Mathematical Optimization a.a. 20/21

Vehicle routing problem with load dependent travel times

Testa Renzo, Tosato Niccolo

6/23/2022

Abstract

This paper is based on the work of Fontaine, P. (2022). We include the minimum information and problem definition from the cited paper to allow for an independent reading, while focusing on the differences of assumptions and implementation of our work.

Problem definition

The vehicle routing problem with load-dependent travel times for cargo bicycles

The problem addressed in this study stems from the increased attention to the concept of two-tier city logistics systems. One main idea in these systems is to use environmental friendly vehicles on the second tier for the last mile delivery, for which a promising option is the cargo bicycle. Compared to classical bicycles, cargo bicycles have a trunk to transport parcels and cargo. Depending on the model, this trunk can be up to 2 m^3 large and can be loaded with up to 200 kg. Furthermore, they are equipped with a small electric engine to support the cyclist. Delivery companies still need to cluster the customers into tours and define the sequence of visiting each customer from the depot. This problem, known as the vehicle routing problem (VRP), has been studied in many variants in the literature but, up till now, all models assume a travel time that is independent of the vehicle load. So far, only load-dependent costs and emissions have been considered in VRPs. Since the VRP is mostly used for scheduling trucks or delivery vans, the effect of the load on the speed is also negligible. However, when using cargo bicycles for the final distribution of goods, the mass of the load is an important factor. Even in flat cities, cargo bicycles cannot drive with full speed if they are fully loaded. Considering further the gradient of the roads, the influence is even stronger. Fontaine, P. (2022) introduced the vehicle routing problem with load dependent travel times (VRPLTT) which, differently from the classical VRP, considers travel times that depend on the mass of the load of the vehicle and the gradient of the street. Accordingly, Fontaine, P. (2022), proposes how to calculate load-dependent travel times and formally defines the VRPLTT, introducing a new mixed-integer programming formulation. We performed an extensive numerical study based on the work of Fontaine, P. (2022) and we extended the test cases to include a thorough scalability analysis and to investigate the importance of considering more detailed path information for effective load-dependent travel times calculations.

The cargo bicycle

As defined by European Union regulations, an e-bike is a pedelec (pedal electric cycle) if the total power of the engine does not exceed 250 W, if the engine support cuts after 25 km/h, and if it does not drive without human power support.

It is assumed that a well trained human can generate a power of 150 W and that a normal person can generate a power of 100 W over a longer period when cycling. This makes the total available power respectively $\overline{P} =$ 400 W and $\overline{P} = 350$ W. In any case, the available power is always a mix of cyclist and battery power, while

it is assumed that the primary source of energy is the cyclist. This means that the battery is only used if the needed power is higher than the power generated by the cyclist.

One example of a cargo bicycle is the Armadillo by Velove (2018), which is currently used by DHL as the Cubicycle. The bicycle has a mass of 60 kg and can carry a load of up to 150-200 kg, depending on the setting. Including the mass of the driver (we assume 80 kg), this results in a total mass between 140 kg (if the bicycle is empty) and 340 kg (if it is fully loaded).

Figure 1: DHL cubicycle

Problem implementation

Following Fontaine, P. (2022), the VRPLTT is defined on a graph $G = (N, E)$. $N = 0, ..., n$ is the set of nodes with 0 being the depot and E is the set of edges (i, j) with i, j $\in$ N. Then the set of customers is given by $N0 = 1, ..., n$. Each customer i $\in N_0$ has a demand with mass q_i, and a time window $[a_i, b_i]$ during which she has to be served. The distance of each edge (i, j) is defined as d_{ij} and the service time at a customer as s_i. Similar to the classical VRP, the goal is to find an efficient tour within the given restrictions. The fleet is assumed homogeneous, both for the cargo bicycle, with capacity Q, and the cyclist.

The problem can be formulated as minimizing the travel time, the total tour length or costs associated with edges. As in Fontaine, P. (2022), we choose to minimize the travel time.

Power calculation for cycling

The power consumption when riding an (electric) bicycle depends on many factors. As in Fontaine, P. (2022), we assume that a cyclist drives on a straight line at constant speed in the considered segment. Load-dependent travel times depend on the gradient of the street, which is assumed constant and given by the difference in elevation of the two edge points.

The forces acting on the bicyle are: the aerodynamic drag resistance F_D, the rolling resistance F_R and the component of the gravity force along the direction of motion F_G:

- $F_D = \frac{\rho C_D A v^2}{2}$
- $F_R = C_R mgcos(arctan(h))$
- $F_G = mgsin(arctan(h))$

so, at equilibrium, we have:

$\eta \mathrm{P}_{\mathrm{traction}} = (\mathrm{F_D} + \mathrm{F_R} + \mathrm{F_G})\mathrm{v}$

where η is the efficiency of the mechanic transmission, which is assumed to be 0.95.

