Very_cool_project.mzn

Valerio Costa, Luca Domeniconi, Claudia Maiolino, Tian Cheng Xia
{valerio.costa, luca.domeniconi5, claudia.maiolino, tiancheng.xia}@studio.unibo.it

1 Introduction

The problem of this project is known in the literature as the vehicle routing problem. We tackle this problem by virtually splitting it into two sub-problems: first we look for an assignment of the items to the couriers and then search for the routes of each of them (i.e., by solving multiple traveling salesman problems). To solve the latter, we follow the approach presented in [1] where the route of a courier is modelled through the variables $P_d \in [1, n+1]$ with $d \in [1, n+1]$ defined as follows:

$$P_{d_1} = \begin{cases} d_2 & \text{iff the location } d_2 \neq d_1 \text{ is visited immediately after } d_1 \\ d_1 & \text{iff } d_1 \text{ is not part of the route of the courier} \end{cases} \tag{1}$$

With proper assignment and subtour elimination constraints, P_d allows to define a Hamiltonian cycle that passes through the items that the courier delivers and the solution can be extracted by following the cycle starting from the depot $n+1$.

The lower-bound of the objective function is common to all models and is defined as the maximum path cost that involves a single package:

$$\max_{p \in \{1,\ldots,n\}} \{D_{n+1,p} + D_{p,n+1}\} \tag{2}$$

As upper-bound, we observed that it does not provide any improvements to the results. Nevertheless, we defined it as:

$$\sum_{d_1 \in [1,n+1]} \max_{d_2 \in [1,n+1]} D_{d_1,d_2} \tag{3}$$

Symmetry breaking constraints are also common to all models. By considering couriers with the same capacity, the following constraints can be used to avoid symmetries:

- By imposing an ordering on the amount of assigned packages:

  $$\forall c_1, c_2 \in [1, m] : (c_1 < c_2 \land l_{c_1} = l_{c_2}) \Rightarrow Q_{c_1} \leq Q_{c_2} \tag{4}$$

  where Q_c is the amount of packages delivered by the courier c.

- By imposing an ordering on indexes of the assigned packages:

  $$\forall c_1, c_2 \in [1, m] : (c_1 < c_2 \land l_{c_1} = l_{c_2}) \Rightarrow A_{c_1} <_{\texttt{lex}} A_{c_2} \tag{5}$$

  where A_c is an ordered vector containing the packages delivered by the courier c.

As the triangle inequality holds, we also identified an implied constraint that consists of imposing that each courier delivers at least a package (a short proof is provided in Section A):

$$\forall c \in [1, m] : Q_c \geq 1 \tag{6}$$

where Q_c is the amount of packages delivered by the courier c. This obviously is applicable only if the size of the packages is compatible with the load capacity of each courier.

All experiments use the same random seed and were run as workflows on GitHub Actions which provides two cores at 2.45 GHz and 7 GB of memory, which we software limited to 5 GB to guarantee a safe margin for the Docker container to run.

The work has been completed in approximately one month and has been roughly split in the following way: Xia did the CP part, Costa worked on SAT, Domeniconi did the SMT models, and Maiolino completed the MIP part. The main difficulties we encountered are the following: (i) lack of proper documentation for many tools we used, (ii) aaa.

2 CP model

2.1 Decision variables

The CP model relies on the following decision variables:

- For each package p, $A_p \in [1, m]$ (`assignments[p]` in MiniZinc) indicates which courier delivers it. More specifically, $A_p = c$ iff the package p is delivered by the courier c.
- For each courier c, $P_{c,d}$ (`path[c, d]` in MiniZinc) is defined following Equation (1).

2.2 Objective function

Once the route of each courier has been found, the objective function is computed as follows:

$$\max_{c \in \{1,\dots,m\}} \sum_{\substack{d \in \{1,\dots,n+1\}: \\ P_{c,d} \neq d}} \mathtt{D}[d,\ P_{c,d}] \tag{7}$$

2.3 Constraints

To respect the capacity limits of each courier, the following constraint can be defined:

$$\forall c \in \{1, \dots, m\} : \sum_{p \in \{1,\dots,n\}: A_p = c} s_p \leq l_c \tag{8}$$

In MiniZinc, the global constraint `bin_packing_capa` also models this. In our experiments, we did not notice significant differences between the two formulations and decided to use the latter following the best practice of preferring global constraints.

