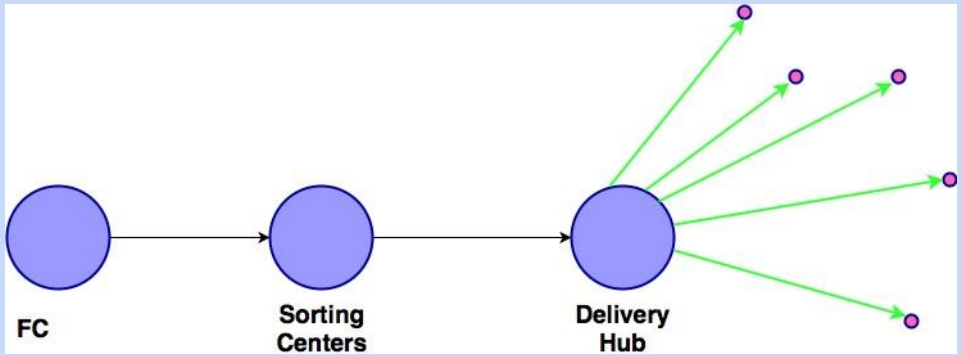
Solving the vehicle routing problem for optimizing shipment delivery

Venkateshan K

FSG TECH
Fulfilment and Services Group

Context: Last Mile Delivery



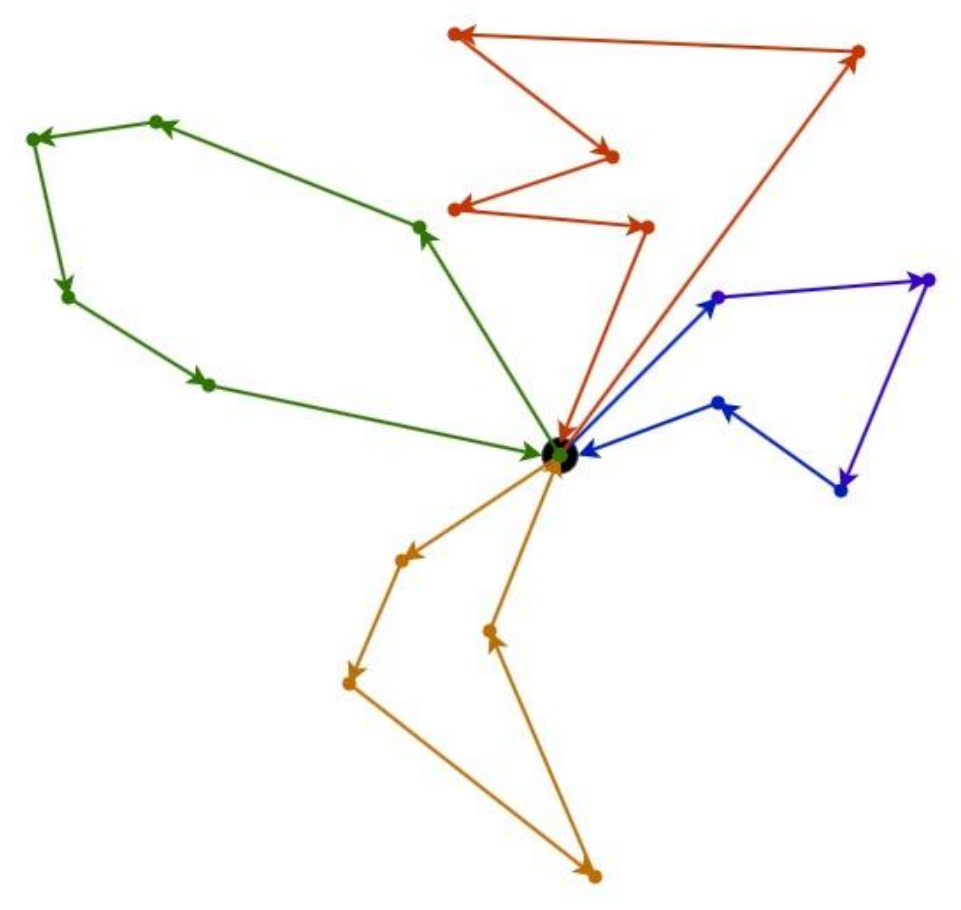
- Delivery Hub: Wishmasters (FEs) carry a **subset of shipments** and **follow a path** delivering the shipments to the customers.
- Heterogeneous hubs: shipments vary from l**ess than 100 to over 1000.**
- Question - How should the shipments be assigned to FEs? Subsequent, what path should the FEs follow to deliver the shipments?

Routing Problem

HOW DO WE FIND THE BEST ASSIGNMENT AND PATH?

- Formulate it as an **optimization problem**
- Associate **a cost function** W with every configuration of routes and find the configuration that minimizes the cost function.
- Standard choice for cost function : **sum of the total travel times** of all FEs/routes.
- A special case of a single FE : **Traveling Salesman Problem** (TSP)

Vehicle Routing Problem



- This is a **generalization of TSP** - the **Vehicle Routing Problem.**
- Computation complexity : TSP is **NP-complete** and naturally so is VRP (formulated as decision problem; is there a solution with total cost less than u?)
- 60 years since publication *The Truck Dispatching Problem* [Dantzig 1959] introducing the VRP.

Planned vs On-Demand delivery

RELEVANT FOR

any business involving delivery of **pre-determined** shipments at **specific locations.** Ex: e-commerce, online grocery ordering, postal service, B2B planning and scheduling.

FULL INFORMATION

CONTRAST WITH

On **demand delivery** - delivery/assignment planning happens **on the go** as soon as the order is placed (online food ordering, hyperlocal deliveries, cab aggregators).

PARTIAL INFORMATION

Variants with additional constraints

Vehicle Routing Problem with Time Window (VRPTW)

- Routing solution when customers are promised **delivery time-window** (say 12:00-14:00)

Capacitated Vehicle Routing Problem (CVRP)

- **Maximum capacity** of each vehicle/FE in terms of number of shipments.