Differently from Fontaine, P. (2022), we introduce speed limits for the bicycle, both for downhill and uphill depending not only the traction regime, but in absolute terms as well. While in uphill, the component of the gravity force along the direction of motion is a resistance force, in downhill it helps the motion of the bicycle and as confirmed by numerical experiments, the speed might reach values far above reasonable ones (over 100km/h). This is also due to the fact this simplified model uses the regime speed. We assume that the bicycle speed is limited in downhill for safety reasons. Within cities, a typical speed limit for motor vehicles is 50km/h. Considering also that the regulation for pedelec bicycle allows a maximum speed of 25km/h in electric assisted mode, we set a maximum speed in downhill of 30km/h. The maximum speed limit is imposed on uphill as well. The strategy for solving the velocity equation is described in the flowchart below:

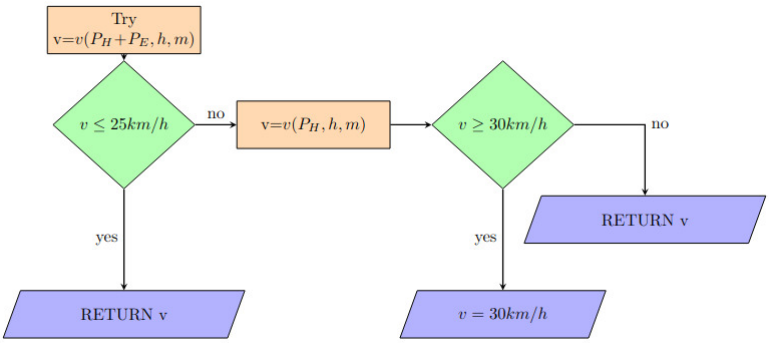

Figure 2: calculation of velocity

In practice we use $P_{electric} = 250$W. For $\overline{P_{cyclist}}$ we do experiments for a few different types of cyclist, as in Fontaine, P. (2022): 75W, 100W, 125W and 150 W.

Mathematical formulation

The problem formulation follows the definition provided by Fontaine, P. (2022). The goal of the model is to minimize the total travel time of the driver. The binary decision variable x_{ij} equals 1 if a vehicle drives on edge (i, j), or 0 otherwise. The continuous decision variable f_{ij} defines the load of the vehicle that is transported between node i and node j. The arrival time at customer $i \in N_0$ is given by the continuous decision variable y_i.

parameter	description	value	unit
N	set of nodes including depot 0		
N_0	set of customers		
E	set of edges		
L	set of load classes		
t_{ij}^l	travel time on edge (i, j) in load class l		
d_{ij}	distance on edge (i, j)		
Q	capacity of the bicycle	150	kg
q_i	demand mass of customer i		kg
s_i	service time at node i		min
$[p^l, r^l]$	mass range of load class l		kg
$[a_i, b_i]$	time window of location i		min
decision variable	**description**		
$x_{ij} \in \{0, 1\}$	binary variable indicating if the vehicle drives on edge (i, j) or not		
$f_{ij} \geq 0$	load of the vehicle on edge (i, j)		
$z_{ij}^l \in \{0, 1\}$	binary variable indicating if load level l on edge (i, j) is used or not		
$y_i \geq 0$	arrival time at customer i		

Figure 3: List of notation of the mathematical model

Similarly to Fontaine, P. (2022), we define a set of load levels L = {1,. . . , l,. . . }. Each load level corresponds to a load interval [p^l,r^l] with $p^1 = 0$ and $r^{|L|}$ = Q. Using this definition, the travel time in each interval l ∈ L is calculated based on the average load level $\frac{p^l+r^l}{2}$ and the characteristics of the road segment (i.e., gradient) as t_{ij}^l. The binary decision variable z_{ij}^l equals 1 if travel time t_{ij}^l is used on edge (i, j) and 0 if not. Using the introduced notation, the problem is formulated as follows:

$$(1) \quad objectivefunction = min \sum_{(i,j)\in E} \sum_{l\in L} t_{ij}^l z_{ij}^l$$

$$(2) \quad \sum_{j\in N} x_{ij} = 1 \quad \forall i \in N_0$$

$$(3) \quad \sum_{i\in N} x_{ij} = 1 \quad \forall j \in N_0$$

$$(4) \quad \sum_{j\in N} f_{ji} - \sum_{j\in N} f_{ij} = q_i \quad \forall i \in N_0$$

$$(5) \quad q_j x_{ij} \leq f_{ij} \leq (Q - q_i)x_{ij} \quad \forall (i, j) \in E$$

$$(6) \quad \sum_{l\in L} z_{ij}^l = x_{ij} \quad \forall (i, j) \in E$$

$$(7) \quad \sum_{l\in L} p^l z_{ij}^l \leq f_{ij} \leq \sum_{l\in L} r^l z_{ij}^l \quad \forall (i, j) \in E$$

$$(8) \quad y_i - y_j + s_i + \sum_{l\in L} t_{ij}^l z_{ij}^l \leq M_{ij}(1 - x_{ij}) \quad \forall i \in N, j \in N_0, i \neq j$$

$$(9) \quad a_i \leq y_i \leq b_i \quad \forall i \in N_0$$

$$(10) \quad x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E$$

$$(11) \quad f_{ij} \geq 0 \quad \forall (i, j) \in E$$

$$(12) \quad z_{ij}^l \in \{0, 1\} \quad \forall (i, j) \in E, \quad l \in L$$

The objective function (1) minimizes the total travel time. Moreover, following Fontaine, P. (2022):

