# Mathematical Optimisation

Federica Azzalini, Alessandro Viol

September 2022

## 1 Introduction

In the current global economic system, the logistics sector is one of the toughest markets to compete in. In order for companies to stand out in this saturated market they have to discover new mechanisms to better exploit their resources, that is to become more economically efficient. The reason is that the major source of inefficiency is the incomplete usage of available resources. The authors of the paper found one possible solution in a centrally organized multi-period collaborative vehicle routing organization that enables collaboration between carriers.
The idea behind is for carriers to be able to exchange customers who have to be serviced on a regular basis, facing the problems of time and service consistency. The former guarantees that the arrival time in two different periods differs of maximum $\delta$. The latter ensures that customers are visited by the same collaboration partner throughout the whole planning horizon. Another important aspect they considered is to ensure carriers a minimum market share so that they are more willing to join the collaboration. This creates a collaborative vehicle routing problem with time and service consistency and workload balance.

# 2 The Collaborative Consistent Vehicle Routing Problem With Workload Balance

This optimization problem involves a set of carriers $K$ that has to serve a set of customers $I$ over a planning horizon composed of $P$ periods.
The improvement this model brings to the typical routing problem is that even if each carrier holds a subset of the total customers, carriers can collaborate and share customers if this leads to an increment of profit for every carrier involved. In fact, as a constraint, each carrier's profit must not be lower than the profit they would obtain in absence of coalition.
For carriers to collect the revenue from a customer it is necessary to perform all the requested visits, as each customer has to be visited in a subset or all the periods in $P$. For each visit the service time required to complete it and the quantity to be delivered are known.
Since carriers can exchange their customers, for consistency purposes all the tasks related to the same customer must be performed by the same carrier. Furthermore each vehicle can perform a single route per period with a maximum duration $T_{max}$ and a maximum cumulative load of $Q_{max}$. For time consistency instead, the difference between the arrival time at customer $i$ in two periods, must differ by at most $\delta$ time units.
To avoid the winner-takes-all effect, there is a lower bound to the number of customers that can be assigned to each carrier.
The goal of the problem is to maximize the total profit of the carriers, which is equal to the difference between the total revenue and the total travel costs.

# 3 Mathematical Formulation

$K$ set of carriers
$I$ set of customers
$A_k$ set of customers of carrier $K$
$P$ set of periods
$D$ set of depots ($D_k$ represents depot associated to carrier $K$)
$N$ set of nodes involved in the network $I \cup D$

$Y_{ik}$ binary variable taking value 1 if customer $i$ is assigned to carrier $k$ and 0 otherwise

$X_{ij}^{kp}$ binary variable taking value 1 if node $j$ is visited immediately after node $i$ by carrier $k$ in period $p$ and 0 otherwise

$T_i^p$ non-negative variable representing visit time of node $i$ on period $p$

$L_i^p$ non-negative variable representing cumulative load at node $i$ in period $p$

$V_{kp}^{min}$ integer variable representing the minimum number of vehicles needed to fulfill the demand assigned to carrier $k$ in period $p$

Our goal is to maximizes the total profit, which is the sum of collected revenues reduced by total travel costs. To this purpose, the objective function is $max \sum_{i \in I} \pi_i - \sum_{k \in K} \sum_{p \in P} \sum_{i \in N} \sum_{j \in N} c_{ij} X_{ij}^{kp}$

Here we list the constraints that are necessary for the implementation of our model.

1. $\sum_{k \in K} Y_{ik} = 1 \quad \forall i \in I$ Each customer is assigned to one and only one carrier.
2. $\sum_{i \in N} X_{ij}^{kp} \leq Y_{jk} \quad \forall j \in I, k \in K, p \in P$ Each customer can be visited by a carrier only if it has been assigned to it.
3. $\sum_{i \in N} X_{ij}^{kp} = \sum_{i \in N} X_{ji}^{kp} \quad \forall j \in I, k \in K, p \in P$ Flow balance constraints.
4. $\sum_{j \in I} X_{jD_k}^{kp} \leq V_k \quad \forall k \in K, p \in P$ Each carrier can use up to $V_k$ vehicles.
5. $T_j^p \geq T_i^p + t_{ij} + s_i^p - T_{max}(1 - \sum_{k \in K} X_{ij}^{kp}) \quad \forall j \in I, i \in N, p \in P$ This constraints tracks arrival time at a customer.

6. $T_j^p + t_{ij} \sum_{k \in K} X_{ji}^{kp} \leq T_{max} \quad \forall j \in I, i \in \cup_{k \in K} D_k, p \in P$ Route duration cannot exceed a maximum allowed value $T_{max}$.

