Eindhoven University of Technology

1CM110 – Decision Making in Transport and Mobility

Assignment 1

Yusuf Sami (Student ID: 2399296)

February 13, 2026

1 Formulating and Solving the Basic Routing Problem

In the first part of the assignment, we start with a simplified version of the problem. We temporarily ignore vehicle capacity, time windows, and service times, and aim only to minimize the total distance traveled.

1.1 TSP Formulations

Notation

Sets

- $N = \{0, 1, \ldots, n\}$: set of all nodes, where node 0 denotes the depot and nodes $1, \ldots, n$ denote customers.

Parameters

- c_{ij}: travel distance (cost) from node i to node j.
- n: number of customers.

Decision Variables

- $x_{ij} = \begin{cases} 1 & \text{if the tour travels directly from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$
- u_i: order of node i in the tour (MTZ formulation only).

1.1.1 (a) Mathematical Models for MTZ and DFJ

Miller–Tucker–Zemlin (MTZ) Formulation

$$\begin{aligned}
\min \quad & \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} & & (1) \\
\text{s.t.} \quad & \sum_{j \in N} x_{ij} = 1, & \forall i \in N \quad & (2) \\
& \sum_{i \in N} x_{ij} = 1, & \forall j \in N \quad & (3) \\
& u_i - u_j + n\, x_{ij} \le n - 1, & \forall i, j \in N,\ i \ne j \quad & (4) \\
& 2 \le u_i \le n, & \forall i \in N \setminus \{0\} \quad & (5) \\
& u_0 = 1, & & (6) \\
& x_{ij} \in \{0, 1\} & \forall i, j \in N \quad & (7)
\end{aligned}$$

Dantzig–Fulkerson–Johnson (DFJ) Formulation

$$\begin{aligned}
\min \quad & \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} & & (8) \\
\text{s.t.} \quad & \sum_{j \in N} x_{ij} = 1, & \forall i \in N \quad & (9) \\
& \sum_{i \in N} x_{ij} = 1, & \forall j \in N \quad & (10) \\
& \sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, & \forall S \subset N,\ |S| \ge 2 \quad & (11) \\
& x_{ij} \in \{0, 1\}, & \forall i, j \in N \quad & (12)
\end{aligned}$$

1.1.2 (b) MTZ Implementation (`q_1_1_b.py`)

The MTZ formulation was implemented in Python using the Gurobi optimizer . The model includes binary arc decision variables, continuous ordering variables u_i, degree constraints, MTZ subtour elimination constraints, and an objective minimizing total travel distance. The formulation solved efficiently for all instance sizes tested.

1.1.3 (c) DFJ Implementation (`q_1_1_c.py`)

The DFJ formulation was implemented by explicitly generating all subsets $S \subseteq N$ and adding subtour elimination constraints for each subset. Due to the exponentially increasing number of constraints, the formulation is only suitable for small instances. Full DFJ models were solved successfully for small sizes, while larger instances approached time and memory limits.

Table 1: Performance comparison of MTZ and DFJ formulations

Instance Size	Formulation	Constraints	Time (s)	Objective	Status
5 customers	MTZ	32	0.01	43.0	Optimal
	DFJ	38	0.01	43.0	Optimal
10 customers	MTZ	112	0.01	56.0	Optimal
	DFJ	1035	0.03	56.0	Optimal
15 customers	MTZ	242	0.23	102.0	Optimal
	DFJ	32,784	2.44	102.0	Optimal
20 customers	MTZ	-	-	-	Optimal
	DFJ	1,048,597	272.37	124.0	Optimal
25 customers	MTZ	652	88.68	133.0	Optimal
	DFJ	33,554,458	-	-	Time Limit

1.1.4 (d) Performance Comparison

The computational results show a clear distinction between the two formulations. The MTZ formulation scales reasonably well and can solve instances up to 25 customers within the time limit. Its performance is not perfectly monotonic (e.g., the solver struggles specifically at $n = 20$ but solves $n = 19$ and $n = 21$ faster), which is expected due to the weak LP relaxation and branch-and-bound sensitivity to instance structure.