Capacitated Vehicle Routing Problem with Time Windows (CVRPTW)

Both constraints applied **simultaneously** (as most real world situations would be).

Out-of-box solver

- Optaplanner (Java package) - currently in use

- Limitations (Results)
  - Routes tend to be **elongated and stretched out.**
  - Large overlap in the regions covered by the routes

- Limitations (Flexibility)
  1. **Probe the details of the algorithm**
  2. **Change the heuristic approaches**
  3. **Examine effects of specific choices of cost functions/hyperparameters.**

Python out-of-box solvers

Local Solver

Google OR Tools

Neither allow custom penalties

Formulating the problem

Given :

1. N customers (and DEPOT) locations (vertices)

$$(x_i, y_i), i = 0, 1, 2, ..N$$

2. Pairwise travel time (or distance) matrix

$$q_{ij},\ i \neq j, i, j = 0, 1, 2, \cdots N$$

3. Serviceable time window (customer) $[t_i^E, t_i^L]$

4. m vehicles each with max capacity Q each

Formulating the problem

Flipkart

Determine:

Routes $r^i = (r^i_1, r^i_2, ..r^i_{n_I})$

$i = 1, 2, \cdots m$

1. each customer is covered by **exactly one route**
2. **time window** compatibility
3. $n_i \leq Q$

Minimize the **cost function**

Feasible solution : All the hard constraints (time-window and capacity are satisfied)

Cost function choices?

Should **reflect the desired characteristics** of the route.

Multiple Objectives

1. Minimize travel time - studied in classical formulation
2. Minimize uneven distribution of shipments
3. Minimize occurrence and magnitude of outliers
4. Minimize stretched or extended routes.

Associate a cost component for each

Cost Function Expansion

1. Total travel time

$$U_{TT} = \sum_{a=1}^{m} \sum_{i=0}^{n_i} q_{a_i, a_{i+1}}$$

2. Even distribution of shipments across the vehicles.

"Fairness" cost : $U_F \propto \sigma/E[n]$

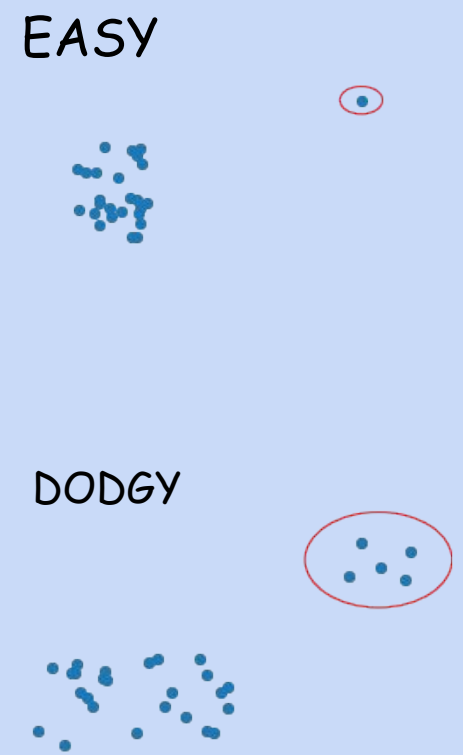
Normalization: $\sqrt{N}$

Dealing with Outliers (Intuition)

Intuition: Neighborhood distance

25th percentile distance (7th nearest neighbor)

[0.81 0.7 1.01 0.86 0.72 0.67 0.69 0.7 0.8 0.92 1.03 1.09 0.83 0.91 0.78 0.88 0.57 1.01 0.76 0.73 0.72 0.62 0.86 1.03 1.42 **4.12 4.29 3.84 4.45 4.2**]



Dealing with Outliers (Formulation)

We determine a z-score for each point in the route depending on the distance to the α^{th} percentile nearest neighbor.

$$z_i = \frac{r_{\alpha,i} - r_\alpha}{\gamma_\alpha}$$

r_α - Median

γ_α - Median absolute deviation (from median)

Route **compactness**

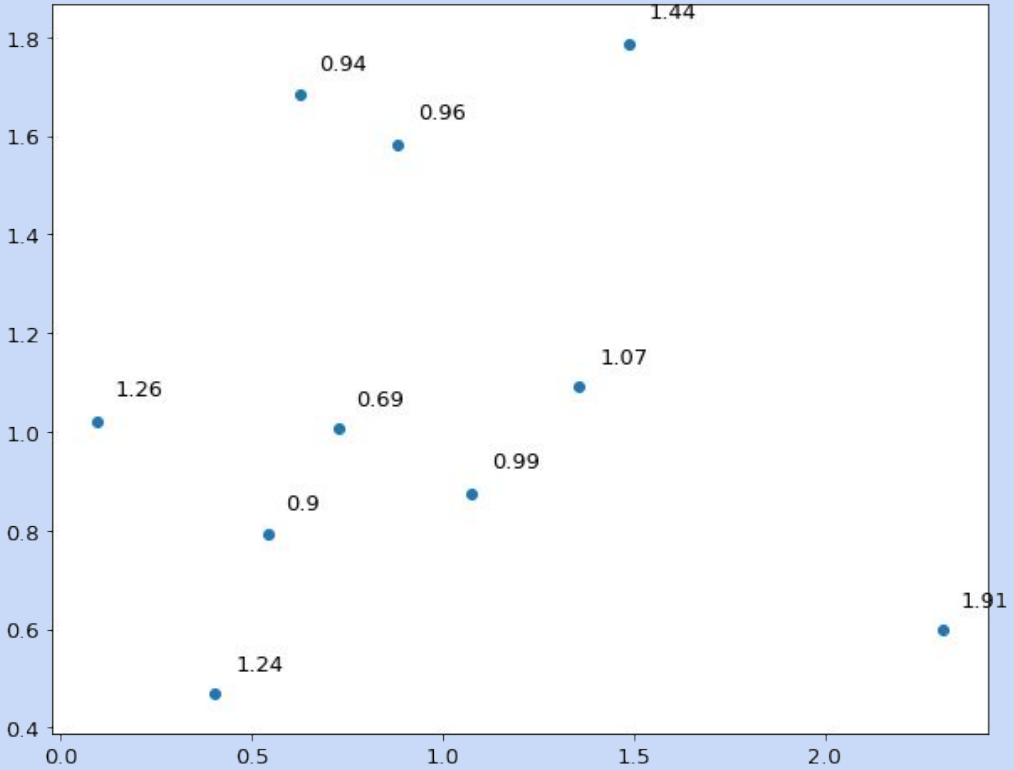
Defining compactness:

80th percentile of the sorted distance for every customer

80th percentile of those numbers : 1.35

$$U_{compact} = \begin{cases} 0 & r_C < R \\ K_c \frac{n}{\sqrt{N}}(r_C - R)(\frac{r_C}{R}) & r_C \geq R. \end{cases}$$

$$R = \sqrt{A/\pi m}$$



TOTAL COST

$$U = U_{TT} + U_F + U_{\text{Out}} + U_{\text{Comp}}$$

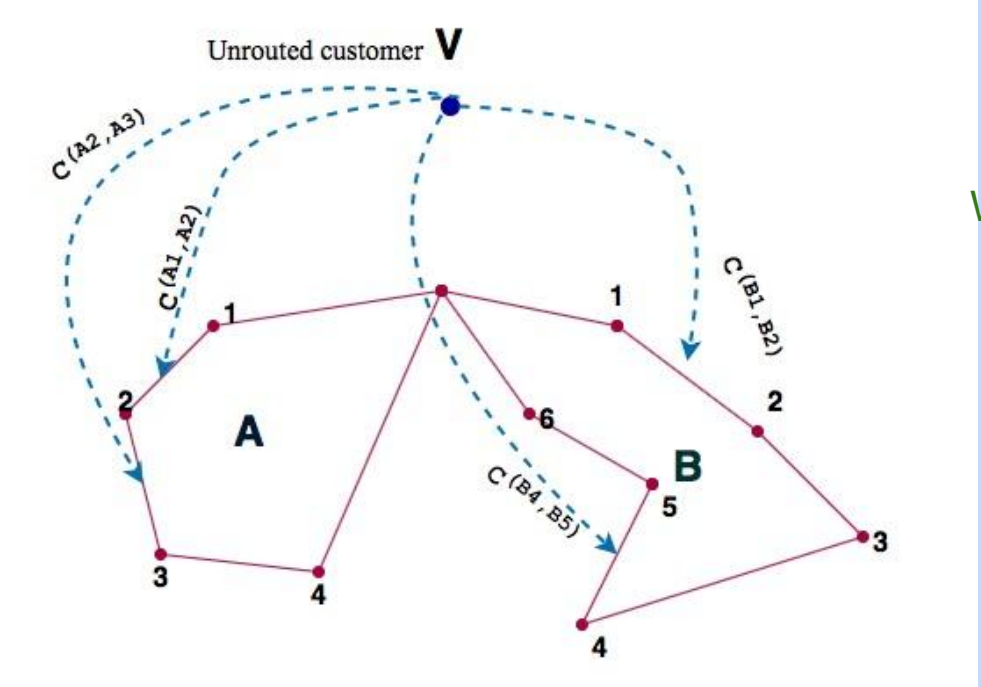
Approximate Algorithms Approach

- Exact algorithms are computationally expensive and we use **approximate algorithms**, i.e., those that need not converge to optimality.
- Broadly, these are characterized by primarily two stages
  - **Construction Heuristics** - construction of initial feasible routes (m routes satisfying the constraints)
  - **Improvement Heuristics** - iteratively improve the routes by intra- and inter-route reconfiguration

Construction Heuristic

At any given iteration, there are a set of partially constructed routes.

For a given unrouted customer, we consider all **feasible insertion positions**



We compute the minimum cost for each vertex

$$c_v^{\min \text{ insert}}$$

Construction Heuristic..cont'd (Algorithm)

$$c_{v_1}^{\text{min insert}} \qquad (j_{v_1}^*, k_{v_1}^*)$$

$$c_{v_2}^{\text{min insert}} \qquad (j_{v_2}^*, k_{v_2}^*)$$

$\ldots$

$\ldots$

$$c_{v_s}^{\text{min insert}} \qquad (j_{v_s}^*, k_{v_s}^*)$$

$$\mathbf{v*} = \arg\min_v \{c_v^{\text{min insert}}\} \qquad (\mathbf{j}_{\mathbf{v}^*}^*, \mathbf{k}_{\mathbf{v}^*}^*)$$

Insert the customer that **minimizes the minimum** insertion cost.

Insertion Cost (Travel Time)

When inserting an unrouted customer we take into account:

(a) increase in total travel time
(b) change in the fairness cost
(c) outlier cost (if any)
(d) compactness cost (if any)

Insertion Cost (Time window shift)

Insert v between i and j in route C:

Costs **proportional to window shrinkage** of:

1. Inserted Customer
2. Next customer early time **push-forward**
3. Previous customer latest time **push back**

[Solomon, 1987]

Insertion Cost (Time window shift)

WHY TIME-WINDOW SHRINKAGE COST?

$$U = U_{TT} + U_F + U_{\text{Out}} + U_{\text{Comp}}$$

Computational cost?

Flipkart

Construction Heuristic:

O(LgN)

1. Recompute the **minimum cost only for the top L customers** and then pick the lowest for insertion.
2. N**arrow the list of feasible insertion** points initially.
   (if considering insertion of v after customer i)

$$t_i^E + T_S \leq t_v^L$$

3. Instead of considering all insertion points, consider only **g** nearest neighbors.

PART 2: Improvement Heuristic

How do we improve the initial feasible routes?