- Constraints (2) and (3) ensure that each customer is visited exactly once.
- The flow balance constraints (4) guarantee that each customer demand is satisfied and the vehicle load is decreased by the customer demand after each customer visit.
- The vehicle capacity is ensured by constraints (5).
- Constraints (6) state that a travel time level on an edge is only selected if a vehicle uses this edge. Since z_{ij}^l is a binary decision variable, constraint (6) further ensures that exactly one travel time level is selected for each traversed edge.
- Constraints (7) guarantee that only the travel time level that corresponds to the load of the vehicle on the edge is selected. Thus, the load lies in between the upper and lower bound of the selected load level.
- Constraints (8) update the visiting times at each customer. If a vehicle travels along an edge, the difference between the two arrival times of the involved customers has to be larger than the service time at the origin node and the selected travel time. We use the formulation suggested by Fontaine, P. (2022): $M_{ij} = max\{0, b_i + s_i + t_{ij}^{|L|} - a_j\}$. Since several travel times per edge are possible, the slowest travel time $t_{ij}^{|L|}$ is used.
- Constraints (9) ensure that the time windows are met.

Travel time calculation

The load-dependent travel time t_{ij}^l is the key element of this paper. We can calculate the travel time with the distance d_{ij} and the speed $v(P, m_l, h_{ij})$, which is a function of the power P, the mass m_l, and the gradient h_{ij} on edge (i, j). The power is a given parameter and the mass is given through the load level l. However, to apply the physical formulas for the calculation of the velocity, a constant gradient between two nodes (i, j) has to be assumed. The velocity calculations described in the previous section assumes that each edge between two nodes has a constant gradient and therefore a constant speed. Especially for larger networks, the connection between two nodes can also have uphill and downhill segments or segments with different gradients. Fontaine, P. (2022) sugggets how load-dependent travel times can be computed in this case. A graph G^l for each load level l has to be used. This graph consists of a set of nodes V that represent not only all nodes N but also all intersections and points in the network where the gradient is changing. Let S further be the set of edges in this extended network. Then all edges in S are constant segments. The travel time on a segment s is then defined as $t_s^l = d_s/v(P, m_l, h_s)$ with d_s being the distance and h_s the gradient on the segment.

Then the fastest path from node i ∈ N to node j ∈ N in load level l can be computed using a shortest path algorithm. Fontaine, P. (2022) suggests that this may result in different paths for different load levels.

Algorithm 1: Advanced travel time calculation.

```
for l ∈ L do
    for s ∈ S do
        t_s^l = d_s/v(P, m_l, h_s)
    for (i, j) ∈ E do
        t_ij^l ← Calculate the shortest path from i to j in G^l(t_s^l)
```

Figure 4: Travel time calculation pseudocode - Fontaine, P. (2022)

We investigate the effect of incorporating intermediate elevation points using a synthetized data generated with the following procedure:

- build a rectangular grid of type street-avenue, with elevation at each grid node;
- place randomly a certain number of customers at intersection points;
- the distance among two points (i, j) is the sum $\Delta x + \Delta y$ of their coordinates.

For defining intermediate path points we used a simple approach:

- if both Δx and Δy are different from zero, the intermediate point is placed at the intersection of the horizontal/vertical lines passing through i and j;
- no intermediate point is considered otherwise

We implement Dijkstra algorithm for searching the shortest path given an instance that includes both the basic distance matrix and the intermediate points:

- base case: use only the upper left part of the distance matrix (and the corresponding portion of the problem parameters)
- intermediate points case: set to zero the upper left portion of the distance matrix. The intermediate point on the edge (i, j) has non-zero distances only to nodes i and j. For each load level, all possible shortest paths between valid nodes are calculated. The result is a tridimensional matrix in which the depth is the number of load levels.

This distance matrix is symmetric as in the case of Google-generated data in Fontaine, P. (2022) and is structured in a way that the upper left sub matrix defines the no-intermediate point case while the rest of the matrix, which is sparse, include the distances of edge (i, j) to its intermediate point.

With this procedure we are able to use a single data set for running comparisons among for the same test case with and without intermediate data points by simply selecting a sub portion of the data matrix.

