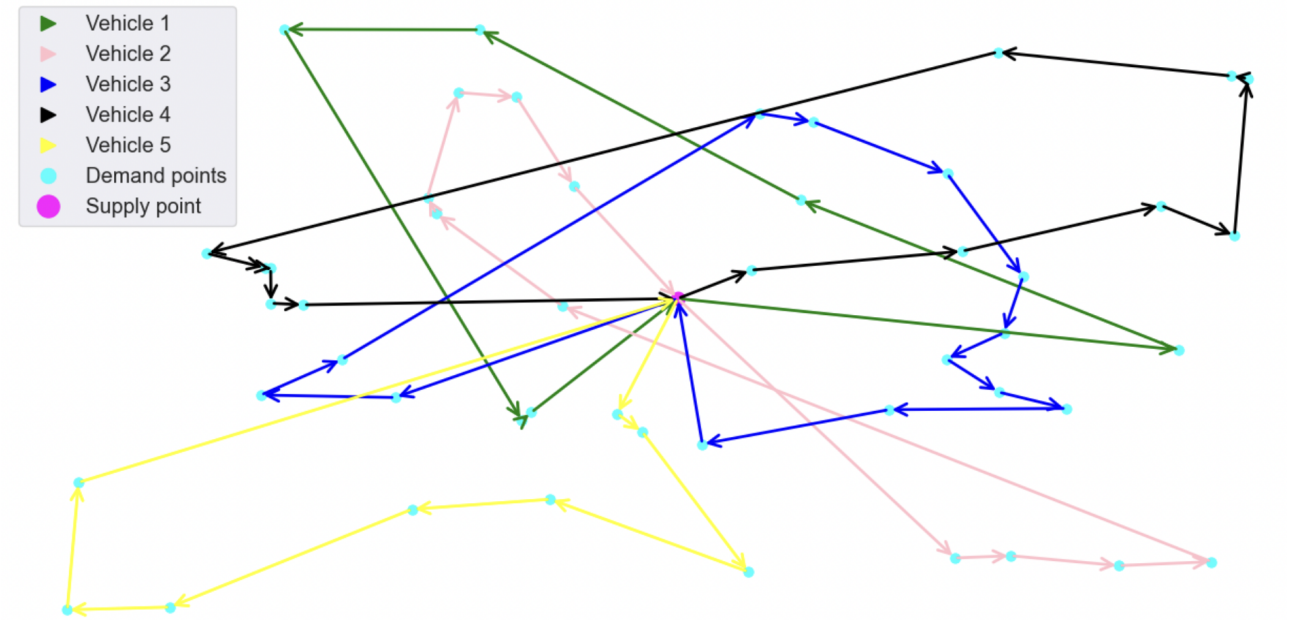

Figure 7: Solution using 2-OPT algorithm for the large standard MVRP without battery constraints which gave the best result among all the algorithms

