Modelling and Solving the Multiple Couriers Problem

Davide Leone - davide.leone@studio.unibo.it

June 2025

1 Introduction

In modeling the Multiple Couriers Problem, we initially adopted a unified formulation across all three optimization paradigms—Constraint Programming (CP), Satisfiability Modulo Theories (SMT), and Mixed-Integer Programming (MIP)—using a 3D binary "travel" matrix. This *travel-based* approach is commonly used in arc-based formulations of the Capacitated Vehicle Routing Problem (CVRP) [1], where each binary variable encodes whether a courier travels directly between two locations [2].

While this formulation proved effective in the SMT and MIP models, it was inefficient in the CP context, especially for complex instances. As a result, for the CP model we adopted a different strategy based on the *predecessor-successor* relationship, known as the *Giant Tour Representation* (GTR), a well-established technique for solving vehicle routing problems [3, 4, 5, 6]. This shift significantly improved the solver's scalability and performance for larger problem instances.

Although the project began as a group effort, it was ultimately completed individually due to persistent coordination challenges. The main difficulty was ensuring feasibility for more complex instances, which demanded extensive research. Additionally, an error on Windows prevented the use of the Gecode solver within the CP model, forcing to switch to a Linux environment for testing. Working alone also meant I had no opportunity to discuss my ideas and no feedback on the reproducibility of my solution. As a result of these challenges, the project extended over several months and involved extensive experimentation.

Given that two different modeling strategies were employed, the only elements common across all formulations are the following.

Instance Variables

- $M = \{1, 2, \ldots, m\}$: Set of couriers
- $N = \{1, 2, \ldots, n\}$: Set of distribution points
- $O = n + 1$: Index representing the origin location
- l_c: Load capacity of courier $c \in M$
- s_i: Size of the item at distribution point $i \in N$
- D_{ij}: Distance between locations i and j, where $i, j \in N \cup \{O\}$

Decision Variables

- $d_c \geq 0$: Total distance traveled by courier c
- $z \geq 0$: Maximum distance traveled by any courier (objective variable)

Objective Function

$$\min z$$

where:

$$z = \max_{c \in 1..m} d_c$$

Objective Bounds

The objective variable z is bounded by lower and upper estimates:

$$z \in \left[\max_{i \in N} \left(D_{O,i} + D_{i,O} \right), \left\lceil \bar{D} + 2\sigma \right\rceil \cdot \left\lceil \frac{n}{m} \right\rceil \right]$$

where:

$$\bar{D} = \frac{1}{(n+1)^2} \sum_{i,j=1}^{n+1} D_{ij}, \quad \sigma^2 = \frac{1}{(n+1)^2} \sum_{i,j=1}^{n+1} (D_{ij} - \bar{D})^2$$

The lower bound corresponds to the worst-case round-trip from the origin to the furthest distribution point and back. The upper bound formulation tries to balance feasibility and tightness, and was selected based on empirical evaluation; it considers the mean inter-location distance plus two times the standard deviation, scaled by the average number of items per courier.

2 CP Model

The CP model is implemented in MiniZinc and tested using two solvers: `Gecode` and `Chuffed`. As already anticipated, to address the limitations encountered with the travel-matrix formulation, we opted for the *predecessor-successor* approach, specifically through the Giant Tour Representation (GTR).

The GTR encodes the entire multi-courier delivery route as a single sequence (a "giant tour") by augmenting the set of nodes with artificial start and end points for each courier. In this formulation, the full path is composed of $n + 2m$ nodes:

- n distribution nodes: $\{1, \ldots, n\}$,
- m start nodes: $\{n+1, \ldots, n+m\}$, one per courier at the origin,
- m end nodes: $\{n+m+1, \ldots, n+2m\}$, one per courier returning to the origin.

This representation simplifies route construction and enables global constraints such as `circuit` to eliminate subtours and ensure route feasibility.