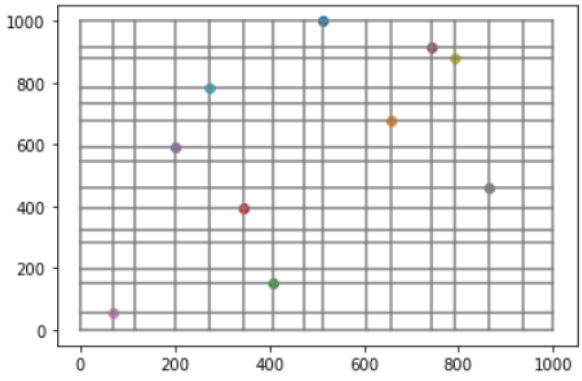

Figure 5: example of synthesized data

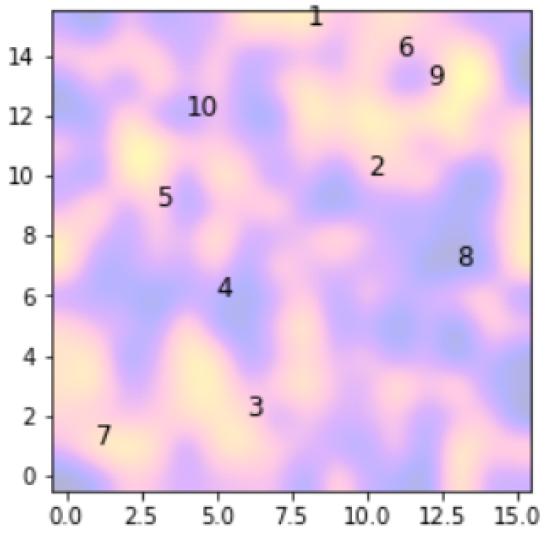

Figure 6: elevation of the grid map

Customer demand is generated in a similar way to what is done by Fontaine, P. (2022) using a uniform distribution and three average values. Regarding the time windows for delivery we generate different scenarios, since it is reasonable to assume that this parameter affects the solution of the VRP problem as well. The synthetic data generation procedure allows to select different types of time windows: randomly distributed, overlapped, disjoint and a single time window for all customers. As confirmed by the results presented in the next section, the time window parameter affects the solution of the problem.

Besides the flexibility of generating many different problem instances, the synthetic data generation is used to assess the scalability of the solution.

Implementation

Differently from Fantaine, P. (2022), we implemented the model using Gurobi. Our numerical study included some of the instances used by Fontaine, P. (2022), synthetic instances generated as described in the previous section and some instances on the city of Trieste, for the purpose of visualizing the paths in a familiar environment. The latter has been accomplished by generating the necessary data with Google API.

	x	y	elevation	demand	tw a	tw b	s	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0	0	27.2	0	0	0	0	0	1.43	1.39	0.47	0.74	0.8	1.29	0.51	0.66	0.41	0.35	0.2	0.74	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	513	913	17.4	24	88	117	5	1.43	0	0.32	0.96	0.68	0.63	0.6	0.91	0	0	0	0	0	0.18	0.86	0	0	0	0	0.52	0	0	0	0.32	0	0.37	0	0
2	657	735	11.4	10	98	113	5	1.39	0.32	0	0.93	0.65	0.6	0.27	0	0.74	0	0	0	0	0.14	0	0.68	0	0	0	0	0.34	0	0	0	0.14	0	0.19	0
3	409	58	21.0	13	144	159	5	0.47	0.96	0.93	0	0.4	0.74	0.82	0	0	0.06	0	0	0	0	0.1	0.25	0.06	0.21	0.3	0	0	0	0	0	0	0	0	0
4	346	396	24.1	5	136	159	5	0.74	0.68	0.65	0.4	0	0.34	0.55	0	0	0	0.4	0	0	0	0	0	0.34	0	0	0.17	0.31	0.14	0.4	0	0	0	0	0
5	202	593	14.7	28	85	101	5	0.8	0.63	0.6	0.74	0.34	0	0.59	0	0	0	0	0.59	0	0	0	0	0	0.54	0	0	0	0.2	0	0.31	0.45	0	0	0.05
6	742	546	9.9	18	118	145	5	1.29	0.6	0.27	0.82	0.55	0.59	0	0	0	0	0	0	0.55	0	0	0	0	0	0.5	0	0	0	0.15	0	0	0.23	0.09	0.54
7	0	0	4.7	-1	0	0	0	0.51	0.91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	20.0	-1	0	0	0	0.66	0	0.74	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	21.9	-1	0	0	0	0.41	0	0	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	7.5	-1	0	0	0	0.35	0	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	6.7	-1	0	0	0	0.2	0	0	0	0	0.59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	9.3	-1	0	0	0	0.74	0	0	0	0	0	0.55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	22.2	-1	0	0	0	0	0.18	0.14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	14.3	-1	0	0	0	0	0.86	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	27.3	-1	0	0	0	0	0	0.68	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	15.6	-1	0	0	0	0	0	0	0.06	0.34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	4.2	-1	0	0	0	0	0	0	0.21	0	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	20.2	-1	0	0	0	0	0	0	0.33	0	0	0.49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	19.2	-1	0	0	0	0	0.52	0	0	0.17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	8.4	-1	0	0	0	0	0	0.34	0	0.31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	16.2	-1	0	0	0	0	0	0	0	0.14	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	16.5	-1	0	0	0	0	0	0	0	0.4	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	4.1	-1	0	0	0	0	0.32	0	0	0	0.31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	27.0	-1	0	0	0	0	0	0.14	0	0	0.45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	11.9	-1	0	0	0	0	0.37	0	0	0	0	0.23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	11.1	-1	0	0	0	0	0	0.19	0	0	0	0.09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	3.1	-1	0	0	0	0	0	0	0	0	0.05	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7: example of synthesized grid data matrix

Results

In this section we report the main results of the numerical simulations and the effects of the various parameters on problem solution. This type of analysis, together with those in this chapter, aim to investigate the model behavior and to provide some confidence in the results, which do not always make immediate sense, especially when the number of customers grows. We report that we found all the instances we worked on as feasible.

