Eindhoven University of Technology

1CM110 – Decision Making in Transport and Mobility

Assignment 3

Yusuf Sami (Student ID: 2399296)

Part 1.1 (a) – Route-based CVRP

We propose using a route-based (set-partitioning) formulation to minimize the total traveled distance, visit each customer exactly once with the five vehicles available.

The following results were obtained:

- **Runtime:** 0.56 seconds
- **Optimality gap:** 0.0000
- **Total cost (objective value):** 8831
- **Selected route costs:** $[228, 1697, 2100, 2343, 2463]$
- **Range (max − min):** $2463 - 228 = 2235$

This solution minimizes the total traveled distance. However, the workload among vehicles is highly unbalanced, since one vehicle is assigned a very short route (228) while another is assigned a very long route (2463), resulting in a large range.

Part 1.1 (b) – Fairness-oriented model

We now apply the vehicle-index formulation with a cost budget of

$$(1 + 0.05)z^* = 9272.55,$$

where $z^* = 8831$ is the optimal cost obtained in Part (a). The results are:

- **Runtime:** 511.6 seconds
- **Optimality gap:** 0.0000
- **Total cost:** 9271
- **Selected route costs:** $[1000, 1529, 1855, 2424, 2463]$
- **Range (max − min):** $2463 - 1000 = 1463$

Comparison between (a) and (b) Table 1 summarizes the results of both models.

Metric	(a) Cost minimization	(b) Fairness model
Total cost	8831	9271
Range	2235	1463
Runtime (s)	0.56	511.6
Optimality gap	0.0%	0.0%

Table 1: Comparison between cost-minimizing and fairness-oriented solutions

The fairness-oriented model increases the total cost by about 5% (from 8831 to 9271), which is within the allowed budget. In return, the range decreases from 2235 to 1463, meaning that the workloads of the vehicles are much more balanced. This demonstrates the trade-off between efficiency and fairness.

Part 1.1 (c)

For each route r we define $b_{ir} = 1$ if customer i is the last customer on route r, and 0 otherwise. Since every selected route has exactly one last customer, for each selected route ($x_r = 1$) there is exactly one customer i such that $b_{ir} = 1$.

Constraint (1e)

$$\sum_{r \in R} p_r \, b_{ir} \, x_r \leq \eta \qquad \forall i$$

ensures that the length of every selected route is at most η, because for a last customer i the left-hand side equals the length of the corresponding route. Therefore, η becomes the maximum route length.

Constraint (1f)

$$M\Big(1 - \sum_{r \in R} b_{ir} x_r\Big) + \sum_{r \in R} p_r b_{ir} x_r \geq \gamma \qquad \forall i$$

forces the length of every selected route to be at least γ, while it is inactive for customers that are not last customers due to the big-M term. Hence, γ becomes the minimum route length.

Since the objective minimizes $\eta - \gamma$, the model correctly minimizes the range of the selected route lengths.

Part 1.1 (d)

From a computational point of view, the last-customer formulation is preferred over the vehicle-index formulation for two main reasons.

First, it uses far fewer binary variables. The vehicle-index formulation requires one variable x_{rk} for every combination of route and vehicle, resulting in $|R| \times |K|$ binary variables. In contrast, the last-customer formulation uses only one binary variable x_r per route, which leads to a much smaller and more tractable mixed-integer program.

Second, the last-customer formulation avoids symmetry between vehicles. In the vehicle-index model, all vehicles are identical, so permuting vehicle indices leads to equivalent solutions. This creates many symmetric solutions and makes the branch-and-bound search much harder. The last-customer formulation eliminates this symmetry because routes are selected directly, without assigning them to vehicle labels, which significantly improves computational performance.

Part 1.1(e) – Last-Customer Formulation

The last-customer formulation was solved for different values of ε. For each run, we report the total cost, the achieved range (max–min route length), the solver gap, and the runtime. Table 2 summarizes the results of the last-customer formulation for different values of ε.

Table 2: Results of the last-customer formulation for different values of ε.

ε	Budget	Total Cost	Range	Runtime (s)
0.01	8919.31	8889	2194	7.15
0.025	9051.77	8883	2194	14.37
0.05	9272.55	9271	1463	22.17
0.075	9493.32	9487	1167	31.41
0.10	9714.10	9710	933	26.82
0.125	9934.88	9905	855	44.86
0.15	10155.65	10098	731	27.65

As ε increases, the budget becomes less restrictive and the range decreases, meaning that the workloads of the vehicles become more balanced. This improvement in fairness is obtained at the cost of a higher total travel distance. For small values of ε, the range does not improve because no more balanced solution is feasible within the tight budget.

