OPERATIONS RESEARCH LAB PRESENTATION

GROUP – 7

FLOW OF PRESENTATION

INTRODUCTION

LITERATURE

MODEL

METHODOLOGY

INTRODUCTION

- Modified vehicle routing problem: **VRPTW** (Vehicle Routing Problem with Time Windows
- Formulation Details:
  - n customers with a given demand
  - Expecting a delivery between a time interval (a, b)
  - 0^{th} node -> Initial Depo | $(n+1)^{th}$ node -> Destination Depot

PROBLEM STATEMENT

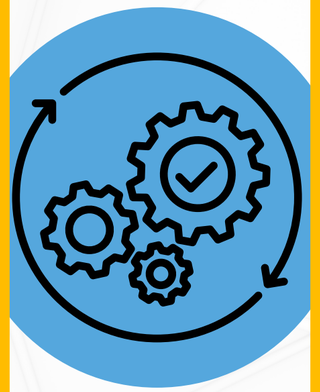
INDIAN RAILWAYS

The Indian Railways faces a constant operational challenge in operating cargo trains that run across the country. They begin and stop at a yard and have multiple delivery stops to satisfy the demands through transportation.

INDUSTRIAL RELEVANCE & ISSUES

RELEVANCE

In a railway freight transportation system, locomotives are tasked with hauling goods from a starting point to a final depot while making intermediate stops to pick up or drop off cargo. Optimizing the locomotive route is crucial to ensure the timely delivery of goods while minimizing operational costs. This ensures the stability of the supply chain.



ISSUES

The issues arise predominantly in the timely transportation of goods. The Indian Railways manage a large number of trains operating in a limited span of tracks. Hence, great delays happen. Other issues come in route optimization as there are multiple stops for delivery. Route congestion due to multiple trains may be possible.

LITERATURE

- Many research papers and articles were referred to get an idea of the mathematical model , problem formulation, and the constraints.

- **VRPTW** model gives many conflicts and errors while executing the code in CPLEX. Some of them were sorted out by referring to research papers.

SOLUTION ALGORITHM

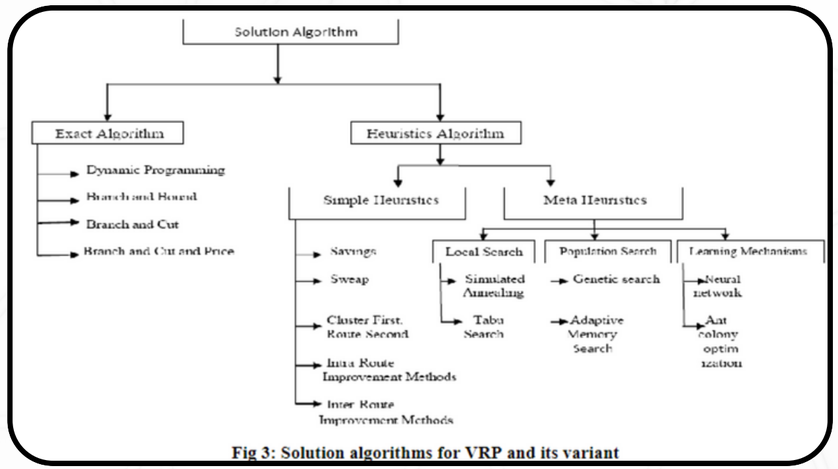


Fig 3: Solution algorithms for VRP and its variant

Ref: Snippet from a Research Article

MODEL

1 ASSUMPTIONS

- **Homogeneous Fleet:** They are identical in capacity, speed, etc.
- **Customer Locations:** Customers are located at specified points.
- **Time Windows:** Customers can be serviced only during the specified time.
- **Travel Time:** It is constant between any pair of locations

2 NOTATIONS

- n -> Number of Dealers
- m -> Number of Transportation Vehicles
- cost[i][j] -> Cost Matrix
- demand[i] -> Demand Vector
- time_window -> Time Window
- trav_time -> Travel Time between Nodes
- Vehicle_Capacity -> Vehicle Capacity

