

TIPE: Modélisation et optimisation d'un réseau de transport

Vadim HEMZELLE-DAVIDSON

Numéro de candidat: 36070

Travail en groupe réalisé avec Maxime BONCOUR

Numéro de candidat: 36814

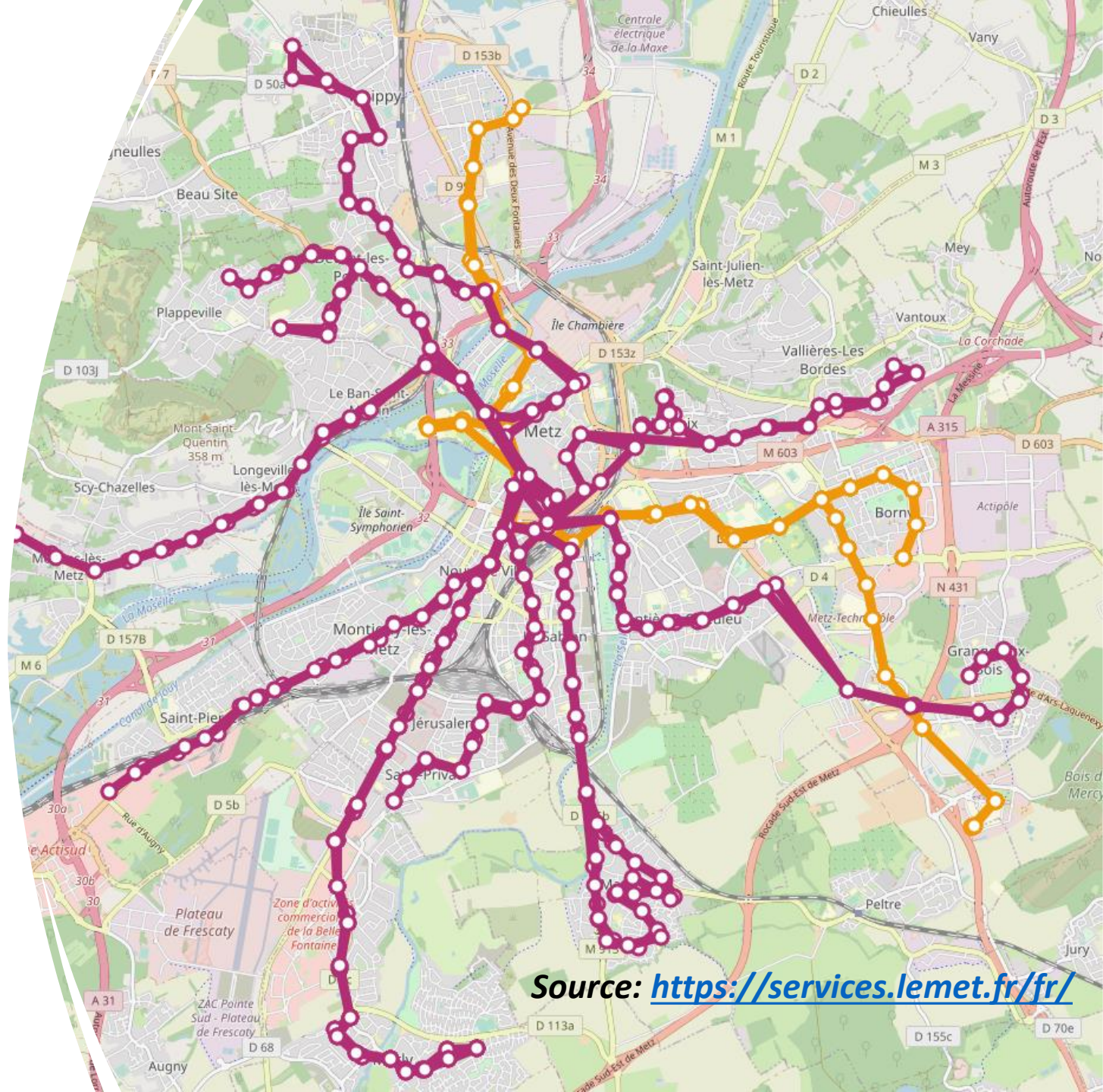
2022-2023

Problématique

- Comment proposer, à l'aide d'une base de données et de modèles connus, une modélisation mathématique et informatique d'un réseau de transport en évaluant sa précision par rapport à la réalité ?

