# Vehicle Routing Scheduling for Cross-docking.

Computer & Industrial Engineering

Submitted To :

Dr. Vikas Kumar

By: Bimal Kumar, Abhishek Chand, Aishwarya Verma.

Department of Data Science & Analytics

Central University of Rajasthan

# AGENDA

- Customer Satisfaction
- TSP
- Vehicle routing
- Cross-docking
- Tabu Search
- Result
- Conclusion

# Reference

## **Research Paper**

Vehicle routing scheduling for cross- docking in the supply chain

**Author** : - Young Hae Lee, Jung Woo Jung, Kyong Min Lee

Department of Industrial Engineering, 1271, Sa-1 Dong, Ansan, Gyeonggi Do 426-791, South Korea,
LG Phillips LCD, 642-3, Jinpyung Dong, Gumi, Gyeongsangbuk Do 730-726, South Korea

Link :- https://www.sciencedirect.com/science/article/abs/pii/S036083520600091X

## Objective :

Maximize the profit of organisation as well as satisfies the customer expectation.

# Customer Satisfaction

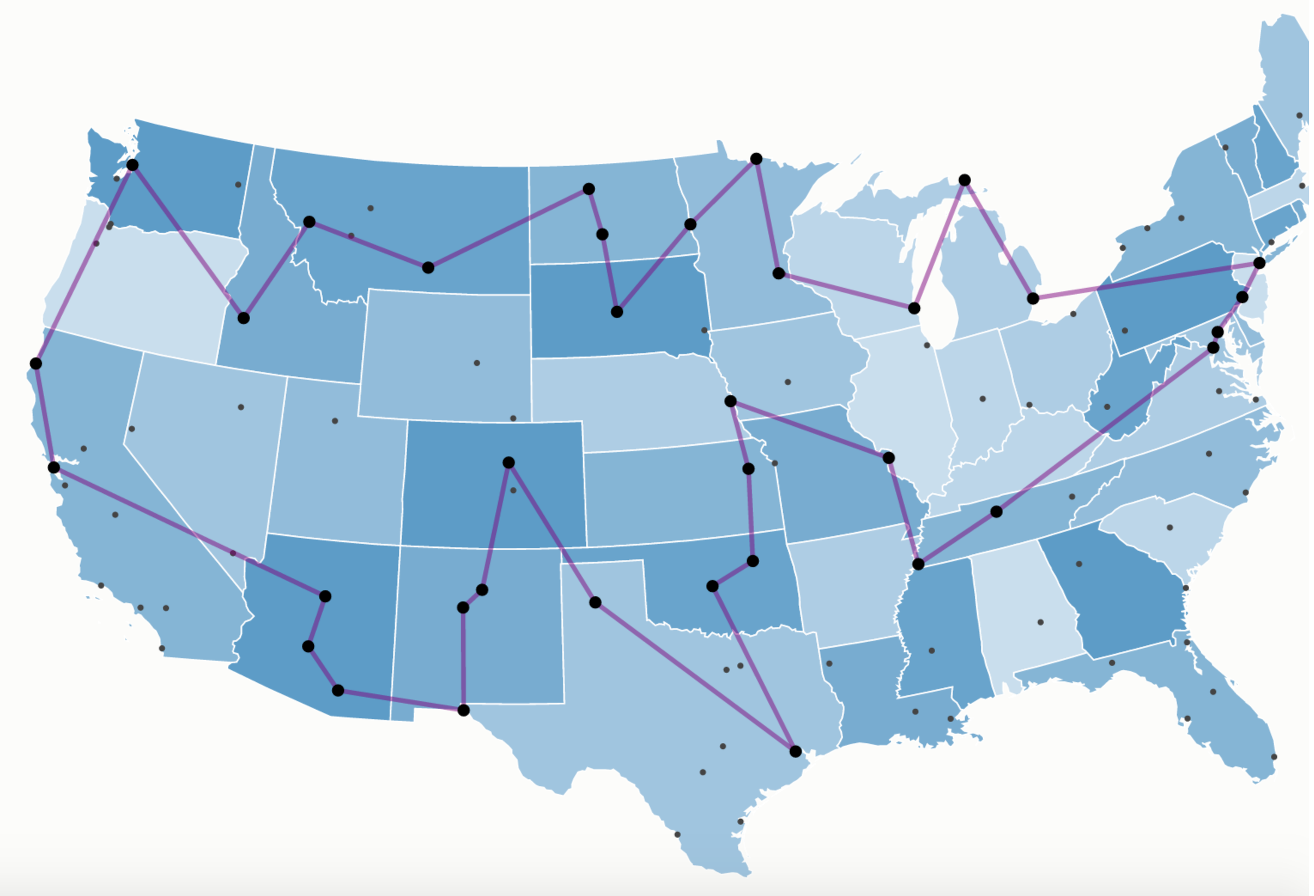
Customer satisfaction is a metric used to quantify the degree to which a customer is happy with a product, service or experience.

# Traveling Salesman Problem

- Given a depot and a set of n customers, find a route (or "tour") starting and ending at the depot, that visits each customer once and is of minimum length.

- One vehicle.
- No capacities.
- Minimize distance.

- No time windows.
- No compatibility constraints.
- No DOT rules



# Symmetric and Asymmetric

- Let $c_{ij}$ be the cost (distance or time) to travel from i to j.

- If $c_{ij} = c_{ji}$ for all customers, then the problem is *symmetric*.

Direction does not affect cost.

- If $c_{ij} \neq c_{ji}$ for some pair of customers, then the problem is *asymmetric*. Direction does affect cost.

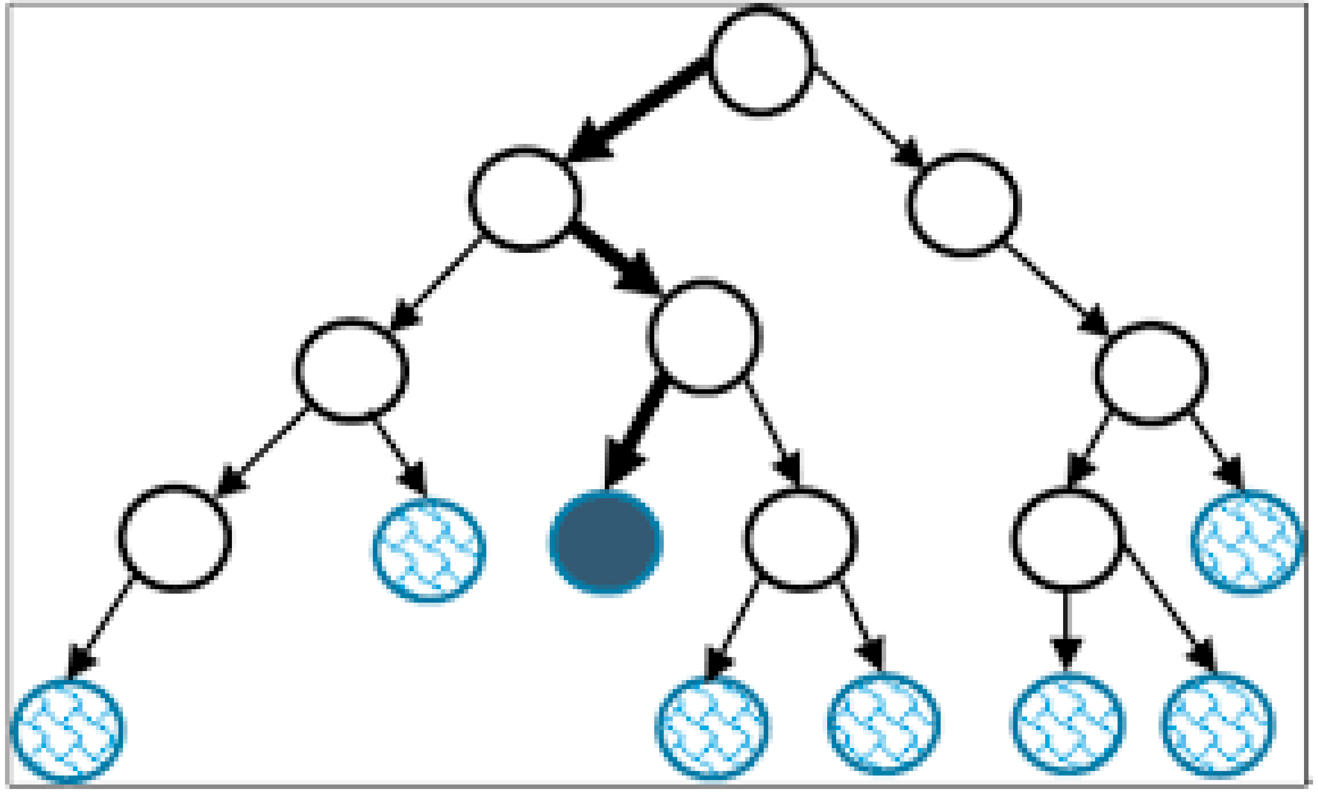
# TSP Solutions

- Heuristics
  - **Construction**: build a feasible route.
  - **Improvement**: improve a feasible route.
    - Not necessarily optimal, but fast.
    - Performance depends on problem.
    - Worst case performance may be very poor.

