Università di Pisa

Dipartimento di Informatica

Corso di Laurea Magistrale in

Data Science and Business Informatics

Model-Driven Decision-Making Methods

By

Francesco Santucciu

Anno accademico 2022/2023

Problem Description

A logistic company must organize the short-haul distribution of goods from its central depot to a set of customers in the same city. This can be represented by a directed graph with a node for the depot and one for each customer, and two directed arcs for each pair of nodes, in both directions, labeled with the distance between the two (which can be different between the two directions due to one-way streets). The company has an unlimited supply of vehicles stationed at the depot and each can carry a given maximum amount of goods in weight and volume. It is assumed that the goods are “flexible” enough so that they can always be arranged in the vehicle provided that their total weight and volume does not exceed the corresponding capacity. Each customer requires a certain amount of goods, measured by their total weight and total volume (the two not being directly proportional), and has to be serviced by exactly one vehicle. However, the company can outsource individual customers at a very high cost per weight (w.r.t. the cost per weight corresponding to using its own vehicles). The problem is to serve all the customers at minimum total cost, which is (proportional to) the total distance traveled by all the vehicles.

The relevant facts of the problem that i need to formalize are the following:

1) directed graph with a node for the depot and one for each customer, and two directed arcs for each pair of nodes, in both directions, labeled with the distance between the two which can be different
2) unlimited supply of vehicles
3) each vehicle has a limit capacity in volume and weight
4) customer requires a certain amount of goods, measured by their total weight and total volume
5) a customer has to be serviced by exactly one vehicle
6) the company can outsource individual customers at a very high cost per weight

Model Formulation

Constants of the model

Graph of the logistic network:
It is a directed connected graph $G(N, E)$ where N is the set of the nodes, each for one customer and one for the depot, and E is the set of the arcs. Every node has two kind of arcs, incoming ones and outgoing ones, but every node is connected with every other node

Distance constants:
Each arc which goes from i to j has an associated distance $d_{ij} > 0$ and we can formalize this set as following:

$$D = \{ d_{ij} : d_{ij} \neq d_{ji} \lor d_{ij} = d_{ji} \}$$

Distance-cost constant:
This constant, $c > 0$, represents the unitary cost per kilometer the company's vehicle should travel.

Outsourced cost constants:
It is a fixed cost, $f > 0$, which represents the unitary cost of transportation per kilogram of weight.

Demands of the customers:
Instead of considering the various different items required by every customer, we can group them with the summation of the volumes/weights of the single items required by each customer.
In other words, the demands are formalized in two sets:

- one for the volumes: $V = \{v_i : v_i \in \mathbb{R}_+\}$ one for each customer
- and one for the weights: $W = \{w_i : w_i \in \mathbb{R}_+\}$ one for each customer

Company's vehicles:
The logistic company has a set K of vehicles where $K = \{k: k = 1, 2, 3,..., h\}$ where these k vehicles are limited between $k_{min} \leq h \leq \infty$ and k_{min} is the minimum number of vehicles needed to have at least a feasible solution. The fleet of vehicle is homogeneous and each vehicle has two associated constants:

C_V limit capacity in terms of volume
C_W limit capacity in terms of weight
(it is assumed that $\nexists v_i \geq C_V$ and $\nexists w_i \geq C_W$)

Since there are an unlimited number of vehicles, an upper bound is needed and it is equal to the cardinality of the demand.

$$|K| = |V| = |W|$$

Variables of the model

Route variables:
This kind of variable decides if an arc is selected to be part of the path for the company's vehicles. The formalization is the following:

$\{x_{ij}: (i,j) \in (NxN)\}$

$x_{ij} = 1$ if the arc x_{ij} belongs to the route, 0 otherwise

Flow variables:
These variables, v_{ij} and w_{ij} ,represent the flow of the load, respectively in terms of volume and weight, transported between the arc (i,j).

$$v_{ij}, w_{ij} \in \mathbb{R}_+.$$

Vehicle variables:
For the company's vehicles, these variables, z_{ik} , put in a relationship the vehicle k and each the customer i.