To model the route, the following constraints have to be imposed:

$$\forall c \in \{1, \dots, m\} : \begin{cases} P_{c,n+1} = n + 1 & \text{if } \nexists p \in \{1, \dots, n\} : A_p = c \\ P_{c,n+1} \neq n + 1 & \text{if } \exists p \in \{1, \dots, n\} : A_p = c \end{cases} \tag{9}$$

$$\forall c \in \{1, \dots, m\}, \forall p \in \{1, \dots, n\} : \begin{cases} P_{c,p} = p & \text{if } A_p \neq c \\ P_{c,p} \neq p & \text{if } A_p = c \end{cases} \tag{10}$$

The constraint defined in Equation (9) imposes that, for each courier, the depot has a successor only if that courier delivers at least a package. On the same note, Equation (10) imposes that only the packages delivered by a specific courier have a successor in the route. Moreover, it is necessary to impose that the route defined by P_c is a Hamiltonian cycle that passes through the relevant destinations. In MiniZinc, this can be easily modelled by using the global constraint `subcircuit`.

2.3.1 Symmetry breaking constraints

For CP, we experimented with both symmetry breaking constraints defined in Equations (4) and (5). Moreover, we experimented with a stronger version of these constraints that is applied between two couriers whose actual loads are interchangeable. This is defined by the following condition:

$$\max \{L_{c_1}, L_{c_2}\} \leq \min \{l_{c_1}, l_{c_2}\} \tag{11}$$

where L_c is the actual load carried by the courier c.

2.4 Validation

2.4.1 Experimental design

We experimented variations of our models using as solvers Gecode, Chuffed, and OR-Tools. The search order for all models assigns the packages (`assignments`) first and then searches for the route (`path`) of each courier in decreasing order of capacity. Regarding search, we experimented several combinations of variable selection and assignment strategies on a subset of instances and used only the best ones to obtain the final complete results.

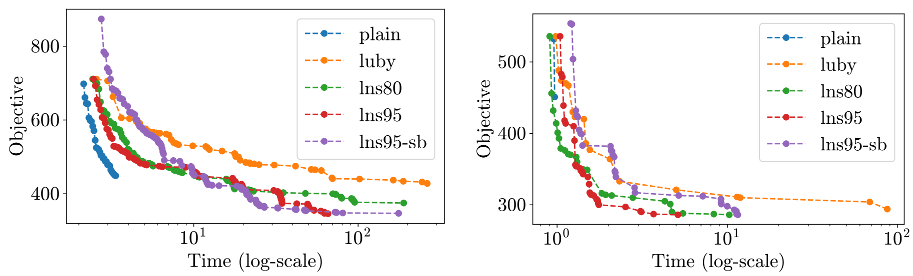
2.4.2 Experimental results

From some preliminary experiments, we observed that first fail and largest domain with weighted degree (`dom_w_deg`) are the best performing search strategies. Therefore, we performed the full experiments using as search strategy `dow_w_deg` for `assignments` and `first_fail` for `path`, respectively. Moreover, between the two symmetry breaking constraints, we observed that the lexicographic ordering defined in Equation (5) has better performances and therefore present only those results. The objective values found by the most significant models are reported in Table 1.

Table 1: Selected subset of CP results. Results in **bold** are solved to optimality. Instances that are all solved to optimality have been omitted.

	Gecode						Chuffed			OR-Tools		
Id	plain	luby	lns80	lns95	lns95 + SB (5)	lns95 + SB (5)(11)	plain	luby	SB (5)	plain	plain + FS	SB (5) + FS
7	201	**167**	**167**	**167**	**167**	**167**	**167**	**167**	**167**	408	**167**	**167**
9	**436**	**436**	**436**	**436**	**436**	**436**	**436**	**436**	**436**	617	**436**	**436**
11	594	597	528	490	503	–	963	–	756	1669	1189	1081
12	449	428	375	**346**	348	–	833	1061	785	1605	706	899
13	648	704	656	616	610	624	1126	914	1126	1734	584	746
14	725	972	794	715	792	–	1449	–	1089	–	–	–
15	659	901	765	738	803	–	1292	–	–	–	–	–
16	451	294	**286**	**286**	**286**	–	487	636	694	998	467	363
17	1324	1468	1119	1076	1155	–	–	–	–	–	–	–
18	691	806	675	662	620	–	1321	–	1779	–	–	–
19	554	412	336	**334**	**334**	–	795	1079	765	1521	623	577
20	1139	1369	1104	1068	1075	–	–	–	–	–	–	–
21	779	687	600	516	529	–	2104	2140	993	2115	1226	–