Reconfiguration of routes

(a) intra- or (b) inter-route

Move or switch customers within a route

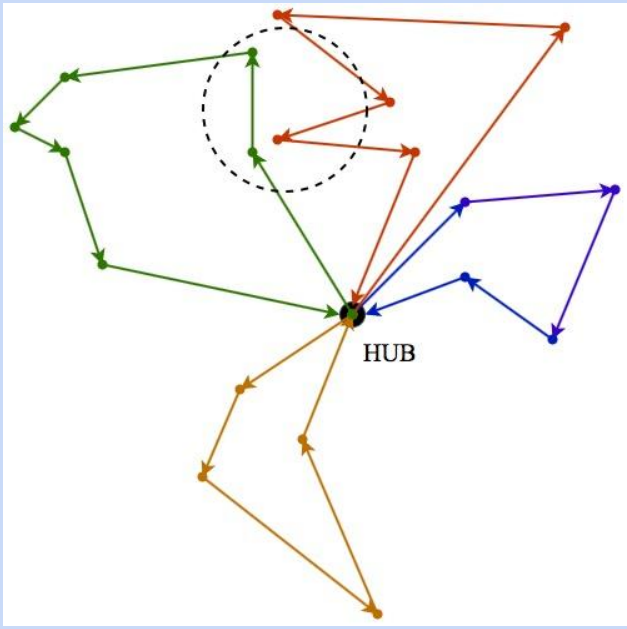
. Change **sequence of visits** over a route segment

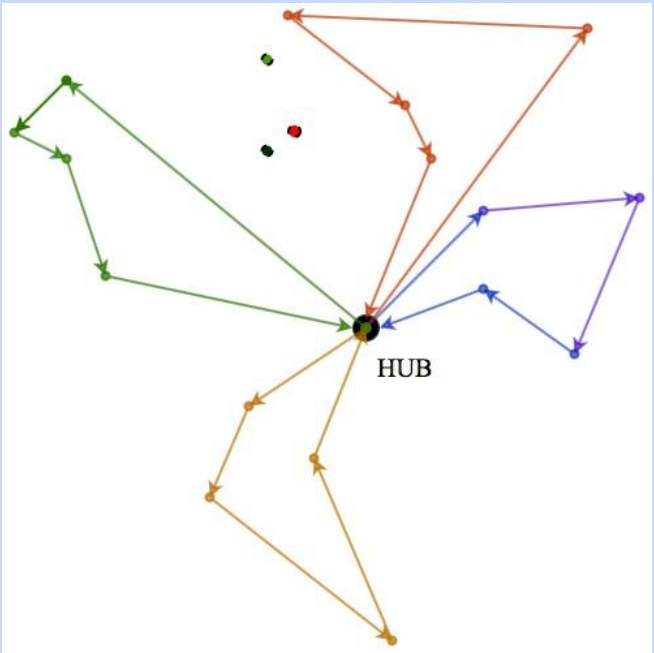
Merge and split pairs of routes

Mode 1: Ruin and Recreate

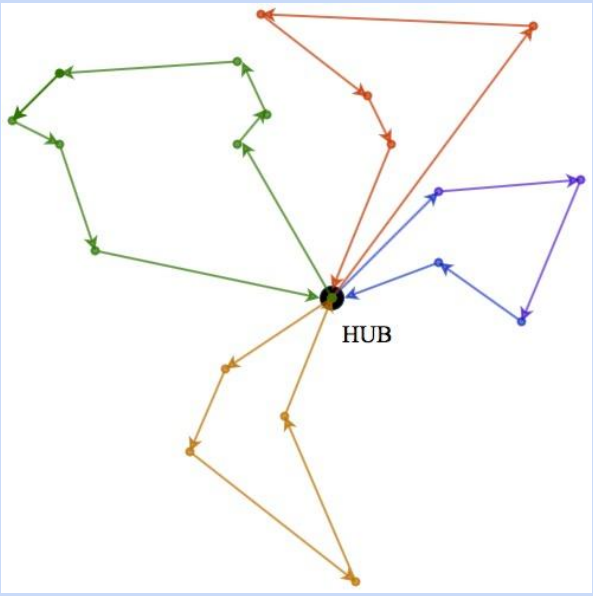
2-PHASES

EJECTION





REINSERT



Ejection Types

Radial : Pick a **customer at random** and eject the nearest B customers.

(Temporal) Radial : Pick a **customer at random** and eject the nearest B customers whose early times are closest.

Random: Eject B of **random customers**.

Cost : Eject B of most **'costly' customers**.

B: U[0, fN]

Time windows of the ejected customers are **reset to original slots** and the routes are reconnected.

f~0.15

(Schrimpf et al. 2000)

Reinsertion

1. **Insertion order : minimization of travel time** and **window shifts to nearest neighbor.**
2. Reinsertion **procedure: same as construction heuristic.**

Computational complexity:

V (No of iterations) *X*
fN/2 (customers to reinsert) *X*
W(top insertion locations based on distance)

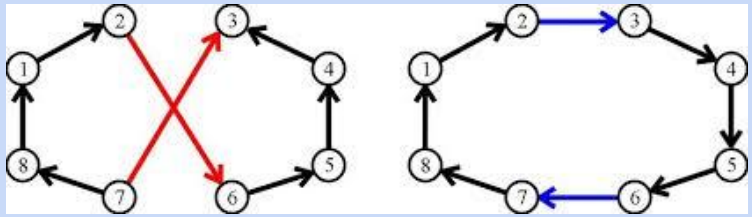
O(fVNW)

Reinsertion with Relocation

- Given the capacity constraint, the insertion at a location in route r may reduce cost but could violate the capacity constraint.
- In this case, we seek to identify a suitable customer on that route and relocate it to another route r_1.
- If that other route's max capacity is exceeded, we would then try to do the same relocation operation for a customer on route r_1.
- We compute the total cost associated with these relocation moves and we proceed with it if that is less than the next best alternative.