- Exact algorithms
  - Integer programming.
  - Branch and bound.
    - Optimal, but usually slow.
    - Difficult to include complications

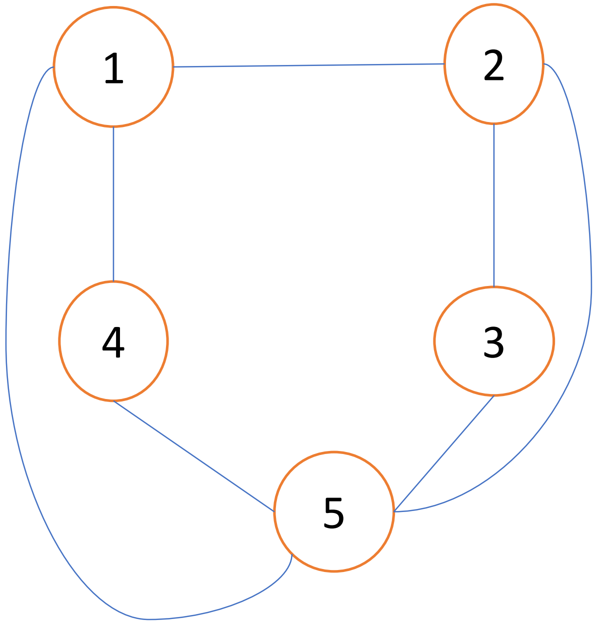
# Branch and Bound

- The branch and bound algorithm is similar to backtracking.
- It applies where the greedy method and dynamic programming fail.
- The general idea of B&B is a BFS-like search for the optimal solution, but not all nodes get expanded (i.e., their children generated).



# Problem Statement

- Consider a milk van delivering milk to four distribution centers (DC) every day morning.
- Objective : - The objective minimize the total cost of providing the services.



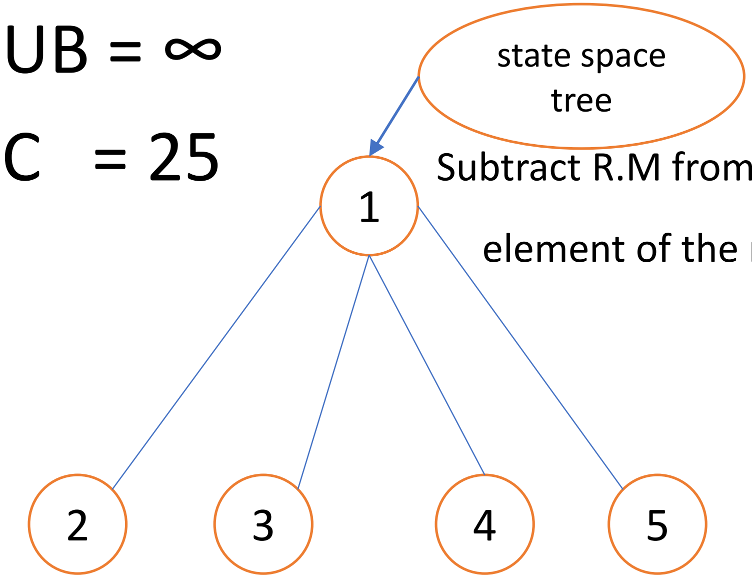

Cost in $

| | | | | |
|---|---|---|---|---|
| — | 20 | 30 | 10 | 11 |
| 15 | — | 16 | 4 | 2 |
| 3 | 5 | — | 2 | 4 |
| 19 | 6 | 18 | — | 3 |
| 16 | 4 | 7 | 16 | — |

# Solution

UB = ∞

C = 25




Subtract R.M from all the element of the matrix

R.M = Raw Minimum

C.M = Column Minimum

Total Reduction cost = Sum(R.M) + Sum(C.M)

= 10+2+2+3+4+1+3 = 25$

This means at least cost would be 25$

## Initial

| | | | | | R.M |
|---|---|---|---|---|---|
| — | 20 | 30 | 10 | 11 | 10 |
| 15 | — | 16 | 4 | 2 | 2 |
| 3 | 5 | — | 2 | 4 | 2 |
| 19 | 6 | 18 | — | 3 | 3 |
| 16 | 4 | 7 | 16 | — | 4 |

fig 1.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | 10 | 20 | 0 | 1 | 10 |
| | 13 | — | 14 | 2 | 0 | 2 |
| | 1 | 3 | — | 0 | 2 | 2 |
| | 16 | 3 | 15 | — | 0 | 3 |
| | 12 | 0 | 3 | 12 | — | 4 |
| C.M | 1 | 0 | 3 | 0 | 0 | |

fig 2.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | 10 | 17 | 0 | 1 | 10 |
| | 12 | — | 11 | 2 | 0 | 2 |
| | 0 | 3 | — | 0 | 2 | 2 |
| | 15 | 3 | 12 | — | 0 | 3 |
| | 11 | 0 | 0 | 12 | — | 4 |
| C.M | 1 | 0 | 3 | 0 | 0 | |

# Continue

## For node 1 → 2

- All values of 1st raw and 2nd column will be infinite.

RM+ CM = 0+0 = 0

- Total cost reduction = C(1,2) + r + r^

  = 10 + 25 + 0 = 35

Total Cost reduction = 35, UB = ∞ , C = 25

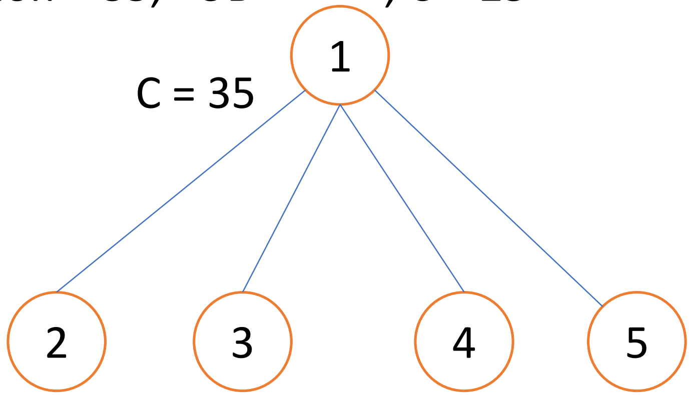



fig **3.**

| | | | | | R.M |
|---|---|---|---|---|---|
| — | — | — | — | — | |
| — | — | 11 | 2 | 0 | 0 |
| 0 | — | — | 0 | 2 | 0 |
| 15 | — | 12 | — | 0 | 0 |
| 11 | — | 0 | 12 | — | 0 |

fig **2.**

| | | | | |
|---|---|---|---|---|
| — | 10 | 20 | 0 | 1 |
| 13 | — | 14 | 2 | 0 |
| 1 | 3 | — | 0 | 2 |
| 16 | 3 | 15 | — | 0 |
| 12 | 0 | 3 | 12 | — |

fig **4.**

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | — | — | 11 | 2 | 0 | 0 |
| | 0 | — | — | 0 | 2 | 0 |
| | 15 | — | 12 | — | 0 | 0 |
| | 11 | — | 0 | 12 | — | 0 |
| C.M | 0 | | 0 | 0 | 0 | |

# Continue

## For node 1 — 3

Repeat same process as we have done above.

Reduction = R.M + C.M = 11+0 = 0

Total cost = C(1,3) + r + r^

= 17+ 25+ 11 = 53

UB = ∞ , C = 25

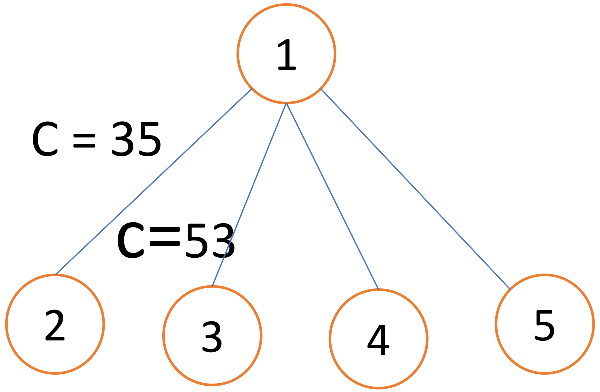



fig 5.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | 12 | — | — | 2 | 0 | 0 |
| | — | 3 | — | 0 | 2 | 0 |
| | 15 | 3 | — | — | 0 | 0 |
| | 11 | 0 | — | 12 | — | 0 |
| C.M | 11 | 0 | | 0 | 0 | |

fig 2.

| | | | | |
|---|---|---|---|---|
| — | 10 | 17 | 0 | 1 |
| 12 | — | 11 | 2 | 0 |
| 0 | 3 | — | 0 | 2 |
| 15 | 3 | 12 | — | 0 |
| 11 | 0 | 0 | 12 | — |

fig 6.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | 1 | — | — | 2 | 0 | 0 |
| | — | 3 | — | 0 | 2 | 0 |
| | 4 | 3 | — | — | 0 | 0 |
| | 0 | 0 | — | 12 | — | 0 |
| C.M | 11 | 0 | | 0 | 0 | |

# Continue

Node 1 — 4

- Repeat same process as we have done above.
- Reduction = R.M + C.M = 0+0 = 0
- Total cost = C(1,4) + r + r^