For Gecode, we experimented different search approaches by testing restart methods, large neighborhood search (LNS), and varying symmetry breaking constraints. We observed that, compared to a depth-first approach, by simply restarting search following the Luby sequence there is a mixed impact on the results with both positive and negative effects. Instead, by also using LNS, results tend to improve globally and we observed that a higher retain probability, with a peak at around 95%, works better. Regarding symmetry breaking constraints, we observed that they tend to worsen the final objective value in the majority of the cases, most reasonably due to the fact that the overhead to impose them is higher than the speed-up in search. For a visual comparison, we show in Figure 1 the evolution of the objective value during search. It can be seen that without restart the objective stops improving in the early stages of search as it most likely gets "stuck" in a branch of the search tree. By introducing some non-determinism, the objective reaches lower values, with LNS being the fastest and best performing.



(a) Instance 12 (b) Instance 16

Figure 1: Intermediate solutions using Gecode

For Chuffed, we only experimented with restarts and symmetry breaking constraints as LNS is not available through MiniZinc. Moreover, as indicated in the documentation[1], we tried enabling free search mode but only obtained worse results. Compared to Gecode, results on bigger instances are generally all worse, while smaller instances tend to be solved faster by Chuffed. As in Gecode, restarts, which we only experimented with random variable selection as it is the only non-deterministic search strategy available, and symmetry breaking constraints both yield mixed effects on the final result.

Finally, for OR-Tools, we conducted fewer experiments as it supports fewer MiniZinc annotations. In this case, we observed that enabling free search mode allows obtaining better results and, similarly to Gecode and Chuffed, symmetry breaking constraints have mixed results. As a side note, we must also observe that, as suggested in the documentation, these results are worse as OR-Tools performs better with multi-threading.

3 SAT model

3.1 Decision variables

For SAT, we defined two different models:

Unified model based on the definition of the two decision variables A (`assignments` in Z3) and P (`paths` in Z3) as defined in Section 1.

Matrix model based on the definition of a single matrix X such that $X\texttt{[}c\texttt{,}\ p\texttt{,}\ k\texttt{]} = 1$ iff courier c delivers item p as its k-th package.

Our discussion will be focused on the former to maintain coherence with the models defined in the other methods and obtain more comparable results. It decision variables are the following:

- A is an $n \times m$ matrix such that $A\texttt{[}p\texttt{,}\ c\texttt{]} = 1$ iff courier c delivers item p.

[1] `https://docs.minizinc.dev/en/stable/solvers.html`

- P is an $m \times (n+1) \times (n+1)$ matrix such that $P\texttt{[}c\texttt{,}\ loc_1\texttt{,}\ loc_2\texttt{]} = 1$ iff courier c moves from location loc_1 to loc_2.
- U is an $m \times n \times n$ matrix to implement the MTZ subtour elimination [3]. $U\texttt{[}c\texttt{,}\ p\texttt{,}\ k\texttt{]} = 1$ iff for the courier c the item p is delivered as the k-th.

3.1.1 Objective function

The objective function is computed as follows:

$$\max_{c \in \{1,\dots,m\}} \sum_{loc_1=1}^{n+1} \sum_{loc_2=1, loc_2 \neq loc_1}^{n+1} \texttt{D[}loc_1\texttt{,}\ loc_2\texttt{]} \cdot P\texttt{[}c\texttt{,}\ loc_1\texttt{,}\ loc_2\texttt{]} \tag{12}$$

3.1.2 Constraints

Assignment related constraints

- Capacity constraint:
  - Sum of sizes of packs delivered by a singular courier must be under its load limit

$$\forall c \in [1,m] : \sum_{p=1}^{n} A\texttt{[}p\texttt{,}\ c\texttt{]} \cdot s\texttt{[}p\texttt{]} \leq l\texttt{[}c\texttt{]} \tag{13}$$

  - Each pack must be delivered only by a courier

$$\forall p \in [1,n], \exists! c \in [1,m] : A\texttt{[}p\texttt{,}\ c\texttt{]} = 1 \tag{14}$$