7. $L_j^p \geq L_i^p + q_j^p - Q_{max} \left(1 - \sum_{k \in K} X_{ij}^{kp}\right) \quad \forall j \in I, i \in N, p \in P$ This constraints tracks cumulative load at a customer.

8. $L_j^p \leq Q_{max} \quad \forall j \in I, p \in P$ Each vehicle has maximum loading capacity $Q_{max}$.

9. $X_{ij}^{kp} = 0 \quad \forall j \in I, i \in D : i \neq D_k, p \in P, k \in K$ Only vehicles owned by a carrier can exit the depot of that carrier.

10. $X_{ji}^{kp} = 0 \quad \forall j \in I, i \in D : i \neq D_k, p \in P, k \in K$
Only vehicles owned by a carrier can enter the depot of that carrier.

11. $\sum_{i \in N} \sum_{k \in K} X_{ij}^{kp} \geq q_j^p \frac{1}{Q_{max}} \quad \forall j \in I, p \in P$ If a customer requires some goods in a given period, they must be served in that period.

12. $T_i^p = 0 \quad \forall i \in D, p \in P$ This constraint fixes the earliest starting time from the depot to 0.

13. $L_i^p = 0 \quad \forall i \in D, p \in P$ This constraint fixes the cumulative load at the depot to 0.

14. $T_j^{p'} - T_j^{p"} \leq \delta \quad \forall j \in I, p', p" \in P : q_j^{p'} > 0$ and $q_j^{p"} > 0$ Ensures arrival time consistency.

15. $\sum_{j \in I} \pi_j Y_{jk} - \sum_{p \in P} \sum_{i \in N} \sum_{j \in I} c_{ij} X_{ij}^{kp} \geq R_k \quad \forall k \in K$ Each carrier's profit must be equal or higher than the profit obtainable without taking part in the coalition.

16. $\sum_{j \in I} Y_{jk} \geq |A_k| - \alpha_k \quad \forall k \in K$ Ensures workload balance, the number of customers assigned to a given carrier cannot be lower than a minimum value imposed by the carrier.

17. $V_{kp}^{min} \geq \sum_{i \in I} \frac{q_i^p Y_{ik}}{Q_{max}^k} \; \forall k \in K, p \in P$ The minimum number of vehicles $V_{kp}^{min}$ required to fulfill the demand of a carrier $K$ in period $P$.

In addition, we present some other valid inequalities that shrink our search space for quicker results.

- $\sum_{i \in I} \pi_i Y_{ik} \geq R_k \quad \forall k \in K$ The sum of the revenues associated with the customers assigned to a specific carrier k must be greater than the best profit obtainable by this carrier without collaboration.

- $\sum_{i \in I} q_i^p Y_{ik} \leq V_k Q_{max}^k \quad \forall k \in K, p \in P$ This constraint prevents assignments to a given carrier, where the total demand for a period exceeds the maximum demand manageable by the carrier in a single period.

- $\sum_{i \in I} Y_{ik} \leq min \left( |I|, |A_k| + \sum_{k' \in K : k' \neq k} \alpha_{k'} \right) \forall k \in K$ Each carrier can be assigned a maximum number of customers.

- $\sum_{i \in I : q_i^p > 0} Y_{ik} + V_{kp}^{min} \leq \sum_{i \in I \cup D} \sum_{j \in I \cup D} X_{ij}^{kp} \leq \sum_{i \in I : q_i^p > 0} Y_{ik} + V_k \quad \forall k \in K, p \in P$ Limits the number of variables $X_{ij}^{kp}$ being simultaneously active for each carrier $K$ and each period $P$, based on the number of customers that have been assigned to $K$ and that have to be served in period $P$.

# 4 Implementation

To implement and solve the proposed model, we used Gurobi 9.5 with Academic License for Python 3.9. The problem instances have been provided by the authors (Mancini, Simona; Gansterer, Margaretha; Hartl, Richard F. (2020), "Collaborative Consistent Vehicle Routing Problem with Workload Balance (Instances))", Mendeley Data, V1, doi: 10.17632/wwmvnkm46h.1).
They consist of 10 small-sized instances (with 20 customers, 4 carriers and 4 periods) and 10 larger instances (with 50 customers, 8 carriers and 5 periods). The dataset was inconsistent in the representation of the $R_k$ values, so it has been pre-processed to make the data homogeneous. After that, Pandas library has been used to import and manipulate each instance. Instead of considering $K$ and $D$ separately, we joined the two tables. $P$ is generated as an array of appropriate length of strings ["p1", "p2",...]. In the information about clients and depots some values are missing, so we chose the following:

- $Q^k_{max} = Q_{max} \quad \forall k \in K$, as it seems by constraints (7) e (8) that the authors made the same choice.