Part 1.1 (f)

When routes must be planned for each day of the week, fairness is no longer defined at the daily vehicle level but at the level of drivers over the entire week. In dynamic scenarios, it's likely that route lengths vary greatly as customers change from day to day, so the daily fairness is harder to ensure. The objective now becomes to balance the total weekly distance driven by each driver.

For this purpose, a vehicle-index type formulation is the most appropriate, because it explicitly models which driver is assigned to which route. This is essential, as it allows the weekly workload of each driver to be tracked and compared. The last-customer formulation, on the other hand, does not explicitly model driver-route assignments, as it uses the route's last customer as an identifier. This makes the model unsuitable for tracking and balancing workloads across multiple days.

In practice, the way to solve the problem depends on the planning horizon and the information available for the week at the time of optimization. If all information for the week is available, the problem can be solved in two stages. First, a standard CVRP is solved for each day to generate a set of feasible routes and their distances. Second, these daily routes are assigned to drivers using an assignment model with decision variables indicating whether a given route on a given day is assigned to a specific driver. The total weekly distance of each driver is then constrained between two variables representing the maximum and minimum weekly workload, and the range between them is minimized.

However, when the planning horizon is shorter than a week, the previous model must be solved in a rolling-horizon fashion: it should also take into account the cumulative distance already driven by each driver in order to compensate for unfairness in the previous days.

Both these approaches ensure that routes are efficient on each day while also being distributed fairly among drivers over the entire week.

Part 1.2(a) – Classical two-index CVRP

The classical two-index CVRP formulation with capacity constraints was implemented and solved using Gurobi with lazy constraints for subtour and capacity elimination. The number of vehicles was fixed to 5.

The obtained cost-efficient solution is summarized in Table 3.

Computing time (s)	Optimality gap (%)	Total cost
213.46	0.00	8742

Table 3: Results for Part 1.2(a): classical two-index CVRP formulation

1 Part 1.2(b)

We simulated $k = 1000$ demand realisations, where for each customer i the true demand $\tilde{q}_i$ was drawn uniformly from $[0.9q_i,\, 1.1q_i]$ (rounded to integers). For each realisation, we computed the total capacity violation as the sum over all routes of $\max\{0, \sum_{i \in r} \tilde{q}_i - Q\}$.

The results are:

- Number of scenarios with capacity violation: 287 out of 1000
- Average total capacity violation: 2.01
- Maximum total capacity violation: 25

Part 1.2(c) – Scenario-based model with cutting planes

We implemented the scenario-based two-index CVRP formulation and ran the cutting-plane procedure for five iterations. For the simulation step, we used $k = 1000$ demand realisations (seed = 42) and solved each iteration with a time limit of 600 seconds. Table 4 summarizes the results.

Table 4: Cutting-plane iterations for the scenario-based CVRP model. "Viol. scen." indicates the number of scenarios with capacity violations. The last column reports the average and maximum total violation across 1000 simulations.

Iter.	$\|S\|$	Cost	Gap (%)	Time (s)	Viol. scen. / 1000	Avg./Max viol.
1	1	8742	15.63	600.03	287	2.01 / 25
2	2	8838	16.04	600.06	212	1.33 / 28
3	3	9245	21.20	600.06	126	0.89 / 23
4	4	9291	22.43	600.07	4	0.01 / 6
5	**5**	**9447**	**23.99**	**600.03**	**0**	**0.00 / 0**

The results demonstrate a clear trade-off between cost and reliability. Adding the most violating demand realisations to S improves robustness substantially: the number of violating scenarios decreases from 287 (28.7%) to 0 (0%). This full immunity against simulated uncertainty comes at the expense of a higher routing cost ($8742 \rightarrow 9447$), corresponding to a **Price of Robustness of approximately 8.1%**. The optimality gap increases in later iterations because the scenario-based model size grows with $|S|$, making it harder to prove optimality within the 600-second time limit.

Part 1.2(d)

We use a simple recourse policy that repairs a route only when infeasibility is revealed.

Recourse policy. While following the planned customer order, when arriving at the next customer and the remaining vehicle capacity is insufficient to serve their realised demand, the vehicle returns to the depot to replenish (reset remaining capacity to Q), then returns to the same unserved customer and the planned route continues from there. This guarantees that (i) all realised demand is served, (ii) customers are visited in the originally planned order, and (iii) decisions use only information revealed up to the current customer.

Only if the extent of uncertainty is known could a policy be adopted which looks a head if the next customer could be served with the current capacity, even in the worst case scenario.

Illustration on a violating scenario. Consider the planned route

$$r: \ 0 \rightarrow 20 \rightarrow 16 \rightarrow 12 \rightarrow 19 \rightarrow 18 \rightarrow 15 \rightarrow 0,$$

with vehicle capacity $Q = 478$. Under one feasible demand realisation (each $\tilde{q}_i \in [0.9q_i, 1.1q_i]$),

$$(\tilde{q}_{20}, \tilde{q}_{16}, \tilde{q}_{12}, \tilde{q}_{19}, \tilde{q}_{18}, \tilde{q}_{15}) = (78, 94, 108, 82, 92, 73),$$

the total demand on the route equals $527 > 478$, hence a capacity violation occurs without recourse.