  = 0 + 25 + 0 = 25

UB = ∞ , C = 25

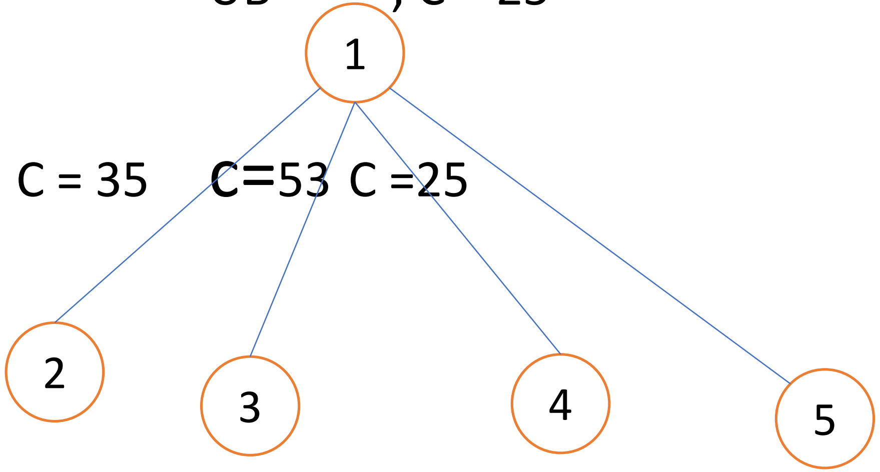



fig 7

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | 12 | — | 11 | — | 0 | 0 |
| | 0 | 3 | — | — | 2 | 0 |
| | — | 3 | 12 | — | 0 | 0 |
| | 11 | 0 | 0 | — | — | 0 |
| C.M | 0 | 0 | 0 | | 0 | |

fig 2.

| | | | | |
|---|---|---|---|---|
| — | 10 | 17 | 0 | 1 |
| 12 | — | 11 | 2 | 0 |
| 0 | 3 | — | 0 | 2 |
| 15 | 3 | 12 | — | 0 |
| 11 | 0 | 0 | 12 | — |

# Continue

Node 1 — 5

- Repeat same process as we have done above.
- Reduction = R.M + C.M = 5+0 = 5
- Total cost = C(1,5) + r + r^ = 1+ 25 + 5

  = 31

UB = ∞ , C = 25

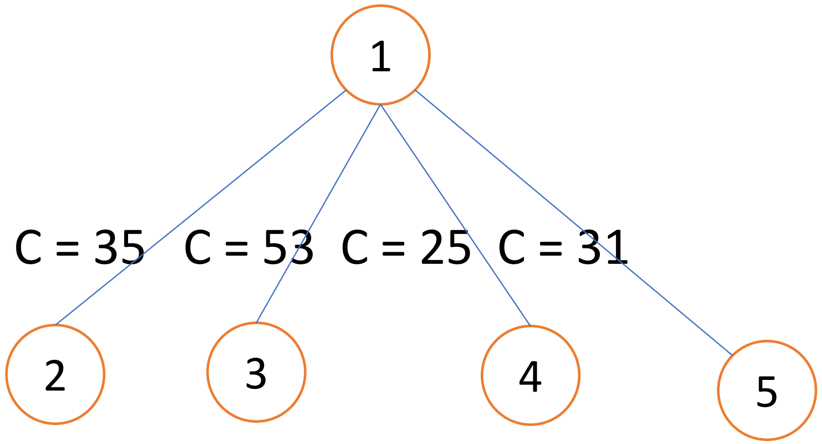



fig 8.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | 12 | — | 11 | 2 | — | 2 |
| | 0 | 3 | — | 0 | — | 0 |
| | 15 | 3 | 12 | — | — | 3 |
| | — | 0 | 0 | 12 | — | 0 |
| C.M | 0 | 0 | 0 | | 0 | |

fig 2.

| | | | | |
|---|---|---|---|---|
| — | 10 | 17 | 0 | 1 |
| 12 | — | 11 | 2 | 0 |
| 0 | 3 | — | 0 | 2 |
| 15 | 3 | 12 | — | 0 |
| 11 | 0 | 0 | 12 | — |

fig 9.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | 10 | — | 9 | 0 | — | 2 |
| | 0 | 3 | — | 0 | — | 0 |
| | 12 | 0 | 9 | — | — | 3 |
| | — | 0 | 0 | 12 | — | 0 |
| C.M | 0 | 0 | 0 | | 0 | |

# Continue

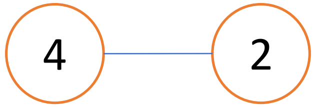
- ➢Exploration of leaf nodes.
- Choose that leaf node which cost is minimum.
- For exploration choose that matrix whose node cost minimum
- Repeat same process as we have done above.

Cost of node 1 — 4 is minimum.

fig 8.

| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| 12 | — | 11 | — | 0 |
| 0 | 3 | — | — | 2 |
| — | 3 | 12 | — | 0 |
| 11 | 0 | 0 | — | — |

# Continue

For node 

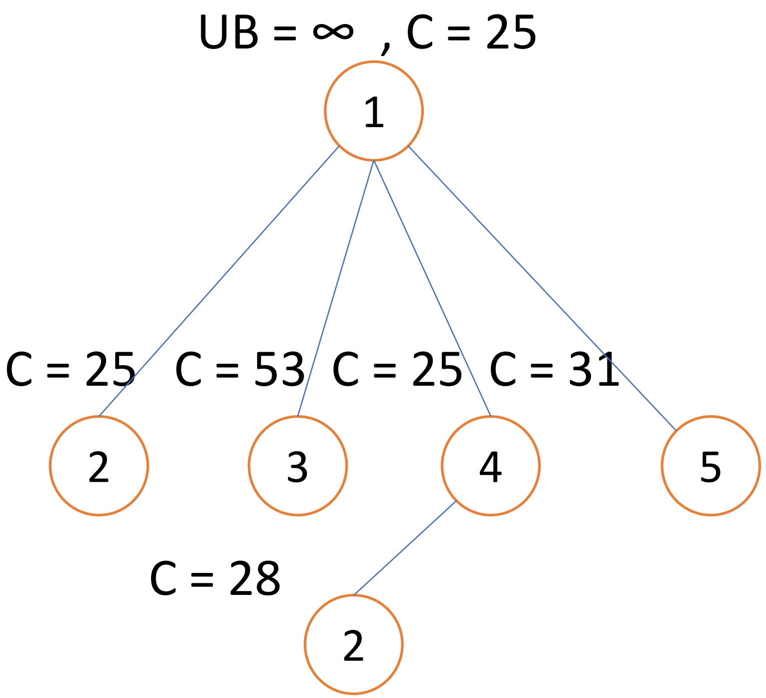




fig10.

| | | | | | | R.M |
|---|---|---|---|---|---|---|
| | — | — | — | — | — | |
| | — | — | 11 | — | 0 | 0 |
| | 0 | — | — | — | 2 | 0 |
| | — | — | — | — | — | |
| | 11 | — | 0 | — | — | 0 |
| C.M | 0 | 0 | 0 | | 0 | |

fig 8.

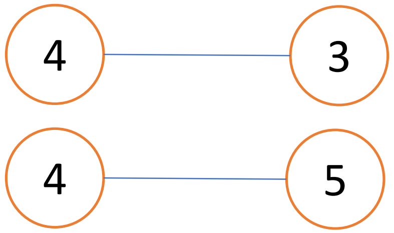
| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| 12 | — | 11 | — | 0 |
| 0 | 3 | — | — | 2 |
| — | 3 | 12 | — | 0 |
| 11 | 0 | 0 | — | — |

- Repeat same process as we have done above.
- Reduction = R.M + C.M = 0+0 = 0
- Total cost = C(4,2) + C(4) + r^

  = 3+ 25 + 0 = 28

# Continue

- Repeat all the process and we get the result.
- For node
- And



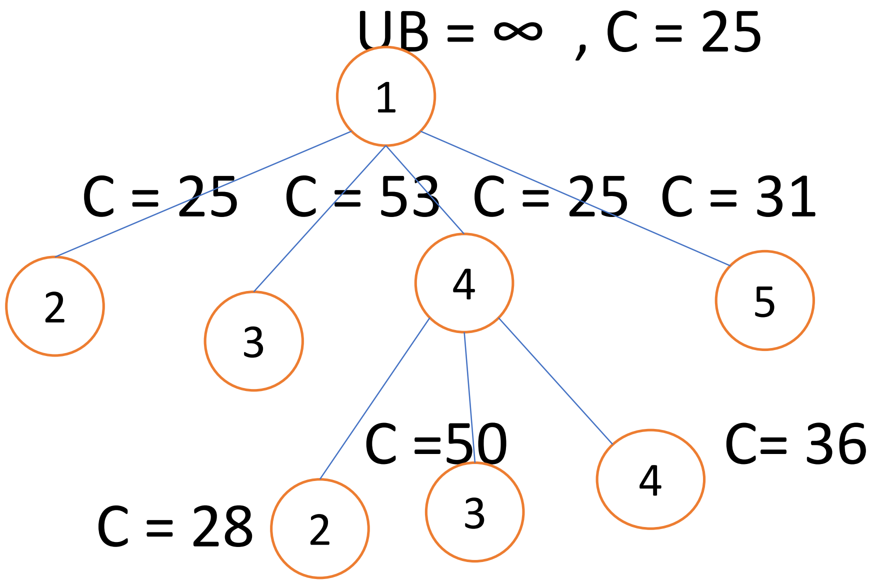




fig 11.

| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| 1 | — | — | — | 0 |
| 0 | 1 | — | — | 0 |
| — | — | — | — | — |
| — | 0 | 0 | — | — |

fig 12.

| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| 1 | — | 0 | — | — |
| 0 | 3 | — | — | — |
| — | — | — | — | — |
| — | 0 | 0 | — | — |

# Continue

- We repeat same process of slide # 16.