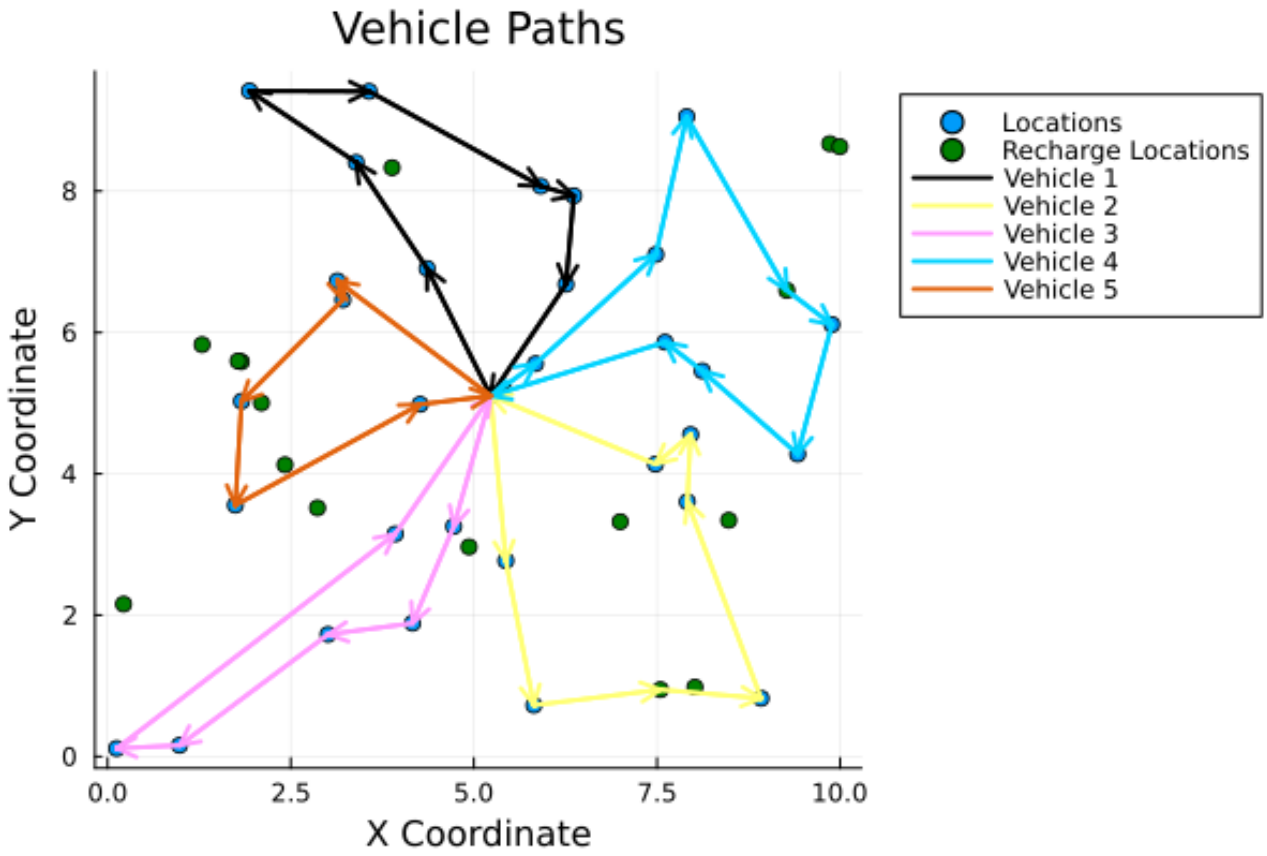

Figure 8: Solution found by Gurobi for the large MVRP problem with battery constraints (31 delivery points, 1 supply depot, 16 charging stations, 5 vehicles, 25 km vehicle range). This solution was run with a time constraint of 3600 seconds, or 1 hour, and Gurobi had not converged to an optimal solution in that time, so this solution is probably not optimal. Running Gurobi for longer would allow better solutions.