Additional Instance Variables

- $V = N \cup \{O+1, \ldots, O+2m\}$: Extended node set.
- D_{ij}^{ext}: Extended distance matrix for all $i, j \in V$, defined as:

$$D_{ij}^{\text{ext}} = \begin{cases} D_{ij}, & \text{if } i, j \in N \\ D_{i,O}, & \text{if } i \in N, j \notin N \\ D_{O,j}, & \text{if } i \notin N, j \in N \\ D_{O,O}, & \text{otherwise} \end{cases}$$

- s_i^{ext}: Extended item size for all $i \in V$, defined as

$$s_i^{\text{ext}} = \begin{cases} s_i, & \text{if } i \in N \\ 0, & \text{otherwise} \end{cases}$$

2.1 Decision Variables

In addition to the global variables defined in Section 1, the CP model introduces:

- **Successor Assignment:** For each node $i \in V$, let:

  $$\text{succ}(i) \in V \setminus \{i\}$$

  be the immediate successor of node i in the delivery tour.

- **Predecessor Assignment:** Similarly, for each node $i \in V$, define:

  $$\text{pred}(i) \in V \setminus \{i\}$$

  as the node immediately preceding i in the tour. This is redundant but may help in model propagation.

- **Courier Assignment:** For each node $i \in V$, let:

  $$a_i \in M$$

  denote the courier responsible for visiting node i. This ensures all nodes are assigned to exactly one courier and supports constraints on load capacity and routing consistency.

- **Intermediate Load:** For each node $i \in V$, let:

  $$L_i \geq 0$$

  denote the load upon arrival at node i.

2.2 Objective Function

The goal is to minimize the maximum distance travelled by any courier, as described formally in Section 1.

2.3 Constraints

The model includes the following constraints:

1. Start and End Connections

- Successors of end nodes are start nodes.

$$\text{succ}(i) = i - m + 1, \quad \forall i \in \{n + m + 1, \ldots, n + 2m - 1\}$$
$$\text{succ}(n + 2m) = n + 1$$

- Predecessors of start nodes are end nodes.

$$\text{pred}(i) = i + m - 1, \quad \forall i \in \{n + 2, \ldots, n + m\}$$
$$\text{pred}(n + 1) = n + 2m$$

2. Courier Initialization

- Associate a courier to each start node.

$$a_i = i - n, \quad \forall i \in \{n + 1, \ldots, n + m\}$$

- Associate a courier to each end node.

$$a_i = i - n - m, \quad \forall i \in \{n + m + 1, \ldots, n + 2m\}$$

3. Successor-Predecessor Consistency

$$\text{succ}(\text{pred}(i)) = i, \quad \forall i \in V$$
$$\text{pred}(\text{succ}(i)) = i, \quad \forall i \in V$$

4. Courier Consistency: Ensure the courier of a node is the same as its predecessor and successor.

$$a_i = a_{\text{succ}(i)} = a_{\text{pred}(i)}, \quad \forall i \in N$$

5. Load Constraints

- Initial load at start nodes is 0.

$$L_i = 0, \quad \forall i \in \{n + 1, \ldots, n + m\}$$

- Final load at end nodes is more than 0 (ensures that all couriers are utilized).

$$L_{n+m+c} > 0, \quad \forall c \in M$$

- Maintain load consistency when moving between nodes.

$$L_{\text{succ}(i)} = L_i + s_i^{\text{ext}}, \quad \forall i \in N$$
$$L_i = L_{\text{succ}(i)}, \quad \forall i \in \{n + 1, \ldots, n + m\}$$

- Ensure load does not exceed the courier's capacity at any node.

$$L_i \leq l_{a_i}, \quad \forall i \in N$$

6. Subtour Elimination

$$\texttt{circuit}(\text{succ})$$

$$\texttt{circuit}(\text{pred})$$

7. Distance Calculation

$$d_c = \sum_{\substack{i \in V \\ a_i = c}} D^{\text{ext}}_{i,\text{succ}(i)}, \quad \forall c \in M$$

Symmetry Breaking Constraints

To reduce the search space caused by symmetric courier assignments, we enforce a lexicographic ordering:

$$\forall c_1, c_2 \in M,\ c_1 > c_2,\ l_{c_1} = l_{c_2} \Rightarrow \text{lex}\left([a_i = c_1]_{i \in N}\right) < \text{lex}\left([a_i = c_2]_{i \in N}\right)$$

2.4 Validation

Experimental Design

Several configurations were tested to evaluate the effect of solver choice, symmetry breaking, and search strategy. We used both the default and a randomized search to improve exploration, each with and without symmetry breaking. This led to six runs per instance (four with Gecode, two with Chuffed). All experiments were conducted on a laptop equipped with an Intel Core i7-6500U processor (2.50GHz, 4 cores) and 12GB of RAM and with a 300-second time limit per run.