- $c_{ij} = 0.1 \cdot d(i,j) \quad \forall i,j \in N$, where $d(i,j)$ is the euclidean distance between nodes $i$ and $j$. Since in the given instances costs are missing but coordinates are known for each node, we supposed that costs between two nodes were directly proportional to their distance. Using the distances itself as a costs though, they resulted too high breaking constraint (16), which made the problem unfeasible. To avoid this problem and lower the costs we introduced a scale factor $c$. We chose the value $c = 0.1$, calculated from the solution of instance pr01_20. In fact, for this instance both the revenue of the best known solution and the plot of its routing are known. We have $1743.43 = \sum_{i \in I} \pi_i - \sum_{k \in K} \sum_{p \in P} \sum_{i \in N} \sum_{j \in N} c_{ij} X^{kp}_{ij}$. By letting $c_{ij} = c \cdot d(i,j)$, and substituting it in the latter equation, we get $c = \frac{\sum_{i \in I} \pi_i - 1743.43}{\sum_{k \in K} \sum_{p \in P} \sum_{i \in N} \sum_{j \in N} d(i,j) \cdot X^{kp}_{ij}}$.
  The equation is well defined because from the plot of the routing we can deduce the indexes values $i, j, k, p$ so the variable $X^{kp}_{ij} = 1$, while the values $\pi_i$ are given in the problem instance.

- $t_{ij} = 0.7 \cdot d(i,j) \quad \forall i,j \in N$. The scale factor $t = 0.7$ was chosen by observing the route durations of the best known solution pr01_20 while varying the scale factor. For each carrier $k$ and period $p$, route duration was calculated as the sum of travel and service time on the entire route, checking that it stays within the limit $T_{max}$. We get
  $T_{max} \geq t \cdot \sum_{i \in N} \sum_{j \in N} d(i,j) \cdot X^{kp}_{ij} + \sum_{i \in N} s^p_i \quad \forall k \in K, p \in P$. The choice $t = 0.7$ is not too restrictive or binding, therefore appropriated.

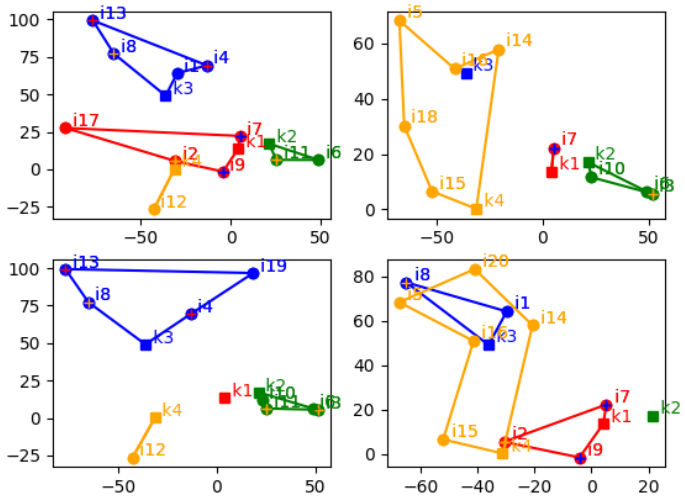
- $s^p_i = 1 \quad \forall i \in D, p \in P$. We assumed to have a minimum service time for

the depots.

Furthermore, we improved the performances by fixing the values of some variables using the following constraints:

- $X_{ii}^{kp} = 0 \quad \forall p \in P, k \in K, i \in N$ keeps a node $i$ to be visited immediately after node $i$.
- $T_i^p = 0 \quad \forall p \in P, i \in I : s_i^p = 0$ if a client doesn't need to be visited in a period $p$, then its visit time is set to 0.
- $L_i^p = 0 \quad \forall p \in P, i \in I : q_i^p = 0$ if a client doesn't need to be visited in a period $p$, then its cumulative load is set to 0.

Here we present the plot of the routing of the optimal solution of pr01_20. The dots represent the clients, the squares represent the depots and the plus symbols represent the original carrier for a costumer. Different colors indicate different carriers.