Path related constraints

- General path constraints
  - If courier delivers at least one package, there must exist a destination from DEPOT[2] with true value, else it can stay in DEPOT

$$\forall c \in [1,m], \forall p \in [1,n] : \begin{cases} P\texttt{[}c\texttt{, DEPOT, DEPOT]} = 0 & \texttt{if } \sum_{p=1}^{n} A\texttt{[}p\texttt{,}\ c\texttt{]} \geq \texttt{1} \\ P\texttt{[}c\texttt{, DEPOT, DEPOT]} = 1 & \texttt{if } \sum_{p=1}^{n} A\texttt{[}p\texttt{,}\ c\texttt{]} = \texttt{0} \end{cases} \tag{15}$$

  - If courier delivers pack p, its destination must be different from p

$$\forall c \in [1,m], \forall p \in [1,n] : \begin{cases} P\texttt{[}c\texttt{,}\ p\texttt{,}\ p\texttt{]} = 0 & \texttt{if } A\texttt{[}p\texttt{,}\ c\texttt{]} = 1 \\ P\texttt{[}c\texttt{,}\ p\texttt{,}\ p\texttt{]} = 1 & \texttt{if } A\texttt{[}p\texttt{,}\ c\texttt{]} = 0 \end{cases} \tag{16}$$

[2] DEPOT is the original position, hypotesized as `n+1`.

– Each pack must be delivered by a single courier only once

$$\forall c \in [1,m], \forall loc_1 \in [1,n+1]: \quad \sum_{loc_2=1}^{n+1} P\texttt{[}c\texttt{,}\ loc_1\texttt{,}\ loc_2\texttt{]} = 1 \tag{17}$$

- Subcircuit constraints
  - Only a single courier must deliver pack in location loc_2 from loc_1

$$\forall c \in [1,m], \forall loc_2 \in [1,n+1]: \quad \sum_{loc_1=1}^{n+1} P\texttt{[}c\texttt{,}\ loc_1\texttt{,}\ loc_2\texttt{]} = 1 \tag{18}$$

  - Subtour elimination
    * u relative to first pack p of courier c path must have value = 1

$$\forall c \in [1,m], \forall p \in [1,n]: P\texttt{[}c\texttt{, DEPOT,}\ p\texttt{]} = 1 \Rightarrow U\texttt{[}c\texttt{,}\ p\texttt{,}\ \texttt{1]} = 1 \tag{19}$$

    * $u_j \geq u_i + 1$

$$\forall c \in [1,m],\ \forall i,j,k \in [1,n]: \quad P\texttt{[}c\texttt{,}\ i\texttt{,}\ j\texttt{]} \wedge U\texttt{[}c\texttt{,}\ i\texttt{,}\ k\texttt{]} \Rightarrow \sum_{l=k+1}^{n} U\texttt{[}c\texttt{,}\ j\texttt{,}\ l\texttt{]} = 1 \tag{20}$$

    * Exatcly one true value for each U$\texttt{[}c,p,\texttt{:]}$ vector

$$\forall c \in [1,m], \forall p_1 \in [1,n]: \sum_{p_2=1}^{n} U\texttt{[}c\texttt{,}\ p_1\texttt{,}\ p_2\texttt{]} = 1 \tag{21}$$

    * MTZ formulation constraint:

$$\begin{gathered} u_i - u_j + 1 \leq (n-1) * (1 - P[c,i,j]) \\ \forall c \in [1,m],\ \forall i,j,k_1,k_2 \in [1,n]: \\ U\texttt{[}c\texttt{,}\ i\texttt{,}\ k_1\texttt{]} \wedge U\texttt{[}c\texttt{,}\ j\texttt{,}\ k_2\texttt{]} \Rightarrow (k_1 - k_2 + 1) \leq (n-1) \cdot (1 - P\texttt{[}c\texttt{,}\ i\texttt{,}\ j\texttt{]}) \end{gathered} \tag{22}$$

Symmetry breaking constraint A possible way to reduce tree exploration is to introduce symmetry breaking constraints. One of them in this case can consist in constraining the order of assignments of packs between couriers with the same amount of load capacity.

But the experimentation didn't lead to major improvements.

