Master in Statistics and Operative Research

Large-Scale Optimization

Frank Wolfe & RSD Algorithms

Víctor Diví i Cuesta

Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

Contents

List of Figures

List of Tables

List of Summaries

1. Introduction

In this assignment, we will solve a Traffic Assignment Problem (TAP), also known as Network Equilibrium Problem, in which we aim to obtain an equilibrium state assigning the flow that must go from every origin to its destinations to the best possible path.

In this Section, the problem and scenario at hand are introduced. In Section 2, the base formulation of the equilibrium problem is solved directly. In Sections 3 and 4, the same problem is solved using the Frank Wolfe (FW) algorithm and the Restricted Simplicial Decomposition (RSD) algorithm, respectively. Then, in Section 5 an analysis of the Wardrop Equilibrium of the solutions is presented. Finally, in Section 6, a small summary is provided as conclusion.

1.1. Problem

The equilibrium problem at hand can be formulated in several ways. Here we will use the origin-based formulation, obtaining the following non-linear optimization problem in scalar notation:

$$\begin{aligned}
\text{Min} \quad & \sum_{a\in A} \int_0^{x_a} S_a(x)dx \\
\text{s.t.} \quad & \sum_{(i,j)\in A} x_{ij}^l - \sum_{(j,i)\in A} x_{ji}^l = T_i^l && \forall i \in N, \forall l \in O \\
& \sum_{l\in O} x_a^l = x_a && \forall a \in A \\
& x_a^l \geq 0 && \forall a \in A, \forall l \in O
\end{aligned} \tag{TAP}$$

Where given an arc a, an origin l and a node i, x_a^l is the flow at a from l, x_a os the total flow at a, S_a is the link-cost function of a, and T_i^l is the net output flow at i for flow from l.

Here, the link-cost function are linear volume-delay functions:

$$S_a(x) = C_a + dx \qquad \forall a \in A$$

In which $C_a \geq 0$ is the base link cost and $d \geq 0$ is a fixed constant. The problem can be expressed as an optimization problem because S_a are all non-negative, non-decreasing and continuous.

1.2. Scenario

The data used in this report is from Scenario 19, with:

$$\begin{aligned}
N &= \{1, ..., 24\} \\
O &= \{3, 11\} \\
D &= \{5, 8, 12, 13, 14, 17, 18, 19, 21, 23\} \\
C_{ij} &= 95 + (x_i - x_j)^2 + 8(y_i - y_j)^2 && \forall (i,j) \in A \\
T_i^l &= \begin{cases} 500K, & \text{if } i = l \\ -50K, & \text{if } i \in D \\ 0, & \text{otherwise} \end{cases} && \forall i \in N, \forall l \in O \\
d &= 0.0035 \\
K &= 200
\end{aligned}$$

To appreciate the difference in performance of the different algorithms, we multiply the flows for each OD pair by a factor K (here set at 200), so the resulting network is significantly congested. The values for all these parameters is provided in the `net.dat` file.

2. Base Formulation

We start solving the problem by solving directly the base formulation (TAP), which will serve as the goal solution to achieve by the FW and RSD algorithms.

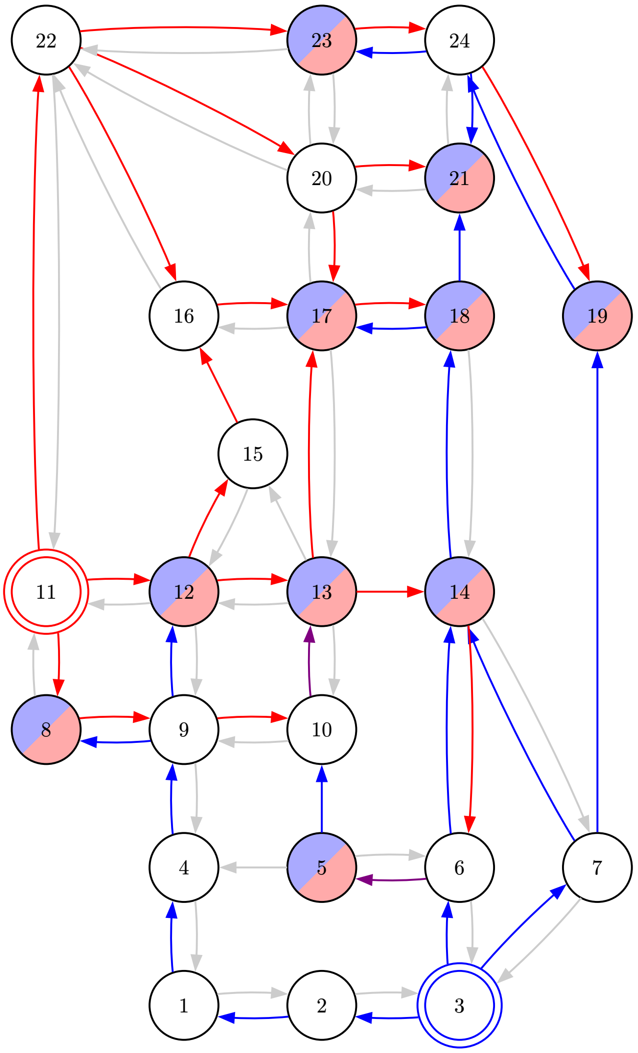
The AMPL code for the model is provided in `base.mod` and `base.run` for the model definition and the runner script respectively.

The obtained solution has a total cost of 101181657, and the detailed flows are presented in Summary 1, along with a visualization of the resulting assignation on the network. On the tables on the left, we can see, for each used arc, the amount of flow they carry from each origin. On the right, we can see the resulting assignation on the network.