Scalability

We performed a series of tests using the number of customers as a proxy for the problem size and the execution time as a proxy for problem scalability. Since there are random elements in the data generation process, different instances of a problem of a given size might require dramatically different execution times. We run eight randomly generated test cases for each problem size within the range [2, 25] at steps of 2. For each problem size we report the average execution time. Although the branch and bound algorithm is upper bounded by an exponential time complexity, we found an average execution time that is often much better.

Moreover, since the solution process includes different algorithms, we report the execution time for the main sub problems:

- Dijkstra algorithm for finding best paths matrix
- Gurobi overhead time
- Gurobi solver time
- total time

Although the solver execution time depends heavily on the specific characteristics of the instance and not only on the problem size, as expected, the total time is dominated by the solver time for problem instances larger than trivial values. For problems with number of customers up to 30, we found it unlikely to have solution times larger than few minutes on a consumer laptop machine.

Problem size

Another measure of scalability is given by the problem size The charts below, show the number of decision variables for different problem sizes.

In particular, we see that the number of binary variables grows quadratically with the number of customers, since the distance matrix, which represents the edges of the graph connecting all the customers has size given by the number of customers. Using intermediate points along the paths affects only Dijkstra execution time.

Figure 8: execution time vs number of customers

Figure 9: log scale of execution time vs number of customers

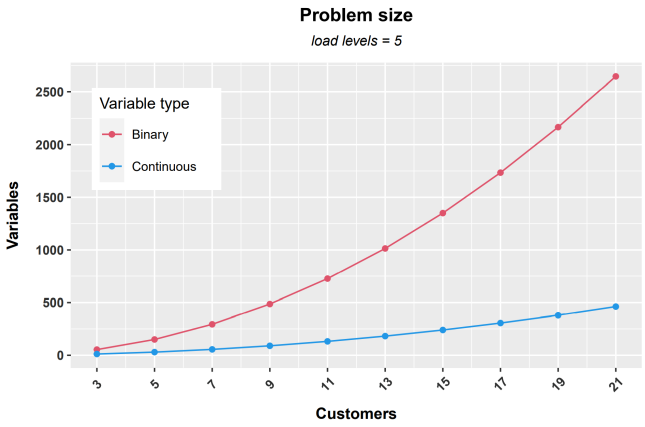

Figure 10: number of variables vs problem size

We investigated as well the effect of increasing the number of load levels. The maximum number of load levels has been set to Q/Q_{min}, where Q is the capacity of the trunck and Q_{min} is the lower value of the interval $[Q_{min}, Q_{max}]$ used to draw uniformly distributed load values. As expected, the number of binary variables is linear with the number of load levels.

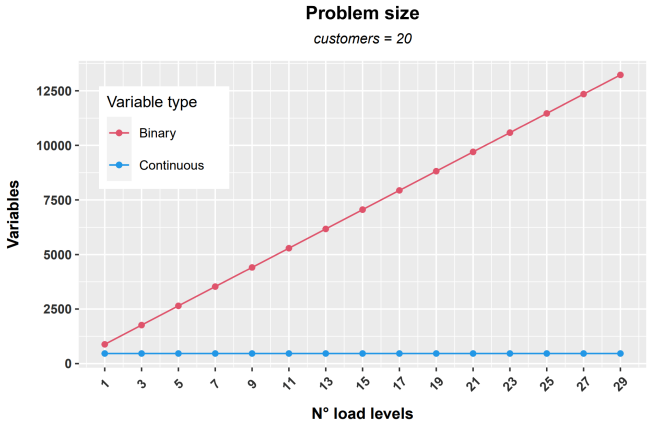

Figure 11: number of variables vs problem size

Effect of the midpoint path elevation information

The synthetic data sets have been generated with the main target of testing the performances of the problem metrics when including intermediate elevation information. Numerical experiments confirmed that including such information has a dramatic impact on the results. The table below shows that there is great variability in the difference of the objective function value:

- T_{no_ext} is the value of the objective function for the base distance matrix;
- T_{ext} is the value of the objective function for the extended distance matrix (including the mid points);
- the third column indicates the percentage difference among the two values.

It seems that in the majority of the cases, at least for the problems tested, the base case underestimates the travel time. We notice that the solution tours change as well.

