Combinatorial Decision Making and Optimization (CDMO) Project Report - Multiple Couriers Problem

Ali Loloee – ali.loloeejahromi@studio.unibo.it
Fatemeh Izadinejad – fatemeh.izadinejad@studio.unibo.it
HamidReza Bahmanyar – hamidreza.bahmanyar@studio.unibo.it

June 2025

1 Introduction

This report addresses the Multiple Couriers Planning (MCP) problem, focusing on three modeling approaches: Constraint Programming (CP), Satisfiability (SAT), and Mixed-Integer Programming (MIP). All models are built upon a shared set of core parameters:

- m: Number of couriers
- n: Number of items
- l: List of courier capacities
- s: List of item sizes
- D: Distance matrix representing travel distances between nodes

The project was carried out over a period of 25 days. HamidReza Bahmanyar was responsible for the development of the CP model, Ali Loloee implemented the SAT-based approach and Docker integration, and Fatemeh Izadinejad worked on the MIP formulation. AI-assisted tools were employed to support Python function documentation and improve the clarity and presentation of the written report.

2 Objective Function

$$\text{minimize } \mathbf{max\_dist} = \max_{c \in \{1,\ldots,m\}} (\mathbf{dist}_c)$$

3 CP Model

The CP model for the MCP problem uses a 2D matrix to represent courier routes, capturing both assignment and delivery order. This allows for a clear and efficient MiniZinc formulation.

3.1 Decision Variables

The model defines the following decision variables:

- $\mathbf{path}_{c,p} \in \{0, 1, \ldots, n+1\} \quad \forall c \in \{1, \ldots, m\},\ p \in \{1, \ldots, p_{\max}\}$
  Represents the location visited by courier c at position p in their route. The value 0 denotes an unused position, values from 1 to n correspond to delivery locations, and $n+1$ indicates the depot.
- $\mathbf{path\_length}_c \in \{2, \ldots, p_{\max}\} \quad \forall c \in \{1, \ldots, m\}$
  The actual length of courier c's route, including depot visits at the start and end.
- $\mathbf{dist}_c \in \mathbb{Z}^+ \quad \forall c \in \{1, \ldots, m\}$
  The total distance traveled by courier c along their assigned route.
- $\mathbf{courier\_load}_c \in \{0, \ldots, \sum_{i=1}^{n} s_i\} \quad \forall c \in \{1, \ldots, m\}$
  The total load assigned to courier c, computed as the sum of item sizes allocated to them.
- $\mathbf{max\_dist} \in \mathbb{Z}^+$
  The maximum distance traveled by any courier, which is the objective function to minimize.

3.2 Constraints

In Constraint Programming, constraints are used to define the rules that valid solutions must satisfy. We adopt a minimal and focused set of constraints to keep the model efficient and easy to interpret. The model includes the following key constraints:

3.2.1 Depot and Route Length Constraint

Each courier's route must start and end at the depot, and its total length must not exceed the allowed maximum. For all $c \in \{1, \ldots, m\}$:

- $\text{path}_{c,1} = n+1$ (start at depot)
- $\text{path}_{c,\text{path\_length}_c} = n+1$ (end at depot)
- $\text{path\_length}_c \leq p_{\max}$ (bounded route length)

3.2.2 Unused Path Position Constraint

After a courier's actual route ends, all remaining positions in their path must be set to 0. This ensures that only valid route positions contain meaningful values.

$$\mathbf{path}_{c,p} = 0 \quad \forall c \in \{1, \ldots, m\},\ \forall p \in \{\mathbf{path\_length}_c + 1, \ldots, p_{\max}\}$$

3.2.3 Unique Item Assignment Constraint

Each item must be assigned exactly once across all non-depot route positions.

$$\sum_{c=1}^{m} \sum_{p=2}^{p_{\max}-1} [\text{path}_{c,p} = i] = 1 \quad \forall i \in \{1, \ldots, n\}$$

3.2.4 Courier Load Calculation Constraint

For each courier, the total load is calculated by summing the sizes of all items assigned to their route (excluding depot positions). Formally, for all $c \in \{1, \ldots, m\}$:

$$\mathbf{courier\_load}_c = \sum_{\substack{p=2 \\ \mathbf{path}_{c,p} \neq 0}}^{\mathbf{path\_length}_c - 1} s_{\mathbf{path}_{c,p}} \quad \forall c \in \{1, \ldots, m\}$$

This load is then used to ensure that each courier does not exceed their capacity limits.

$$\mathbf{courier\_load}_c \leq l_c \quad \forall c \in \{1, \ldots, m\}$$

3.2.5 Courier Distance Calculation Constraint

The total distance traveled by each courier c is computed by summing the distances between consecutive locations along their route:

$$\mathbf{dist}_c = \sum_{p=1}^{\mathbf{path\_length}_c - 1} D_{\mathbf{path}_{c,p}, \mathbf{path}_{c,p+1}} \quad \forall c \in \{1, \ldots, m\}$$

where $D_{i,j}$ denotes the distance from location i to location j.