Total Cost	101181657

Flows for $l = 3$	
Arc	**Flow**
1 → 4	20000.00
2 → 1	20000.00
3 → 2	20000.00
3 → 6	40037.27
3 → 7	39962.73
4 → 9	20000.00
5 → 10	10000.00
6 → 5	20000.00
6 → 14	20037.27
7 → 14	19540.37
7 → 19	20422.36
9 → 8	10000.00
9 → 12	10000.00
10 → 13	10000.00
14 → 18	29577.64
18 → 17	10000.00
18 → 21	9577.64
19 → 24	10422.36
24 → 21	422.36
24 → 23	10000.00

Flows for $l = 11$	
Arc	**Flow**
6 → 5	10000.00
8 → 9	1340.66
9 → 10	1340.66
10 → 13	1340.66
11 → 8	11340.66
11 → 12	49334.38
11 → 22	39324.96
12 → 13	34885.40
12 → 15	4448.98
13 → 14	20000.00
13 → 17	6226.06
14 → 6	10000.00
15 → 16	4448.98
16 → 17	9499.22
17 → 18	10000.00
20 → 17	4274.73
20 → 21	10000.00
22 → 16	5050.24
22 → 20	14274.73
22 → 23	20000.00
23 → 24	10000.00
24 → 19	10000.00



Summary 1: Results of the base model. Origin nodes have a colored double border, while destination nodes are filled with the origin color. Used links are painted with the color of the origin whose flow they carry (purple for both), while unused links are greyed. Exact flows are shown in flow tables.

3. Frank Wolfe Algorithm

Now, we will solve the same problem using the Frank Wolfe Algorithm. The algorithm follows these steps:

1) Start with a feasible solution x^0.
2) Repeat until a stopping criteria is met (at each iteration k):
    1) Solve linear approximation of the problem to obtain new point $\hat{x}^k$.
    2) Compute the descent direction: $d^k = \hat{x}^k - x^k$.
    3) Calculate the optimal step size α^k minimizing the objective function.
    4) Update the solution: $x^{k+1} = x^k + \alpha^k \hat{x}^k$.

In our case, the linear approximation of the (TAP) is the following:

$$\begin{array}{lll} \text{Min} & \sum_{a\in A} x_a\left(C_a + dx^k\right) & \\ \text{s.t.} & \sum_{(i,j)\in A} x_{ij}^l - \sum_{(j,i)\in A} x_{ji}^l = T_i^l & \forall i \in N, \forall l \in O \\ & \sum_{l\in O} x_a^l = x_a & \forall a \in A \\ & x_a^l \geq 0 & \forall a \in A, \forall l \in O \end{array} \tag{LTAP}$$

An we stop the algorithm when we reach 500 iterations or a relative gap lower than ε (set to $5e^{-4}$), computed as follows:

$$\text{gap} = \frac{-\sum_{a\in A} S_a\left(x^k\right)\left(\hat{x}^k - x^k\right)}{\sum_{a\in A} S_a(x^k)x^k}$$

Moreover, due to the nature of the link-cost functions, we can compute the optimal α^k analytically as the following expression:

$$\alpha^k = \min\left(1, \frac{-\sum_{a\in A} S_a\left(x^k\right)\left(\hat{x}^k - x^k\right)}{d\sum_{a\in A}\left(\hat{x}^k - x^k\right)^2}\right)$$

The full AMPL model and runner script is provided in the `fw.mod` and `fw.run` files respectively.

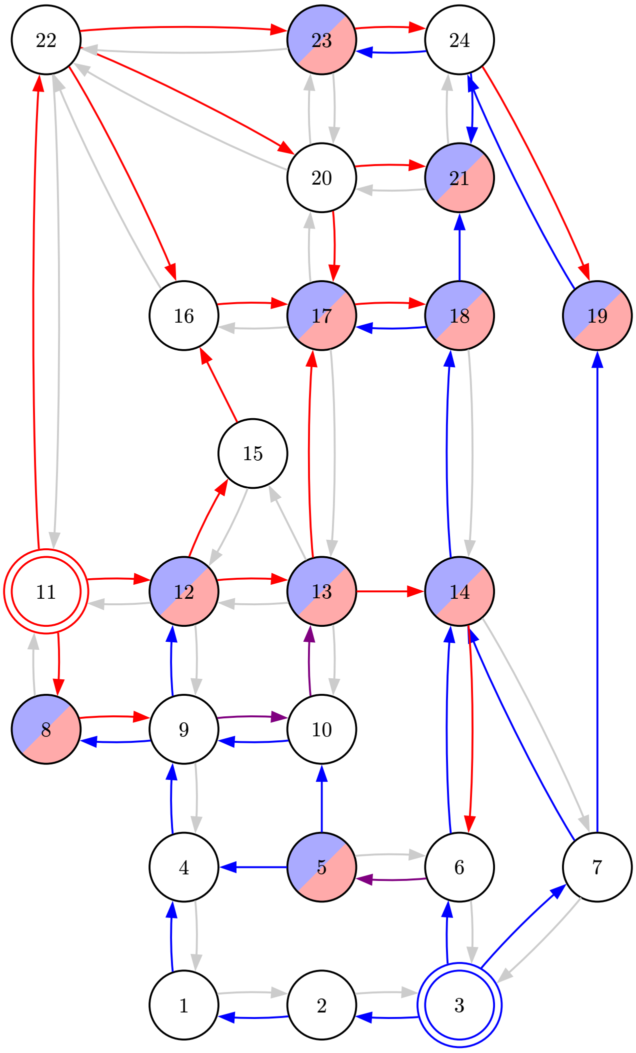
After a total of 112 iterations, the result obtained has a total cost of 101218966, which is 37309 higher than the optimal cost (101181657) obtained with the base formulation (see Section 2). This gap is within acceptable limits, since in relative terms it is a gap of $3.68e^{-4}$, which is lower than the accepted tolerance $\varepsilon = 5e^{-4}$ (note, however, that this is not the gap that was used to stop the algorithm).

The obtained flows can be seen in Summary 2, both listed on the flow tables on the left, as well as visualized in the network on the right. As can be observed, the generated flows differ slightly (as expected) from the optimal solution generated before. For example, there is a cycle in the flow of origin 3 between nodes 9 and 10 that doesn't exist in the previous solution. Arcs ($9 \to 10$) and ($10 \to 9$) both carry very little flow and only increase the overall cost without contributing anything, so most probably this cycle would be removed in subsequent iterations. For origin 11, the values of the flows vary slightly, but the used arcs are the same as in the optimal solution.