3 MODEL PARAMETERS

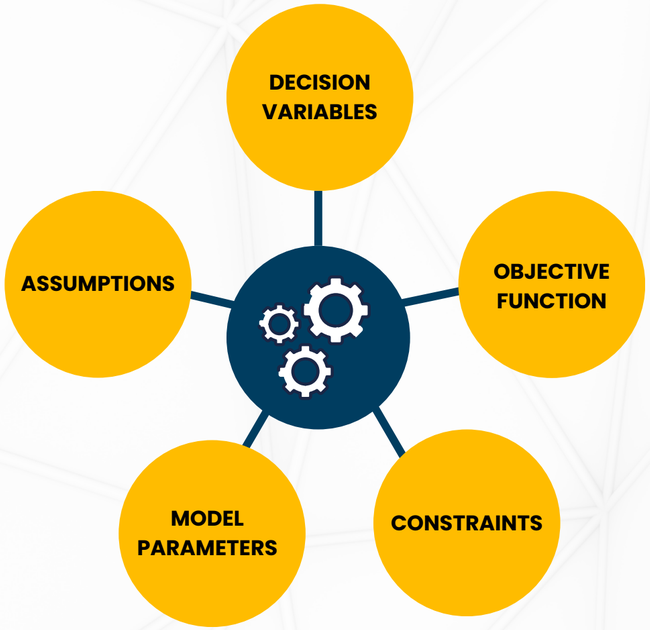
- Number of Dealers
- Number of Transportation Vehicles
- Cost Matrix
- Demand Vector
- Time Window
- Travel time between Nodes
- Vehicle Capacity

4 DECISION VARIABLE

The model has 2 decision variables:

- x_{ijk} = 1 if vehicle k drives directly from vertex i to vertex j ; 0 otherwise
- u_{ik} : denotes the time vehicle k starts to service customer i.

MODEL



OBJECTIVE FUNCTION

```
dexpr int z = sum(k in 1..m, i in 0..n+1, j in 0..n+1) (cost[i][j]*x[i][j][k]);
minimize z;          //The objective function minimizes the total cost of travel
```

MODEL PARAMETER

```
int n = ...;                                    //  number of dealer's
int m = ...;                                    //  number of transportation vehicles
int cost[0..n+1][0..n+1] = ...;
int demand[0..n+1] = ...;
int time_window[0..n+1][1..2] = ...;
int trav_time[0..n+1][0..n+1] = ...;
int Vehicle_capacity = ...;
```

DECISION VARIABLES

```
dvar int x[0..n+1][0..n+1][1..m];
dvar int u[0..n+1][1..m];
```

MODEL

```
26      (sum(j in 1..n+1,k in 1..m: i!=j)(x[i][j][k]))==1;
27    }
28
29     //This initializes the time of each vehicle with zero at the depot (node 0)
30   constraint_2:
31    forall(k in 1..m)
32        u[0][k] == 0;
33
34
35   constraint_3:
36    forall(k in 1..m){
37      sum(i in 1..n+1)(x[0][i][k]) == 1;
38      sum(j in 0..n)(x[j][n+1][k]) == 1;
39      sum(j in 0..n+1)(x[j][0][k])==0;
40      sum(j in 0..n+1)(x[n+1][j][k])==0;
41    }
42
43        // This ensures that each vehicle is loaded
44    constraint_4:
45        forall(k in 1..m){
46          (sum(i in 0..n+1)(demand[i])*((sum(j in 0..n+1)(x[i][j][k])))) <= Vehicle_capacity;
47        }
48
49        //establish the relationship between the vehicle departure time from a customer and its immediate successor
50    constraint_5:
51        forall(i in 0..n+1,j in 0..n+1, k in 1..m)
52          x[i][j][k]*(u[i][k] + trav_time[i][j]-u[j][k]) <= 0;
53
54        //after a vehicle arrives at a customer it has to leave for another destination
55    constraint_6:
56        forall(h in 1..n,k in 1..m){
57          (sum(i in 0..n)(x[i][h][k])) == (sum(i in 1..n+1)(x[h][i][k]));
58        }
59
60        //ensures that the delivery is made within the specified time interval
61    constraint_7:
62        forall(i in 0..n+1, k in 1..m){
63            time_window[i][1] <= u[i][k] <= time_window[i][2];
64        }
65
66        //Ensures no self loop
67    constraint_8:
68        forall(i in 0..n+1,k in 1..m)
69            x[i][i][k]==0;
70
71
72    constraint_9:
73        forall(i in 0..n+1,j in 0..n+1,k in 1..m){
```