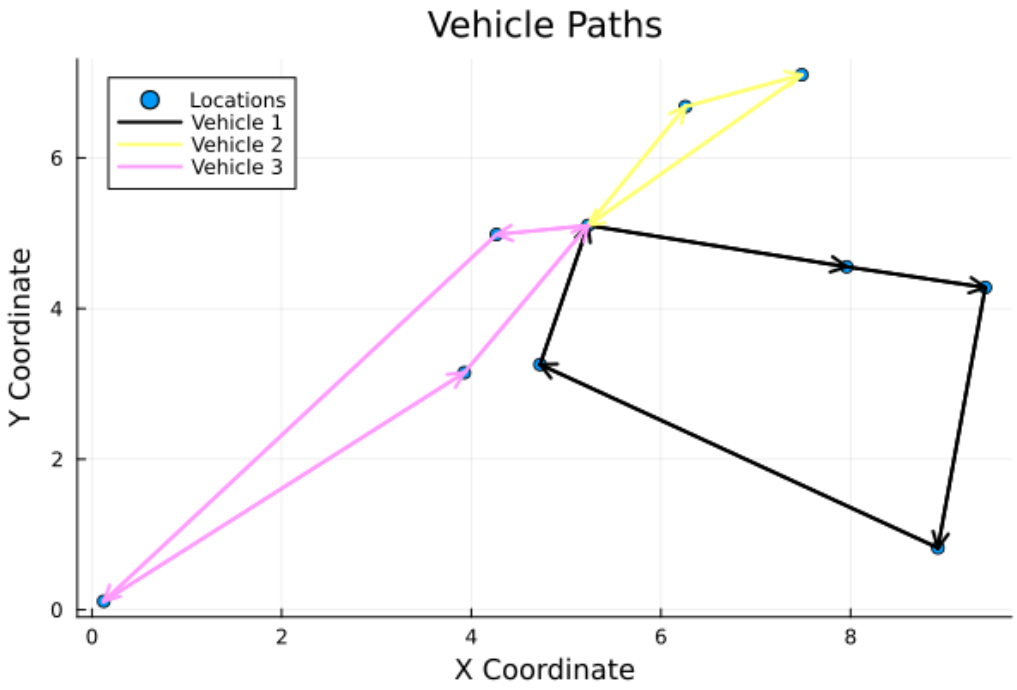

Figure 9: Plot of standard MVRP solution using Gurobi, with no battery constraints, and average energy tradeoff parameter $w = 0.25$. Compare this to the next figure, where we increase w.

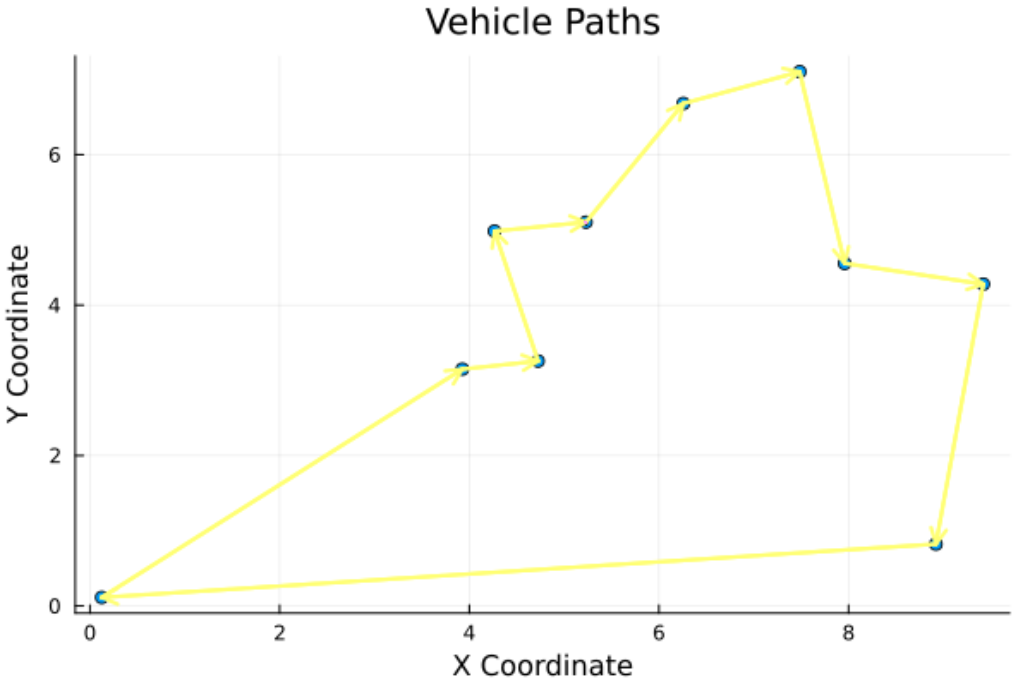

Figure 10: Plot of standard MVRP solution, with no battery constraints, and average energy tradeoff parameter $w = 100$. This means that the average energy cost dominates the objective function, which makes the optimal solution be to use a single vehicle. This minimimizes the total energy cost, and therefore the average energy cost among all 3 vehicles in the fleet. Compare this plot to the plot above, where w is smaller.