Experimental Results

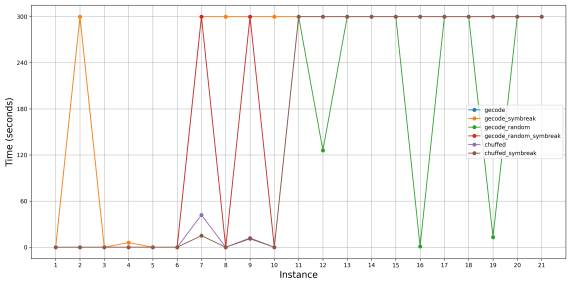

Figure 1: Runtime comparison between various configurations

The results of Table 1, clearly show how important is to use a more sophisticated search strategy, which allows to find a solution for almost all the instances. Also Figure 1 shows how using random search strategy reduces the runtime required to find an optimal solution for some of the more complex instances.

Instance	Naive Search				Randomized Search	
	Gecode	Gecode + SB	Chuffed	Chuffed + SB	Gecode	Gecode + SB
1	**14**	**14**	**14**	**14**	**14**	**14**
2	277	235	**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	179	179	**167**	**167**	**167**	**167**
8	229	-	**186**	**186**	**186**	**186**
9	-	-	**436**	**436**	**436**	**436**
10	-	-	**244**	**244**	**244**	**244**
11	-	-	-	-	644	-
12	-	-	-	-	**346**	-
13	-	-	-	-	1160	1284
14	-	-	-	-	1333	-
15	-	-	-	-	1063	-
16	-	-	-	-	**286**	-
17	-	-	-	-	-	-
18	-	-	-	-	1071	-
19	-	-	-	-	**334**	-
20	-	-	-	-	-	-
21	-	-	-	-	667	-

Table 1: Objective values obtained by different model configurations

3 SMT Model

The SMT model is formulated using Z3 and relies on boolean and integer variables to represent routing decisions and enforce constraints.

3.1 Decision Variables

In addition to the global variables defined in Section 1, the SMT model introduces:

- $t_{cij} \in \{\text{true}, \text{false}\}$: Indicates whether courier $c \in M$ travels from node i to node j, where $i, j \in N \cup \{O\}$.
- $u_i \in \mathbb{Z}$: Helper variable representing the position of the distribution point $i \in N$ in the route (used for subtour elimination).

3.2 Objective Function

The goal is to minimize the maximum distance traveled by any courier, as described formally in Section 1.

3.3 Constraints

The model enforces the following constraints.

1. Origin Start/End:

- Each courier must leave the origin exactly once.

$$\sum_{j \in N} t_{cOj} = 1, \quad \forall c \in M$$

- Each courier must return to the origin exactly once.

$$\sum_{j \in N} t_{cjO} = 1 \quad \forall c \in M$$

2. Unique Node Visit:

- Each distribution point must be entered exactly once.

$$\sum_{c \in M} \sum_{i \in N \cup \{O\}} t_{cij} = 1, \quad \forall j \in N$$

- Each distribution point must be exited exactly once.

$$\sum_{c \in M} \sum_{j \in N \cup \{O\}} t_{cij} = 1, \quad \forall i \in N$$

3. Flow Conservation: The number of incoming and outgoing arcs must be equal at each node for each courier.

$$\sum_{i \in N \cup \{O\}} t_{cij} = \sum_{k \in N \cup \{O\}} t_{cjk}, \quad \forall c \in M, j \in N \cup \{O\}$$

4. Avoid Self-loops: No node can be visited twice in a row.

$$t_{cii} = 0, \quad \forall c \in M, i \in N$$

5. Capacity Constraint: Total item sizes collected by each courier must not exceed its capacity.

$$\sum_{i \in N} \sum_{j \in N \cup \{O\}} s_i \cdot t_{cij} \leq l_c, \quad \forall c \in M$$