Total Cost	101218966
Optimal Cost	101181657
Gap	37309
Relative Gap	3.68e-4
Iterations	112

Flows for $l = 3$	
Arc	**Flow**
1 → 4	19761.32
2 → 1	19761.32
3 → 2	19761.32
3 → 6	40104.21
3 → 7	40134.47
4 → 9	20005.78
5 → 4	244.46
5 → 10	9994.22
6 → 5	20238.68
6 → 14	19865.53
7 → 14	19678.36
7 → 19	20456.11
9 → 8	10000.00
9 → 10	141.90
9 → 12	10000.00
10 → 9	136.13
10 → 13	10000.00
14 → 18	29543.89
18 → 17	10000.00
18 → 21	9543.89
19 → 24	10456.11
24 → 21	456.11
24 → 23	10000.00

Flows for $l = 11$	
Arc	**Flow**
6 → 5	10000.00
8 → 9	1397.44
9 → 10	1397.44
10 → 13	1397.44
11 → 8	11397.44
11 → 12	49254.14
11 → 22	39348.42
12 → 13	34811.16
12 → 15	4442.97
13 → 14	20000.00
13 → 17	6208.60
14 → 6	10000.00
15 → 16	4442.97
16 → 17	9547.11
17 → 18	10000.00
20 → 17	4244.28
20 → 21	10000.00
22 → 16	5104.14
22 → 20	14244.28
22 → 23	20000.00
23 → 24	10000.00
24 → 19	10000.00



Summary 2: Results of the FW algorithm. Origin nodes have a colored double border, while destination nodes are filled with the origin color. Used links are painted with the color of the origin whose flow they carry (purple for both), while unused links are greyed. Exact flows are shown in flow tables.

Figure 1 shows the evolution of the objective function value (Obj. Fun.), the relative gap (Rel. Gap), and the step length (Step (α)) over the iterations (I) of the execution.

As can be seen, the objective function constantly decreases, although slowly, while the relative gap has some fluctuation, although always keeps a descending tendency. The step length (α) is always comprised between $6e^{-1}$ and $2e^{-3}$, and fluctuates heavily over the iterations, although in general, it gets smaller as the algorithm progresses.

I	Obj. Fun.	Rel. Gap	Step (α)
1	1.0337e+08	2.1133e-01	5.3724e-01
2	1.0260e+08	3.0525e-02	3.8251e-01
3	1.0224e+08	3.8921e-02	1.4092e-01
4	1.0205e+08	1.4793e-02	1.9576e-01
5	1.0193e+08	1.9750e-02	9.9047e-02
6	1.0185e+08	9.4300e-03	1.3051e-01
7	1.0171e+08	1.3272e-02	1.5595e-01
8	1.0162e+08	1.1033e-02	1.3498e-01
9	1.0154e+08	9.1320e-03	1.2334e-01
10	1.0150e+08	7.4255e-03	9.9912e-02
11	1.0147e+08	5.5951e-03	7.4235e-02
12	1.0145e+08	5.0882e-03	6.9019e-02
13	1.0143e+08	4.8634e-03	7.1315e-02
14	1.0141e+08	4.6738e-03	6.6118e-02
15	1.0140e+08	4.3682e-03	1.8553e-02
16	1.0139e+08	3.0279e-03	4.2057e-02
17	1.0138e+08	3.3568e-03	4.8943e-02
18	1.0137e+08	3.5394e-03	5.3149e-02
19	1.0136e+08	3.6784e-03	5.0830e-02
20	1.0135e+08	2.9092e-03	4.0164e-02
21	1.0134e+08	2.8623e-03	4.1931e-02
22	1.0134e+08	3.1906e-03	1.4013e-02
23	1.0134e+08	1.7919e-03	2.8080e-02
24	1.0133e+08	2.6793e-03	3.9321e-02
25	1.0132e+08	2.5283e-03	3.5140e-02
26	1.0132e+08	2.6143e-03	3.6752e-02
27	1.0131e+08	2.3013e-03	3.5314e-02
28	1.0131e+08	2.2538e-03	3.3313e-02
29	1.0130e+08	2.1444e-03	2.9982e-02
30	1.0130e+08	2.0137e-03	2.8665e-02
31	1.0130e+08	1.8449e-03	2.8590e-02
32	1.0129e+08	1.9697e-03	2.9254e-02
33	1.0129e+08	2.3513e-03	1.0073e-02
34	1.0129e+08	1.2374e-03	1.7638e-02
35	1.0129e+08	1.6312e-03	2.3448e-02
36	1.0128e+08	1.6601e-03	2.4372e-02
37	1.0128e+08	1.8529e-03	2.9159e-02
38	1.0128e+08	1.7070e-03	2.3958e-02
39	1.0128e+08	1.4410e-03	2.1650e-02
40	1.0127e+08	1.8534e-03	7.8284e-03
41	1.0127e+08	1.1564e-03	1.8194e-02
42	1.0127e+08	1.5793e-03	2.3599e-02
43	1.0127e+08	1.4611e-03	2.0653e-02
44	1.0127e+08	1.5198e-03	2.1770e-02
45	1.0127e+08	1.4223e-03	2.2151e-02
46	1.0126e+08	1.4253e-03	2.1355e-02
47	1.0126e+08	1.3395e-03	5.7752e-03
48	1.0126e+08	9.7634e-04	1.3172e-02
49	1.0126e+08	1.1925e-03	1.7250e-02
50	1.0126e+08	1.2547e-03	1.8525e-02
51	1.0126e+08	1.3616e-03	2.1622e-02
52	1.0126e+08	1.3572e-03	1.9134e-02
53	1.0125e+08	1.0950e-03	1.5863e-02
54	1.0125e+08	1.2241e-03	1.8065e-02
55	1.0125e+08	1.2558e-03	5.6014e-03
56	1.0125e+08	8.4456e-04	1.3610e-02
57	1.0125e+08	1.2103e-03	1.7856e-02
58	1.0125e+08	1.1076e-03	1.5985e-02
59	1.0125e+08	1.1717e-03	1.6614e-02
60	1.0125e+08	1.0901e-03	1.7410e-02
61	1.0125e+08	1.1184e-03	1.6528e-02
62	1.0124e+08	1.0343e-03	1.4943e-02
63	1.0124e+08	1.0257e-03	4.4276e-03
64	1.0124e+08	6.7574e-04	1.0066e-02
65	1.0124e+08	1.0000e-03	1.5096e-02
66	1.0124e+08	1.0587e-03	1.5037e-02
67	1.0124e+08	9.7228e-04	1.4058e-02
68	1.0124e+08	9.3791e-04	1.4732e-02
69	1.0124e+08	1.0396e-03	1.5662e-02
70	1.0124e+08	9.5150e-04	4.1081e-03
71	1.0124e+08	6.5867e-04	9.4320e-03
72	1.0124e+08	8.5965e-04	1.2487e-02
73	1.0124e+08	8.8721e-04	1.3172e-02
74	1.0124e+08	9.9810e-04	1.5943e-02
75	1.0124e+08	9.3370e-04	1.3258e-02
76	1.0123e+08	7.4060e-04	1.0788e-02
77	1.0123e+08	8.9799e-04	1.3319e-02
78	1.0123e+08	9.2039e-04	4.1113e-03
79	1.0123e+08	6.2971e-04	1.0168e-02
80	1.0123e+08	8.9897e-04	1.3327e-02
81	1.0123e+08	8.0229e-04	1.1632e-02
82	1.0123e+08	8.3586e-04	1.1918e-02
83	1.0123e+08	8.0979e-04	1.2994e-02
84	1.0123e+08	8.4238e-04	1.2502e-02
85	1.0123e+08	7.5856e-04	1.1006e-02
86	1.0123e+08	7.7050e-04	3.2723e-03