3.2 Validation

Experimental design

Some modifications were applied to the basic model in order to visualize eventuale differences in performances. The original Unified Model was modified in three different versions:

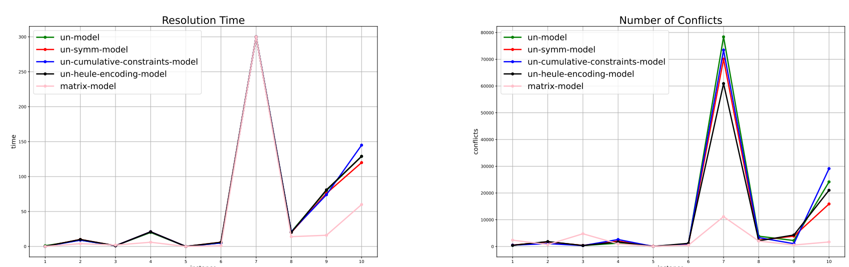
- Unified Model with Symmetry Breaking Constraint
- Unified Model with Cumulative Constraint Application
- Unified Model with Heule Encoding approach for `at_most_one()`

Experimental results

As we can notice from the following table the performances were not very much different from the basic model. Only in the first 10 instances it has been possible to rach at least a suboptimal solution, while for the remaining ones the construction of the model required too much time causing the exceeding of the timeout limit.

Table 2: Objective value through instances

Id	un-model	un-symm-model	un-cum-constr-model	un-heule-enc-model	matrix-model
1	**14**	**14**	**14**	**14**	**14**
2	**226**	**226**	**226**	**226**	**226**
3	**12**	**12**	**12**	**12**	**12**
4	**220**	**220**	**220**	**220**	**220**
5	**206**	**206**	**206**	**206**	**206**
6	**322**	**322**	**322**	**322**	**322**
7	232	238	222	296	292
8	**186**	**186**	**186**	**186**	**186**
9	**436**	**436**	**436**	**436**	**436**
10	**244**	**244**	**244**	**244**	**244**



(a) Resolution time for each instance

(b) Number of conflicts for each instance

Figure 2: Statistics about the resolution of the problem for the first 10 instances

3

4 SMT model

4.1 Decision variables

The SMT model uses the logic of quantifier-free linear integer arithmetic (`QF_LIA`) and relies on the following decision variables:

- For each package j, $A_j \in [1, m]$ (`ASSIGNMENTS[j]` in Z3) indicates which courier delivers package j where $A_j = i$ indicates that courier i delivers package j.
- For each courier i, $P_{i,d} \in [1, n+1]$ for $d \in [1, n+1]$ (`PATH[i][d]` in the code) follows the definition of Equation (1).

4.2 Objective function

The objective function is defined as follows:

$$\max_{i \in \{1,\ldots,m\}} \texttt{DISTANCES}_i$$

where $\texttt{DISTANCES}_i$ is equal to the distance traveled by each courier.

4.3 Constraints

- Weight constraint:

$$\forall i \in \{1,\ldots,m\} : \sum_{j \in \{1,\ldots,n\} : A_j = i} s_j \leq l_i \tag{23}$$

- Assignment and path constraint:

$$\forall j \in \{1,\ldots,n\} : \quad A_j = i \iff P_{i,j} \neq j \tag{24}$$

$$\forall j \in \{1,\ldots,n\} : \quad A_j \neq i \iff P_{i,j} = j \tag{25}$$

- All the elements of each row of P should be distinct:

$$\forall i \in \{1,\ldots,m\} : \quad P_{i,j_1} \neq P_{i,j_2} \quad \forall j_1, j_2 \in \{1,\ldots,n+1\}, j_1 \neq j_2 \tag{26}$$

- Subcircuit constraint: each row P_i should define a subcircuit, that is a Hamiltonian path that ignores all the elements $P_{i,j} = j$, namely the packages that the courier i don't deliver. This can be modelled through MTZ subtour elimination as defined in Equation (22).

4.3.1 Implied constraints

For SMT, we experimented the implied constraint defined in Equation (6).

4.3.2 Symmetry breaking constraints

For SMT, we experiment both symmetry breaking approaches as presented in Equations (4) and (5).

4.4 Validation

4.4.1 Experimental design

The experimental setup consists of two steps: first, we developed a Python package to automate the generation of SMT-LIB code from a high-level interface, allowing us to easily experiment and compare different solvers. More specifically, as solvers we experimented with Z3, cvc5, OpenSMT, SMTInterpol, and Yices 2. Then, to improve the performances on larger instances, we experimented with two different search strategies using the Z3 Python library (`z3py`):

Two solvers approach As SMT solvers do not allow to assign a priority to the variables, this approach attempts to guide the exploration of the search space by alternating two solvers: the first one finds A and the second one finds P given A. In other words, the former decides which courier delivers which package and the latter decides the route taken by each courier, basically solving m different Travelling Salesman Problems.

Local search approach Similarly to the previous one, this approach also uses two solvers to first find A and then P. However, instead of letting the second solver find an optimal solution for P on its own, it is manually guided by performing a local search starting from a trivial solution (i.e., a path that delivers the items ordered by index).