6. Subtour Elimination: Using the Miller-Tucker-Zemlin formulation [7, 8], we eliminate subtours.

- Enforce correct ordering in the tour.

$$u_i + 1 \leq u_j + n \cdot (1 - t_{cij}), \quad \forall c \in M, i \neq j \in N$$

- Enforce valid position indices for all nodes (between `1` and n).

$$1 \leq u_i \leq n, \quad \forall i \in N$$

7. Objective Tracking:

- Compute the route length of each courier.

$$d_c = \sum_{i,j \in N \cup \{O\}} D_{ij} \cdot t_{cij}, \quad \forall c \in M$$

- Set z accordingly.

$$z \geq d_c, \quad \forall c \in M$$

3.4 Validation and Results

Experimental Design

To enhance solver efficiency, we introduce an optional heuristic-based pruning strategy that limits the search space to a subset of promising arcs. Since Z3 does not support warm-starting with an initial solution, we approximate this behavior by disabling arcs unlikely to contribute to high-quality solutions. This strategy is configurable, and while aggressive pruning may eliminate optimal paths, it significantly improves scalability and runtime. To ensure reproducibility, heuristics can be generated dynamically or loaded from disk. The detailed procedure is described in Appendix A.

When pruning is enabled:

- The upper bound on the objective z is defined as the minimum between the analytical bound described in Section 1 and the maximum route length found in the best heuristic solution.
- Variables t_{cij} for arcs $(c, i, j) \notin A'$ are fixed to `false`, where A' denotes the set of arcs extracted from heuristic solutions.

For the experiments, only arcs from the ten best heuristic solutions were retained—an intentionally low threshold used to demonstrate the method's impact.

All experiments were conducted with Z3 and a 300-second time limit per instance, on a machine with an Intel Core i7-10750H (2.60GHz, 6 cores) and 16GB RAM.

Experimental Results

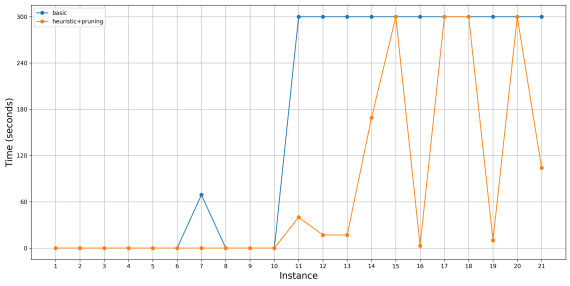

Figure 2: Runtime comparison between various configurations

Instance	Basic	Heuristic+Pruning
1	**14**	**14**
2	**226**	**226**
3	**12**	**12**
4	**220**	**220**
5	**206**	**206**
6	**322**	**322**
7	**167**	**167**
8	**186**	**186**
9	**436**	**436**
10	**244**	**244**
11	-	304
12	-	346
13	-	474
14	-	364
15	-	-
16	-	**286**
17	-	-
18	-	-
19	-	334
20	-	-
21	-	374

Table 2: Objective values obtained by different model configurations

As shown in Table 2, the heuristic+pruning strategy significantly enhances the model's ability to solve larger instances within the time limit (Figure 2), enabling the solution of seven additional instances compared to the basic model. However, our experiments revealed that in most cases, the objective value returned by the solver matches the heuristic upper bound exactly. This suggests that the model often reproduces the best heuristic solution rather than improving upon it. This limitation derives from the small number of heuristic solutions considered, which restricts the model's ability to explore alternative solutions. Striking a balance between pruning for feasibility and preserving enough flexibility for genuine optimization is non-trivial. While this approach is effective to find a solution, the result is often closer to a guided reconstruction of the heuristic rather than a true optimization. Nevertheless, we report these results for completeness.

4 MIP Model

The MIP model is formulated using both `PuLP` (with CBC) and `GurobiPy` (with Gurobi), with a consistent logic across implementations.

4.1 Decision Variables

In addition to the global variables defined in Section 1, both MIP models introduce:

- $t_{cij} \in \{0, 1\}$: Binary variable indicating whether courier $c \in M$ travels from node i to node j, $i, j \in N \cup \{O\}$.
- $u_{ci} \in \mathbb{R}_{\geq 0}$: Load carried by courier c upon reaching distribution point $i \in N$ (used for subtour elimination).