3.2.6 Lower Bound on Courier Distance

To tighten the search space, we impose a lower bound on each courier's travel distance based on the shortest round-trip from the depot to any item:

$$\text{LowestDist} = 2 \cdot \min_{i \in \{1, \ldots, n\}} D_{\text{depot},i}$$

$$\text{dist}_c \geq \text{LowestDist} \quad \forall c \in \{1, \ldots, m\}$$

This excludes clearly suboptimal solutions and enhances solver pruning.

3.2.7 Symmetry Breaking for Identical Couriers

To eliminate symmetric solutions, couriers with identical capacities are ordered lexicographically by their paths:

$$l_i = l_j \wedge i < j \Rightarrow [\text{path}_{i,1}, \ldots, \text{path}_{i,p_{\max}}] \preceq_{\text{lex}} [\text{path}_{j,1}, \ldots, \text{path}_{j,p_{\max}}]$$

This prevents redundant exploration of equivalent courier assignments.

3.2.8 Symmetry Breaking for Smallest Point First

To avoid symmetric solutions differing only by item order, we require the smallest item index in each courier's route (excluding the depot) to be at the second position. Formally, for all $c \in \{1, \ldots, m\}$:

$$\text{path}_{c,2} \leq \text{path}_{c,p} \quad \forall p \in \{3, \ldots, \text{length}_c - 1\}$$

This enforces a canonical item order and reduces redundant search.

3.3 Validation

Experimental Design: We evaluate our CP models via a Python script running MiniZinc with a 300-second timeout per instance. Three solver configurations with increasing constraint sophistication are tested:

- **CP SB LB HEURISTIC**: Includes symmetry breaking, lower bounds, and heuristic search; implemented with Gecode and Chuffed (latter uses a different value selection heuristic).
- **CP SB LB**: Baseline constraints plus symmetry breaking and lower bounds, without heuristics.
- **Base CP**: Only fundamental problem constraints.

Each configuration is run with both Gecode and Chuffed to assess the impact of enhancements across solvers. The results are summarized below:

Table 1: Solver Results for Different Configurations

ID	Gecode			Chuffed		
	Base	SB+LB	SB+LB+HR	Base	SB+LB	SB+LB+HR
1	**14**	**14**	**14**	**14**	**14**	**14**
2	**226**	**226**	**226**	**226**	**226**	**226**
3	**12**	**12**	**12**	**12**	**12**	**12**
4	220	**220**	**220**	**220**	**220**	**220**
5	**206**	**206**	**206**	**206**	**206**	**206**
6	**322**	**322**	**322**	**322**	**322**	**322**
7	167	167	167	167	167	167
8	186	**186**	**186**	**186**	**186**	**186**
9	436	N/A	N/A	436	436	436
10	244	N/A	N/A	244	244	244

The results indicate that most instances (1–6, 8) were solved optimally within 300 seconds. Instances 7, 9, and 10 remained suboptimal. For instances 9 and 10, only the base configurations and advanced Chuffed variants found feasible but suboptimal solutions, while advanced Gecode configurations failed to produce any solutions.

4 SAT Model

Z3 was used to model and solve the MCP problem by encoding all numerical data in binary, as it operates solely on Boolean variables. This makes it particularly suited for problems formulated in propositional logic.

4.1 Decision Variables

A set of Boolean decision variables was defined to model the MCP problem:

- **Assignment Matrix** ($Asg \in \{0,1\}^{m\times(n+1)}$): Indicates item-to-courier assignments. $Asg_{i,j} = 1$ means courier i carries item j; column $n+1$ denotes whether courier i is active (departs depot).
- **Ordering Matrix** ($Ordering \in \{0,1\}^{(n+2)\times(n+2)}$): Specifies the stepwise visit order. $Ordering_{i,j} = 1$ means item j is visited at step i. Extra rows/columns represent start and end at depot.
- **Couples Matrix** ($Couples \in \{0,1\}^{(n+1)\times(n+1)}$): Represents transitions $i \to j$ in a tour. The last row and column represent movements to and from the depot n.
- **Courier_loads** ($\in \{0,1\}^{m\times(n+1)\times load\_bits}$): Tracks cumulative load per courier over visited items.