$z_{ik} = 1$ if the vehicle k serves the customer i, 0 otherwise.

Regarded the outsourced vehicles, it is just needed a binary variable:

$l_i = 1$ if the customer i is served by outsourcing, 0 otherwise.

Objective function

$$min\ \ c * \sum_{(i,j)\in E} (d_{ij} * x_{ij})\ +\ f * \sum_{i\in N\setminus\{0\}} (w_i * l_i)$$

This objective function is composed with two kind of costs,

- the company's cost for the transportation of the goods: $c\ * \sum_{(i,j)\in E} (d_{ij} * x_{ij})$, which is the unitary cost per kilometer multiplied by the distances associated to the arcs selected to be part of the path,
- and the outsourcing cost: $f\ * \sum_{i\in N\setminus\{0\}} (w_i * l_i)$ which is the unitary fixed cost per kilogram multiplied by the weight of the demanded goods of the customer served with outsourcing.

Set of constraints

1) $\sum_{(i,j)\in BS(j)} x_{ij} = 1 \qquad \forall j\ \in\ N\setminus\{0\}$

2) $\sum_{(i,j)\in FS(j)} x_{ji} = 1 \qquad \forall j\ \in\ N\setminus\{0\}$

These constraints allow the model to select just one incoming arc and just one outgoing arc for every node, except the depot node.

($BS(j)$ stands for backward start of j and $FS(j)$ stands for frontward start of j)

3) $\sum_{(i,0)\in E} x_{i0} \leq |K|$

4) $\sum_{(0,i)\in E} x_{0i} \leq |K|$

The depot (or node 0) is a special node which is the only one to be allowed to accept more than one incoming arc and more than one outgoing arc. It is modeled to be equal to K (the number of the available company's vehicles).

5) $\sum_{(j,i)\in BS(i)} v_{ji} - \sum_{(i,j)\in FS(i)} v_{ij} = v_i \qquad \forall j\ \in\ N\setminus\{0\}$

6) $\sum_{(j,i)\in BS(i)} w_{ji} - \sum_{(i,j)\in FS(i)} w_{ij} = w_i \qquad \forall j\ \in\ N\setminus\{0\}$

7) $0 \leq v_{ij} \leq C_V * x_{ij} \qquad \forall(i,j)\ \in\ E$

8) $0 \leq w_{ij} \leq C_W\ * x_{ij} \qquad \forall(i,j)\ \in\ E$

9) $\sum_{(j,0)\in BS(0)} v_{j0} - \sum_{(0,j)\in FS(0)} v_{0j} = - \sum_{j\in N\setminus\{0\}} v_i + \sum_{j\in N\setminus\{0\}} (l_i * v_i)$

10) $\sum_{(j,0)\in BS(0)} w_{j0} - \sum_{(0,j)\in FS(0)} w_{0j} = - \sum_{j\in N\setminus\{0\}} w_i + \sum_{j\in N\setminus\{0\}} (l_i * w_i)$

The constraints 5 and 6 assure that every customer i receives its demand in terms of volume and weight, while the constraints 7 and 8 are just bounds for the variables v_{ij} and w_{ij} (which are the flow variables, not the volume/weight constants v_i and w_i).

The constraints 9 and 10 ensure, both in terms of weight and volume, that what leaves the depot (the outgoing flow of the depot) is exactly equal to the negative total demand except the demand served by outsourcing, and furthermore, it is guaranteed that the vehicles return empty (the incoming flow of the depot).

11) $\sum_{k\in K} z_{ik} = 1 - l_i \qquad \forall i \in N\setminus\{0\}$

This assures that just one vehicle k is allowed to serve the customer i, and it is not allowed the overlapping between outsourcing vehicles and company's vehicles.≤

12) $\sum_{k\in K} z_{0k} \leq \sum_{(0,i)\in E} x_{0i}$

Constraint 12 is used to model the fact that the path of every vehicle should start from the depot

13) $\sum_{i\in N\setminus\{0\}} v_i * z_{ik} \leq C_V \qquad \forall k \in K$

14) $\sum_{i\in N\setminus\{0\}} w_i * z_{ik} \leq C_W \qquad \forall k \in K$

The summation of the demands (both in terms of volume and weight) of every customer assigned to the path do not exceed the capacity of the vehicle.