Plot of the optimal solution for pr01_20

# 5 Matheuristics

To solve the CCVRP, the authors propose a matheuristic MH, where a Mixed Integer Program solver is used to explore very large neighborhoods. The main idea is to fix a part of the solution and re-optimize the remaining sub-problem. The method starts with an initial non-optimal solution found by running the MIP for a short amount of time $TL_{init}$ and using the best found solution. When MIP is not able to find a feasible solution within $TL_{init}$, the initial solution remains the one where each carrier is assigned only its original customers. Then, for all possible pairs of carriers $(k_1, k_2)$, the authors apply a local search mechanism in which they fix all the others customers-to-carriers assignments for all the remaining carriers, while re-optimizing the sub-problem for $k_1$, $k_2$. In this sub-problem, available customers are assigned either to $k1$ or to $k2$. The MIP is run for the sub-problem with a short time limit, TL.
If the best solution obtained is better than the current best, we substitute it and start again exploring the entire set of carrier pairs. Otherwise, we proceed with the exploration of the pairs which have not been selected so far. Once the solution stops improving after having explored all the pairs of carriers, the procedure ends.
For large problem instances, the authors have developed a slightly different version of MH, the MH*, in which the whole set of possible carrier pairs is not exhaustively explored but at each micro-iteration two random carriers are jointly re-optimized. The algorithm stops after $N_{noimp}$ unsuccessful micro-iterations. This version of the algorithm is useful when the number of carriers is large. Otherwise, when the number pairs is limited it is much more effective to explore all of the pairs exhaustively.

# 6 Iterated Local Search

In order to move toward a different area of the solution space to escape local minima, the authors designed an approach based on ILS. At each iteration of the ILS, MH*, used as the local search operator, is run. Once we find a local minimum, we perform a perturbation in the customer to carrier assignment and restart MH* again. The perturbation consists of randomly drawing $N_{pert}$ customers and assigning them to a different carrier. This phase plays a crucial role, if we blindly move too far in the solution space by changing too many customer assignments, we risk restarting from a poor solution and facing a long computational time to move towards good quality solutions. But if the perturbation is too small, we risk remaining trapped in the same local minimum. If the solution obtained after the perturbation is infeasible, we discard it and perform another perturbation. The procedure terminates after a given number of restarts, $N_{iter}$.

# 7 Scalability Analysis

Here we perform an analysis on the scalability of our model and algorithms. We tested all of them on both small and big instances problems. In the small instances case, we obtain a model with 9504 variables and 7482 constraints (7522 in the case with valid inequalities). In the large instances case, we obtain a model with 135580 variables and 66516 constraints (66612 in the case with valid inequalities). We notice that the number of variables and constraints has considerably increased. In fact the number of $X_{ij}^{kp}$ variables increases quadratically with the number of nodes. Therefore a small increase in the dimensions of the instance causes a massive increase in complexity. Gurobi's MIP solver isn't able to solve the problem in reasonable time for the 50 clients instances. MH* and ILS can instead reach good feasible solutions. We can observe that ILS algorithm can obtain an average improvement of 0.77% at the expense of a small increase timewise, which makes it a good alternative to MH*.

# 8 Performances

To run our algorithms we used a laptop with a Intel-i7-7700HQ cpu running at an average of 3.1 GHz and 16 GB of RAM. Firstly, we make a comparison between MIP performance with and without the valid inequalities proposed in the paper. We limit the execution time to 10 minutes for simplicity. The MIP with the valid inequalities reaches optimal solutions way before the end of ten minutes, but for the MIP without the valid inequalities ten minutes aren't enough to complete the search. Then we compare MH and ILS performances against MIP with valid inequalities for small instances. Because ILS isn't deterministic, it's executed 10 times and the average performances are considered. We can see that even if MH not always reaches an optimal solution, it reaches good feasible solutions in a much shorter time than MIP. ILS slightly improves MH's solutions but it's slower than MIP, therefore is not convient for smaller instances. Lastly, we compare MH* with ILS for big instances. These last two algorithms, which aren't deterministic, are executed 5 times. We provide the average values. What we can observe is that MH* is able to obtain feasible solutions in reasonable times, while ILS improves the solution value by 0.77% in almost the same time.

| | MIP | | | MIP + Valid Inequalities | | |
|---|---|---|---|---|---|---|
| Instance | Solution Value | Gap (%) | Time (s) | Solution Value | Gap (%) | Time (s) |
| Pr01_20 | 1731,14 | 2,80 | **600,00** | 1744,98 | 0,00 | 66,27 |
| Pr02_20 | 1875,27 | 2,31 | **600,00** | 1882,18 | 0,00 | 93,16 |
| Pr03_20 | 1732,56 | 1,01 | **600,00** | 1732,57 | 0,00 | 24,76 |
| Pr04_20 | 1904,45 | 1,21 | **600,00** | 1910,04 | 0,00 | 46,96 |
| Pr05_20 | 1884,14 | 1,90 | **600,00** | 1884,14 | 0,00 | 43,20 |
| Pr06_20 | 1854,08 | 1,92 | **600,00** | 1857,38 | 0,00 | 56,75 |
| Pr07_20 | 1667,93 | 2,18 | **600,00** | 1670,11 | 0,00 | 44,58 |
| Pr08_20 | 1747,16 | 1,22 | **600,00** | 1747,16 | 0,00 | 25,15 |
| Pr09_20 | 2018,92 | 1,62 | **600,00** | 2021,33 | 0,00 | 163,65 |
| Pr10_20 | 1623,28 | 1,27 | **600,00** | 1623,38 | 0,00 | 24,99 |