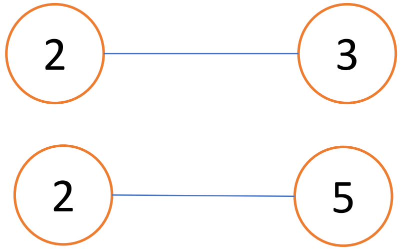


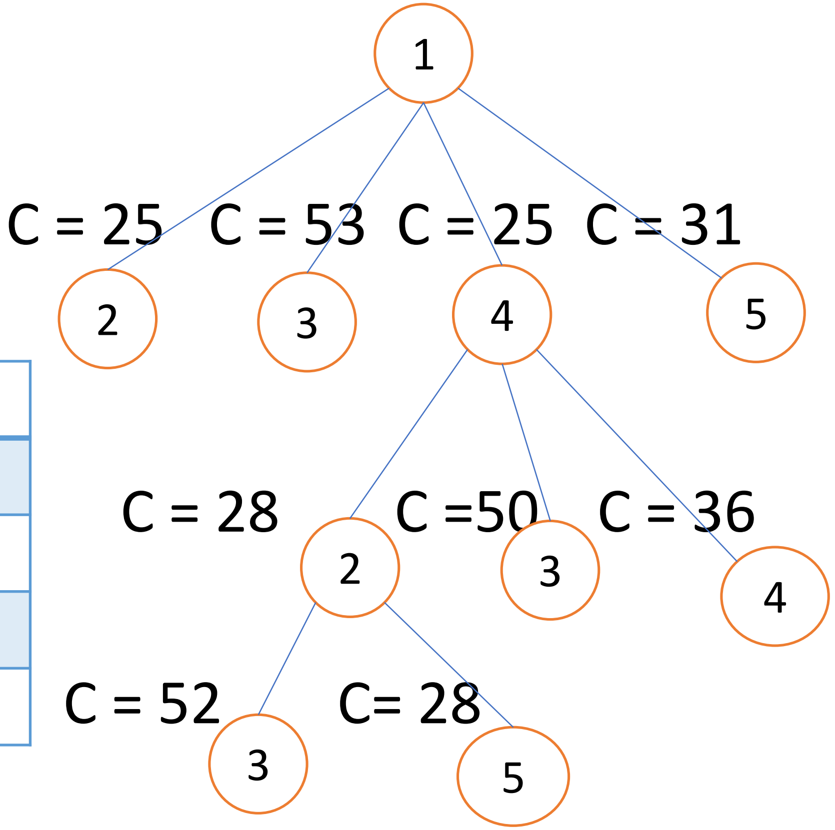

fig 13

| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| — | — | — | — | — |
| — | — | — | — | 0 |
| — | — | — | — | — |
| 0 | — | — | — | — |

fig 14

| | | | | |
|---|---|---|---|---|
| — | — | — | — | — |
| — | — | — | — | — |
| 0 | — | — | — | — |
| — | — | — | — | — |
| — | — | 0 | — | — |

UB = ∞ , C = 25



# Continue

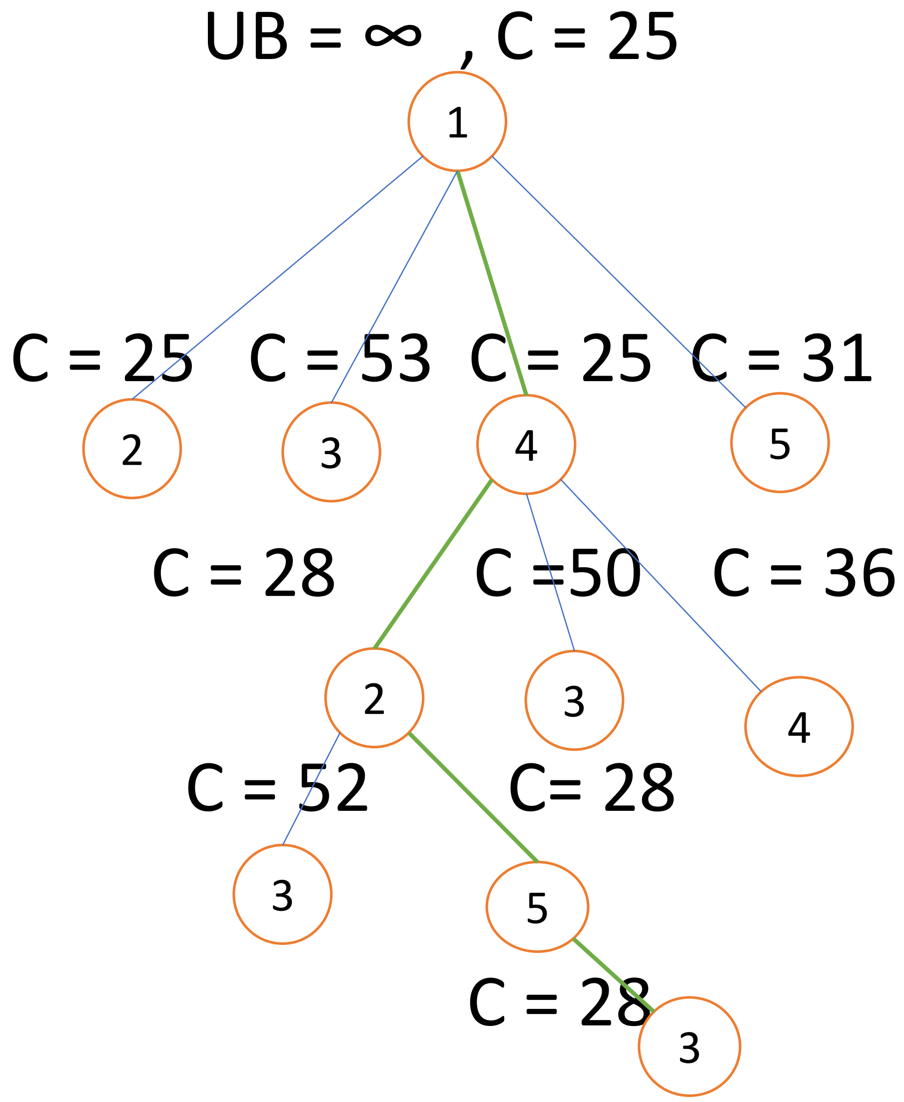
For node 5 — 3

UB = 25

Optimal Sol

1 — 4 — 2 — 5 — 3 — 1

25 + 25 +28 + 28 + 28 = 134 $

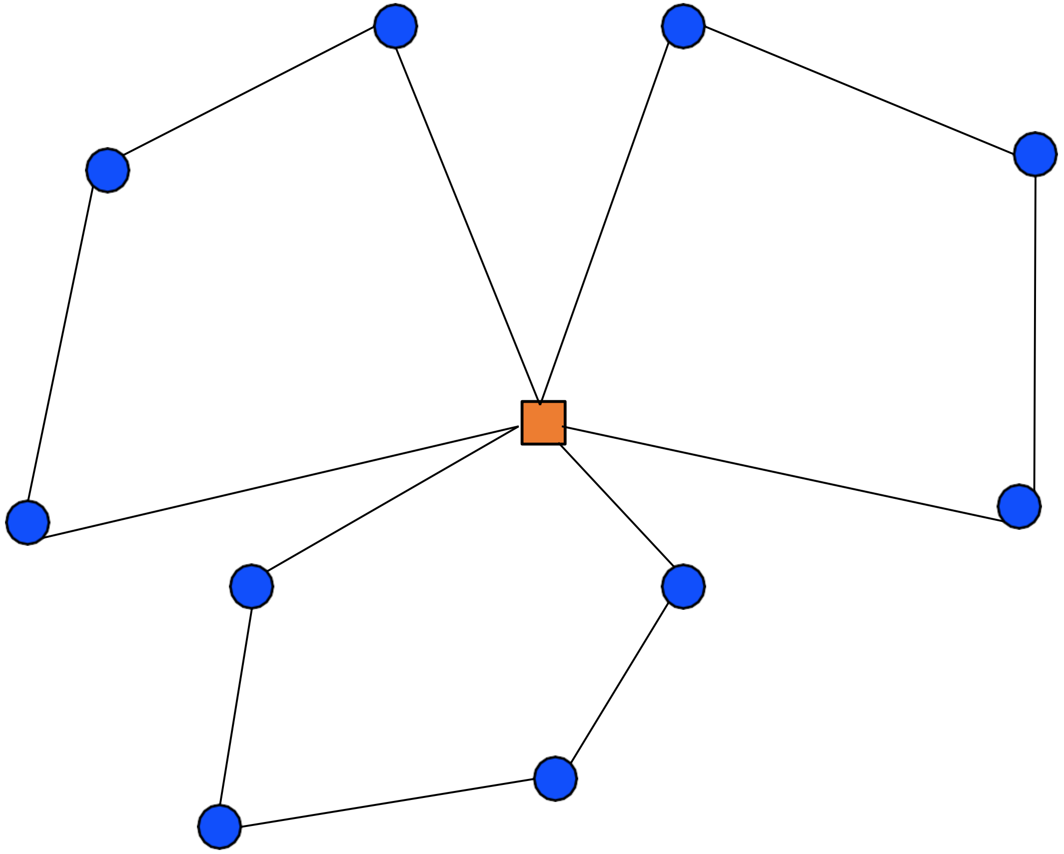


# Vehicle Routing Problem

- In the VRP a number of vehicles located at a central depot has to serve a set of geographically dispersed customers. Each vehicle has a given capacity and each customer has a given demand. The objective is to minimize the total distance travelled
- Type of decisions:
  - Assigning
  - Routing