I	Obj. Fun.	Rel. Gap	Step (α)
87	1.0123e+08	5.0689e-04	8.0462e-03
88	1.0123e+08	8.2085e-04	1.2415e-02
89	1.0123e+08	7.9481e-04	1.1337e-02
90	1.0123e+08	7.3211e-04	1.0624e-02
91	1.0123e+08	6.9827e-04	1.1013e-02
92	1.0123e+08	7.8102e-04	1.1821e-02
93	1.0123e+08	7.5587e-04	3.2098e-03
94	1.0123e+08	5.1059e-04	8.0975e-03
95	1.0123e+08	6.8591e-04	1.0408e-02
96	1.0122e+08	6.8088e-04	9.7362e-03
97	1.0122e+08	7.8370e-04	1.1348e-02
98	1.0122e+08	6.9570e-04	1.0965e-02
99	1.0122e+08	6.6669e-04	1.0118e-02
100	1.0122e+08	6.7782e-04	9.1603e-03
101	1.0122e+08	6.0271e-04	8.7695e-03
102	1.0122e+08	6.3685e-04	1.0052e-02
103	1.0122e+08	6.4732e-04	9.8260e-03
104	1.0122e+08	8.0964e-04	3.4953e-03
105	1.0122e+08	5.0620e-04	7.2515e-03
106	1.0122e+08	6.1136e-04	9.2882e-03
107	1.0122e+08	6.3884e-04	9.1372e-03
108	1.0122e+08	6.4418e-04	1.0363e-02
109	1.0122e+08	6.5122e-04	9.2877e-03
110	1.0122e+08	5.9058e-04	8.9752e-03
111	1.0122e+08	7.2903e-04	3.1489e-03
112	1.0122e+08	4.1612e-04	6.6078e-03

Figure 1: Iteration values of the FW algorithm execution.

The evolution of the Objective Function and the Relative Gap can be better observed in Figures 1 and 2 respectively.

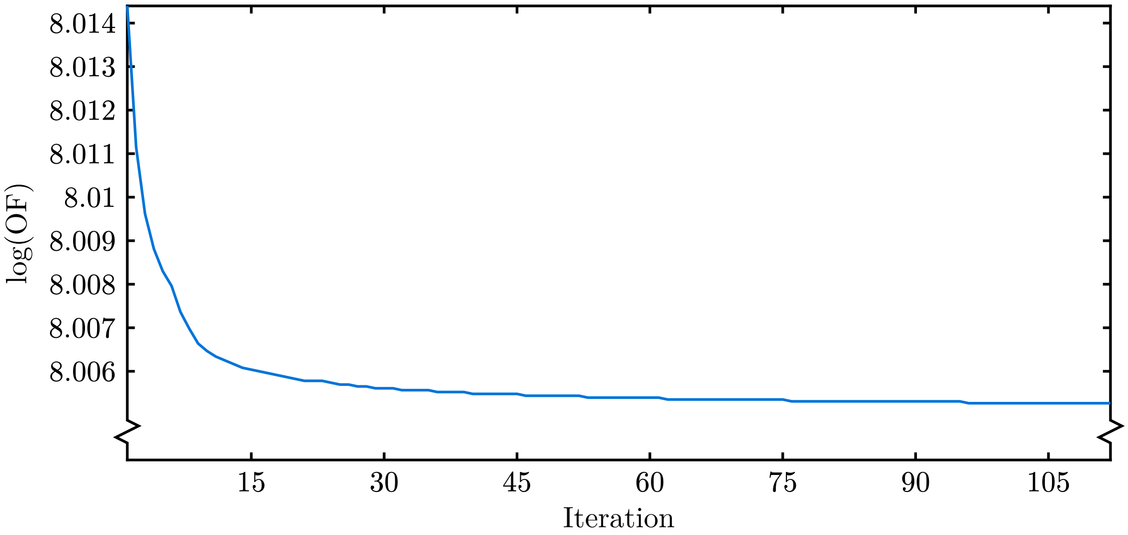


Figure 1: Evolution of the Objective Function (log) over the FW algorithm execution.