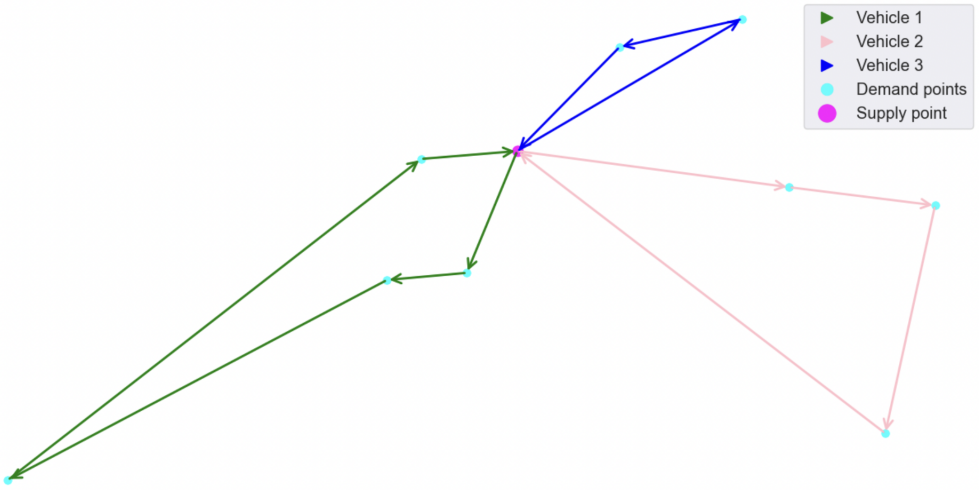

Figure 11: Plot of standard MVRP solution using 2-OPT algorithm, with no battery constraints, and average energy tradeoff parameter $w = 0.25$

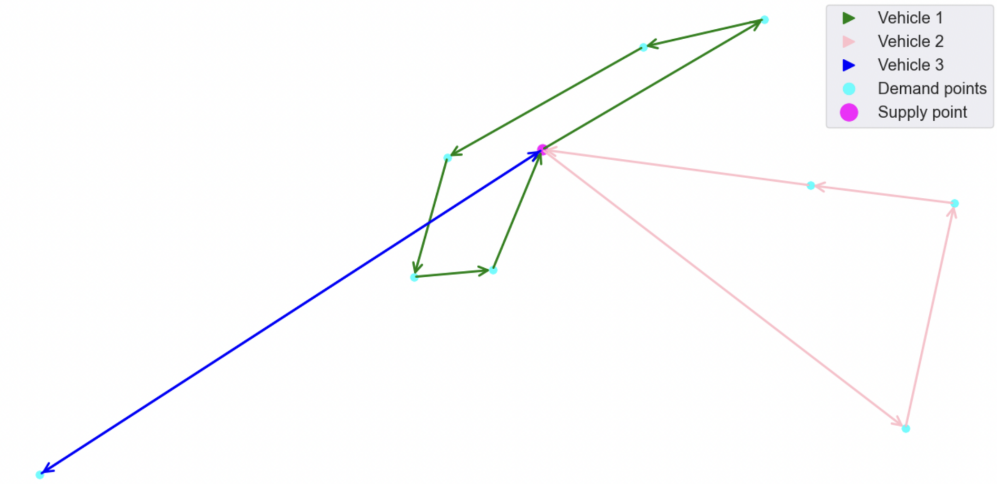

Figure 12: Plot of standard MVRP solution using hybrid Genetic and 2-OPT algorithm, with no battery constraints, and average energy tradeoff parameter $w = 0.25$