In contrast, the DFJ formulation becomes slow very quickly. Because subtour elimination constraints must be added lazily and their number grows exponentially, DFJ performs well only for small instances. For 25 customers, the model either cannot be built within the time limit or fails to find a solution.

Overall, MTZ outperforms DFJ for medium-sized instances: it has a fixed number of constraints and remains computationally manageable, whereas DFJ provides a tighter relaxation but becomes impractical as the number of customers increases.

1.2 Improving DFJ

1.2.1 (a) Detecting subtours.(`q_1_2_a.py`)

Our subtour detection exploits the property that TSP solutions form disjoint cycles due to flow conservation constraints ($\sum_j x_{ij} = 1$ for all i).

The algorithm maintains a set of unvisited nodes and iteratively constructs cycles by following successor links: starting from node i, we trace $i \to j \to k \to \ldots$ where $x_{ij} = x_{jk} = 1$ until returning to i. Each complete cycle is checked for depot inclusion—cycles without the depot are illegal subtours that violate the TSP requirement.

This approach requires $O(n)$ time as each node is visited once. Validation on small instances confirms the method correctly identifies all subtours when subtour elimination constraints are absent.

1.2.2 (b) Extending to only eliminate emerging subtours.(`q_1_2_b.py`)

We implement lazy constraint generation: solve the model with flow constraints, detect subtours using our method from (a), add SEC for violated subtours, and repeat until convergence. This exploits that only $O(n)$ constraints typically bind despite $O(2^n)$ possibilities, maintaining strong LP bounds while avoiding exponential constraint growth. Detailed performance analysis is in Section 1.2.4(d).

1.2.3 (c) Plot of the objective value per iteration. (`q_1_2_b.py`)

Figure 1 shows the objective value of the MTZ formulation during the branch-and-bound search for the largest instance we were able to solve (25 customers).

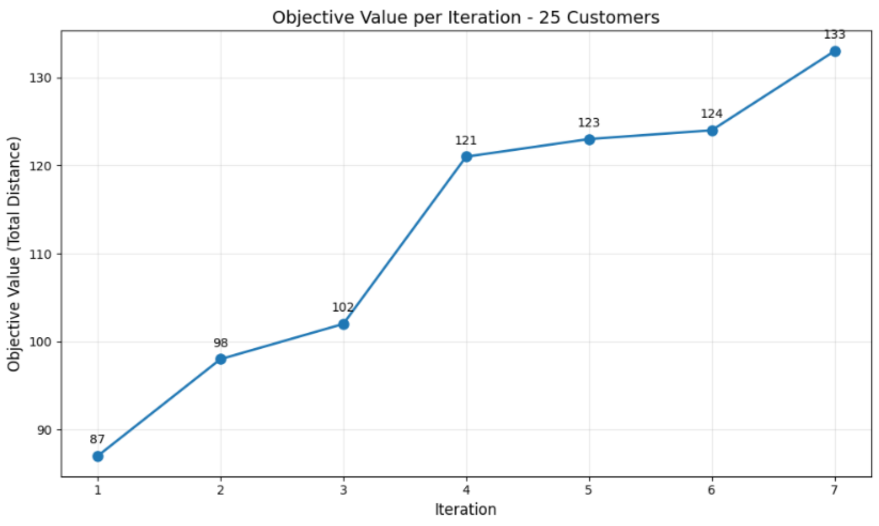

Figure 1: Objective value per iteration for the 25-customer instance.

For the 25-customer instance, the objective value increases at every iteration. This is expected: in local search–based heuristics, each iteration applies a move that improves feasibility or reduces constraint violations, but not necessarily the total distance. As a result, the algorithm may move away from short routes in order to eliminate subtours or satisfy connectivity. Once the solution becomes feasible, the objective stabilizes but remains higher than the initial values.