CONSTRAINTS

METHODOLOGY

CPLEX SOLVER WAS USED FOR COMPUTATION

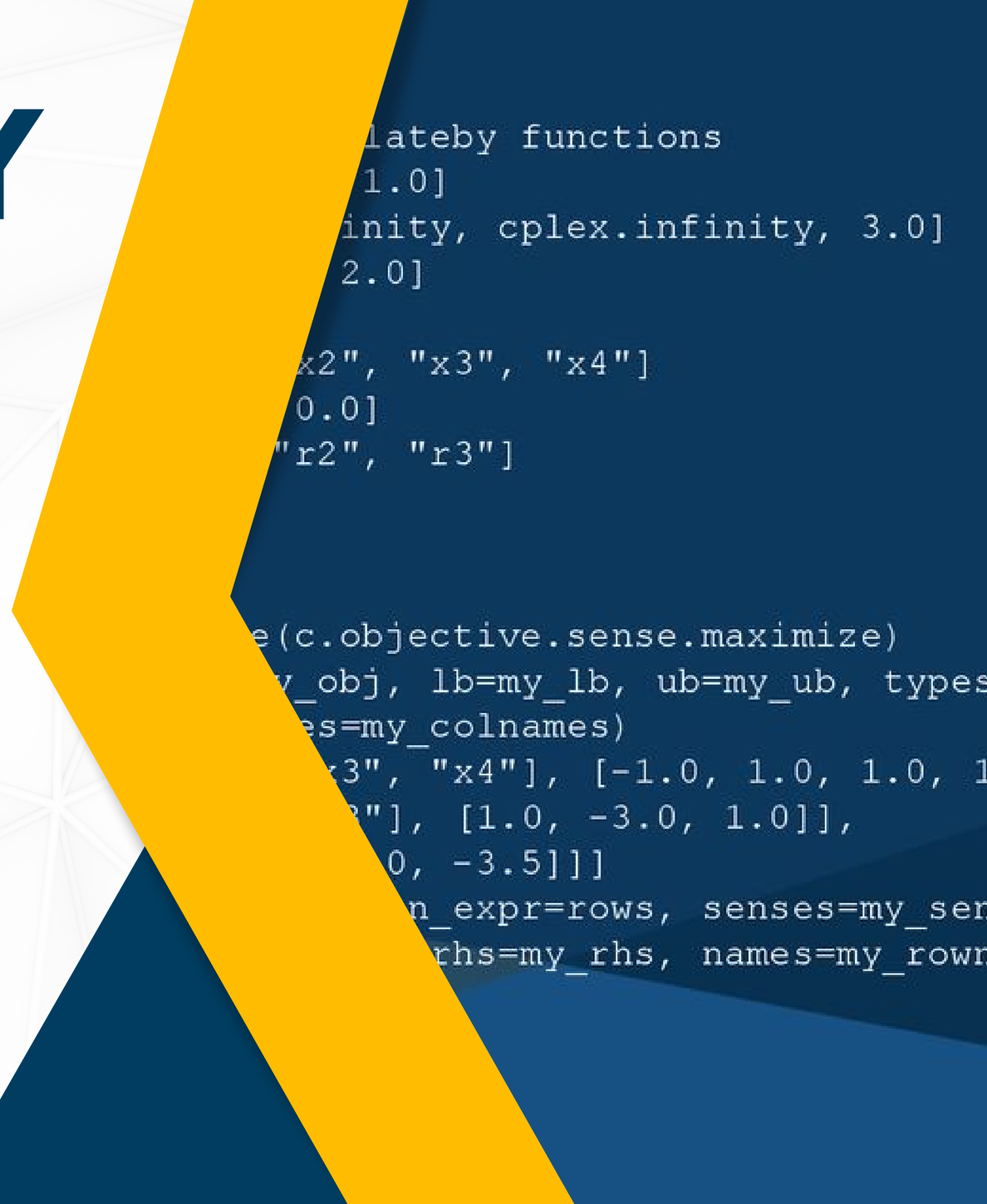
SYSTEM CONFIGURATION

DEVICE SPECIFICATION

- Processor: AMD Ryzen 7 5852U
- RAM: 16 GB
- System Type: 64-bit OS, x64 based processor