As mentioned before, the value of the objective function has a constant decreasing behavior, while the gap is much more chaotic and fluctuates constantly, although the overall tendency is still decreasing.

It is important to not how both values decrease significantly fast on the first iterations (~15), and then stagnates and progresses very slowly until reaching the accepted gap.

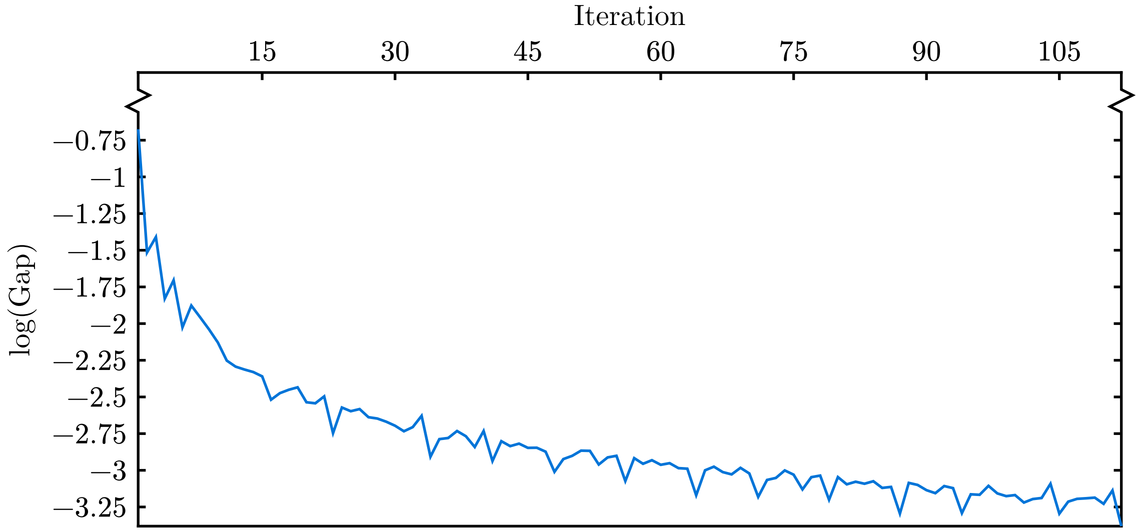


Figure 2: Evolution of the Relative Gap (log) over the FW algorithm execution.

4. RSD Algorithm

Now, we will solve the problem with the Restricted Simplicial Decomposition Algorithm, an extension of the FW algorithm used in the previous Section.

The RSD algorithm consists of these steps:

1) Initialize:
   - Initial feasible point x^0.
   - Lower Bound $BLB = -\infty$.
   - Working Sets $W_x^0 = \{x^0\}$ and $W_s^0 = \emptyset$.
2) Repeat while finishing criteria is not met (at iteration k):
   1) Solve linear approximation of the problem to obtain new point $\hat{x}^k$.
   2) Update Working Sets:
      - If $|W_s^k| < \rho$, then $W_s^{k+1} = W_s^k + \{\hat{x}^k\}$ and $W_x^{k+1} = W_x^k$.
      - Otherwise, let $\tilde{x}$ be the element in W_s^k for which x^k has the smaller barycentric coordinate, $W_s^{k+1} = W_s^k - \{\tilde{x}\} + \{\hat{x}^k\}$ and $W_x^{k+1} = \{x^k\}$.
   3) Obtain new point x^{k+1} within the convex hull of $W_s^k \cup W_x^k$ that minimizes the objective function.
   4) Purge elements from working sets for which x^{k+1} has null barycentric coordinates.

Step 1 of the loop is the same as in the FW algorithm, and is preformed solving the same optimization problem (LTAP), called the SubProblem in this algorithm. For step 3, we solve the so-called Master Problem, which for the problem at hand has the following formulation:

$$\begin{aligned}
\text{Min} \quad & \sum_{a\in A}(C_a x_a + 0.5 d x_a^2) & \\
\text{s.t.} \quad & \sum_{i=1}^{|W|} \alpha^i = 1 & \\
& \sum_{i=1}^{|W|} \alpha^i W_a^i = x_a & \forall a \in A \\
& x_a \geq 0 & \forall a \in A \\
& \alpha^i \geq 0 & i = 1, \ldots, |W|
\end{aligned} \tag{MP}$$

Where the α variables are the barycentric coordinates used in steps 2 and 4 of the loop. To avoid precision errors, the purging of the working sets has been performed with a tolerance of $1e^{-8}$ (i.e. if the its corresponding barycentric coordinate is lower than $1e^{-8}$, the vertex is purged).

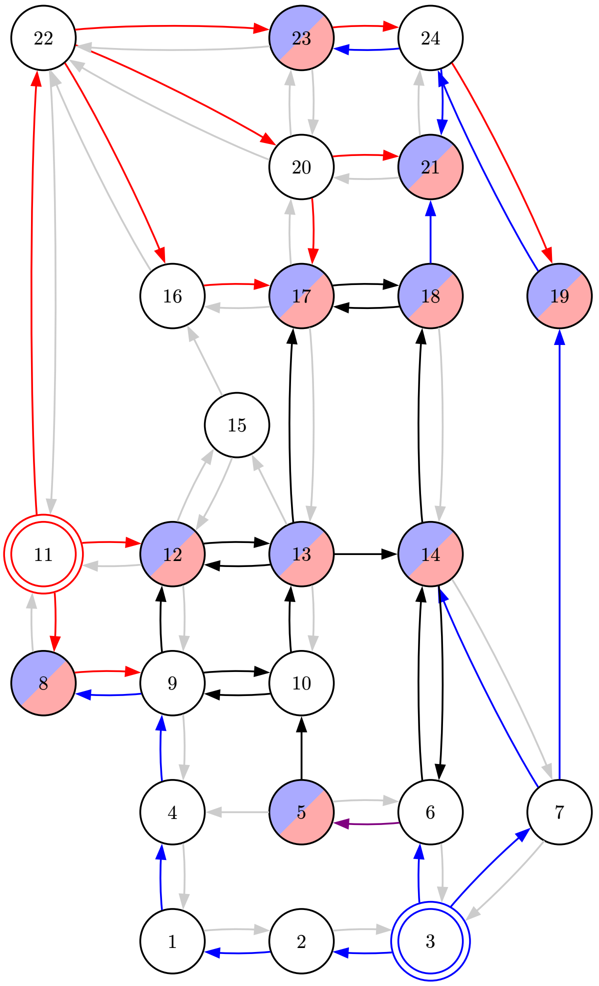
The solution obtained required 22 iterations (against the 112 of the FW), and has a total cost of 101192206, only 10549 higher than the optimal solution found in Section 2 (and 26760 lower than the FW solution), which corresponds to a relative gap of $1.04e^{-4}$.