Applying the policy yields the sequence reported in Table 5 (remaining capacity after service is shown).

Table 5: Application of the recourse policy to a capacity-violating route.

Step	Customer	$\tilde{q}_i$	Remaining capacity	Action
Start	Depot	–	478	start route
1	20	78	400	serve
2	16	94	306	serve
3	12	108	198	serve
4	19	82	116	serve
5	18	92	24	serve
6	15	73	$24 < 73$	return to depot, replenish
7	15	73	405	serve after replenishment
End	Depot	–	–	return to depot

Therefore, the realised (repaired) route becomes

$$0 \to 20 \to 16 \to 12 \to 19 \to 18 \to 15 \to 0 \to 15 \to 0,$$

which serves all realised demand and preserves the original customer order.

Part 1.2(e)

For each of the five solutions obtained in Part 1.2(c), we simulated the recourse policy described in Part 1.2(d) for $k = 1000$ demand realisations (seed = 42). For every realisation, we computed (i) the total routing cost of the planned routes (before recourse), and (ii) the total routing cost after applying the recourse actions when capacity violations occurred.

Table 6 reports the average values over the 1000 simulations.

Table 6: Average routing costs before and after applying the recourse policy for the five cutting-plane solutions.

Iteration	Cost from (c)	Avg. cost before	Avg. cost after	Avg. extra cost
1	8742	8742.00	9179.94	437.94
2	8838	8838.00	9124.63	286.63
3	9245	9245.00	9460.46	215.46
4	9291	9291.00	9297.98	6.98
5	**9447**	**9447.00**	**9447.00**	**0.00**

The results clearly illustrate the trade-off between cost efficiency and robustness. The first solution (Iteration 1) is the cheapest in nominal terms, but it is highly sensitive to demand uncertainty: on average, recourse actions increase the routing cost by about 438 units. As more scenarios are added to the scenario set in the cutting-plane algorithm, the planned routes become increasingly robust, which is reflected in the rapidly decreasing average extra cost.

By Iteration 5, the solution is fully robust with respect to the simulated demand uncertainty: no recourse actions are needed in any of the 1000 scenarios, and the average cost after recourse equals the planned cost. Thus, while the robust solution has a higher nominal cost, it eliminates recourse costs and guarantees feasibility under all simulated demand realisations.

However, it should also be considered that a comparison between average total cost before and after recourse is an unfair one, as the first does not take into account that several routes contained in the average would be infeasible and thus incur some kind of penalty in reality.

Part 2.1(a)

Let $G_0 = (N, E)$ denote the underlying road network, where each edge $(u, v) \in E$ has a non-negative travel distance d_{uv}. Given a driving range R, we construct an auxiliary *range-network* (or implicit network) $G = (N, A)$ by including a directed arc (i, j) in A if and only if an electric vehicle can travel from i to j without recharging. Formally:

$$(i, j) \in A \quad \Longleftrightarrow \quad \mathrm{dist}_{G_0}(i, j) \leq R,$$

where $\mathrm{dist}_{G_0}(i, j)$ denotes the shortest-path distance between nodes i and j in the road network G_0.

From a computational perspective, A can be generated efficiently via a preprocessing step. Since road networks are typically sparse, we can run a *radius-bounded* Dijkstra shortest-path search starting from each node $i \in N$. The search from node i can be terminated once the tentative distances to all remaining unvisited nodes exceed R. We then add arcs (i, j) for all nodes j reached within distance R. This construction ensures that any arc in A represents a feasible non-stop trip with respect to the range limit, significantly reducing the complexity of the subsequent optimization model.

Part 2.1(b)

We extend the formulation from the lecture to account for a finite capacity $Q = 10$ at each charging facility. As in the lecture, the objective is to minimize the total number of installed charging units.

Since OD volumes can exceed the capacity of a single facility ($Q = 10$), we reinterpret the decision variable y_i as the *number of charging units* installed at node i, i.e., $y_i \in \mathbb{Z}_{\geq 0}$. The original coupling constraint ($x_{ij}^k \leq y_j$), which allowed flow to pass through node j only if a facility was installed there, is replaced by a capacity constraint that limits the total recharging flow handled at each node.

$$\min \quad \sum_{i \in N} y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = d_i^k \qquad \forall i \in N,\ \forall k \in K \tag{2}$$

$$\sum_{k \in K:\ D_k \neq j} \ \sum_{i:(i,j) \in A} x_{ij}^k \leq Q\, y_j \qquad \forall j \in N \tag{3}$$

$$x_{ij}^k \geq 0 \qquad \forall (i, j) \in A,\ \forall k \in K \tag{4}$$

$$y_i \in \mathbb{Z}_{\geq 0} \qquad \forall i \in N. \tag{5}$$

Constraint (3) enforces that the total passenger flow that needs to recharge at node j (i.e., all flow arriving at j that does not terminate there) does not exceed the installed capacity $Q\, y_j$.