4.4.2 Experimental results

The results of our experiments are presented in Table 3. Analyzing SMT-LIB results, we can observe that all five solvers perform more or less similarly. The best performing is Yices 2 which solves `QF_LIA` logic based on the simplex algorithm [4]. On the other hand, the worst performing is SMTInterpol which relies on Craig interpolation [2]. Furthermore, experiments in `z3py` show that symmetry breaking and implied constraints do not provide significant contribution in improving the results.

By analyzing the two search approaches, we can observe that using two separate solvers have a negligible impact on the final results with mixed effects when using symmetry breaking constraints. Instead, local search enables the model to find a solution in a reasonable amount of time, even for the largest instances.

Table 3: SMT results. Results in **bold** are solved to optimality. Instances that are all solved to optimality have been omitted.

	SMT-LIB (plain)					`z3py`						
Id	Z3	cvc5	OpenSMT	SMTInterpol	Yices 2	plain	plain + SB	plain + IC	2 solvers	2 solvers + SB	2 solvers + IC	Local search
7	228	210	218	372	**167**	174	168	181	**167**	**167**	**167**	**167**
9	**436**	**436**	**436**	437	**436**	**436**	**436**	**436**	**436**	**436**	**436**	**436**
10	**244**	**244**	**244**	381	**244**	**244**	**244**	**244**	**244**	**244**	**244**	**244**
11	–	–	–	–	–	–	–	–	–	–	–	547
12	–	–	–	–	–	–	–	–	–	–	–	435
13	1446	–	–	–	1490	–	–	–	1812	1346	1832	632
14	–	–	–	–	–	–	–	–	–	–	–	1177
15	–	–	–	–	–	–	–	–	–	–	–	1140
16	–	–	–	–	1032	–	–	–	1510	1944	1861	303
17	–	–	–	–	–	–	–	–	–	–	–	1525
18	–	–	–	–	–	–	–	–	–	–	–	917
19	–	–	–	–	–	–	–	–	–	870	–	398
20	–	–	–	–	–	–	–	–	–	–	–	1378
21	–	–	–	–	–	–	–	–	–	–	–	648

5 MIP model

5.1 Decision variables

The MIP models also follow the same idea presented in Section 1. The models are based on the following three variables:

- A binary tensor $X \in \{0,1\}^{(n+1)\times(n+1)\times m}$, where $X[i,j,k] = 1$ if and only if the courier k depart from the i-th delivery point and arrive at the j-th one. This formulation is inspired from the AMPL book [5].

- A binary matrix $A \in \{0,1\}^{n\times m}$, where $A[i,k] = 1$ if and only if the the package i is delivered by the courier k. We observe that these variables are not strictly necessary for the model, but they allowed us to write some constraints in an easier way.

- An auxiliary matrix $u \in \{1,\ldots,n+1\}^{(n+1)\times m}$ that keeps track of the order in which nodes are visited by each courier starting from 1. It is necessary, following the MTZ formulation, for the sub-circuits elimination. The interpretation is that, fixing the courier k, $u[i,k] < u[j,k]$ implies that the node i is visited before the node j by the courier k.

5.2 Objective function

As objective function, defined from the assignment as the maximum distance travelled by any courier, we use the following:

$$\max_{k \in 1 \dots m} \sum_{i=1}^{n+1} X[i,j,k] \cdot D[i,j] \tag{27}$$

5.3 Constraints

We defined the following constraints:

- Constraint to link the variables A and X:

$$\sum_{j=1}^{n+1} X[i,j,k] = A[i,k] \qquad \forall i \in 1 \dots n, \forall k \in 1 \dots m. \tag{28}$$

$$\sum_{i=1}^{n+1} X[i,j,k] = A[j,k] \qquad \forall j \in 1 \dots n, \forall k \in 1 \dots m. \tag{29}$$

- Constraint to guarantee that each package has to be assigned:

$$\sum_{k=1}^{m} A[i,k] = 1 \qquad \forall i \in 1 \dots n. \tag{30}$$

- Constraint for the capacity of each courier:

$$\sum_{i=1}^{n} A[i,k] s[i] \leq l[k] \qquad \forall k \in 1 \dots m. \tag{31}$$

- Constraint to avoid that each courier departs and arrives at the same point:

$$X[i,i,k] = 0 \qquad \forall i \in 1 \dots n+1, \forall k \in 1 \dots m. \tag{32}$$