- **Courier_distances** ($\in \{0,1\}^{m\times((n+1)^2+1)\times distance\_bits}$): Tracks cumulative travel distance of each courier.
- **Sum_asg** ($\in \{0,1\}^{(m+1)\times bits\_sum}$): Binary counter of active couriers.
- **Sum_first_row / Sum_first_col** ($\in \{0,1\}^{(n+1)\times bits\_sum}$): Binary counters for couriers starting from or returning to the depot.

4.2 Constraints

Constraints are expressed in propositional logic and grouped according to the decision variables they govern.

4.2.1 Constraints on Assignment matrix

Constraint	**Logical Expression**
Each item assigned to exactly one courier	$\bigwedge_{j=1}^{n} \text{exactly\_one}(\{asg_{i,j} \mid 1 \leq i \leq m\})$
If a courier is assigned any item, it must depart from depot	$\bigwedge_{i=1}^{m} \left(\left(\bigvee_{j=1}^{n} asg_{i,j} \right) \rightarrow asg_{i,n+1} \right)$
If *sub_tour* holds, all couriers must be active	$sub\_tour \rightarrow \bigwedge_{i=1}^{m} asg_{i,n+1}$

Table 2: Constraints related to the Assignment matrix

4.2.2 Constraints on Ordering matrix

Constraint	**Logical Expression**
Each item (and the depot) must be visited at exactly one step	$\bigwedge_{j=1}^{n+2} \text{exactly\_one}(\{ordering_{i,j} \mid 0 \leq i < n+2\})$
At each step, exactly one location must be visited	$\bigwedge_{i=0}^{n+1} \text{exactly\_one}(\{ordering_{i,j} \mid 0 \leq j < n+2\})$
The tour must start and end at the depot	$ordering_{0,n} \wedge ordering_{n+1,n+1}$

Table 3: Constraints related to the Ordering matrix

4.2.3 Constraints on Couples matrix

Constraint	Logical Expression
Each item must have exactly one outgoing transition	$\bigwedge_{i=0}^{n-1} \text{exactly\_one}(\{couples_{i,j} \mid 0 \leq j \leq n\})$
Each item must have exactly one incoming transition	$\bigwedge_{j=0}^{n-1} \text{exactly\_one}(\{couples_{i,j} \mid 0 \leq i \leq n\})$
No self-loops (i.e., no $i \to i$)	$\bigwedge_{i=0}^{n} \neg couples_{i,i}$
Transitions imply shared courier assignment	$\bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \left(couples_{i,j} \to \bigvee_{k=1}^{m} (asg_{k,i} \wedge asg_{k,j}) \right)$
Transitions respect item visit order in *ordering*	$couples_{i,j} \to \text{greater\_eq}(O_i, O_j)$, where $O_i = \{ordering_{k,i} \mid 0 \leq k < n+2\}$ $O_j = \{ordering_{k,j} \mid 0 \leq k < n+2\}$
Each courier departs and returns to depot at most once	$\bigwedge_{k=1}^{m} \big(\text{at\_most\_one}(\{couples_{n,j} \wedge asg_{k,j} \mid 0 \leq j < n\}) \wedge \text{at\_most\_one}(\{couples_{i,n} \wedge asg_{k,i} \mid 0 \leq i < n\})\big)$

Table 4: Constraints related to the Couples matrix

4.2.4 Constraints on Courier_loads matrix

Constraint	Logical Expression
Initialization: all couriers start with zero load	$\bigwedge_{k=1}^{m} (Courier\_loads_{k,0} = \mathbf{0})$
Final load must not exceed courier capacity	$\bigwedge_{k=1}^{m} (Courier\_loads_{k,n} \leq Courier\_capacity_k)$
Symmetry breaking on final loads	$\bigwedge_{k=1}^{m-1} (Courier\_loads_{k,n} \geq Courier\_loads_{k+1,n})$

Table 5: Constraints related to the Courier_loads matrix

4.2.5 Constraints on Courier_distances matrix

Constraint	Logical Expression
Initialization: all couriers start with zero distance	$\bigwedge_{k=1}^{m} (Courier\_distances_{k,0} = \mathbf{0})$