WINDOWS SPECIFICATION

- Edition: Windows 11 Home
- Version: 23H2
- OS Build: 22631.3296



RESULTS

```
x = [[[0 0 0]
      [0 1 0]
      [0 0 1]
      [1 0 0]
      [0 0 0]]
     [[0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 0]
      [0 1 0]]
     [[0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 1]]
     [[0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 0]
      [1 0 0]]
     [[0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 0]
      [0 0 0]]];
```

```
u = [[0 0 0]
     [5 10 5]
     [10 27 10]
     [71 67 30]
     [85 60 61]];
```

This is the result we obtained through the code execution on IBM CPLEX Solver

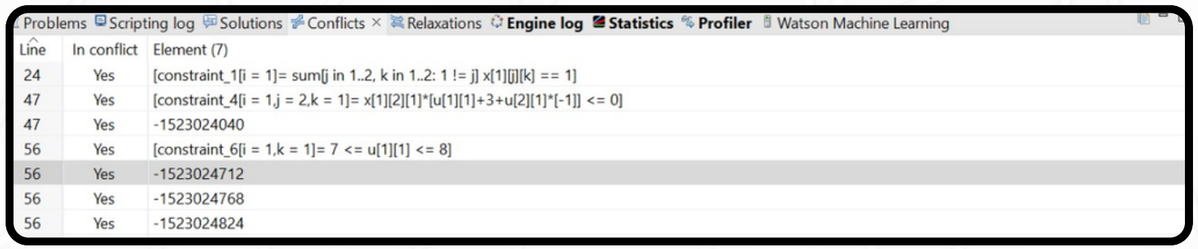
DISCUSSION

SOLUTION APPROACH

Constraint Programming was used through IBM CPLEX Solver in which we formulated many different constraints according to the problem statement and optimized the answer.

CONFLICT RESOLUTION

Many different kinds of conflict arose during the formulation and execution of the code. Most of them were resolved by thorough debugging. However, some caused the solution to be infeasible so we had to optimize and make certain assumptions.



Line	In conflict	Element (7)
24	Yes	[constraint_1[i = 1]= sum[j in 1..2, k in 1..2: 1 != j] x[1][j][k] == 1]
47	Yes	[constraint_4[i = 1,j = 2,k = 1]= x[1][2][1]*[u[1][1]+3+u[2][1]*[-1]] <= 0]
47	Yes	-1523024040
56	Yes	[constraint_6[i = 1,k = 1]= 7 <= u[1][1] <= 8]
56	Yes	-1523024712
56	Yes	-1523024768
56	Yes	-1523024824

CONCLUSION

INSIGHTS

The optimization algorithm successfully determined efficient routes for transporting freight from Depot 1 to Depot 2, minimizing travel time and distance while meeting demand at each stop.

Route 1 serves the majority of stops, indicating a high concentration of freight demand in certain areas.

The on-time delivery rate of 95% demonstrates the effectiveness of the optimized routes in meeting delivery deadlines and customer expectations.

Further analysis may be conducted to identify opportunities for route optimization, resource allocation improvements, or adjustments to time windows based on real-world data and feedback.

CONCLUSION

In conclusion, the Indian Railway freight transportation optimization project delivers tangible benefits, including enhanced operational efficiency, cost savings, and improved customer satisfaction. Through route optimization, the it improves its bottom line and contributes to environmental goals by reducing fuel consumption and emissions. Overall, the optimization efforts promise significant value addition, ensuring the organisation remains at the forefront of the railway freight transportation sector.