MIP without valid inequalities compared to MIP with valid inequalities

| | MIP | | | MH | | | ILS | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Solution Value | Gap (%) | Time (s) | Solution Value | Gap (%) | Time (s) | Solution Value | Gap (%) | Time (s) |
| Pr01_20 | 1744,98 | 0,00 | 66,27 | 1743,43 | -0,09 | 24,68 | 1743,43 | -0,09 | 70,12 |
| Pr02_20 | 1882,18 | 0,00 | 93,16 | 1877,34 | -0,26 | 17,91 | 1879,41 | -0,15 | 78,06 |
| Pr03_20 | 1732,57 | 0,00 | 24,76 | 1724,63 | -0,46 | 15,90 | 1724,63 | -0,46 | 68,89 |
| Pr04_20 | 1910,04 | 0,00 | 46,96 | 1909,84 | -0,01 | 12,33 | 1909,84 | -0,01 | 60,38 |
| Pr05_20 | 1884,14 | 0,00 | 43,20 | 1884,14 | 0,00 | 18,63 | 1883,65 | -0,03 | 66,96 |
| Pr06_20 | 1857,38 | 0,00 | 56,75 | 1857,38 | 0,00 | 19,56 | 1857,38 | 0,00 | 69,10 |
| Pr07_20 | 1670,11 | 0,00 | 44,58 | 1665,33 | -0,29 | 13,88 | 1667,86 | -0,13 | 66,27 |
| Pr08_20 | 1747,16 | 0,00 | 25,15 | 1747,16 | 0,00 | 14,07 | 1747,16 | 0,00 | 65,89 |
| Pr09_20 | 2021,33 | 0,00 | 163,65 | 2021,33 | 0,00 | 27,25 | 2021,83 | 0,00 | 280,87 |
| Pr10_20 | 1623,38 | 0,00 | 24,99 | 1623,38 | 0,00 | 11,71 | 1623,38 | 0,00 | 57,38 |

MIP with valid inequalities compared with both MH and ILS

| | MH* | | ILS | | |
|---|---|---|---|---|---|
| Instance | Solution Value | Time (s) | Solution Value | Time (s) | Gap (%) |
| Pr01_50 | 4706,00 | 128,00 | 4804,41 | 426,78 | 2,05 |
| Pr02_50 | 4139,68 | 113,64 | 4120,00 | 495,57 | -0,48 |
| Pr03_50 | 4263,73 | 190,09 | 4265,58 | 487,45 | 0,04 |
| Pr04_50 | 4567,45 | 128,15 | 4649,11 | 490,23 | 1,76 |
| Pr05_50 | 4604,35 | 717,58 | 4618,99 | 507,16 | 0,32 |
| Pr06_50 | 4157,73 | 184,42 | 4175,01 | 513,26 | 0,41 |
| Pr07_50 | 4159,76 | 200,62 | 4160,42 | 430,44 | 0,02 |
| Pr08_50 | 4605,52 | 165,54 | 4619,89 | 486,12 | 0,31 |
| Pr09_50 | 4469,28 | 219,24 | 4537,82 | 453,39 | 1,51 |
| Pr10_50 | 4149,23 | 242,02 | 4225,11 | 458,45 | 1,80 |

MH* compared to ILS for large instances

# 9 Conclusion

We showed that for small instances both MIP and MH reach good solutions in reasonable times. To be precise, MIP reaches better solutions but if we are looking for sub-optimal solutions MH is much faster. For large instances both MH* and ILS are viable algorithms, MH* is faster but reaches worse solutions while ILS is slower but provides better solutions. It's worth noticing that apart from the missing values for the parameters discussed in the implementation paragraph, there are also some small mistakes spread through the paper. For instance, the variable $T_i^p$ represents the arrival time at a node instead of a customer. The constraint (8) is true for all $p \in P$ not only for all $j \in I$, while the constraint (19) of the paper, which we omitted, is redundant because is the same as constraint (14). We also noticed that the valid inequality (17) is actually a required constraint. Overall it was a very interesting but challenging study.