4.2 Objective Function

The goal is to minimize the maximum distance traveled by any courier, as described formally in Section 1.

4.3 Constraints

The constraints are the same as the ones described in Section 3.3. In particular:

- **Origin Start/End:** Each courier starts and ends their route at the origin.
- **Unique Node Visit:** Each distribution point is visited and exited exactly once.
- **Flow conservation:** The number of entries to a node equals the number of exits for each courier.
- **Capacity Constraint:** The total load delivered by each courier does not exceed their maximum capacity.
- **Subtour Elimination:** Using the Miller-Tucker-Zemlin (MTZ) formulation.
- **Objective Tracking:** Each courier's route must not exceed the overall objective value z.

4.4 Validation

Experimental Design

As anticipated, we evaluated two MIP-based solvers:

1. A baseline model implemented with `PuLP` using the default `CBC` solver, supporting an optional pruning mechanism.
2. An advanced model using `Gurobi`, supporting optional warm-start and pruning mechanisms.

For configurations involving warm-start or pruning, we employed an heuristic to generate a feasible solution. The heuristic procedure, described in Appendix A, is shared with the SMT model. In this context, it provides only one high-quality initial solution for warm-start and another upper bound on the objective z. As in the SMT model, heuristics can be generated dynamically or loaded from disk for reproducibility.

To reduce computational complexity, we also applied a pruning strategy based on pairwise distances between nodes. The idea is to eliminate arcs between distribution points that are too distant to belong to an efficient tour. We compute several thresholds and choose the pruning threshold as the mean of these values. For each courier, any arc exceeding this threshold and not included in the heuristic candidate set is disabled.

Given the considerations above, each instance was tested under the following six configurations:

- **CBC-Basic:** PuLP model with CBC, without enhancements.
- **CBC-Basic+Pruning:** PuLP model with CBC, plus arc pruning based on distance thresholds.
- **Gurobi-Basic:** Gurobi model without warm-start or pruning.
- **Gurobi+Warm Start:** Gurobi model initialized with a heuristic solution.
- **Gurobi+Pruning:** Gurobi model with arc pruning based on distance thresholds.
- **Gurobi+Warm Start+Pruning:** Combined configuration using both strategies.

All experiments were executed with a 300-second time limit per instance on a laptop equipped with an Intel Core i7-10750H (2.60GHz, 6 cores) and 16GB of RAM. During testing, we observed that the first two configurations frequently exceeded the time limit for instances beyond ten. As a result, these configurations can be optionally disabled. Additionally, since Gurobi requires a valid license, its related configurations can also be optionally excluded to ensure full reproducibility across environments.

Experimental Results

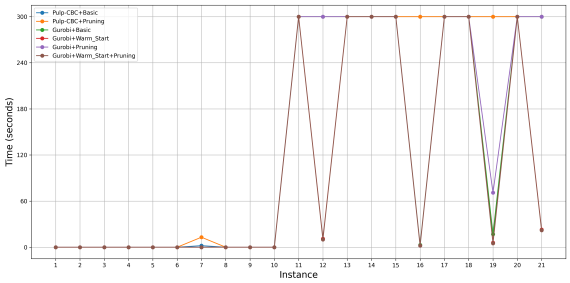

Figure 3: Runtime comparison between various configurations

From Table 3, we observe that for smaller instances, all configurations successfully reach the optimal solution. As instance complexity increases, the CBC solver struggles to return results within the time limit, whereas the Gurobi solver is able to solve more instances—even in its basic configuration. While pruning alone has limited impact on solvability, the use of warm start significantly improves performance, enabling the solver to find solutions across all tested instances. Also, combining warm start with pruning can, in some cases, lead to improved solution quality. Furthermore, as shown in Figure 3, warm start also reduces the solver's runtime, allowing it to converge more quickly.

5 Conclusions

This project addressed the Multiple Couriers Problem by modeling it using three different optimization paradigms: Constraint Programming (CP), Satisfiability Modulo Theories (SMT), and Mixed-Integer Programming (MIP).