15) $b_{ij}^{k} \geq z_{ik} + z_{jk} - 1 \qquad \forall (i,j) \in N\setminus\{0\}, \forall k \in K$

16) $b_{ij}^{k} \leq z_{ik} \qquad \forall (i,j) \in N\setminus\{0\}, \forall k \in K$

17) $b_{ij}^{k} \leq z_{jk} \qquad \forall (i,j) \in N\setminus\{0\}, \forall k \in K$

This is the formulation of a logic AND, in this way b_{ij}^{k} is forced to be 1 if both the customers i and j belong to the path of the vehicle k.

18) $x_{ij} + x_{ji} \leq \sum_{k \in K} b_{ij}^{k}$ $\quad\quad \forall (i, j) \in N \backslash \{0\}, \forall k \in K$

18 assures that if either the arc (i, j) or arc (j, i) are in a feasible solution, then the customers i and j are served by just one vehicle. Basically, in conjunction with 15, 16 and 17, these are a sort of linking constraint between the vehicles and the arcs

19) $z_{ik} \leq z_{0k}$ $\quad\quad \forall i \in N, \ \forall k \in K$

19 says that if a customer is served by vehicle k, then that vehicle must be filled by the depot otherwise it cannot serve any customer.

20) $z_{ik} \in \{0, 1\}$ $\quad\quad \forall k \in K, \ \forall i \in N$

21) $x_{ij} \in \{0, 1\}$ $\quad\quad \forall (i, j) \in E$

22) $l_{i} \in \{0, 1\}$ $\quad\quad \forall i \in N \backslash \{0\}$

20, 21 and 22 are just integrality constraints, more precisely they must be binary

Performance of the model

Methodology followed and Performance

This study aims to conduct a thorough analysis of the solver's performance through a structured approach. To achieve robust and variable results, three datasets were utilized, generated using the script Generate_data.py, each characterized by a unique random seed. Constants related to vehicle capacities were kept identical across all datasets, ensuring uniformity in test conditions. Furthermore, the solver used is CBC (even for the reoptimization chapter at page 13), without any parameter specifications except for a time limit.

Furthermore, it should be noted that constants pertaining to transportation and outsourcing costs are directly determined by the seed used. This implies that each dataset, while sharing the same vehicle capacities, exhibits unique variations in costs due to the diversity of seeds.

For each dataset, customer cardinality was incrementally increased in 5 stages, ranging from 5 customers to 40. Thus, each of the three main datasets transformed into a series of sub-datasets, each characterized by a different customer size. This methodology allows for a detailed analysis of the model's performance in terms of scalability, elucidating its behavior in expanding scenarios.

For each increment in the number of customers, I meticulously recorded the solver's computational time, along with the assessment of any existing gap. This systematic tracking not only provides insights into the temporal efficiency of the solver but also enables the identification and evaluation of any disparities between the model's solution and the optimal solution, referred to as the "gap. The table below shows the obtained results:

	5	**10**	**20**	**30**	**40**
5311	time = 0.05 gap = 0	time = 0.21 gap = 0	time = 93.62 gap = 0	time = 282.10 gap = 0.01	time = 561.21 gap = 0.03
3649	time= 0.03 gap = 0	time = 1.06 gap = 0	time = 283.84 gap = 0.04	time = 591.91 gap = 0.14	time = 557.26 gap = 0.23
7535	time = 0.03 gap = 0	time = 0.35 gap = 0	time = 24.16 gap = 0	time = 264.50 gap = 0.02	time= 222.81 gap = 0.05

(time in seconds)