Part 2.1(c)

We solved the capacitated charging facility location model with facility capacity $Q = 10$ for each company. Table 7 reports the computing time, the optimal number of charging facilities (total installed units), and the optimal charging locations (nodes with $y_i > 0$).

Table 7: Results for the capacitated charging facility location problem ($Q = 10$).

Company	Time (s)	# Facilities (units)	Optimal charging locations ($y_i > 0$)
A	1.8210	8	$\{1, 2, 5, 12, 24, 31, 33, 39\}$
B	0.8060	8	$\{5, 10, 11, 12, 17, 33, 37, 39\}$
C	2.9490	9	$\{1, 5, 10, 12, 16, 17, 33, 37, 39\}$

Part 2.1(d)

We solve the capacitated charging facility location problem for every coalition $S \subseteq \{A, B, C\}$ (with facility capacity $Q = 10$) and denote the optimal total number of installed charging units by $c(S)$. The coalition costs are reported in Table 8.

Table 8: Optimal coalition costs $c(S)$ (total installed charging units) for all subsets of companies.

Coalition S	Cost $c(S)$
$\emptyset$	0
$\{A\}$	8
$\{B\}$	8
$\{C\}$	9
$\{A, B\}$	10
$\{A, C\}$	9
$\{B, C\}$	10
$\{A, B, C\}$	11

Using these values, the Shapley cost allocation ϕ_i for each company $i \in \{A, B, C\}$ is given in Table 9. The allocation satisfies $\phi_A + \phi_B + \phi_C = c(\{A, B, C\}) = 11$.

Table 9: Shapley value allocation and savings compared to building individually.

Company	Standalone cost $c(\{i\})$	Shapley share ϕ_i	Savings $c(\{i\}) - \phi_i$
A	8	3.33	4.67
B	8	3.83	4.17
C	9	3.83	5.17

Hence, all companies benefit from collaboration: relative to building their own facilities, Company A saves approximately 4.67 units, Company B saves 4.17 units, and Company C saves 5.17 units.

Part 2.2(a) Selfish routing versus system-optimal routing

Using the charging station locations obtained in Part 2.1(d), we compare the system-optimal routing with selfish routing behavior of travelers. The installed charging infrastructure consists of 11 stations located at nodes $\{1, 2, 3, 4, 5, 7, 12, 15, 22, 31, 33\}$, each with capacity $Q = 10$.

In the *system-optimal* solution, routes are chosen to minimize the total travel time in the network while respecting the charging capacity constraints. The resulting total travel time is

$$T^{\text{SO}} = 25{,}855.46.$$

Under *selfish routing*, each traveler independently chooses the shortest charge-feasible path from its origin to its destination, ignoring the congestion effects on charging stations. The total travel time under selfish routing equals

$$T^{\text{SR}} = 28{,}942.56.$$

Hence, selfish routing increases the total travel time by

$$T^{\text{SR}} - T^{\text{SO}} = 3{,}087.10,$$

demonstrating a clear efficiency loss due to uncoordinated user behavior.

Effect on charging capacities. Selfish routing leads to uneven usage of charging stations, causing capacity violations at several locations. In particular, the following stations exceed their available capacity:

- Node 7: violation of 1.08
- Node 12: violation of 2.17
- Node 15: violation of 2.18
- Node 31: violation of 1.26

All other charging stations operate within their capacity limits. These results show that selfish routing not only increases total travel time but also induces local congestion at popular charging locations, potentially leading to infeasible or unreliable system operation if capacities are not strictly enforced.

Part 2.2(b)

To account for selfish routing behavior when determining cost-optimal charging locations, the companies could use an iterative cutting-plane approach that incorporates user equilibriums into the charging facility location model.

First, the charging facility location problem is solved assuming centrally coordinated routes, generating an initial set of locations for charging stations. With these locations, selfish routing is simulated by assigning each start-end pair to its shortest feasible (charge wise) path. The flow through each charging station is checked against its capacity.

If violations occur in any station, additional constraints are added to the model. These constraints could be, for example, forcing the opening of new stations in the area, or limit the total flow that can be routed through these overloaded locations. The model is then re-optimized with these new constraints.

This procedure is repeated until there are no violations in any stations and it respects every constraint. In this way, the facility locations are optimized while anticipating selfish behavior.