Table 6: Constraints related to the Courier_distances matrix

4.2.6 Constraints on summation matrices

Constraint	Logical Expression
Initialization: all summation counters start at 0	$Sum\_asg_0 = \mathbf{0}, \quad Sum\_first\_row_0 = \mathbf{0}, \quad Sum\_first\_col_0 = \mathbf{0}$
Incrementing active couriers in *Sum_asg*	$\bigwedge_{k=0}^{m-1} \text{binary\_increment}(Sum\_asg_k, asg_{k,n}, Sum\_asg_{k+1})$
Incrementing depot departures in *Sum_first_row*	$\bigwedge_{i=0}^{n-1} \text{binary\_increment}(Sum\_first\_row_i, couples_{n,i}, Sum\_first\_row_{i+1})$
Incrementing depot returns in *Sum_first_col*	$\bigwedge_{i=0}^{n-1} \text{binary\_increment}(Sum\_first\_col_i, couples_{i,n}, Sum\_first\_col_{i+1})$
Final consistency: all three counters must match	$Sum\_asg_{m+1} = Sum\_first\_row_{n+1} = Sum\_first\_col_{n+1}$

Table 7: Constraints related to summation structures

4.3 Validation

Once the SAT model is constructed with all variables and constraints, the optimization objective cannot be solved directly by a pure SAT solver. Instead, two search strategies were implemented on top of the SAT encoding to iteratively guide the solver toward better solutions.

- **Linear Search:** Starts from a high upper bound on the maximum courier distance. If a feasible solution exists, the bound is reduced iteratively until unsatisfiable. The last feasible solution is returned.
- **Binary Search:** Maintains upper and lower bounds on the objective. At each step, the midpoint is tested. If satisfiable, the upper bound is reduced; otherwise, the lower bound is increased, continuing until convergence.

The table compares linear and binary search strategies, with and without symmetry breaking. All reported results are optimal except Instance 7, which exceeded the 300s timeout (extra time shown in parentheses). Solutions for unlisted instances are unknown.

ID	Linear + SB	Linear w/out SB	Binary + SB	Binary w/out SB
1	**14**	**14**	**14**	**14**
2	**226**	**226**	**226**	**226**
3	**12**	**12**	**12**	**12**
4	**220**	**220**	**220**	**220**
5	**206**	**206**	**206**	**206**
6	**322**	**322**	**322**	**322**
7	194 **(+27)**	175 **(+8)**	200 **(+23)**	179 **(+17)**
8	**186**	**186**	**186**	**186**
9	**436**	**436**	**436**	**436**
10	**244**	**244**	**244**	**244**

Table 8: Results obtained by the SAT model using Linear and Binary search.

5 MIP Model

5.1 Decision Variables

- $x_{i,j}^k \in 0, 1$: Binary variable; 1 if courier k travels directly from location i to j, 0 otherwise.
- $u_i^k \in \mathbb{Z}$: Integer variable used for MTZ subtour elimination for courier k at node i.
- weight$k \in \mathbb{Z}$: Integer variable; total weight carried by courier k.
- dist$k \in \mathbb{R} \geq 0$: Continuous variable; total distance traveled by courier k.
- $D\text{max} \in \mathbb{R}_{\geq 0}$: Continuous variable representing the maximum distance traveled among all couriers **(objective variable)**.
- order$j \in \mathbb{Z}$: Integer variable representing the position of item j in the courier route **(for symmetry breaking)**.
- ordmat$i, j \in 0, 1$: Binary permutation matrix; 1 if item i is in position $j + 1$ (to ensure each item has a unique order).

5.2 Constraints

1. Capacity per courier:

Each courier has a limited capacity. We calculate the total size of items courier k delivers by summing over all arcs (i, j) where courier k visits item j. The total weight must not exceed the courier's capacity l_k.

$$\text{weight}_k = \sum_{i=0}^{n} \sum_{j=0}^{n-1} x_{i,j}^k \cdot s_j \quad \text{and} \quad \text{weight}_k \leq l_k \quad \forall k = 0, \ldots, m-1$$

2. Each item visited exactly once:

Every item j must be visited exactly once by a single courier. This is ensured by summing over all incoming arcs (i, j) across all couriers.

$$\sum_{i=0}^{n} \sum_{k=0}^{m-1} x_{i,j}^{k} = 1 \quad \forall j = 0, \ldots, n-1 \quad \text{and} i \neq j$$

3. Flow conservation:

For each courier and item node j, the number of incoming arcs must equal the number of outgoing arcs. This ensures continuous flow through the tour.