The solution can be seen in Summary 3. Due to the fact that (MP) deals with the aggregated flows, only the aggregated arc flows are shown, instead of the segregated ones as in the previous sections. In the network, black arcs represent links carrying flow whose origin is not known for certain (may be only one or both).

Total Cost	101192206
Optimal Cost	101181657
Gap	10549
Relative Gap	1.04e-4
Iterations	22

General Flows	
Arc	**Flow**
1 → 4	19796.42
2 → 1	19796.42
3 → 2	19796.42
3 → 6	39136.57
3 → 7	41067.02
4 → 9	19796.42
5 → 10	10203.58
6 → 5	30203.58
6 → 14	18932.98
7 → 14	16456.90
7 → 19	24610.11
8 → 9	44.59
9 → 8	10000.00
9 → 10	44.59
9 → 12	9871.24
10 → 9	74.82
10 → 13	10173.35
11 → 8	10044.59
11 → 12	41895.81
11 → 22	48059.60

General Flows	
Arc	**Flow**
12 → 13	31895.81
13 → 12	128.76
13 → 14	20000.00
13 → 17	1940.40
14 → 6	10000.00
14 → 18	25389.89
16 → 17	11519.28
17 → 18	10000.00
18 → 17	10000.00
18 → 21	5389.89
19 → 24	14610.11
20 → 17	6540.31
20 → 21	10000.00
22 → 16	11519.28
22 → 20	16540.31
22 → 23	20000.00
23 → 24	10000.00
24 → 19	10000.00
24 → 21	4610.11
24 → 23	10000.00



Summary 3: Results of the RSD algorithm. Origin nodes have a colored double border, while destination nodes are filled with the origin color. Used links are painted with the color of the

origin whose flow they carry (purple for both, black for unknown), while unused links are greyed. Exact flows are shown in flow tables.

Table 1 shows the evolution of the objective function (Obj. Fun.), relative gap (Rel. Gap), number of vertices used (#V), and the barycentric coordinates (α_0 to α_4) through the iterations (I) of the RSD algorithm execution.

In the AMPL model implemented, the working set W for the Master Problem is a static set of 5 points with a bit matrix indicating which points are being used. Thus, we see in the table gaps in the barycentric coordinates such as for iteration 8, in which there are no α_2 nor α_3 but α_4 does exist.

I	Obj. Fun.	Rel. Gap	#V	α_0	α_1	α_2	α_3	α_4
1	1.0678e+08	2.3375e-01	2	6.5181e-01	3.4819e-01	-	-	-
2	1.0325e+08	6.1067e-02	3	4.6005e-01	1.7407e-01	3.6589e-01	-	-
3	1.0217e+08	2.6871e-02	4	3.1679e-01	1.4355e-01	3.9284e-01	1.4682e-01	-
4	1.0197e+08	1.6027e-02	5	3.2064e-01	8.8664e-02	2.6981e-01	1.0771e-01	2.1318e-01
5	1.0180e+08	1.4318e-02	5	5.8407e-01	6.2395e-03	1.5814e-01	3.8712e-08	2.5156e-01
6	1.0158e+08	1.2190e-02	4	7.7403e-01	3.1376e-03	-	4.8819e-02	1.7401e-01
7	1.0146e+08	9.2601e-03	5	6.9795e-01	1.6802e-08	5.1003e-02	3.8587e-02	2.1246e-01
8	1.0144e+08	8.8463e-03	3	8.7409e-01	1.5904e-02	-	-	1.1001e-01
9	1.0140e+08	6.1110e-03	4	4.9129e-01	1.4913e-02	2.2569e-01	-	2.6810e-01
10	1.0133e+08	5.4214e-03	5	3.1506e-01	2.1577e-02	2.9519e-01	1.2551e-01	2.4267e-01
11	1.0129e+08	5.0341e-03	4	7.2928e-01	1.9426e-08	-	1.1928e-01	1.5144e-01
12	1.0123e+08	3.5978e-03	5	5.3837e-01	2.3875e-02	1.1834e-01	1.1505e-01	2.0436e-01
13	1.0121e+08	1.5930e-03	5	9.3659e-01	6.6742e-03	1.5691e-03	2.3967e-02	3.1196e-02
14	1.0121e+08	1.2870e-03	5	8.3094e-01	1.5047e-03	5.5636e-02	2.8608e-02	8.3306e-02
15	1.0120e+08	1.1609e-03	5	8.2805e-01	1.1110e-02	4.6207e-02	4.2427e-02	7.2207e-02
16	1.0120e+08	8.5769e-04	5	8.7960e-01	1.1505e-03	3.9665e-02	2.0365e-02	5.9216e-02
17	1.0120e+08	7.7632e-04	5	9.4461e-01	3.6736e-03	1.4303e-02	1.3200e-02	2.4216e-02
18	1.0120e+08	5.9812e-04	5	9.6450e-01	3.7055e-03	1.5995e-02	1.3126e-05	1.5789e-02
19	1.0120e+08	5.6483e-04	5	9.1969e-01	5.2066e-03	2.3485e-02	1.7469e-02	3.4153e-02
20	1.0120e+08	5.5661e-04	5	9.0087e-01	8.9335e-04	3.2634e-02	1.6772e-02	4.8831e-02
21	1.0119e+08	5.3966e-04	5	9.0573e-01	6.1300e-03	2.5371e-02	2.3251e-02	3.9517e-02
22	1.0119e+08	4.7592e-04	-	-	-	-	-	-

Table 1: Iteration values of the RSD algorithm execution.