1.2.4 (d) Runtime Comparison

Table 2 compares the three formulations across instance sizes. The improved DFJ consistently outperforms both alternatives for $n \geq 15$.

Table 2: Runtime comparison (seconds) across formulations

n	MTZ	DFJ Original	DFJ Improved
5	0.01	0.01	0.01
10	0.01	0.03	0.01
15	0.23	2.44	0.02
20	45.32	272.37	0.05
25	88.68	TL (>600)	0.08

Why the differences? MTZ has $O(n^2)$ constraints but weak LP relaxation, causing extensive branching. Original DFJ has strong bounds but $O(2^n)$ constraints become prohibitive beyond $n = 20$. Improved DFJ combines strong bounds with lazy generation, adding only $O(n)$ violated constraints in practice.

Instance size dependency: The speedup grows exponentially with n. For small instances ($n \leq 10$), all methods solve quickly. At $n = 15$, improved DFJ is $10\times$ faster than MTZ and $100\times$ faster than original DFJ. By $n = 25$, the gaps widen to $10^3\times$ and $10^4\times$ respectively, demonstrating superior scalability.

2 Extending the Model: Capacity and Time Windows

Lastly, we integrate capacity and time constraints into the models

2.1 Practical Constraints

(a) Extended MTZ Formulation with Capacity and Time Windows

Sets and Parameters.

- $N = \{0, 1, \dots, n\}$: set of nodes, where 0 is the depot.
- c_{ij} : travel cost (or distance) from node i to node j.
- t_{ij} : travel time from node i to node j.
- q_i : demand of customer i (with $q_0 = 0$).
- Q : vehicle capacity.
- a_i, b_i : earliest and latest arrival times (time window of customer i).
- s_i : service time at node i.
- M^T : sufficiently large constant for time-window constraints.

Decision Variables.

- $x_{ij} \in \{0, 1\}$: 1 if a vehicle travels directly from i to j.
- u_i : amount of goods in vehicle after visiting customer i.
- T_i : arrival time at node i.

Model.

$$\min \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{13}$$

$$\sum_{j \in N} x_{ij} = 1, \qquad \forall i \in N \setminus \{0\} \quad \text{(one outgoing arc per customer)} \tag{14}$$

$$\sum_{i \in N} x_{ij} = 1, \qquad \forall j \in N \setminus \{0\} \quad \text{(one incoming arc per customer)} \tag{15}$$

$$u_i - u_j + Q x_{ij} \le Q - q_j, \qquad \forall i, j \in N \setminus \{0\},\ i \ne j \quad \text{(capacity-based subtour elimination, MTZ)} \tag{16}$$

$$q_i \le u_i \le Q, \qquad \forall i \in N \setminus \{0\} \quad \text{(load bounds)} \tag{17}$$

$$u_0 = 0 \quad \text{(vehicle starts empty at depot)} \tag{18}$$

$$T_i + s_i + t_{ij} - T_j \le M^T (1 - x_{ij}), \qquad \forall i, j \in N,\ i \ne j \quad \text{(time propagation, MTZ-style)} \tag{19}$$

$$a_i \le T_i \le b_i, \qquad \forall i \in N \quad \text{(time windows)} \tag{20}$$

$$T_0 = 0 \quad \text{(start time at depot)} \tag{21}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in N. \tag{22}$$

2.2 Improving Customer Service

2.2.1 (a) Why the TSP formulations provides a lower bound on the optimal solution for every time window width.

The TSP formulations developed previously (DFJ or MTZ) only enforce that each customer is visited exactly once and that the route forms a single Hamiltonian cycle. They do not include any time-window restrictions.

Because of this, the TSP is allowed to choose the *shortest possible tour* .