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{i,j}^{k} = \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{j,i}^{k} \quad \forall j = 0, \ldots, n-1, \ \forall k = 0, \ldots, m-1$$

4. Depot departure and return:

Each courier must start and end their tour at the depot (node n). We ensure one departure from and one return to the depot.

$$\sum_{j=0}^{n-1} x_{n,j}^{k} = 1 \quad \text{(departure)}$$

$$\sum_{j=0}^{n-1} x_{j,n}^{k} = 1 \quad \text{(return)} \quad \forall k = 0, \ldots, m-1$$

5. Subtour elimination (MTZ constraints)[1]:

The Miller-Tucker-Zemlin (MTZ) constraints prevent disconnected subtours that do not include the depot. We assign each item i a position u_i^k in the route of courier k, and enforce feasible ordering only in valid tours.

$$u_i^k - u_j^k + (n+1) \cdot x_{i,j}^{k} \leq n \quad \forall i \neq j = 0, \ldots, n-1, \ \forall k = 0, \ldots, m-1$$

$$1 \leq u_i^k \leq n$$

6. Distance tracking and maximum bound:

We compute the total distance traveled by courier k based on the arcs used,

and ensure that $D_{\max}$ is at least as large as any courier's travel distance.

$$\text{dist}_k = \sum_{i=0}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} D_{i,j} \cdot x_{i,j}^k \quad \forall k = 0, \ldots, m-1$$

$$D_{\max} \geq \text{dist}_k \quad \forall k = 0, \ldots, m-1$$

7. Coupled deliveries (same courier):

Some items may be coupled, meaning they must be delivered by the same courier. This is enforced by equating the entry arc sums for the coupled items.

$$\sum_{a \neq i} x_{a,i}^k = \sum_{a \neq j} x_{a,j}^k \quad \forall k = 0, \ldots, m-1, \text{ if items } i, j \text{ are coupled}$$

8. Coupled ordering dominance:

If item i must be delivered after item j, we enforce this by requiring that the order index of i is greater than that of j.

$$\text{order}_i \geq \text{order}_j + 1 \quad \text{if item } i \text{ must follow item } j$$

9. Ordering uniqueness via permutation matrix: We use a binary matrix $\text{ordmat}_{i,j}$ to assign each item i to a unique delivery position $j+1$. Every item must occupy exactly one position, and every position must be assigned to exactly one item.

$$\sum_{j=0}^{n-1} \text{ordmat}_{i,j} = 1 \quad \forall i = 0, \ldots, n-1$$

$$\sum_{i=0}^{n-1} \text{ordmat}_{i,j} = 1 \quad \forall j = 0, \ldots, n-1$$

$$\text{order}_i = \sum_{j=0}^{n-1} (j+1) \cdot \text{ordmat}_{i,j} \quad \forall i = 0, \ldots, n-1$$

10. Symmetry breaking:

To avoid equivalent symmetric solutions, we enforce that courier distances are sorted in non-decreasing order.

$$\text{dist}_k \leq \text{dist}_{k+1} \quad \forall k = 0, \ldots, m-2$$

5.3 Validation

The problem was implemented and tested in two configurations: one with the symmetry-breaking constraint enabled, and one without it. This was done to evaluate the impact of symmetry breaking on solver performance and solution quality. Two solvers, CBC and GLPK, were used to solve the models. Due to hardware and time limitations, the experiments were restricted to the first 10 problem instances provided by the course.

For each configuration, we recorded the runtime, whether the instance was solved to optimality, and the value of the objective function ($D_{\max}$). The results were saved in '.json' files as required, and a comparison is presented in the following section.

ID	CBC + SB	CBC w/out SB	GLPK + SB	GLPK w/out SB
1	**14**	**14**	**14**	**14**
2	**226**	**226**	249	**226**
3	**12**	**12**	**12**	**12**
4	**220**	**220**	**220**	N/A
5	**206**	**206**	**206**	**206**
6	**322**	**322**	**322**	**322**
7	212	168	N/A	180
8	234	**186**	**186**	**186**
9	**436**	**436**	N/A	**436**
10	278	**244**	N/A	**244**

Table 9: Results obtained by the MIP model using CBC and GLPK solvers.

6 Conclusions

This project evaluated CP, SAT, and MIP approaches for solving the MCP problem under a 300-second time limit. CP consistently performed well, with advanced techniques like symmetry breaking and heuristics improving solution quality—though Gecode occasionally struggled on harder cases. SAT achieved optimal solutions across nearly all instances, with binary search offering stronger performance. MIP solvers were reliable overall, and symmetry breaking notably reduced runtime in complex scenarios. These results confirm CP's strength as a baseline and highlight SAT and MIP as robust alternatives in different contexts.

References

[1] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulations and a method for solving traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, 1960.