A more clear view of the evolution of the objective function and the relative gap can be seen Figures 3 and 4. As in the FW algorithm the objective functions decreases very steeply at the beginning (until iteration 4), and much slower afterwards. The behavior of the relative gap is also similar, having slight fluctuations, although not as pronounced as before (also probably due to the reduced number of iterations in the plot).

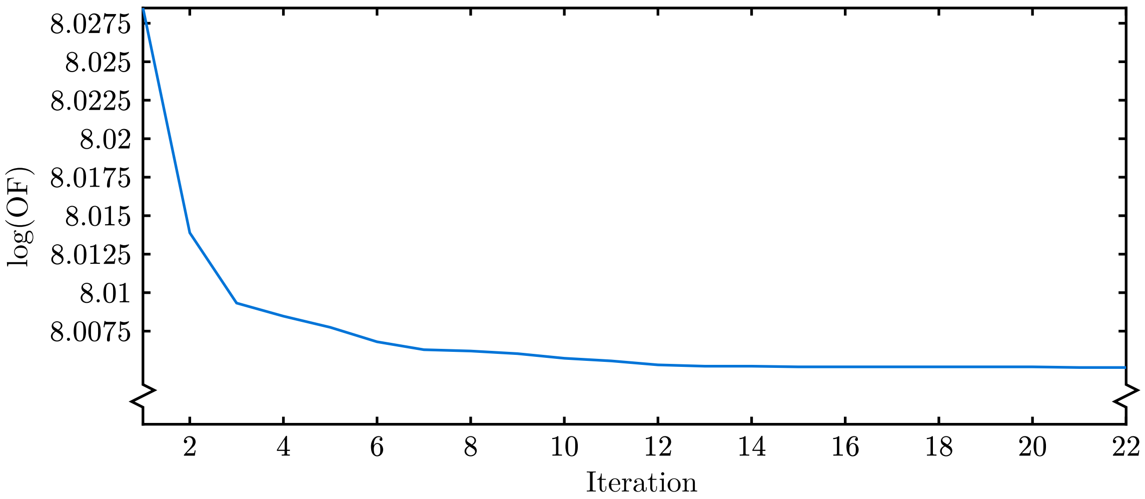


Figure 3: Evolution of the Objective Function (log) over the RSD algorithm execution.

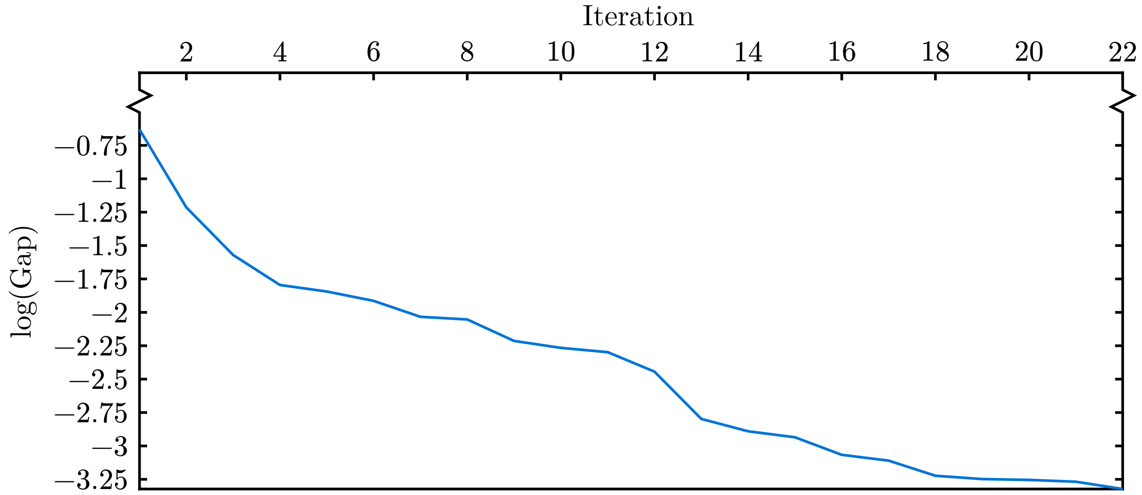


Figure 4: Evolution of the Relative Gap (log) over the RSD algorithm execution.

Figure 5 shows the evolution of the objective function for both the FW and the RSD algorithms, and we can clearly see the drastic difference in performance between the two algorithms for the problem solved.

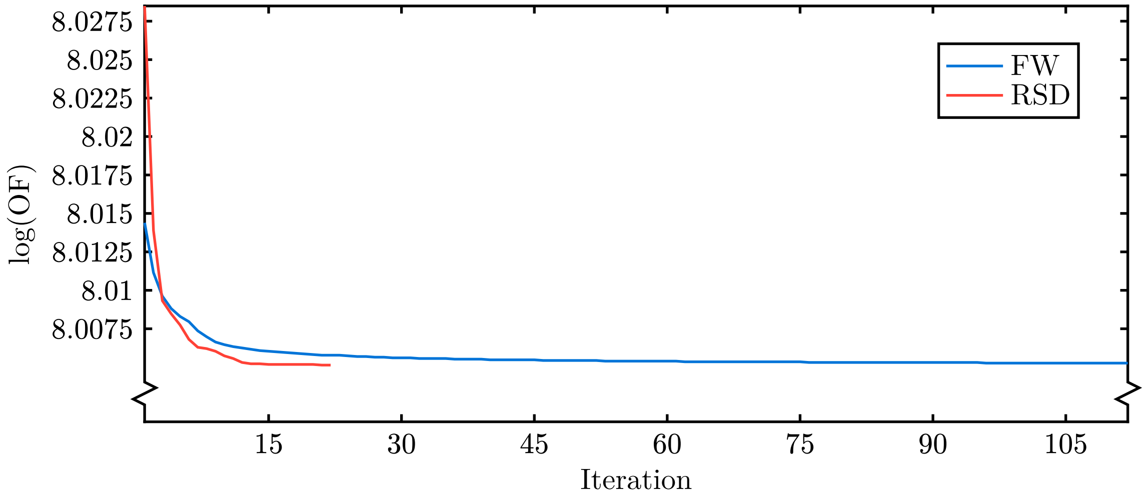


Figure 5: Comparison between the Objective Function (log) over the FW and RSD algorithms executions.