Instance	Pulp-CBC		Gurobi			
	Basic	Pruning	Basic	Warm Start	Pruning	WS+Pruning
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**	**436**	**436**	**436**	**436**	**436**
10	**244**	**244**	**244**	**244**	**244**	**244**
11	-	-	-	305	-	305
12	-	-	533	346	-	346
13	-	-	454	442	428	434
14	-	-	-	362	-	362
15	-	-	-	364	-	364
16	-	-	286	286	286	286
17	-	-	-	444	-	444
18	-	-	-	341	-	341
19	-	-	334	334	334	334
20	-	-	-	471	-	471
21	-	-	1300	374	-	374

Table 3: Objective values obtained by different model configurations

In the CP model, the transition from a 3D travel matrix to a Giant Tour Representation (GTR) based on predecessor-successor relationships significantly improved the solver's ability to handle larger instances. The use of a randomized search strategy proved particularly effective in reducing runtime and increasing the number of solvable instances.

The SMT model, built with Z3, demonstrated solid performance when augmented with a heuristic-based pruning strategy. This approach enabled the model to solve larger instances by limiting the search space to promising arcs. However, it often replicated heuristic solutions rather than discovering improved ones, indicating that the pruning strategy may overly constrain the solution space.

The MIP model, implemented using both PuLP (CBC solver) and Gurobi, benefited the most from advanced techniques such as warm-start initialization and arc pruning. Gurobi consistently outperformed CBC on larger instances, and the use of warm starts led to faster convergence and better solution quality.

In conclusion, despite their differing approaches, all models produced comparable objective values on simpler instances, confirming the validity of their formulations. Among them, the MIP model with warm-start and pruning strategies demonstrated the most consistent performance across a wide range of problem sizes, allowing to find a feasible solution to all instances.

A Appendix

Heuristic Strategy

The heuristic used to support pruning and warm start is structured in four stages:

1. **Clustering:** Distribution points are grouped into clusters, one per courier, based on proximity to the origin and load constraints. The method uses greedy randomized assignment [5, 9, 10]. to generate a valid clustering without empty groups.
2. **Trip Construction:** Feasible trips are constructed by greedily visiting delivery points from the origin and returning to it, ensuring courier capacity constraints are respected.
3. **Feasibility Verification:** Each generated trip is checked to ensure that:
   - It starts and ends at the origin.
   - No delivery point is repeated.
   - Load limits are not exceeded.
4. **Tour Evaluation:** Each valid trip set is evaluated by computing the tour for each courier and measuring the maximum tour length. The best n_{sol} solutions are retained.

The output consists of:

- A *heuristic upper bound*, representing the best maximum distance among the top solutions.
- An *arc set*, $A' \subset M \times (N \cup \{O\})^2$, comprising the union of arcs used across the selected n_{sol} solutions. For the SMT model, n_{sol} is specified as a parameter (default: 10), so we can influence how many arcs will be retained after pruning. For the MIP model, $n_{\text{sol}} = 1$, since we need only the best solution to warm start the solver.

References

[1] Wikipedia contributors. Vehicle routing problem — wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Vehicle_routing_problem, 2024. Accessed: 2025-05-28.

[2] AIMMS. Formulation for the capacitated vehicle routing problem (cvrp). https://how-to.aimms.com/Articles/332/332-Formulation-CVRP.html, 2024. Accessed: 2025-05-28.

[3] J. Gromicho, J. J. van Hoorn, A. L. Kok, and J. M. J. Schutten. Restricted dynamic programming: A flexible framework for solving realistic vrps. *Transportation Science*, 46(3):288–302, 2012.

[4] W. David Fröhlingsdorf. Using constraint programming to solve the vehicle routing problem with time windows. https://publications.lib.chalmers.se/records/fulltext/256792/256792.pdf, 2018. Master's Thesis, Chalmers University of Technology.

[5] Christian Prins, Philippe Lacomme, and Caroline Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179–200, 2014.

[6] Birger Funke, Tore Grünert, and Stefan Irnich. Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of Heuristics*, 11(4):267–306, 2005.

[7] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.

[8] AIMMS. Miller-tucker-zemlin formulation. https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html, 2024. Accessed: 2025-05-28.

[9] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[10] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 31(7):1073–1091, 2004.