# Vehicle Routing Problem

3 - vehicle



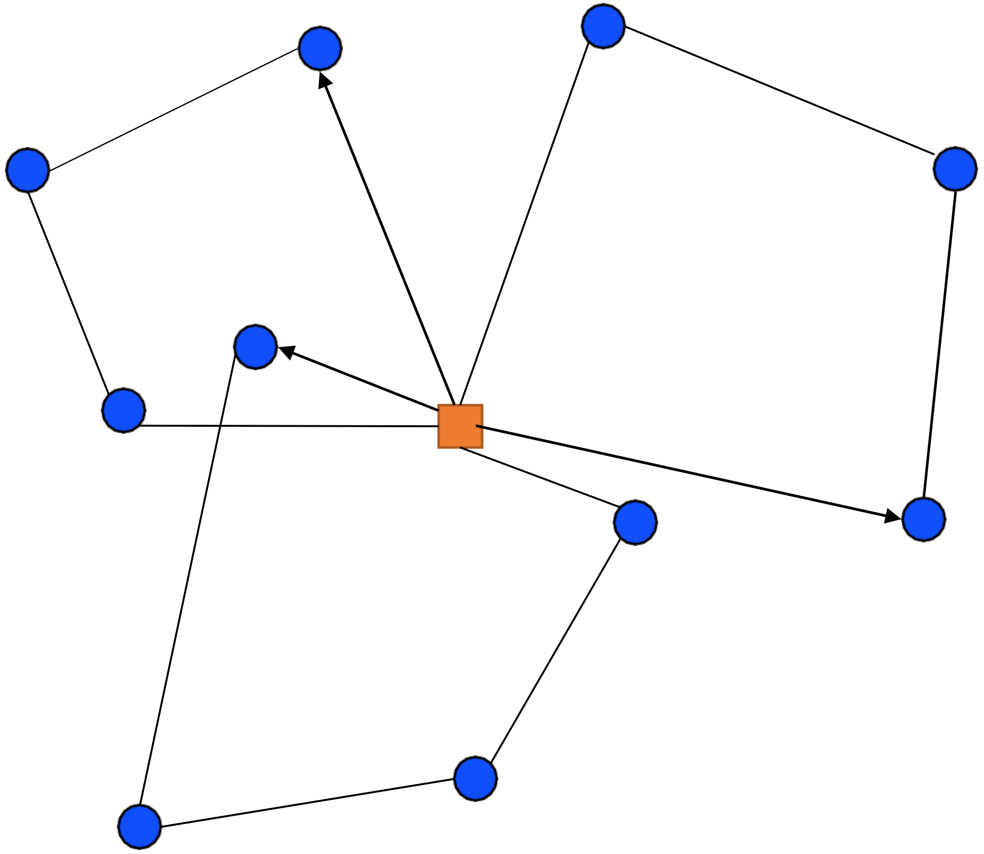
# Vehicle Routing Problem with Time Windows

- In the VRPTW a number of vehicles is located at a central depot and has to serve a set of geographically dispersed customers. Each vehicle has a given capacity. Each customer has a given demand and has to be served within a given time window.

- Type of decisions:
  - assigning
  - routing
  - scheduling

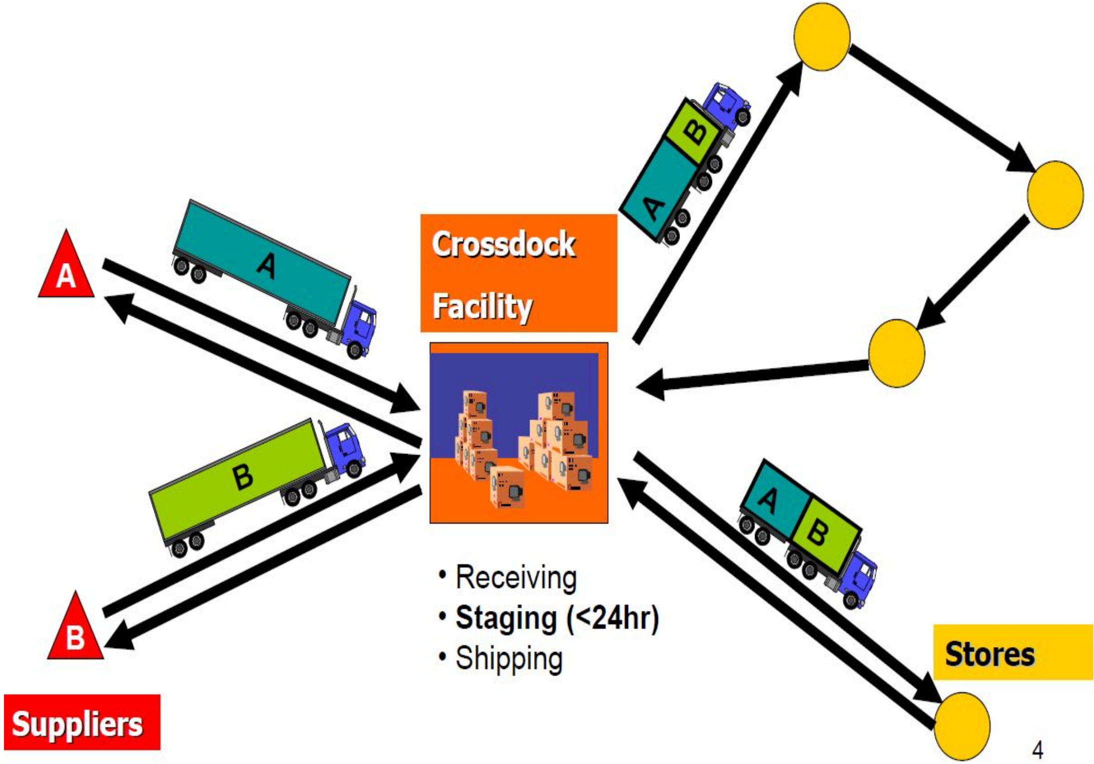
# Vehicle Routing Problem with Time Windows