In contrast, the Vehicle Routing Problem with Time Windows (VRPTW) requires that each customer is visited within its specified time window, which makes the feasible region strictly smaller. Many of the shortest TSP tours will violate time windows and are therefore infeasible in the VRPTW. Since the optimal VRPTW solution must satisfy the additional time constraints, its objective value can only be *greater than or equal to* the objective value of the unconstrained TSP. Thus, the TSP solution provides a valid lower bound on the VRPTW optimal objective value for *any* time-window width.

2.2(b) Trade-off Between Driven Distance and Time Window Width

In this analysis, customer time windows were systematically scaled using multiplier λ while keeping their geometric centers fixed. For each customer i with original window $[a_i, b_i]$:

$$a_i' = c_i - \frac{(b_i - a_i)\lambda}{2}, \quad b_i' = c_i + \frac{(b_i - a_i)\lambda}{2}, \quad c_i = \frac{a_i + b_i}{2}$$

This isolates the effect of window width without shifting delivery moments. Vehicle capacity constraints were removed, and MTZ subtour elimination was applied.

The model was solved for five different values of λ. A summary of the results is provided in Table 3.

Window Multiplier λ	**Avg. Width**	**Vehicles**	**Distance (km)**
0.25	≈ 15	6	299
0.50	≈ 30	5	270
1.00 (baseline)	≈ 60	3	192
2.00	≈ 120	3	192
7.00	≈ 420	3	189

Table 3: Effect of time window width on fleet size and driven distance.

The impact is highly asymmetric. Narrower windows ($\lambda < 1$) severely degrade performance: at $\lambda = 0.25$, six vehicles and 299 km are required as tight constraints prevent grouping spatially close customers. The baseline ($\lambda = 1.0$) achieves 3 vehicles and 192 km. Widening to $\lambda = 2.0$ yields no improvement, indicating that geographical structure dominates once sufficient scheduling flexibility exists. Very wide windows ($\lambda = 7.0$) produce marginal gain (189 km) as the problem approaches pure TSP, but further increases offer diminishing returns. Overall, tightening windows quickly worsens solutions, while relaxing them beyond baseline provides limited benefit due to inherent spatial constraints.

2.2(c) Time-Expanded Network Formulation

In a time-expanded network, each physical node is replicated over discrete time periods. Arcs (i, j, t) represent departing from node i and arriving at node j at time t, so temporal feasibility (travel times, service durations, and time windows) is built directly into the network structure, eliminating the need for continuous time variables and Big-M constraints.

Sets and Parameters.

- $N = \{0, 1, \ldots, n\}$: nodes (0 = depot)
- $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, T_{\max}\}$: discrete time grid
- c_{ij}: distance from i to j
- τ_{ij}: travel time from i to j
- s_i: service duration at node i (with $s_0 = 0$)
- $[a_i, b_i]$: time window for customer i

Decision Variables.

- $x_{ijt} \in \{0, 1\}$: 1 if a vehicle travels from i to j, arriving at time t
- $y_{it} \in \{0, 1\}$: 1 if service at customer i starts at time t

Feasible Arc Set.

$$\mathcal{A} = \{(i,j,t) : i \neq j,\ t \in \mathcal{T},\ t \in [a_j, b_j],\ t = \tau_{0j} \text{ if } i = 0,\ t \leq T_{\max} \text{ if } j = 0\}.$$

Arcs exist only when arrival times satisfy travel times and time windows.

Objective.

$$\min \sum_{(i,j,t)\in\mathcal{A}} c_{ij} x_{ijt} \tag{Obj}$$

Constraints.