Mode 2: k-opt edge exchanges

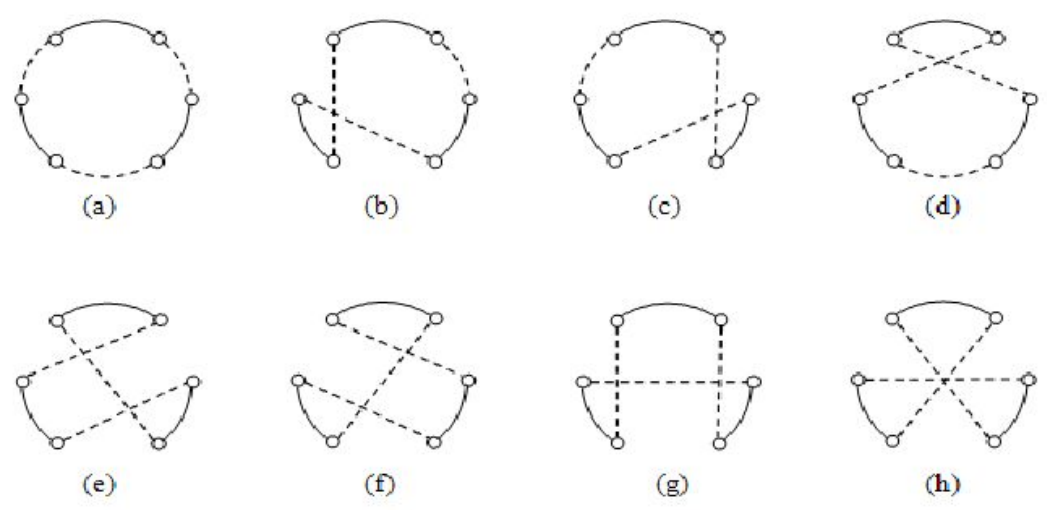
- 2-Opt



Remove two links in the route and rewire them if the combined distance has been reduced.

Mode 2: k-opt edge exchanges

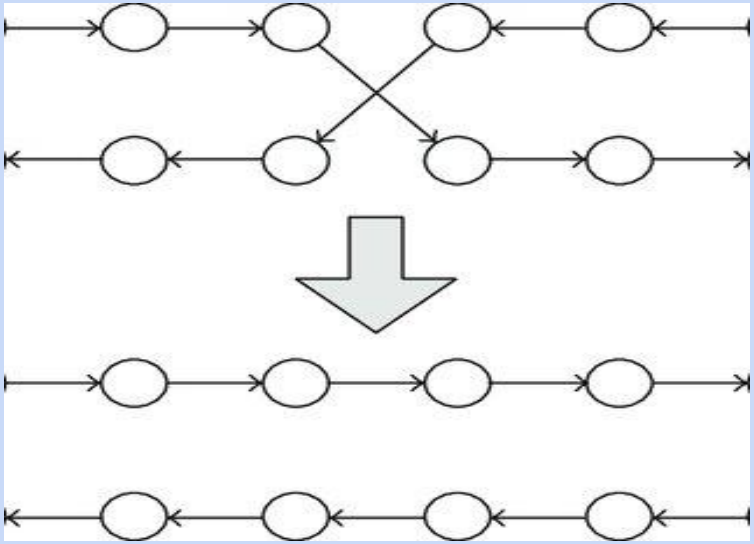
- 3-Opt



Likewise remove three links and rewire the route.

Mode 2: k-opt edge exchanges

- 2-Opt*



Start with two distinct but nearby routes and swap a sequence of customer visits between them as shown.

Mode 3: Lambda Exchange

- For a given pair of routes, r_A and r_B, remove s_A and s_B customer from each (with $\max(|s_A|,|s_B|) < \lambda$) and insert it into the other route .

- The reinsertion again follows a similar procedure of identifying the location that minimizes the total cost.

- The criteria for identifying customers to remove is based on contribution to increase in travel time or total cost.

Time window violations

- Scenarios where there are seemingly no feasible solutions.
- We consider time windows as a soft-constraint and permit breaches but at the same time minimize their occurrence and magnitude.
- At insertion (or reinsertion), if there are no feasible locations, then we identify the location that minimizes the total combined breach.
- In fact, we consider the combination of total breach and extra travel time on the route.

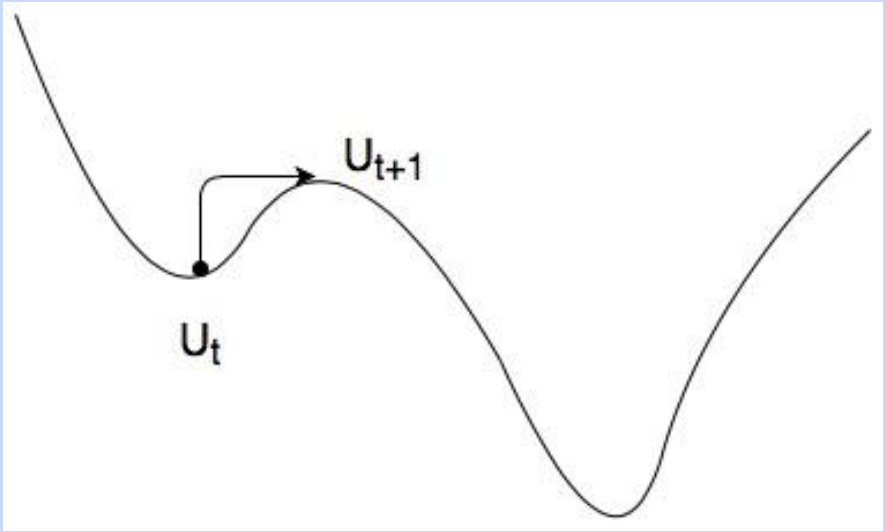
Simulated Annealing

After reinsertion of ejected customers, we recalculate the change in the cost $\Delta U = U_{t+1} - U_t$

If cost **decreases**, we **accept** the new configuration and proceed to the next iteration.