Customers	$T_{no\_ext}[min]$	$T_{ext}[min]$	$\frac{T_{ext}-T_{no\_ext}}{T_{ext}}[\%]$
3	37.22	56.48	34.09
5	133.05	248.48	46.45
7	106.10	248.98	57.38
9	259.95	157.68	-64.86
11	234.62	205.97	-13.91
13	148.24	237.57	37.60
15	71.23	150.28	52.60
17	124.58	260.03	52.09
19	157.96	252.67	37.48
21	119.65	193.49	38.16
23	144.73	213.38	32.17
25	138.57	283.81	51.17

Figure 12: total travel time without and with intermediate path slope information

Effect of time windows

We investigated as well the effect of providing different types of time windows for the delivery to customers. It is reasonable to assume that providing a unique, large, time window for all the customers eliminates de facto one of the constraints, allowing the search for the solution to be based only on the fastest edge sequence. The experimental results confirm the hypothesis and are reported below:

- T_{random} is the value of the objective function for a randomly generated time window;
- T_{single} is the value of the objective function for a single time window.

We notice as well that using a single time window seems to consistently require longer execution times.

Customers	$T_{random}[s]$	$T_{single}[s]$	$\frac{T_{random}-T_{single}}{T_{random}}[\%]$
3	126.62	126.62	0.00
5	179.52	116.02	35.37
7	184.89	118.10	36.13
9	126.08	107.35	14.86
11	168.11	101.36	39.71
13	167.10	102.20	38.84
15	129.86	89.13	31.36
17	158.23	96.80	38.82
19	131.62	96.26	26.87
21	136.17	93.67	31.21
23	132.82	88.28	33.54
25	137.69	94.84	31.12

Figure 13: total travel time single and random time windows

Figure 14: execution time vs type of time window

Effect of payload weight

As expected, increasing the total payload of the bicycle increases the total travel time.

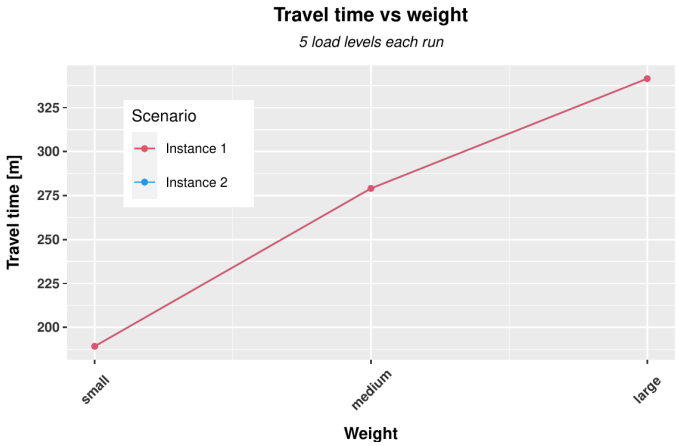

Figure 15: total travel time for three levels of average load

Effect of power

Given that differences of 25W in ciclyst power used in this paper represent only 6-7% on the total power, we decide to use a few higher but still reasonable values, to explore a larger domain of the solution dependence on P. As expected, increasing the cyclist power, the total travel time is reduced.

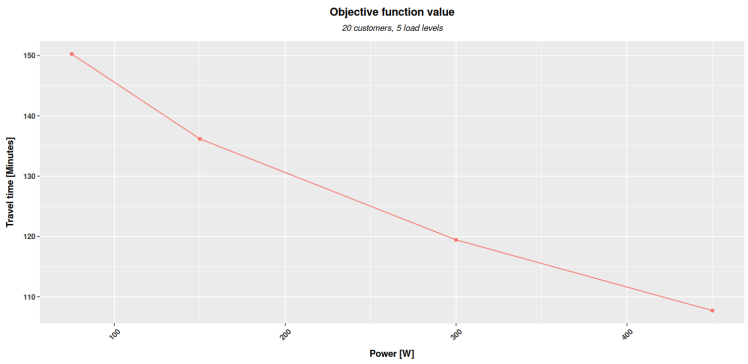

Figure 16: effect of power on total travel time

Effect of load levels

We investigate the effect of the number of load levels starting from the consideration that using few load levels (3 or 4) for a capacity of 160kg might produce inaccurate results. We ran different problem instances using different seeds for the routines using random numbers within the synthetic data procedure and we average the results. Increasing the number of load levels linearly increases the depth dimension of the distance matrix, which confirms a linear effect on Dijkstra execution time. Gurobi overhead and solver time seem to show a linear trend as well. In terms of travel time, we notice that the objective function value increases when moving from 1 load level to more fine discretization, which makes sense since the velocity calculation uses the average load level point. It is interesting to observe that the effect on the tours is also significant (see figures in Appendix).