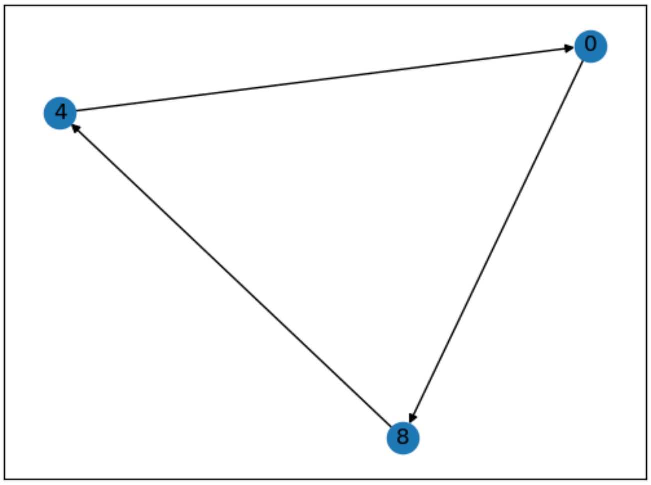
The solver exhibits limited scalability as the number of customers increases incrementally. Notably, a substantial disparity in computational time emerges between scenarios with 10 and 20 customers, and this gap further intensifies with an increase in the customer count.

To substantiate these observations, additional experiments were conducted with 50 customers, utilizing a predefined time limit of 1 hour. In these instances, the solver encountered difficulties, either becoming unresponsive or failing to produce a feasible solution within the stipulated time. It's important to note that while the solver may eventually find an admissible solution, it exceeds the one-hour time limit. This underscores the substantial impact of customer volume on the solver's performance, particularly in the context of larger datasets.

Possible Solver Improvements

Since the primary challenge of this solver is scalability, addressing this issue involves examining the presence of different patterns related to vehicle selection. These patterns often lead to the same solution in terms of the solver's objective function value, even though the vehicle route patterns may vary due to different patterns resulting in the same solution. Indeed, by using the **show_graph** function and adjusting the random seed in the solver options, the same solution can be achieved, but with each vehicle potentially assigned to different routes or customers. This understanding of variability in vehicle assignments could pave the way for refining the solver's scalability and addressing its current limitations.

As an illustration, when we execute the solver using the dataset with a random seed of 7353 and 10 customers, using the default random seed of the solver, we can observe the following route for vehicle 3:



if we change the solver’s random seed to 481, this route will be assigned to the vehicle 8

Optimization

To solve the symmetry issue of the previous model, it proposed a reformulation of the same problem where instead of focusing on finding the best set of edges and assigning them to a possible vehicle, now it takes into consideration the set of possible routes and through a column generation approach to generate these columns.

At the beginning the model is given a set of initial patterns, where every pattern is the path from depot to a single node and from that node to the depot. In other words we have a vehicle for each customer.

To accelerate computation for medium-large and large instances of the problem, a heuristic can be applied to solve it on the subproblem level. While this approach yields a solution, it may not always guarantee optimality; nevertheless, it provides a strong approximation.

Reduced Master Problem

As anticipated before, this model gives us the best set of paths that minimize the transportation cost, taking into account that serving the customers through outsourcing is a possible choice.

Constants of the model:

1) c_r cost of the route r. With the initial set of patterns these constants are calculated using the Distance-cost constant (in the previous modelation it is denoted by c) times the sum of the distances of the arcs $\{ (0, i), (i, 0) \}$. what's more there is a route where no node is satisfied and it has a cost equal to 0
2) f which represents the outsourcing cost.
3) w_i which is the demand of the customer i in terms of weight
4) A matrix A where each row is a pattern of the route and each column represents a customer. So a_{ri} if it is equal to 1 it means that for the route r the customer i is served by one of our vehicles, 0 otherwise

Variables of the model:

1) x_r is a variable that is equal to 1 if the route is selected by the model or 0 otherwise for every route r belonging to the set of routes R
2) l_i is a variable that is 1 if the customer i is served by outsourcing, 0 otherwise

Objective function:

$$min \sum_{r\in R}(c_r * x_r) + f * \sum_{i\in V*}(w_i * l_i)$$

$$(\textit{where}\ V* = \{i \in V:\ i \neq 0\})$$

This function is composed by two types of costs:

- the cost of each route that one vehicle has to take: $\sum_{r\in R}(c_r * x_r)$
- and the outsourcing cost for each customer: $f * \sum_{i\in V*}(w_i * l_i)$

Set of constraints:

1) $l_i + \sum_{r\in R}(a_{ri} * x_r) \geq 1 \qquad \forall i \in V*$

   This constraint says that every customer must be satisfied or by the outsourcing service l_i, or by a pattern x_r

2) $\sum_{r\in R} x_r \leq |R|$

   with this constraint the number of activated pattern is bounded to be at most the cardinality of the set of patterns

Sub-Problem

The set of route R is updated iteratively by the column generation approach.
In order to be able to generate a possible route, we can solve a integer linear problem that can give us a path which has the minimum cost and at the same time it can reduce the cost of the Reduced master problem. To do so, from the reduced master problem, we take the dual variables (we denote these variable as $\pi_i\ \forall i \in V*$) of the first set of constraints

$$l_i + \sum_{r\in R}(a_{ri} * x_r) \geq 1 \qquad \forall i \in V*$$

and through these variables, we can build a prize collecting traveling salesman problem. Since the π_i are given by the reduced master problem, we can treat them as constants.

Variables of the Sub-problem:

1) x_{ij} is a binary variable which denotes if an arc in the graph is chosen or not.
2) y_i will be equal to 1 if a customer is chosen to be part of the path and 0 otherwise
3) w_{ij} and v_{ij}, as in the previous formulation, are flow variables used for the Miller-Tucker-Zemlin constraint, a variant of the GSEC, which serves as a subtour-elimination constraint.

Formulation of the Sub-Problem:

$$min\ \ c * \sum_{(ij)\in A} (d_{ij} * x_{ij}) - \sum_{i\in V*} (\pi_i * y_i)$$

(where c is the distance-cost constant)

subject to:

1) $y_0 = 1$

2) $\sum_{(i,j)\in BS(j)} x_{ij} = y_i \qquad \forall i \in V*$

3) $\sum_{(i,j)\in FS(j)} x_{ji} = y_i \qquad \forall i \in V*$

4) $\sum_{(j,i)\in BS(i)} v_{ji} - \sum_{(i,j)\in FS(i)} v_{ij} = (v_i * y_i) \qquad \forall i \in V*$

5) $\sum_{(j,i)\in BS(i)} w_{ji} - \sum_{(i,j)\in FS(i)} w_{ij} = (w_i * y_i) \qquad \forall i \in V*$

6) $0 \le v_{ij} \le C_V * x_{ij} \qquad \forall (i, j) \in A$

7) $0 \le w_{ij} \le C_W * x_{ij} \qquad \forall (i, j) \in A$

8) $\sum_{(j,0)\in BS(0)} v_{j0} - \sum_{(0,j)\in FS(0)} v_{0j} = -\sum_{i\in V*} (v_i * y_i)$

9) $\sum_{(j,0)\in BS(0)} w_{j0} - \sum_{(0,j)\in FS(0)} w_{0j} = -\sum_{i\in V*} (w_i * y_i)$

10) $\sum_{i\in V*} (v_i * y_i) \le C_V$

11) $\sum_{i\in V*} (w_i * y_i) \le C_W$

where the constraint 1 imposes that the depot must always be selected, 2 and 3 say that if a node y_i is selected, then it must have one in-coming arc and 1 out-going arc. From 4 to 9 there are the MTZ constraints which impose that a path must start from the depot and must end there and with 10 and 11 we cannot violate the capacity limit (C_V and C_W respectively for volume and weight).

Introduction to a heuristic approach

Given the likelihood of the solver becoming stuck with very large sub-problems, it's essential to find ways to speed up the solution process. One effective method involves providing a set of activated nodes, which can be achieved by maximizing the second component of the objective function.

$$max \sum_{i \in V*} (\pi_i * y_i)$$

Utilizing the 10th and 11th constraints and imposing to have more than one node activated (except the depot), a multidimensional knapsack problem is constructed. This approach enables us to leverage the solution to find a path that minimizes transportation costs, utilizing the nodes derived from solving the knapsack problem.