Accept the solution with probability $e^{-\frac{\Delta U}{T}}$

Motivation: Avoid getting **stuck in a local minimal**



'Temperature' $T = T_0 e^{-\beta t/V}$ where t is the current iteration number ;
V is the total number of iterations

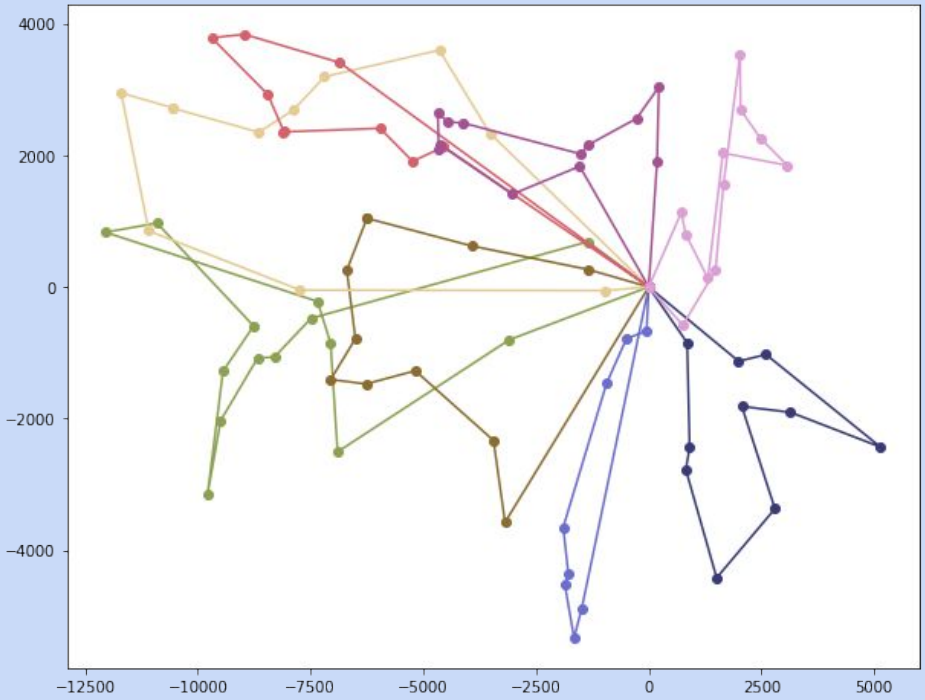
As T decreases, we the search becomes narrower and at T=0, we **recover the greedy algorithm**.

Simulated Annealing (Breach)

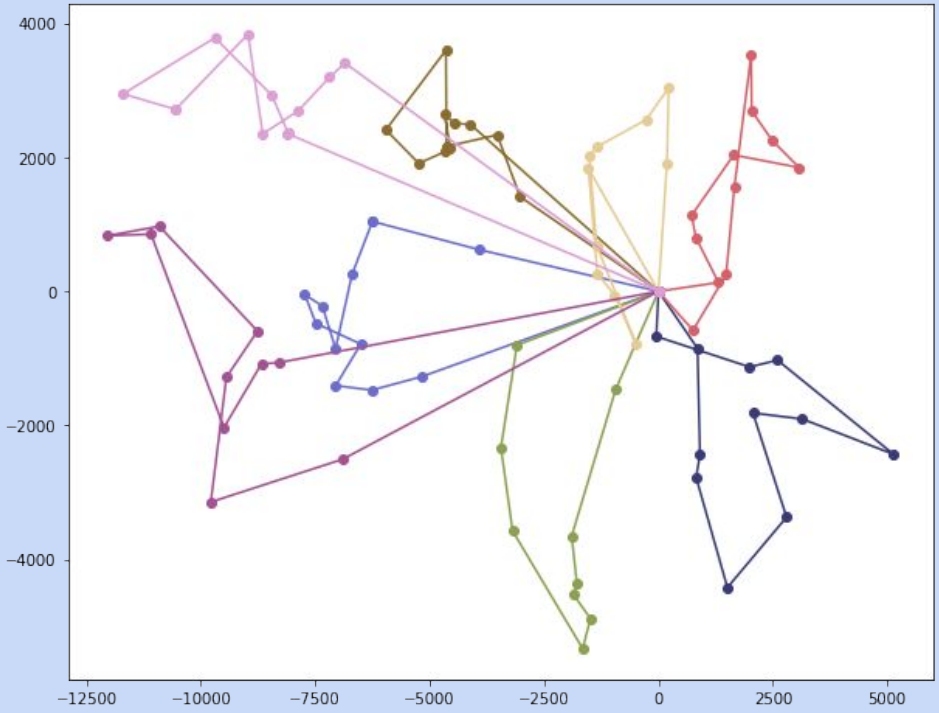
A separate simulated annealing step is carried out for the total breach.

The new configuration should clear both the simulated annealing operations; else the previous configuration is restored.

Results (Time Windows)



Out of box solver



Our algorithm

Results (Time Windows)