- Two constraints to ensure that there is one arrival and one departure for each node, respectively. This concerns only the internal nodes because the depot is visited by each courier:

$$\sum_{i \in 1 \dots n+1, k \in 1 \dots m} X[i,j,k] = 1 \qquad \forall j \in 1 \dots n. \tag{33}$$

$$\sum_{j \in 1 \dots n+1, k \in 1 \dots m} X[i,j,k] = 1 \qquad \forall i \in 1 \dots n. \tag{34}$$

- Constraint for the preservation of the flow (if one courier arrives at one node, he departs from the same one):

$$\sum_{i=1}^{n+1} X[i,j,k] = \sum_{i=1}^{n+1} X[j,i,k] \qquad \forall j \in 1 \dots n, \forall k \in 1 \dots m. \tag{35}$$

- Constraints to ensure that each courier starts and ends its route at the depot:

$$\sum_{j=1}^{n} X[n+1,j,k] = 1 \qquad \forall k \in 1\dots m. \tag{36}$$

$$\sum_{j=1}^{n} X[j,n+1,k] = 1 \qquad \forall k \in 1\dots m. \tag{37}$$

- Constraints for MTZ subtour elimination:

$$u[i,k]-u[j,k]+1 \leq n(1-X[i,j,k]) \qquad \forall i \in 1\dots n, \forall j \in 1\dots n+1, \forall k \in 1\dots m. \tag{38}$$

$$u[i,k] \leq X[n+1,i,k]+(n+1)(1-X[n+1,i,k]) \qquad \forall i \in 1\dots n, \forall k \in 1\dots m. \tag{39}$$

$$u[j,k] \geq (u[i,k]+1)X[i,j,k] \qquad \forall i \in 1\dots n, \forall j \in 1\dots n+1, \forall k \in 1\dots m. \tag{40}$$

5.3.1 Implied Model

For the implied model we added one more constraint with the same meaning of Equation (6):

$$\sum_{i=1}^{n} A[i,k] \geq 1 \qquad \forall k \in 1\dots m. \tag{41}$$

5.3.2 Symmetry Model

For the symmetry model, we added the symmetry-breaking constraint related to the number of delivered packages as defined in Equation (4):

$$\sum_{i=1}^{n} A[i,k] \leq \sum_{i=1}^{n} A[i,j], \tag{42}$$

where $k, j \in 1,\dots,m$ with $k < j$ and $l[k] = l[j]$.

5.4 Validation

5.4.1 Experimental design

For the MIP models, we choose to use the solver-independent language AMPL. The workflow is based on the construction of three different models: the initial one, the implied one and the symmetry one.

For reproducibility, we decided to only use open-source solvers such as HiGHS, SCIP, and GCG, provided by the AMPL framework.

5.4.2 Experimental results

Starting from some preliminary experiments, we immediately observed that GCG performs poorly on the first ten instances and therefore decided to discard it from the full experiments.

In Table 4, we present the results of the MIP models. We can observe that the SCIP solver with the addition of the implied and symmetry breaking constraints improves in performance. On the other hand, this behavior is not the same for HiGHS. Nevertheless, the HiGHS solver performances are in general significantly better than the SCIP ones.

Table 4: Results of the SCIP and HiGHS solvers for the three models. Instances without a result have been omitted.

Id	initial-scip	initial-highs	implied-scip	implied-highs	symmetry-scip	symmetry-highs
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**
13	642	726	616	694	526	692
16	–	**286**	–	557	–	320

For the first ten instances, we plotted the graphs about the execution time (in seconds), the number of simplex iterations, and branching nodes explored for these two solvers.