$$\sum_{t\in\mathcal{T}} y_{it} = 1 \qquad \forall i \in \{1,\ldots,n\} \quad \text{(each customer is served exactly once)} \tag{C1}$$

$$y_{it} = 0 \qquad \forall t \notin [a_i, b_i] \quad \text{(Service may only start within time window)} \tag{C2}$$

$$\sum_{j\in N} x_{jit} = y_{it} \qquad \forall i \in \{1,\ldots,n\},\ t \in \mathcal{T} \quad \textit{(Arrival determines service start)} \tag{C3}$$

$$\sum_{k\in N:\, t+s_i+\tau_{ik}\in\mathcal{T}} x_{ik,\, t+s_i+\tau_{ik}} = y_{it} \qquad \forall i \in \{1,\ldots,n\},\ t \in \mathcal{T} \quad \textit{(Departure occurs after service and travel time)} \tag{C4}$$

$$\sum_{j=1}^{n} \sum_{t\in\mathcal{T}} x_{0jt} = \sum_{i=1}^{n} \sum_{t\in\mathcal{T}} x_{i0t} \quad \textit{(Depot departure and return balance)} \tag{C5}$$

This formulation enforces time feasibility structurally by allowing only arcs that respect travel times and time windows. Unlike the MTZ formulation, it does not require Big-M constraints or continuous service time variables. The trade-off is model size: with $|\mathcal{T}|$ time layers, the number of binary variables grows to $O(n^2|\mathcal{T}|)$, but the linear relaxation becomes significantly stronger and the routing structure becomes fully time-consistent by construction.

2.2(d) Time-Discretization Requirements for Exactness

The time-expanded network formulation relies on discretizing continuous time into a finite grid $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, T_{\max}\}$. The accuracy of this formulation therefore depends on the choice of the discretization step Δt.

To obtain an *exact* solution that is fully equivalent to the continuous-time VRPTW, the discretization must be fine enough so that all relevant event times can be represented on the grid. In particular:

- service start times,
- travel times τ_{ij},
- service durations s_i,
- and time window boundaries a_i, b_i

must all align with at least one time point in $\mathcal{T}$.

If Δt is too large, some feasible service start times cannot be represented and the model may incorrectly exclude feasible routes. In the extreme case, the optimal continuous-time solution may lie strictly between two discretization points, causing the time-expanded formulation to become only an approximation.

A discretization is therefore exact when:

$$\Delta t \text{ divides all relevant times, i.e., } a_i,\ b_i,\ s_i,\ \tau_{ij} \in \mathcal{T} \quad \forall i, j.$$

In practice, this means choosing Δt to be at most the greatest common divisor of all event times in the instance. A very fine grid (e.g., $\Delta t = 1$) always ensures exactness, but increases the size of the time-expanded network to $O(n^2\, T_{\max}/\Delta t)$, making the model computationally expensive.

Thus, the time discretization must be sufficiently fine to represent all feasible arrival and service times, but there is a trade-off: smaller Δt improves accuracy and guarantees exactness, while larger Δt reduces the model size but provides only an approximation of the continuous-time problem.

2.2(e) Which formulation performs better?

Table 4 compares the MTZ and time-expanded formulations across different time window widths.

Formulation	λ	Δt	**Distance (km)**	**Status**
MTZ	0.5	-	270	Optimal (25/25)
MTZ	1.0	-	192	Optimal (25/25)
MTZ	7.0	-	189	Optimal (25/25)
Time-Expanded	0.5	50	-	Infeasible (14/25)
Time-Expanded	1.0	50	348	Optimal (25/25)
Time-Expanded	2.0	50	261	Optimal (25/25)
Time-Expanded	0.5	1	-	Infeasible (17/25)
Time-Expanded	1.0	1	216	Optimal (25/25)
Time-Expanded	2.0	1	216	Optimal (25/25)

Table 4: MTZ vs time-expanded performance across window widths.

MTZ clearly outperforms. At baseline ($\lambda = 1.0$), MTZ achieves 192 km while time-expanded requires 216 km with fine discretization ($\Delta t = 1$) or 348 km with coarse discretization ($\Delta t = 50$)—representing 12.5% and 81% quality loss respectively. More critically, MTZ remains feasible across all tested window widths while time-expanded becomes infeasible at $\lambda = 0.5$.