5. Wardrop Equilibrium

A traffic network reaches Wardrop Equilibrium if changing the path of any single flow unit (traveler, vehicle, etc.) only increases its travel time.

While the solutions found with both FW and RSD algorithms may be close to Wardrop Equilibrium, it is clear the FW one is not: it contains cycles (see arcs $(9 \to 10)$ and $(10 \to 9)$) that could be removed to decrease the overall cost of the solution. However, the solution found with the base formulation in Section 2 should be in Wardrop Equilibrium.

We will perform a partial check on the optimal solution and analyze several paths to see if they are optimal paths for their OD pairs and whether they are used. In particular, for each OD pair in our scenario, Table 2 shows two possible paths, their cost, and whether they are used in the optimal solution found (see Summary 1).

As can be seen, for all the OD pairs, the paths selected either have the same cost and are both used in the solution, or they have different cost and the more expensive one is not used (it also happens to happen that the cheaper one is used, but it is because in all cases, it happens to be the optimal one).

O	D	Path	Cost	Used
3	5	3 → 6 → 5	444.130435	✅
3	5	3 → 7 → 14 → 6 → 5	803.260870	❌
3	8	3 → 2 → 1 → 4 → 9 → 8	809.000000	✅
3	8	3 → 6 → 5 → 10 → 9 → 8	809.130435	❌
3	12	3 → 2 → 1 → 4 → 9 → 12	816.000000	✅
3	12	3 → 6 → 5 → 10 → 13 → 12	820.822742	❌
3	13	3 → 6 → 5 → 10 → 13	724.822742	✅
3	13	3 → 2 → 1 → 4 → 9 → 10 → 13	921.384615	❌
3	14	3 → 6 → 14	440.260870	✅
3	14	3 → 7 → 14	440.260870	✅
3	17	3 → 6 → 5 → 10 → 13 → 17	873.613951	❌
3	17	3 → 6 → 14 → 18 → 17	801.782609	✅
3	18	3 → 6 → 14 → 18	670.782609	✅
3	18	3 → 7 → 14 → 18	670.782609	✅
3	19	3 → 7 → 19	538.347826	✅
3	19	3 → 6 → 14 → 18 → 21 → 24 → 19	1073.304348	❌
3	21	3 → 6 → 14 → 18 → 21	807.304348	✅
3	21	3 → 7 → 19 → 24 → 21	807.304348	✅
3	23	3 → 7 → 19 → 24 → 23	833.826087	✅
3	23	3 → 6 → 14 → 18 → 21 → 20 → 23	1006.304348	❌
11	5	11 → 12 → 13 → 14 → 6 → 5	1015.769231	✅
11	5	11 → 8 → 9 → 4 → 1 → 2 → 3 → 6 → 5	1085.515050	❌
11	8	11 → 8	142.692308	✅
11	8	11 → 12 → 9 → 8	502.670330	❌
11	12	11 → 12	268.670330	✅

O	D	Path	Cost	Used
11	12	11 → 8 → 9 → 12	381.384615	✗
11	13	11 → 12 → 13	486.769231	✅
11	13	11 → 8 → 9 → 10 → 13	486.769231	✅
11	14	11 → 12 → 13 → 14	652.769231	✅
11	14	11 → 8 → 9 → 4 → 1 → 2 → 3 → 6 → 14	1081.645485	✗
11	17	11 → 12 → 15 → 16 → 17	635.560440	✅
11	17	11 → 22 → 16 → 17	635.560440	✅
11	18	11 → 22 → 16 → 17 → 18	766.560440	✅
11	18	11 → 12 → 13 → 14 → 18	883.290970	✗
11	19	11 → 22 → 23 → 24 → 19	823.637363	✅
11	19	11 → 12 → 13 → 14 → 18 → 21 → 24 → 19	1285.812709	✗
11	21	11 → 22 → 23 → 24 → 21	765.115624	✗
11	21	11 → 22 → 20 → 21	648.598901	✅
11	23	11 → 22 → 23	529.637363	✅
11	23	11 → 12 → 13 → 17 → 20 → 23	841.560440	✗

Table 2: Path analysis for Wardrop Equilibrium Check on Optimal Assignment.

The script to get the costs of all the paths is provided in `paths.run`[1].

A quick show that the FW solution doesn't have Wardrop Equilibrium is that the paths chosen for the OD pair (3, 13) have much different cost, and are still both used (see Summary 2):

O	D	Path	Cost	Used
3	13	3 → 6 → 5 → 10 → 13	723.160714	✅
3	13	3 → 2 → 1 → 4 → 9 → 10 → 13	908.839286	✅

Table 3: FW Wardrop Equilibrium Check.

6. Conclusions

In this report we have solved a scenario of a Traffic Assignment Problem using three different methods: solving directly non-linear optimization formulation, using the Frank Wolfe Algorithm (FW), and using the Restricted Simplicial Decomposition Algorithm (RSD).

Experimental results show that the RSD is significantly more efficient than the FW, and reaches a better solution in a fraction of the iterations. However, none of the algorithms provide a solution for the problem that satisfy the Wardrop Equilibrium, which the non-linear optimization problem does. This is checked by performing a path analysis between all the OD pairs in the scenario.

[1] Which has been generated with a Python script from a series of manually created paths, also provided in `paths.py`