Figure 17: travel time in function of number of load levels

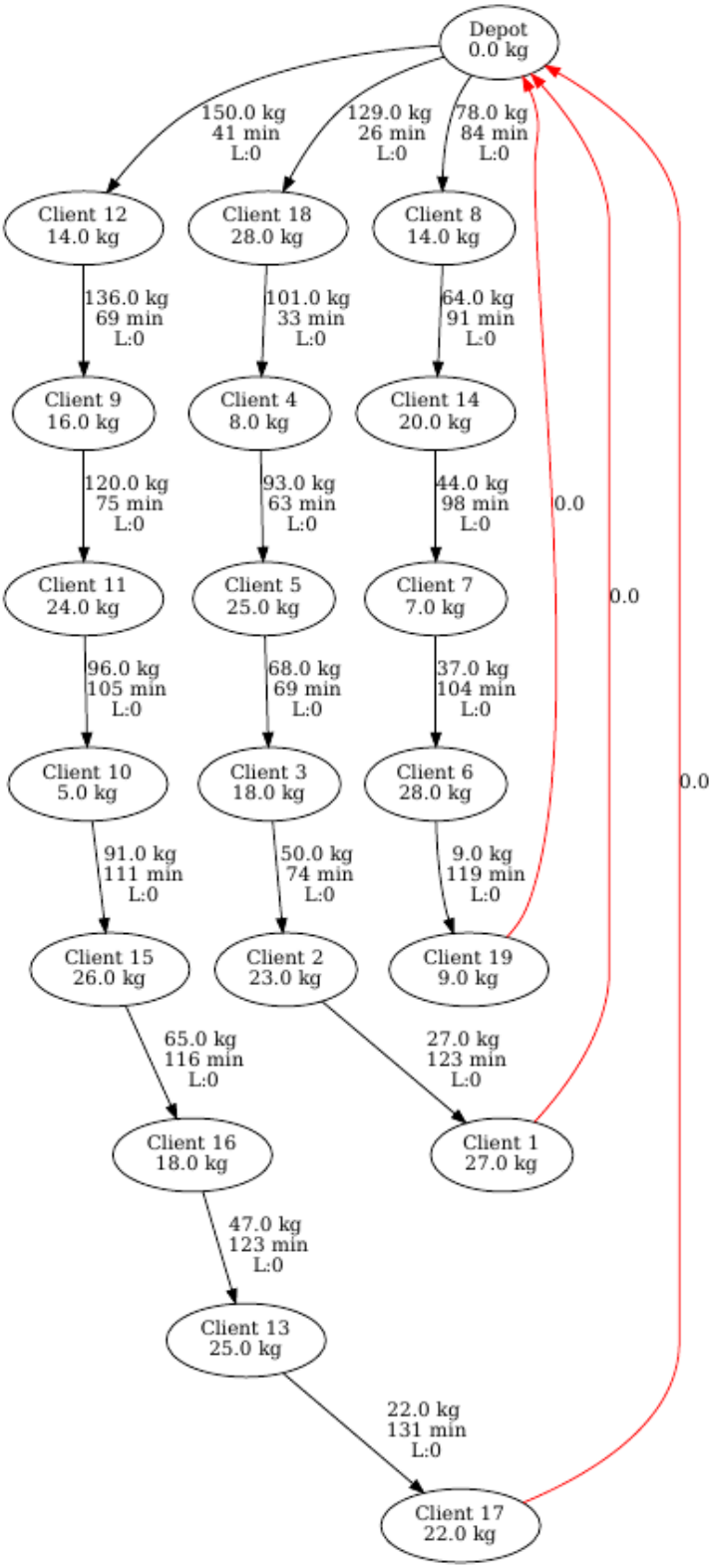

Figure 18: tours with 1 load level

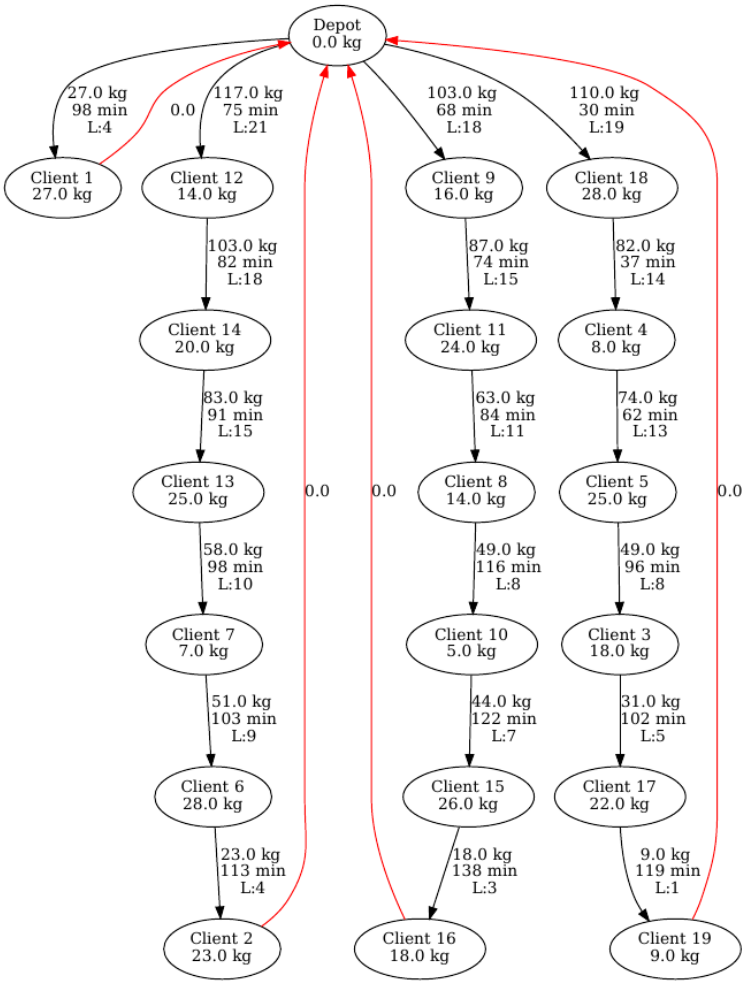

Figure 19: tours with 31 load levels

Example path solution for the city of Trieste

For the purpose of analyzing the problem within a familiar environment, we set up an interface with Google API and created some small problem instances for the city of Trieste, which could be used for any other city. The solution tours are reported in Appendix.

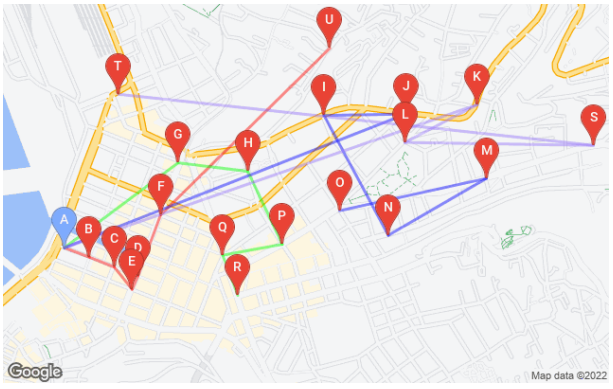

Figure 20: example of instance solution for the city of Trieste

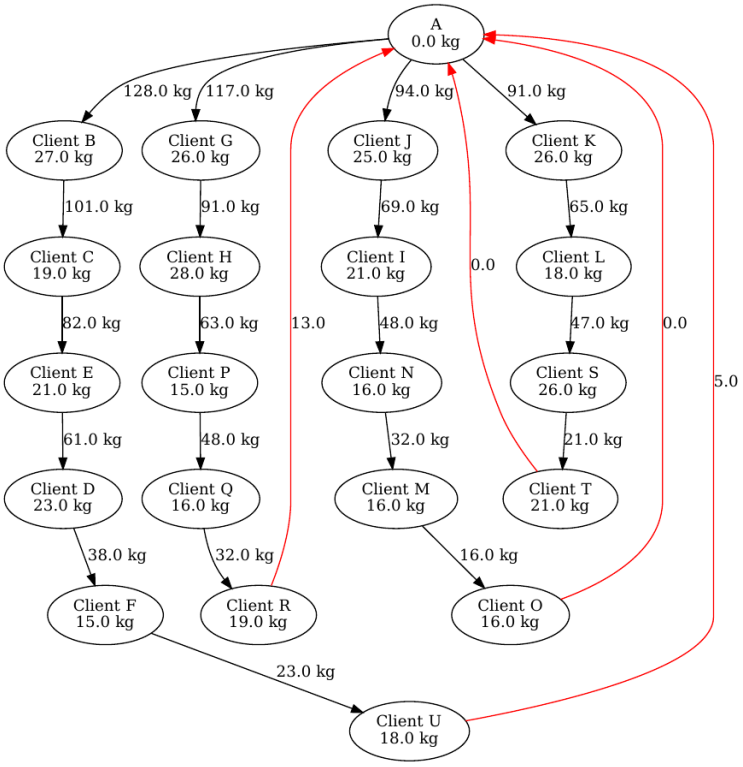

Figure 21: example of instance solution for the city of Trieste

Conclusions

The significant effect on total travel time showed by including even some basic information about the intermediate elevation of the paths indicates that realistic solutions might vary significantly from those obtained using only the elevations of customer points. It is our opinion that a valid solution to the practical problem introduced in the paper of Fontaine, P. (2022) is one that estimates the travel times as accurately as possible. A possible line of research is then related to studying the integration of topographic maps for finding the appropriate level of path detail to include, which can reasonably be assumed to depend on the city in question. Another possible line of improvement can be the use of transient velocity calculations, instead of regime values. Depending on the length of the edge, on the slope and on its sign, there might be significant differences among the transit time using the two velocities. This is even more important in a realistic environment, in which the bicycle might have frequent stops or slowdowns due to traffic conditions. Such an analysis is also related to setting the appropriate level of detail of the intermediate elevation points. Moreover, energy consumption and related battery charge constraints might be considered as well. This might affect not only the length of possible tours but also the total number of customers that can be served in a given time frame, such as a working day.

References

Fontaine, P. (2022). The vehicle routing problem with load-dependent travel times for cargo bicycles. European Journal of Operational Research, 300(3), 1005-1016.