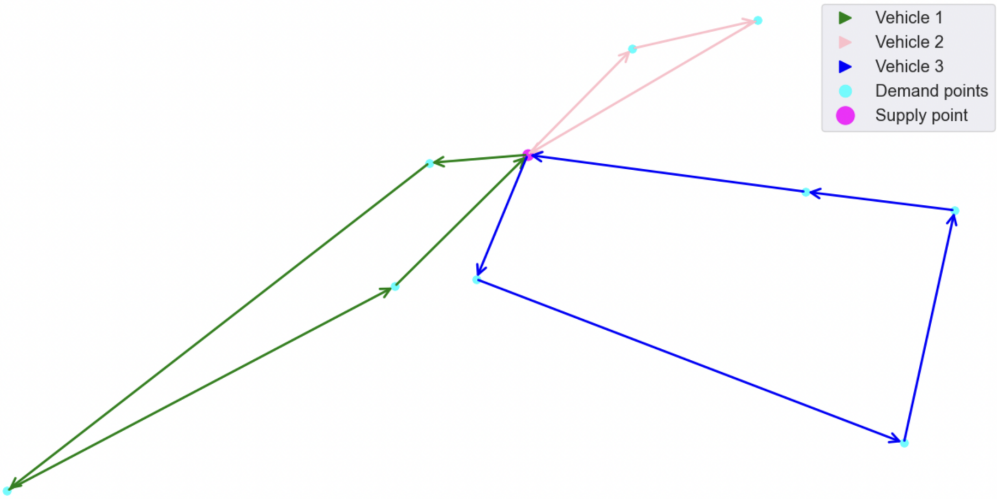

Figure 13: Plot of standard MVRP solution using Genetic algorithm, with no battery constraints, and average energy tradeoff parameter $w = 0.25$ which is equivalent to Gurobi's solution shown above

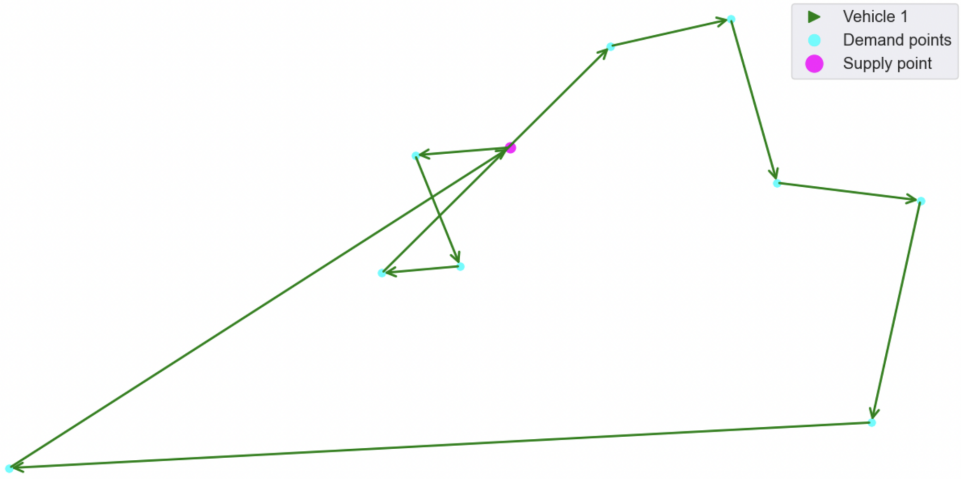

Figure 14: Plot of standard MVRP solution using Greedy approach algorithm, with no battery constraints, and average energy tradeoff parameter $w = 0.25$