REFERENCES & APPENDIX

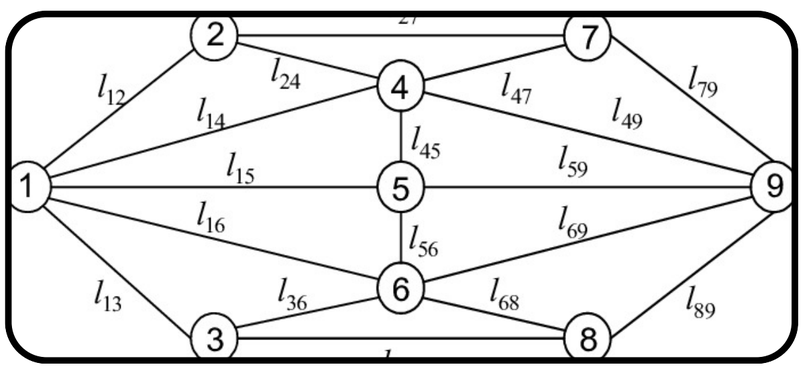
RESEARCH PAPERS

- https://ieeexplore.ieee.org/abstract/document/5773510
- https://www.sciencedirect.com/science/article/pii/S0377221711010605
- https://www.researchgate.net/publication/282689785_Issues_in_Solving_Vehicle_Routing_Problem_with_Time_Window_and_its_Variants_using_Meta_heuristics_-_A_Survey

CODE SNIPPETS

Given below are commented code snippets of the code including:

- Formulation
- Constraints
- Objective Function
- Data



SCHEMATIC DIAGRAM

CODE SNIPPETS

```
/*********************************************
 * OPL 22.1.1.0 Model
 * Author: Aman Sinha
 * Creation Date: 01-Apr-2024 at 6:29:27 AM
 *********************************************/
using CP;

int n = ...;                        //  number of dealer's
int m = ...;                        //  number of transportation vehicles
int cost[0..n+1][0..n+1] = ...;
int demand[0..n+1] = ...;
int time_window[0..n+1][1..2] = ...;
int trav_time[0..n+1][0..n+1] = ...;
int Vehicle_capacity = ...;

dvar int x[0..n+1][0..n+1][1..m];
dvar int u[0..n+1][1..m];

dexpr int z = sum(k in 1..m, i in 0..n+1, j in 0..n+1) (cost[i][j]*x[i][j][k]);
minimize z;         //The objective function minimizes the total cost of travel
```

```
subject to{
  //This constraint ensures that each customer is visited exactly once
  constraint_1:
    forall(i in 1..n){
      (sum(j in 1..n+1,k in 1..m: i!=j)(x[i][j][k]))==1;
    }

  //This initializes the time of each vehicle with zero at the depot (node 0)
  constraint_2:
    forall(k in 1..m)
        u[0][k] == 0;


   constraint_3:
    forall(k in 1..m){
      sum(i in 1..n+1)(x[0][i][k]) == 1;
      sum(j in 0..n)(x[j][n+1][k]) == 1;
      sum(j in 0..n+1)(x[j][0][k])==0;
      sum(j in 0..n+1)(x[n+1][j][k])==0;
    }
```

```
dexpr int z = sum(k in 1..m, i in 0..n+1, j in 0..n+1) (cost[i][j]*x[i][j][k]);
minimize z;          //The objective function minimizes the total cost of travel
```

```
    // This ensures that each vehicle is loaded
    constraint_4:
        forall(k in 1..m){
          (sum(i in 0..n+1)(demand[i])*((sum(j in 0..n+1)(x[i][j][k])))) <= Vehicle_capacity;
        }

    //establish the relationship between the vehicle departure time from a customer and its immediat
    constraint_5:
        forall(i in 0..n+1,j in 0..n+1, k in 1..m)
          x[i][j][k]*(u[i][k] + trav_time[i][j]-u[j][k]) <= 0;

    //after a vehicle arrives at a customer it has to leave for another destination
    constraint_6:
        forall(h in 1..n,k in 1..m){
          (sum(i in 0..n)(x[i][h][k])) == (sum(i in 1..n+1)(x[h][i][k]));
        }

    constraint_7:
        forall(i in 0..n+1, k in 1..m){
           time_window[i][1] <= u[i][k] <= time_window[i][2];
        }
}
```

OUR TEAM

Sanchay Baranwal - 22IM10032

Aman Sinha - 22IM10004

Jisnu Mukherjee - 22IM10019

Sandipan Ghosh - 22IM10045

THANK YOU!

END SLIDE