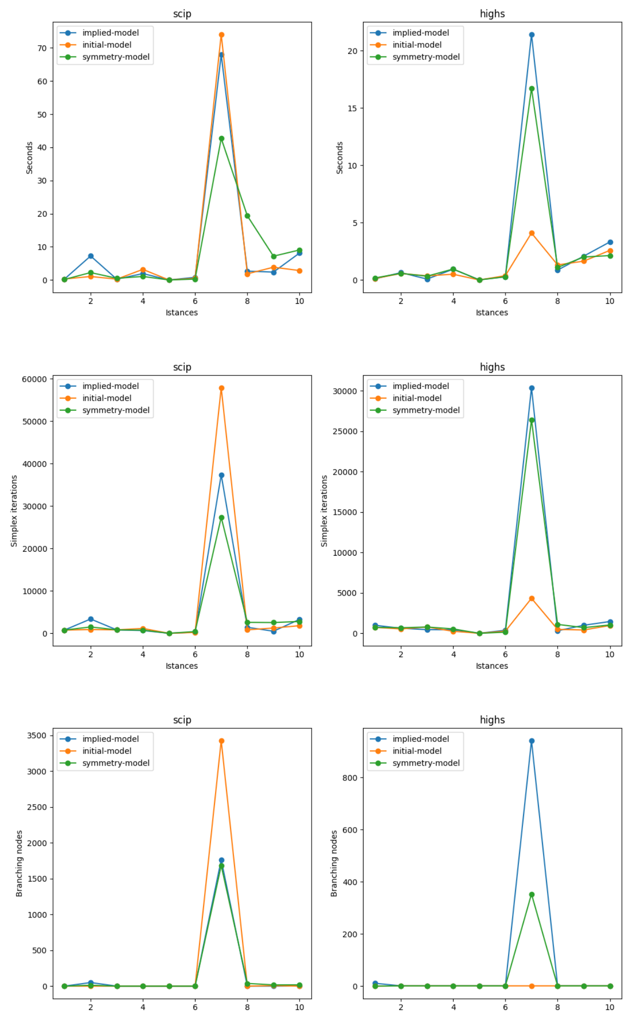


Figure 3: Compared statistics of the performances of the three models, divided by SCIP and HiGHS solvers.

On a theoretical level, SCIP and HiGHS try to initially solve the relaxed-version of the problem (LP) using the revised simplex-method and finding a lower bound for the solution; then, if the solution found is not integer, they start to solve the MIP part using branch-and-cut (SCIP) or branch-and-bound (HiGHS). For the resolution of the sub-problems generated by these two methods, they proceed recursively by applying the same algorithm. From the plots, we can observe that a lower number of simplex iterations and branching nodes corresponds to a faster resolution time. This is in line with our previous observation regarding HiGHS as a better performing solver.

6 Conclusions

References

[1] Bruno De Backer et al. “Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics”. In: *Journal of Heuristics* 6.4 (2000), pp. 501–523. ISSN: 1572-9397. DOI: 10.1023/A:1009621410177.

[2] Jürgen Christ, Jochen Hoenicke, and Alexander Nutz. “SMTInterpol: An Interpolating SMT Solver”. In: *Model Checking Software - 19th International Workshop, SPIN*. Vol. 7385. Lecture Notes in Computer Science. Springer, 2012, pp. 248–254. DOI: 10.1007/978-3-642-31759-0_19.

[3] Martin Desrochers and Gilbert Laporte. “Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints”. In: *Operations Research Letters* 10.1 (1991), pp. 27–36. ISSN: 0167-6377. DOI: 10.1016/0167-6377(91)90083-2.

[4] Bruno Dutertre. “Yices 2.2”. In: *Computer-Aided Verification (CAV'2014)*. Vol. 8559. Lecture Notes in Computer Science. Springer, 2014, pp. 737–744. DOI: 10.1007/978-3-319-08867-9_49.

[5] David M. Gay Robert Fourer and Brian W. Kernighan. In: *AMPL: A Modeling Language for Mathematical Programming*. 1993. URL: https://ampl.com/learn/ampl-book/.

A Implied constraint proof

Claim 1. *If the solution is optimal then every courier has to bring at least one package.*

Proof. Let's assume for the sake of contradiction that we have found an optimal solution where there are one or plus couriers which don't bring any package, calling him k_1. Let's suppose that the courier k_j is the one that cover the maximum distance D_j. If we assign one package, calling i, that k_j brought, to k_1 then, due to the triangular inequality, the two new distances, D_1, travelled by the courier k_1, and D_2, travelled by the courier k_j, are less or equal to D_j, in fact:

$$D_1 = D[depot, i]+D[i, depot] \leq D[depot, i_1]+\cdots+D[i_r, i]+D[i, i_s]+\cdots+D[i_t, depot] = D_j. \tag{43}$$

$$D_2 = D[depot, i_1] + \ldots D[i_r, i_s] + \ldots D[i_t, depot] \leq D[depot, i_1] + \ldots \tag{44}$$

$$\cdots + D[i_r, i] + D[i, i_s] + \cdots + D[i_t, depot] = D_j. \tag{45}$$

Therefore, D_j is not an optimal solution and we have found a contradiction. □