While this method may not guarantee an optimal solution, it serves as a robust bound. As detailed in the results chapter, it significantly enhances computational speed compared to approaches lacking this optimization.

Results

To evaluate the solver's effectiveness in terms of both time taken and precision in converging towards an optimal solution, a series of tests was conducted. Initially, tests were carried out without any time constraints. Subsequently, they were repeated with time limits set at 5/10 minutes. The assessment considered the solver's speed in reaching an optimal solution, taking into consideration both execution time and the calculated gap, as per the following formula:

$$(\frac{best\ solution - actual\ solution}{best\ solution}) * 100$$

First, to verify the correctness of the new model, comparisons were made in terms of both results and time with the old model using the same instances outlined in the chapter titled 'Performance of the Model' (page 8).

		Old model		**New mode**		**New model (heur)**	
ID	**Size**	**Time**	**Result**	**Time**	**Result**	**Time**	**Result**
5311	40	550.52	3361.08	113.75	3327.47	2.11	3949.4
5311	30	283.28	2382.57	37	2382.57	0.87	2914.01
5311	20	146.62	1639.43	10.31	1639.42	0.63	1913.14
3649	40	543.44	2911.19	472.96	2760.92	12.69	3008.21
3649	30	584.97	2142.77	129.98	2136.75	8.96	2355.61
3649	20	283.49	1392.06	22.47	1389.52	2.07	1549.78
7535	40	247.06	11721.23	105.21	11481.73	9.23	11912.37
7535	30	265.49	8574.9	25.04	8574.9	4.68	8744.75
7535	20	132.16	6244.74	10.92	6244.74	1.69	6394.62

(time in seconds, they can be different from previous chapter because of different computer)

From this table, it is evident that the new model outperforms the previous one, achieving faster convergence for problems with a size of 40. While results are essentially identical for smaller sizes, the new model's significantly faster performance is notable

Then, tests were carried out for the standard version (non-heuristic):

ID	**Size**	**Time (seconds)**	**Result**	**Result (5 min)**	**Relative gap**
765	50	919.08	245.46	258.39	-5.26
7289	50	486.35	2576.48	2577.99	-0.05
6774	50	1443	3715.25	3860.47	-3.90
2891	75	5075.6	10605.63	11324.32	-6.77
6576	75	2255.81	18695.44	19745.11	-5.61
9	75	6060.05	5200.3697	6316.19	-21.45

(difference between standard and standard with time limit)

As it can be seen, for medium-small instances of the problem, there is a significant disparity in terms of time taken, albeit with a relatively contained relative gap most of the time. In order to examine the solver's behavior with larger instances and considering time constraints, tests were conducted comparing the standard version, limited to 5 minutes of execution, with the heuristic version, without any time limit.

ID	**Size**	**Result** **(5 minutes)**	**Time (seconds)**	**Result** **(heur)**	**Relative gap**
765	50	258.39	12.76	299.29	-15.82
7289	50	2577.99	3.86	2959.78	-14.80
6774	50	3860.47	6.07	4247.93	-10.03
2891	75	11324.32	4.24	13101.63	-15.69
6576	75	19745.11	45.61	19729.36	0.07
9	75	6316.19	41.01	6196.28	1.89
5381	100	9705.97	24.16	9901.93	-2.01
1942	100	17198.17	66.07	15528.88	9.70
9569	200	Unsolved	796.07	15823.67	NA
6901	200	Unsolved	704.05	8676.6	NA

(difference between standard with time limit and heuristic approach without time limit)

ID	**Size**	**Time** **(seconds)**	**Result**	**Result** **(10 minutes)**	**Relative gap**
9569	200	796.07	15823.67	15823.67	0
6901	200	704.05	8676.6	8778.92	-1.17
8755	300	5158.84	16185.12	18830.16	-16.34
3149	300	14803.62	30290.49	42409.075	-40.00

(difference between heuristic approach with and without time limit)

As it is possible to see, the reformulation made to the previous solver helped handle and sometimes solve accurately larger problems much faster than before. In contrast, the old solver often got stuck on smaller versions of the same problem.