3- Vehicle



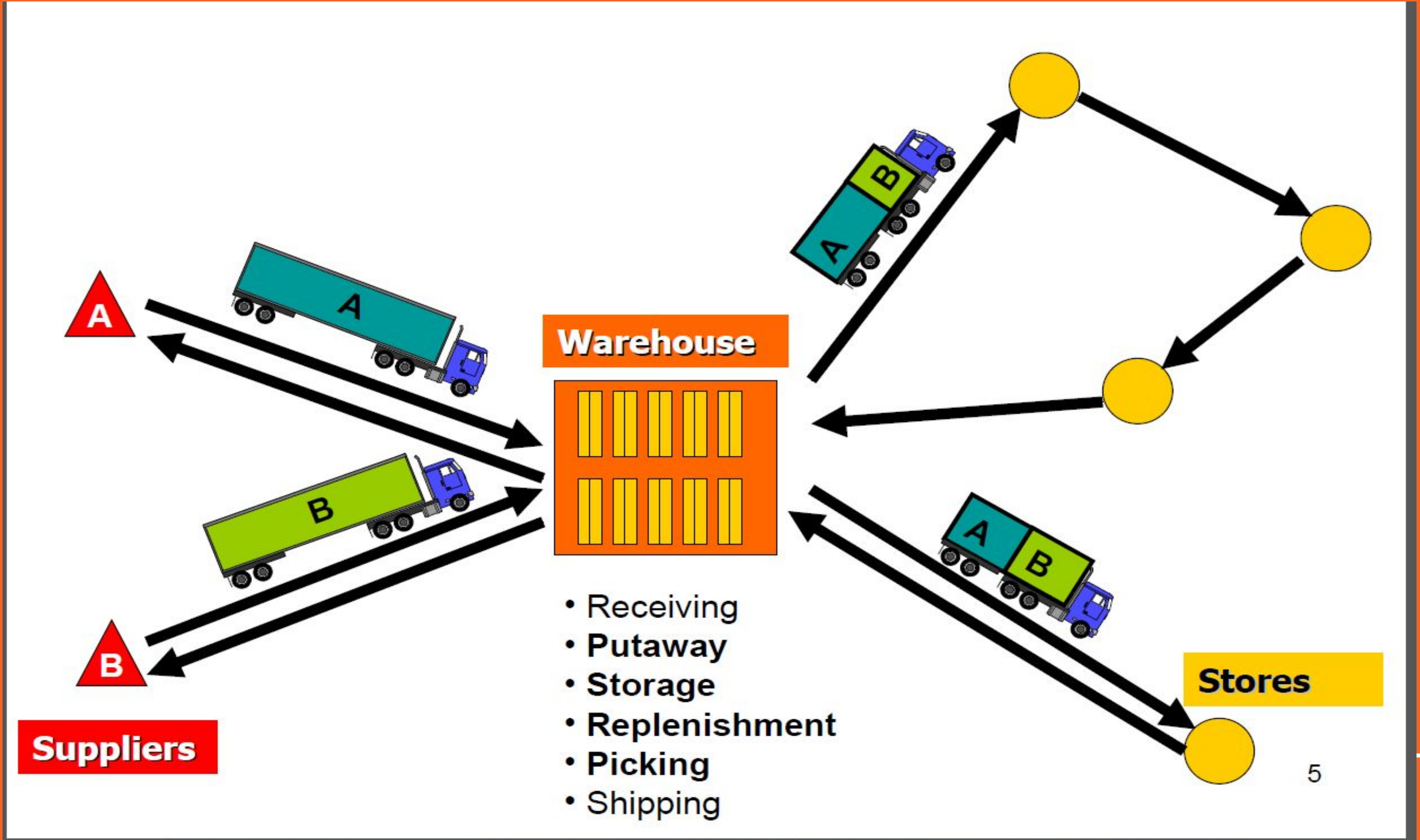
# Cross Docking

It is a logistics procedure where products from a supplier or manufacturing plant are distributed directly to a customer or retail chain with marginal to no handling or storage time.

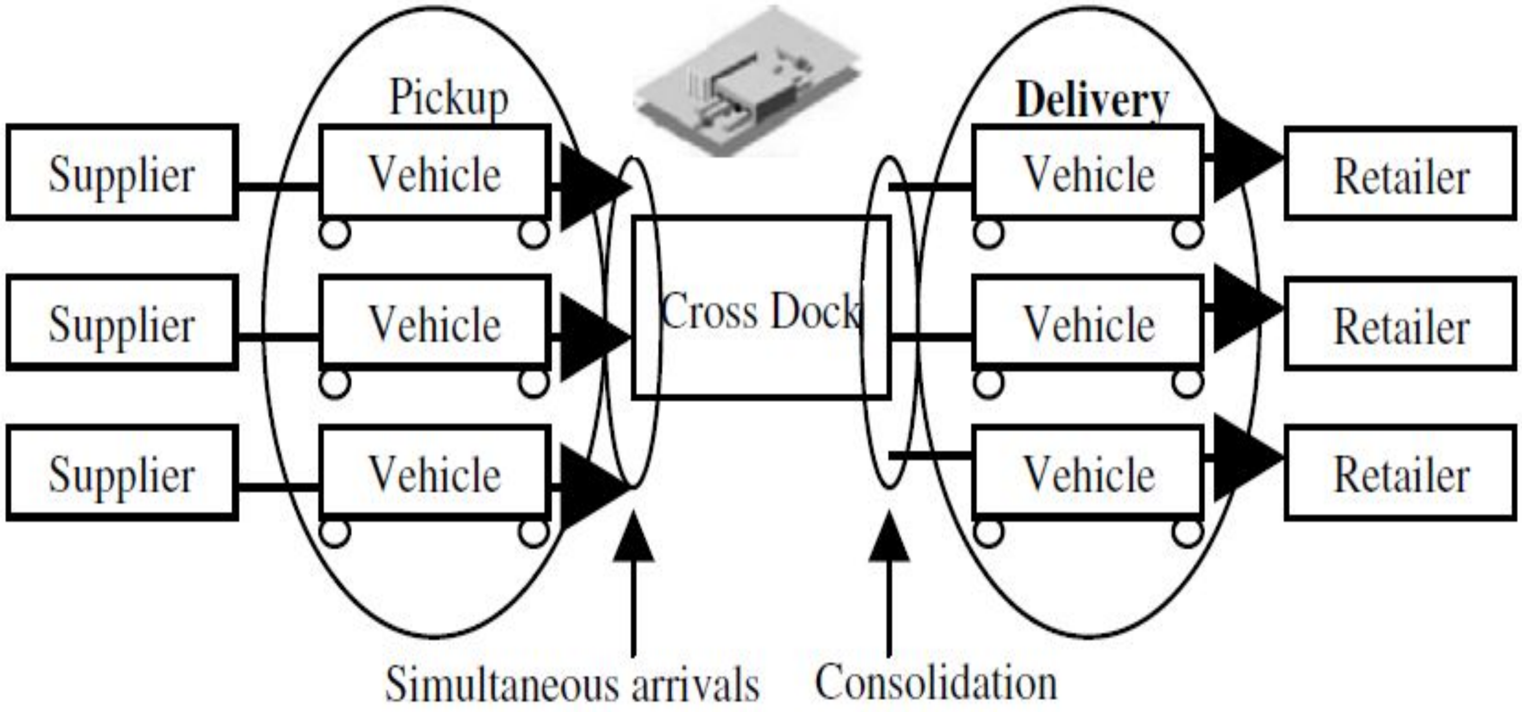


# Why Cross Docking?

# Traditional Way



# Cross Docking



# Introduction To Problem

# Vehicle Routing Scheduling

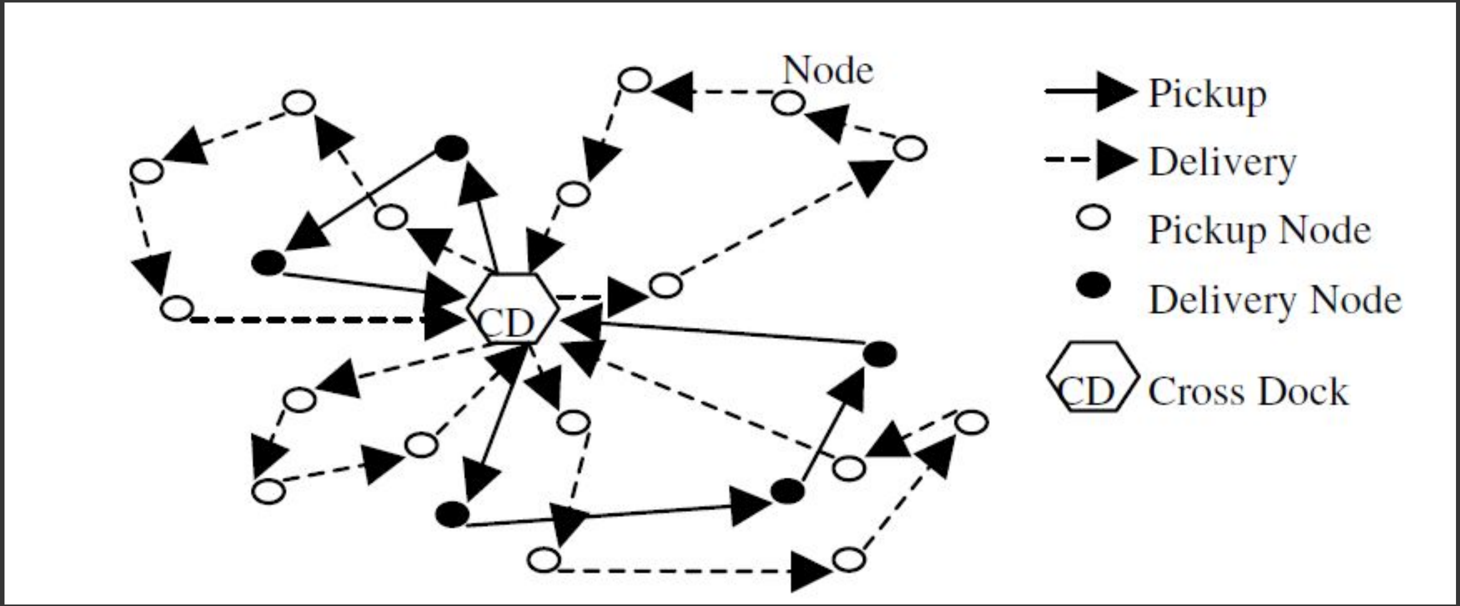
# +

# Cross Docking

# In a

# Supply Chain

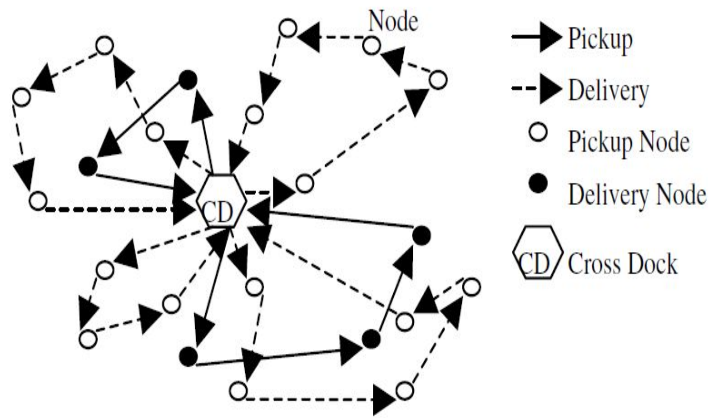
# Problem Description:



$P$ set of nodes in the pickup process
$D$ set of nodes in the delivery process
0 cross-dock
$n$ number of nodes (manufacturers or retailers)
$m$ number of available vehicles
$Q$ maximum capacity of the vehicle
$p_i$ loading quantity in the pickup node $i$
$d_i$ unloading quantity in the delivery node $i$
$tc_{ij}$ transportation cost from node $i$ to node $j$
$c_k$ operational cost of the vehicle $k$
$y_{ij}$ transported quantity of products from node $i$ to node $j$ in the pickup process
$z_{ij}$ transported quantity of products from node $i$ to node $j$ in the delivery process
$t_i$ length of a visit for the vehicle in node $i$
$et_{ij}$ time for the vehicle to move from node $i$ to node $j$
$DT_i^k$ departure time of vehicle $k$ from node $i$
$AT^k$ arrival time of vehicle $k$ at a cross-dock (ending time of the pickup process)

Decision Variable

$$x_{ij}^k : \begin{cases} 1, & \text{if the vehicle } k \text{ moves from the node } i \text{ to the node } j \\ 0, & \text{otherwise} \end{cases}$$



# Application of a Tabu Search Algorithm

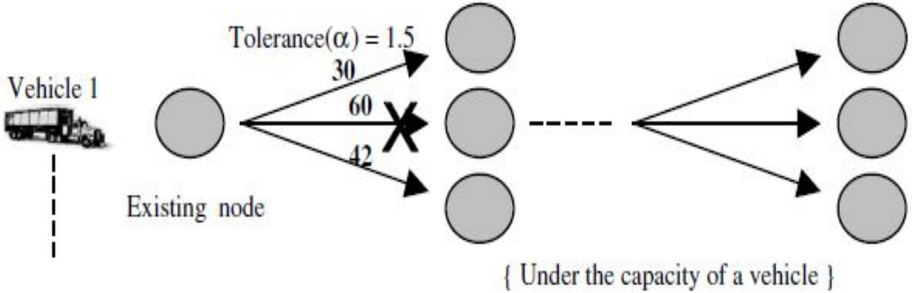
# Tabu Search Algorithm

**Meta-Heuristic Algorithm**

It means it is possible to escape the local solution by the creation of various neighborhoods and to prevent return to the local solution.

# Generating an initial solution

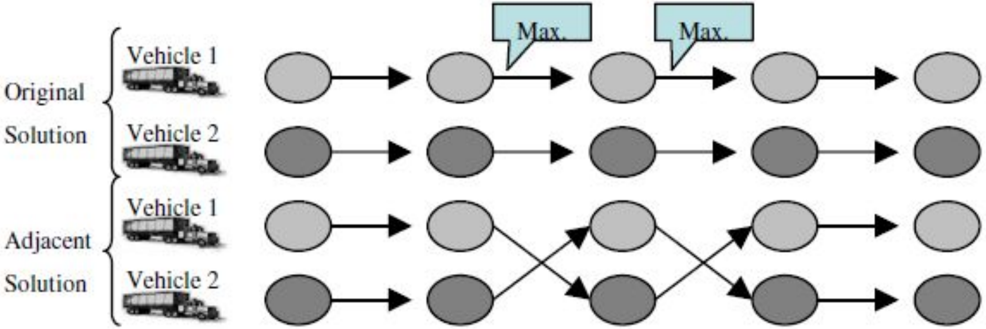
1. **Initialization**
    1.1. Initialize **α** to generate solutions.
2. **Generation of an initial solution**
    2.1. Pickup process
        2.1.1. Select a vehicle to route.
        2.1.2. Search all available routes from the existing node. Generate the candidate list made up of routes whose ratio of transportation cost from the existing location has a minimum transportation cost of less than **α**.



# Generating an initial solution (continued)

2. **Generation of an initial solution (continued)**
    2.1. Pickup process (continued)
        2.1.3. Select a route and its related node randomly from the candidate list.
        2.1.4. Allocate a product in the selected node to a selected vehicle, and replicate 2.1.2 and 2.1.3 under the capacity of a vehicle. If the capacity of a vehicle is exceed, execute 2.1.5.
        2.1.5. Select another vehicle to route. Replicate 2.1.2–2.1.4 for all remaining vehicles.
    2.2. Delivery process
        2.2.1 Replicate the same steps in the pickup process.

# Tabu Search Algorithm




3. **Generation of an adjacent solution**
   3.1. Pickup process
      3.1.1. Find two routes whose transportation cost between two nodes is more than the other nodes.
      3.1.2. Exchange the two routes with the routes in corresponding routes for another randomly selected vehicle. If there are not any corresponding routes or additional vehicle capacity is needed, the sequence of the two routes is changed.
      3.1.3. If the total cost is improved, keep the present solution as the best solution. If not, execute 3.1.1.
      3.1.4. Find the best solution by repeating 3.1.1–3.1.3

# Tabu Search Algorithm (continued)

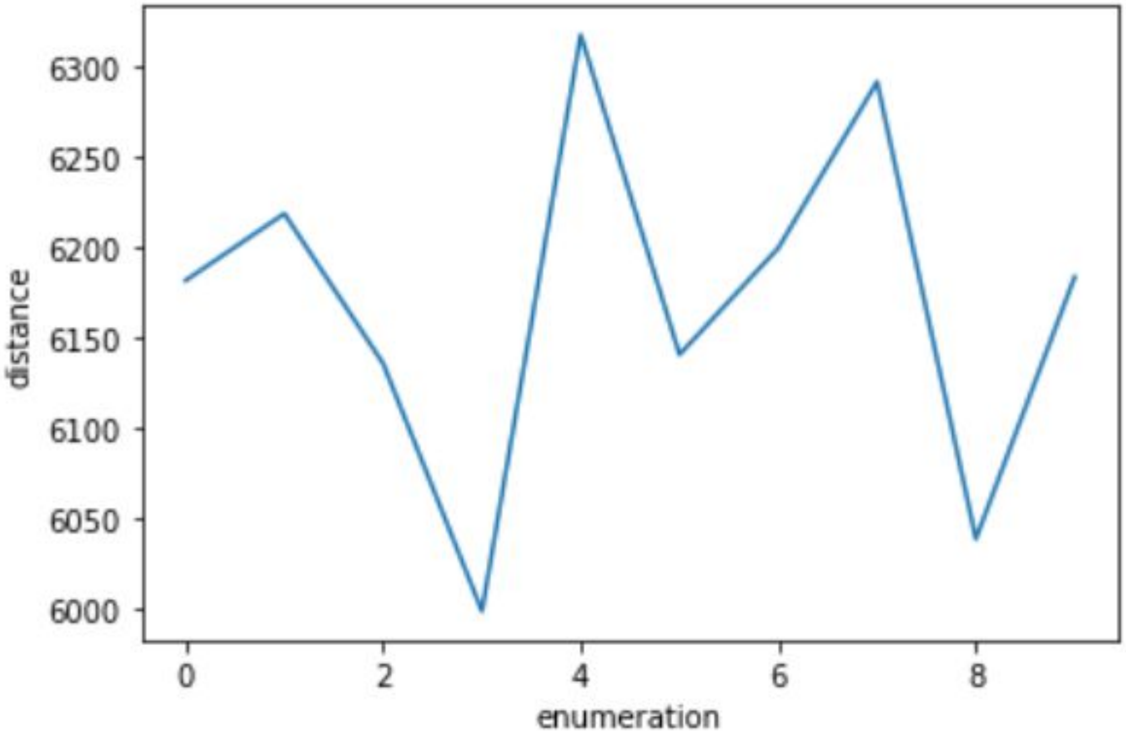
3. **Generation of an adjacent solution (continued)**
   3.2. Delivery process
      3.2.1. Replicate the same steps as in the pickup process.
4. **Generation of the tabu list**
   4.2. Found pairs (route, vehicle), which are the result in 3.1.4, are added to the tabu list.
5. **Terminating condition 1**
   5.2. If the limited number of tabu searches is not exceeded, replicate Step 3.
6. **Terminating condition 2**
   6.2. If the limited number of generated initial solutions was not exceeded, replicate Steps 2 and 3.