Root cause: The fundamental difference is time representation. MTZ uses continuous variables T_i allowing precise optimization of arrival times within windows. Time-expanded restricts arrivals to discrete grid points $\{0, \Delta t, 2\Delta t, \ldots\}$, creating two problems:

First, narrow windows can fall outside the discretized time horizon, making the problem structurally infeasible regardless of route optimization. This occurs at $\lambda = 0.5$ for both discretization levels.

Second, even when feasible, forcing continuous optimal arrival times onto discrete grid points breaks timing sequences and degrades solution quality. Finer discretization reduces but cannot eliminate this gap.

Conclusion: MTZ is superior—better solutions, guaranteed feasibility across window widths, and compact model size. Time-expanded formulations suit problems with naturally discrete time periods, which does not apply here.

2.3 Driver Constraints

2.3.1 (a) Mathematical formulation of the duration constraint.

To ensure that no driver's total working time exceeds 7 hours (420 minutes), we extend the MTZ formulation by explicitly tracking the return time to the depot for each possible route endpoint.

Additional Parameters.

- $H = 420$: maximum allowable working time (minutes)
- M: a sufficiently large constant (e.g. $M = T_{\max} = 1236$)

Additional Decision Variables.

- $R_i \geq 0$: return time to the depot if the route ends at customer i, for all $i \in \{1, \ldots, n\}$

Working Time Definition. If a vehicle finishes its route at customer i, the total working time equals

$$\text{Working time} = T_i + s_i + \tau_{i0},$$

where T_i is the arrival time at i, s_i is its service duration, and τ_{i0} is the travel time back to the depot.

Mathematical Formulation. **(D1a) Lower bound on return time:**

$$R_i \geq T_i + s_i + \tau_{i0} - M(1 - x_{i0}) \qquad \forall i \in \{1, \ldots, n\} \tag{D1a}$$

(D1b) Upper bound on return time:

$$R_i \leq T_i + s_i + \tau_{i0} + M(1 - x_{i0}) \qquad \forall i \in \{1, \ldots, n\} \tag{D1b}$$

(D2) Working time limit:

$$R_i \leq H \qquad \forall i \in \{1, \ldots, n\} \tag{D2}$$

Explanation. Constraints (D1a)–(D1b) force R_i to match the true return time when customer i is the final stop of a route:

- If $x_{i0} = 1$ (route ends at i):
$$T_i + s_i + \tau_{i0} \leq R_i \leq T_i + s_i + \tau_{i0},$$
hence $R_i = T_i + s_i + \tau_{i0}$.
- If $x_{i0} = 0$ (route does not end at i): the $\pm M$ terms relax both inequalities, and R_i becomes irrelevant.

Constraint (D2) guarantees that the working duration of any route does not exceed 420 minutes.

Integration with MTZ. The formulation extends the MTZ model from Section 2.1(a) by:

- reusing MTZ arrival-time variables T_i and routing variables x_{ij},
- adding n continuous variables R_i,
- adding $3n$ new constraints: (D1a), (D1b), and (D2),
- requiring no modification to the existing MTZ constraints.

2.3.2 (b) How this affects the solutions.

Adding the shift duration constraint has several implications that ultimately worsen the solutions. This constraint negatively impacts the flexibility of the route. It breaks long routes that exceed this 7-hour time limit into smaller ones, which leads to a bigger fleet size being required to visit all the nodes within the allowed working time.

As a result, solutions become more fragmented: instead of a few long and efficient subtours, the model tends to generate many shorter routes that visit nodes that are close to each other. Moreover, nodes that are located farther away from the depot, and which were part of a big subtour, may now require nearly dedicated routes with very few additional nodes included. The time-window constraint further amplifies these effects, breaking apart subtours that were previously feasible, because the waiting time now counts towards each driver's total working hours. Although this constraint presents a more realistic scenario, the factors previously mentioned overall have a negative impact on the solutions found previously, as they lead to an increase in the total distance traveled.