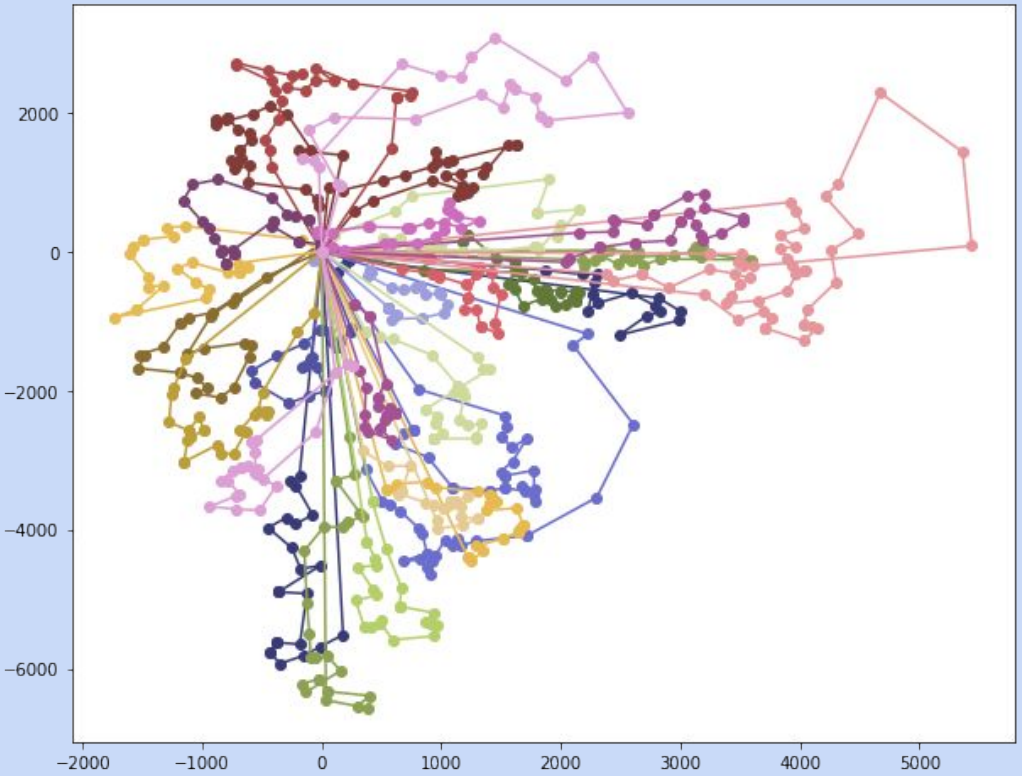
Out-of-box (8 vehicles)

Travel : 4542.85
Fairness : 266.14
Outlier :916.1
Compactness: 2900.22
Total: 8625.29

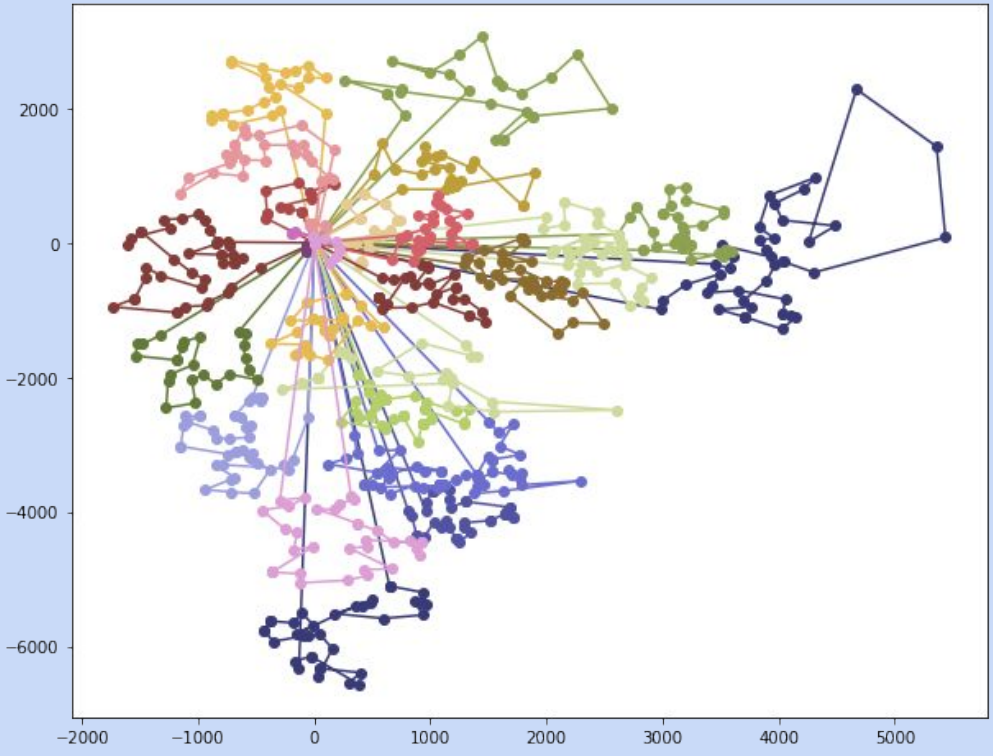
Our algorithm (8 vehicles)

Travel : 4481.95
Fairness : 150.34
Outlier :422.49
Compactness: 768.1
Total: 5822.86

Results (Without Time Windows)



Out of box solver



Our algorithm

Results (Without Time Windows)

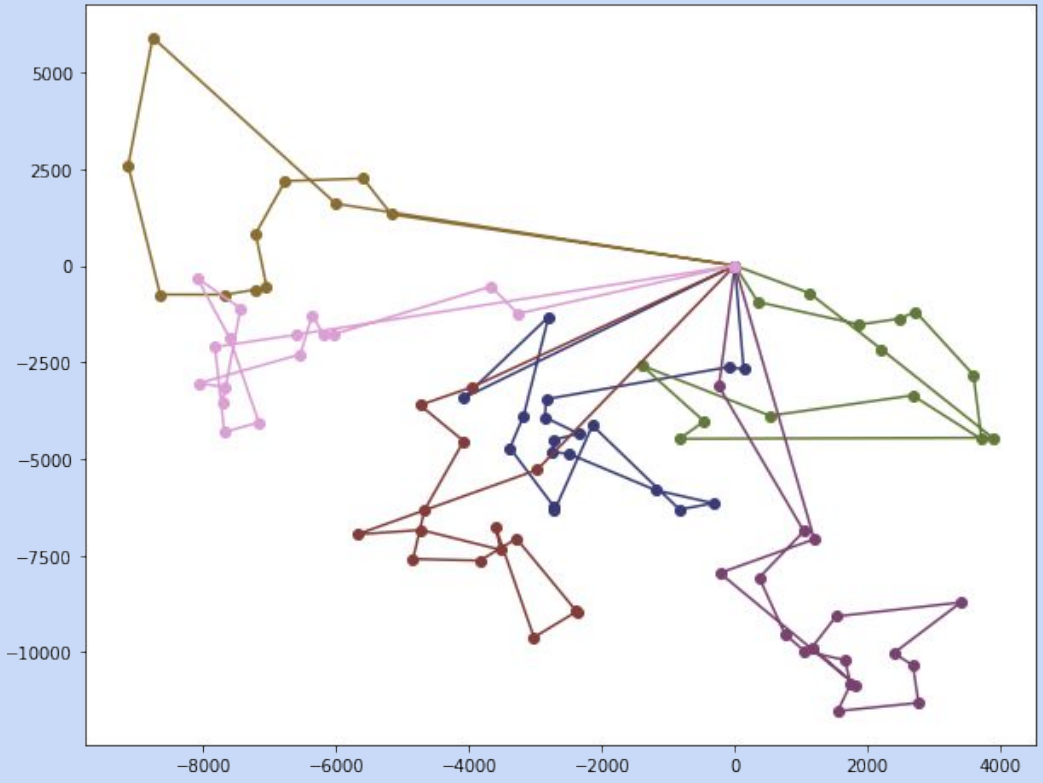
Out-of-box (29 vehicles)