# Results

```
Route for vehicle 0:
 0 ->  7 ->  2 ->  8 -> 0
Distance of the route: 1732m

Route for vehicle 1:
 0 ->  5 ->  9 -> 0
Distance of the route: 1424m

Route for vehicle 2:
 0 ->  10 ->  1 ->  6 -> 0
Distance of the route: 1633m

Route for vehicle 3:
 0 ->  4 ->  3 -> 0
Distance of the route: 1352m

Total Distance of all routes: 6141m
```

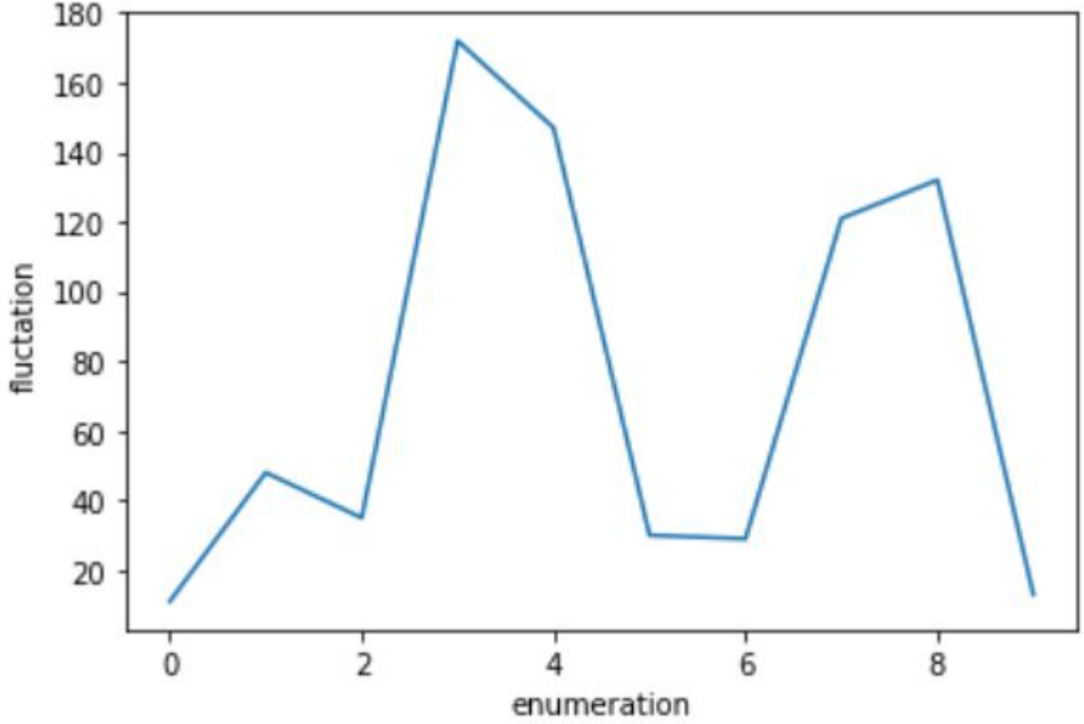
**Single enumeration**
**No. of nodes = 10**
**No. of vehicles = 4**

# Results




**Total Cost in 10 enumeration**
**No. of nodes = 10**
**No. of vehicles = 4**

# Results




**Fluctuation in Cost in 10 enumeration**
**No. of nodes = 10**
**No. of vehicles = 4**

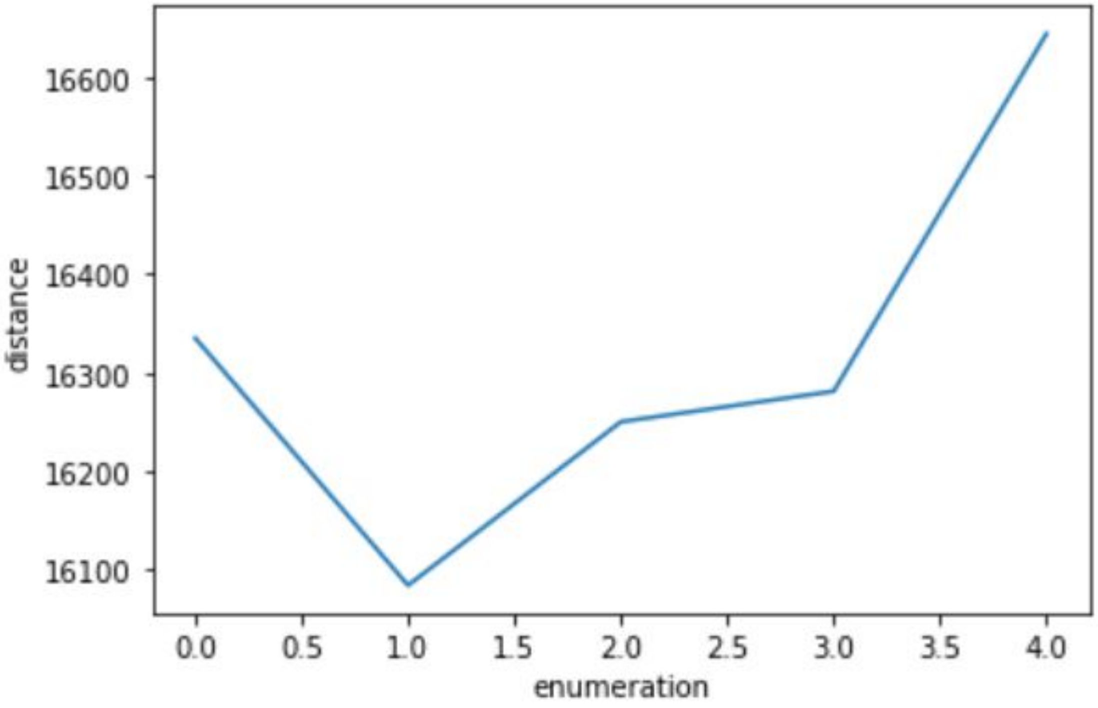
# Results

```
Route for vehicle 0:
 0 ->  15 ->  21 ->  11 ->  25 -> 0
Distance of the route: 2136m

Route for vehicle 1:
 0 ->  4 ->  5 ->  9 ->  3 -> 0
Distance of the route: 2145m

Route for vehicle 2:
 0 ->  26 ->  24 -> 0
Distance of the route: 1387m

Route for vehicle 3:
 0 ->  27 ->  22 ->  29 ->  23 -> 0
Distance of the route: 2154m

Route for vehicle 4:
 0 ->  1 ->  2 ->  6 ->  10 -> 0
Distance of the route: 2155m
```

```
Route for vehicle 5:
 0 ->  18 ->  19 ->  17 ->  30 -> 0
Distance of the route: 2124m

Route for vehicle 6:
 0 ->  7 ->  16 ->  8 ->  14 -> 0
Distance of the route: 2123m

Route for vehicle 7:
 0 ->  13 ->  28 ->  12 ->  20 -> 0
Distance of the route: 2111m

Total Distance of all routes: 16335m
```

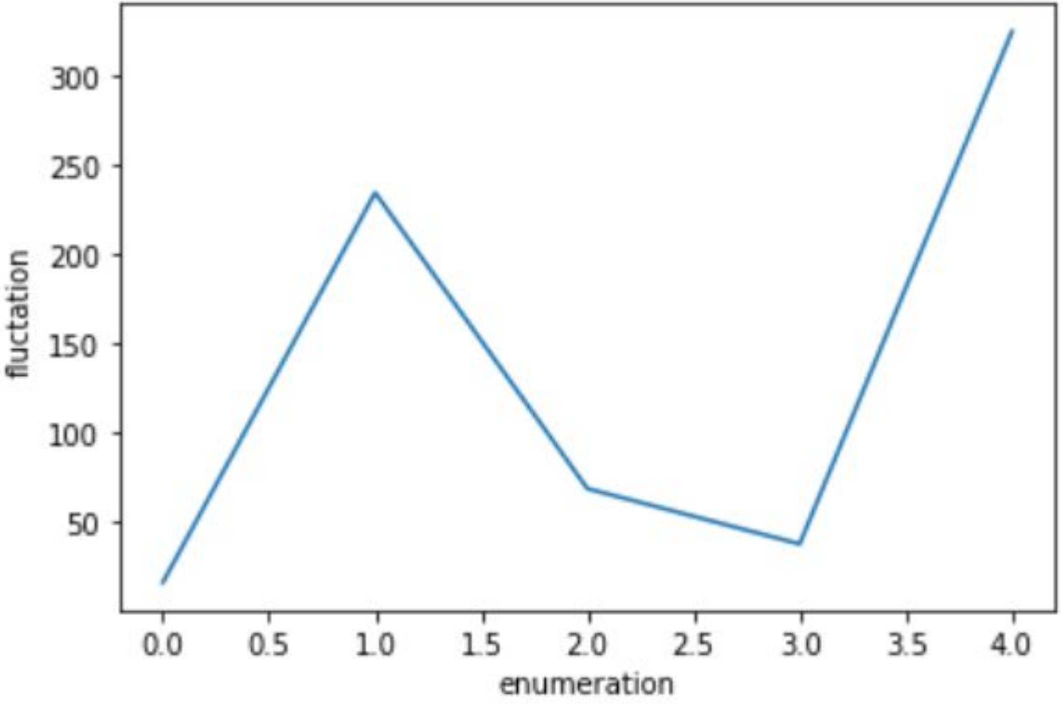
**Single enumeration**
**No. of nodes = 30**
**No. of vehicles = 8**

# Results




**Total Cost in 5 enumeration**
**No. of nodes = 30**
**No. of vehicles = 8**

# Results




**Fluctuation in Cost in 5 enumeration**
**No. of nodes = 30**
**No. of vehicles = 8**

# References


- Vehicle routing scheduling for cross-docking in the supply chain by Young Hae Lee, Jung Woo Jung, Kyong Min Lee
- https://developers.google.com/optimization/
- https://en.wikipedia.org/wiki/Tabu_search