7.3 Julia JuMP Gurobi Code

```
# Data and parameters

# Use 5 vehicles
V = 5
N = 32 # There are 32 delivery locations
Rs = 16 # there are 16 available recharge stations
# S is the number of supply depots, and these are included in N. So
↪ there are N - S total demand locations / recharge stations
S = 1
# Maximum number of timesteps to use - we shouldn't need more than N+Rs
↪ +1
T = N + Rs + 1 #32 + 16 + 1 (plus 1 for supply). If we know we can use
↪ less time, we can reduce this for a potentially faster solution
# big M - should be fine for a maximum battery capacity of 25
M = 1000
# Energy usage - initial battery level of all vehicles in kilometers
G_0 = zeros(V) .+ 25 #(every vehicle can travel up to 25 kilometers)
# time cost - assume we are moving 60 km/hr so that number of
↪ kilometers traveled is equal to minutes, so cost matrix is in
↪ minutes
cost = zeros(N+Rs, N+Rs)
for i in 1:(N+Rs)
    for j in 1:(N+Rs)
        X_i = locations_df[i,1]
        Y_i = locations_df[i,2]
        X_j = locations_df[j,1]
        Y_j = locations_df[j,2]
        cost[i,j] = sqrt((X_i - X_j)^2 + (Y_i - Y_j)^2)
    end
end
#energy cost to travel from place to place
energy_cost = cost
#Recharge stations the last Rs points
recharge_stations = zeros(N+Rs, N+Rs)# + I; #for now, we can fully
↪ recharge by staying in the same place
for i in (N+1):(N+Rs)
    recharge_stations[i,i] = 1
end

#Use a recharge rate:
recharge_rate = 10; #10 kilometers of charge per minute

#################################################################
```

```
#Begin making the model
model = Model(Gurobi.Optimizer)

# x represents whether the path from location i to location j is
↪ traveled by vehicle v at timestep t
@variable(model, 0 <= x[i=1:(N+Rs), j=1:(N+Rs), v=1:V, t=1:T] <= 1, Bin
↪ )
# y is the linearized version of the cost function
@variable(model, y >= 0)
# Battery variable, the battery has a certain capacity and cant have
↪ less than 0 charge
@variable(model, 0 <= B[v=1:V, t=1:T] <= G_0[v])
# time for vehicle v to recharge at timestep t
@variable(model, t_R[v=1:V, t=1:T] >= 0)

w = 0.25
#The objective value is a weighted sum of the total time and the
↪ average energy used by all vehicles
@objective(model, Min, y + w*(1/V)*sum(sum(sum(sum(energy_cost[i,j]*x[i
↪ ,j,v,t] for i in 1:(N+Rs)) for j in 1:(N+Rs)) for v in 1:V) for t in
↪  1:T))

# Linearizing the max in the objective
for v in 1:V
    @constraint(model, y >= sum(sum(sum(cost[i,j]*x[i,j,v,t] + t_R[v,t]
    ↪  for j in 1:(N+Rs)) for i in 1:(N+Rs)) for t in 1:T))
end
# Flow balance constraints - except for time step 1 and the last time
↪ step, where we have special constraints with the depots
# Every vehicle that enters node i at time t must leave to a node j at
↪ time t+1
@constraint(model, flow_balance_constraint[i=1:(N+Rs), t=1:(T-1)], #
↪ there are N x T total nodes to have flow constraints on
    sum(sum(x[k,i,v,t] for v in 1:V) for k in 1:(N+Rs)) == #the sum of
    ↪ vehicles entering from a node k to this node i equals
    sum(sum(x[i,k,v,t+1] for v in 1:V) for k in 1:(N+Rs)) #the sum of
    ↪ vehicles exiting from this node i to a node k at the next
    ↪ timestep
    )