Travel : 5383.570362
Fairness : 625.252579
Outlier :11078.721303
Compactness: 8005.953538
Total: 25093.497783

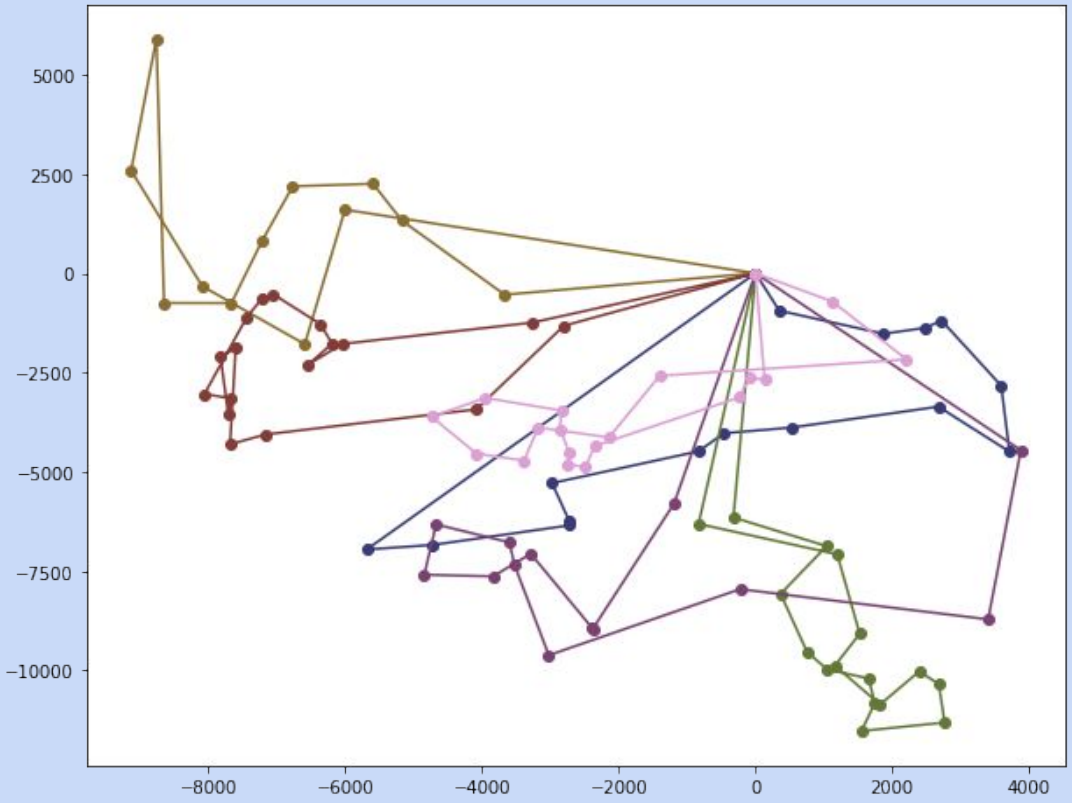
Our algorithm (29 vehicles)

Travel : 5269.445307
Fairness : 2386.762732
Outlier :3113.384236
Compactness: 2408.239887
Total: 13177.832162

Time Windows with breaches



Out of box solver



Our algorithm

Time Windows with breaches

Out-of-box (6 vehicles)

Breach (704 minutes, 11 customers)

Travel : 5310.361508
Fairness : 256.647875
Outlier :950.873044
Compactness: 983.895939
Total: 7501.778366

Our algorithm (6 vehicles)

Breach(476 minutes, 7 customers)

Travel : 5318.646974
Fairness : 239.422382
Outlier :840.246680
Compactness: 1812.622152
Total: 8210.938188

References

- [Desrochers, 2008] M. Desrochers, J. Desrosiers, and M. Solomon, "**A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows**," *Operations Research*, vol. 40, pp. 342–354, apr 2008.
- [Fischetti,1994]M. Fischetti, P. Toth, and D. Vigo, **"A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs,**" *Operations Research*, vol. 42, pp. 846–859, oct 1994.
- [Hashimoto,2008] H. Hashimoto, M. Yagiura, and T. Ibaraki, **"An iterated local search algorithm for the time-dependent vehicle routing problem with time windows,"***Discrete Optimization*, vol. 5, pp. 434–456, may 2008
- [Kohl,1997]N. Kohl and O. B. G. Madsen, **"An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation**,"*Operations Research*, vol. 45, pp. 395–406, jun 1997
- [Solomon, 1987] M. M. Solomon, "**Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints**," *Operations Research*, vol. 35, pp. 254–265, apr 1987
- [Schrimpf, 2000] **G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck**, "**Record Breaking Optimization Results Using the Ruin and Recreate Principle**,"*Journal of Computational Physics*, vol. 159, pp. 139–171, apr 2000.