@constraint(model, ind_flow_balance_constraint[i=1:(N+Rs), v=1:V, t=1:(
↪ T-1)], #there are N x V X T total nodes-vehicles to have flow
↪ constraints on
    sum(x[k,i,v,t] for k in 1:(N+Rs)) == #a vehicle entering from a
    ↪ node k to this node i equals
```

```
    sum(x[i,k,v,t+1] for k in 1:(N+Rs)) #a vehicles exiting from this
    ↪ node i to a node k at the next timestep
    )

# Flow balance constraint for supply node - on time step 1, V vehicles
↪ leave the supply depot
# Vehicles may choose to stay at the depot though, and just "leave" to
↪ this supply node at the next timestep
@constraint(model, depot_supply_constraint, sum(sum(sum(x[i,j,v,1] for
↪ j in 1:(N+Rs)) for i in 1:S) for v in 1:V) == V)
# On time step T, V vehicles enter the supply depot at that time step
@constraint(model, depot_demand_constraint, sum(sum(sum(x[i,j,v,T] for
↪ j in 1:S) for i in 1:(N+Rs)) for v in 1:V) == V)

# add a constraint that the vehicle is in exactly 1 location per
↪ timestep per vehicle slice
@constraint(model, vehicle_single_location_constr[v=1:V, t=1:T], sum(
↪ sum(x[i,j,v,t] for j in 1:(N+Rs)) for i in 1:(N+Rs)) == 1)

# Constraint that all demand nodes must be visited
# The sum of all incoming nodes to delivery point D must be greater
↪ than 1.
# It is an incoming arc if it is in the column corresponding to this
↪ delivery point
@constraint(model, delivery_constraint[j=1:N], sum(sum(sum(x[i,j,v,t]
↪ for i in 1:(N+Rs)) for v in 1:V) for t in 1:T) >= 1)

#Battery constraints - bound from above (battery needs to have less
↪ energy than cumulatively used and recharged.
# M's ignore the constraints when we don't travel there or when we
↪ travel and recharge there. If we just travel, the battery decreases
@constraint(model, battery_constraint_1[i=1:(N+Rs), j=1:(N+Rs), v=1:V,
↪ t=1:1],
    B[v,t] <= G_0[v] -
    energy_cost[i,j]*x[i,j,v,t] +
    M*recharge_stations[i,j]*x[i,j,v,t] +
    M*(1-x[i,j,v,t])
)
@constraint(model, battery_constraint[i=1:(N+Rs), j=1:(N+Rs), v=1:V, t
↪ =2:T],
    B[v,t] <= B[v, t-1] -
    energy_cost[i,j]*x[i,j,v,t] +
    M*recharge_stations[i,j]*x[i,j,v,t] +
    M*(1-x[i,j,v,t])
)
```

```
#Battery constraints - bound from below. The battery level must be
↪ greater thean the previous battery level minus what we used to get
↪ there
@constraint(model, min_battery_constraint_1[i=1:(N+Rs), j=1:(N+Rs), v
↪ =1:V, t=1:1],
    B[v,t] >= G_0[v] -
    energy_cost[i,j]*x[i,j,v,t] - (1-x[i,j,v,t])*M
)
@constraint(model, min_battery_constraint[i=1:(N+Rs), j=1:(N+Rs), v=1:V
↪ , t=2:T],
    B[v,t] >= B[v, t-1] -
    energy_cost[i,j]*x[i,j,v,t] - (1-x[i,j,v,t])*M
)
# Recharge time constraints: the time to recharge is the difference in
↪ battery levels, if we are at a recharge station and recharged
@constraint(model, recharge_time_1[i=1:(N+Rs), j=1:(N+Rs), v=1:V, t
↪ =1:1], t_R[v,t] >=
    (1/recharge_rate)*(B[v,t] - G_0[v])*recharge_stations[i,j] -
    M*(1-x[i,j,v,t])
)
@constraint(model, recharge_time[i=1:(N+Rs), j=1:(N+Rs), v=1:V, t=2:T],
↪  t_R[v,t] >=
    (1/recharge_rate)*(B[v,t] - B[v,t-1])*recharge_stations[i,j] -
    M*(1-x[i,j,v,t])
)

set_optimizer_attribute(model, "TimeLimit", 3600.0)
optimize!(model)
```