Sommaire

- I. Présentation du réseau, organisation des données
- II. Modélisation par un graphe
- III. Algèbre tropicale et plus court chemin



Source: <https://services.lemet.fr/fr/>

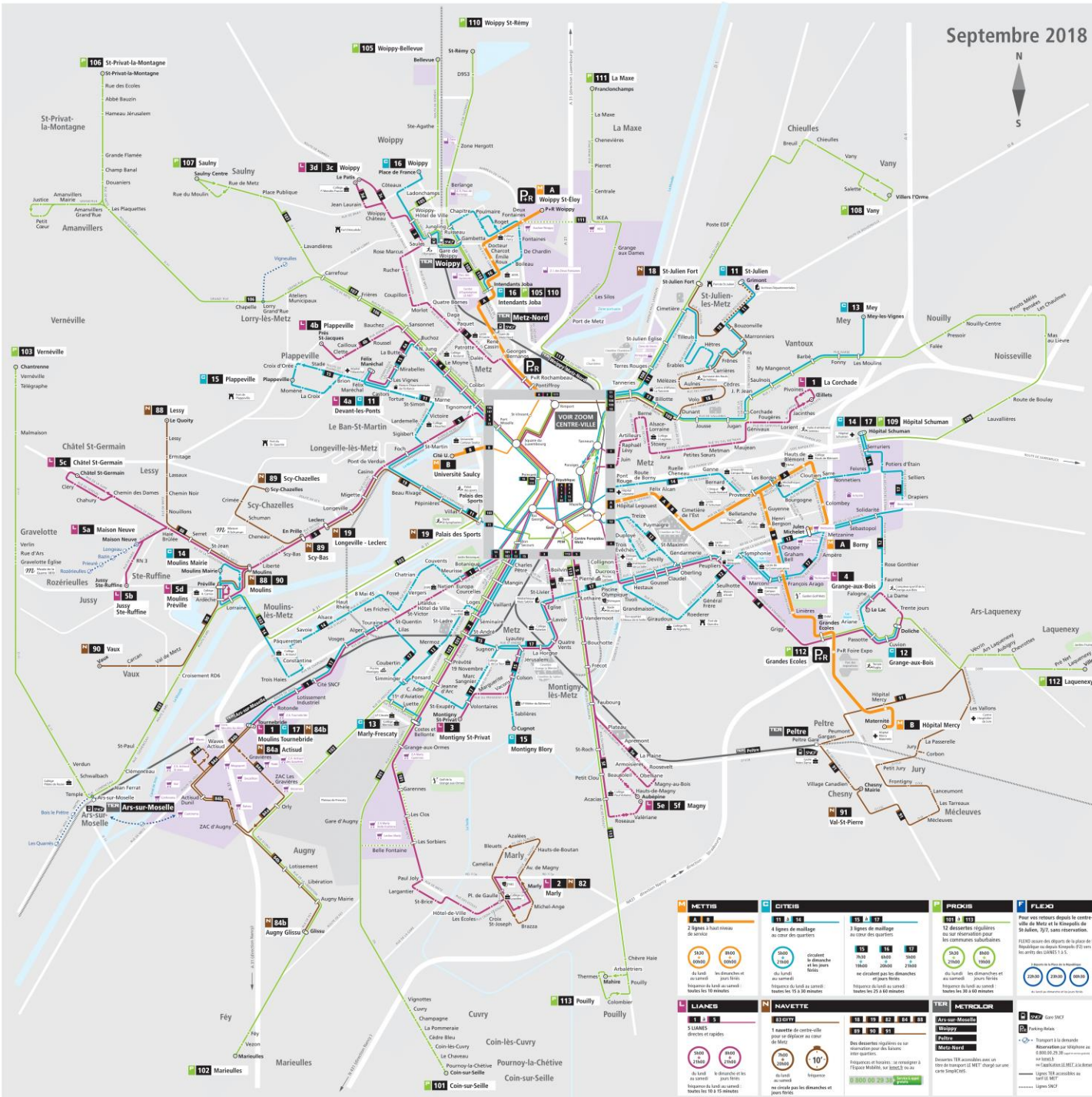
Présentation du réseau

- 26 lignes pour le réseau principal:

- Mettis
- Lianes (sans différenciations)
- Citeis
- Proxis

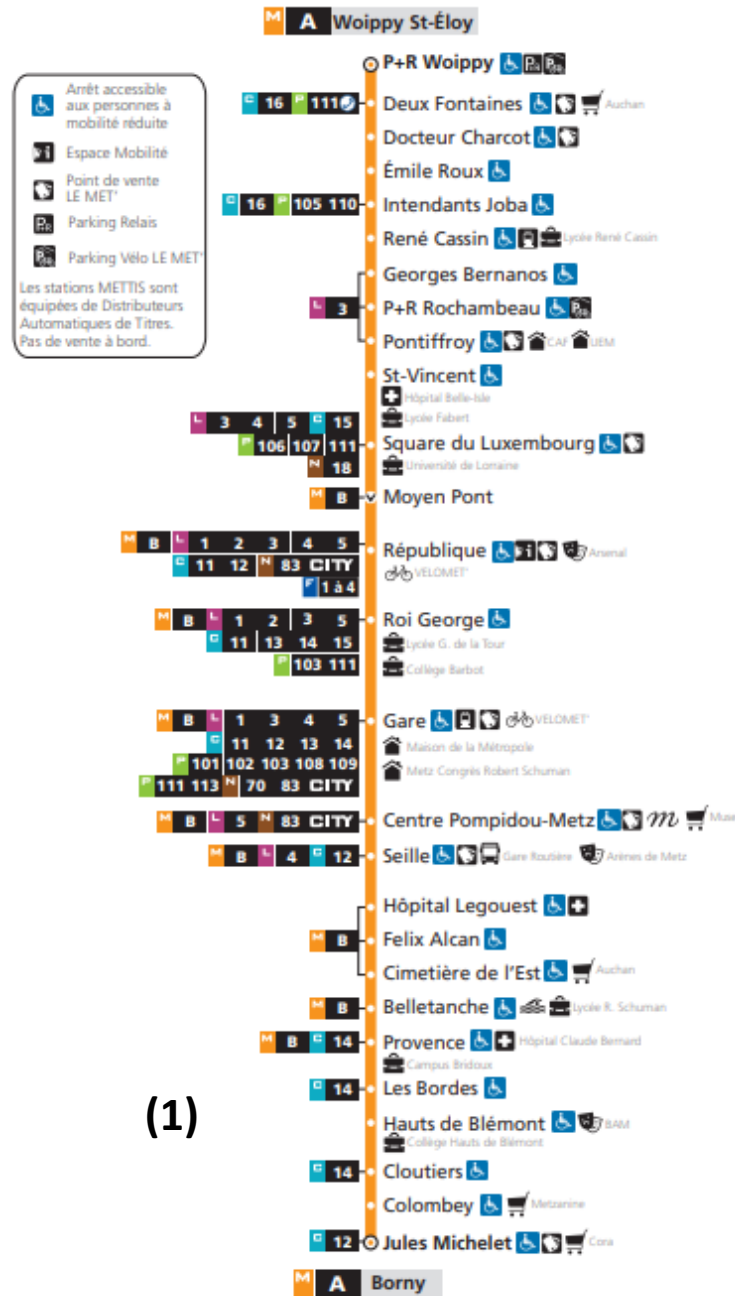
- 18 lignes secondaires:

- Flexo
- Navette
- Metz'O



Données relatives au réseau

5



Ressources GTFS

Transport - Données GTFS

10/03/2022 → 03/07/2022

05/05/2022

100%

49 informations lors de la validation

Visualisation des données disponible !

bus

GTFS autres formats

+ détails

Télécharger

Sources:

<https://www.lemet.fr/horaires/> (1)

<https://transport.data.gouv.fr/datasets/fichiers-gtfs-eurometropole-de-metz> (2)

(2)

Lignes retenues pour le visuel (lignes à plus fortes fréquences)

- Mettis A
- Mettis B
- L1
- L2
- L3
- L4a
- L4b
- L5e
- L5f

Exemple: 8 premiers arrêts de la ligne 2: fichier l2.csv

	A	B	C
1	stop_id,stop_code,stop_name,stop_de:		
2	6120,, "REPUBLIQUE",,49.114995,6.17338		
3	6130,, "ROI GEORGE",,49.110651,6.17161		
4	92,, "BON SECOURS",,49.107180,6.168711		
5	93,, "CHARLES PETRE",,49.105362,6.16672		
6	94,, "MANGIN",,49.102570,6.164369,,http		
7	95,, "VAILLANT",,49.099700,6.161980,,htl		
8	96,, "SEMINAIRE",,49.097280,6.159970,,h		

Fichier stops.txt

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_lon,  
57,, "8 MAI 45",,49.099805,6.138669,,https://services.lem  
46,, "8 MAI 45",,49.099927,6.138371,,https://services.lem  
450,, "11ème D'AVIATION",,49.085890,6.146239,,https://serv  
852,, "11ème D'AVIATION",,49.086159,6.145297,,https://serv  
928,, "19 NOVEMBRE",,49.091774,6.161323,,https://services  
929,, "19 NOVEMBRE",,49.092028,6.161481,,https://services  
519,, "DEUX FONTAINES",,49.148061,6.171578,,https://servi  
451,, "DEUX FONTAINES",,49.148956,6.171576,,https://servi  
530,, "TRENTÉ JOURS",,49.096271,6.245860,,https://services  
536,, "TRENTÉ JOURS",,49.096403,6.245779,,https://services  
179,, "TROIS EVECHES",,49.106303,6.191888,,https://servic  
101,, "TROIS HAIES",,49.090659,6.123135,,https://services  
115,, "TROIS HAIES",,49.090713,6.122693,,https://services  
161,, "QUATRE BORNES",,49.139202,6.153451,,https://servic  
157,, "QUATRE BORNES",,49.139403,6.153449,,https://servic
```

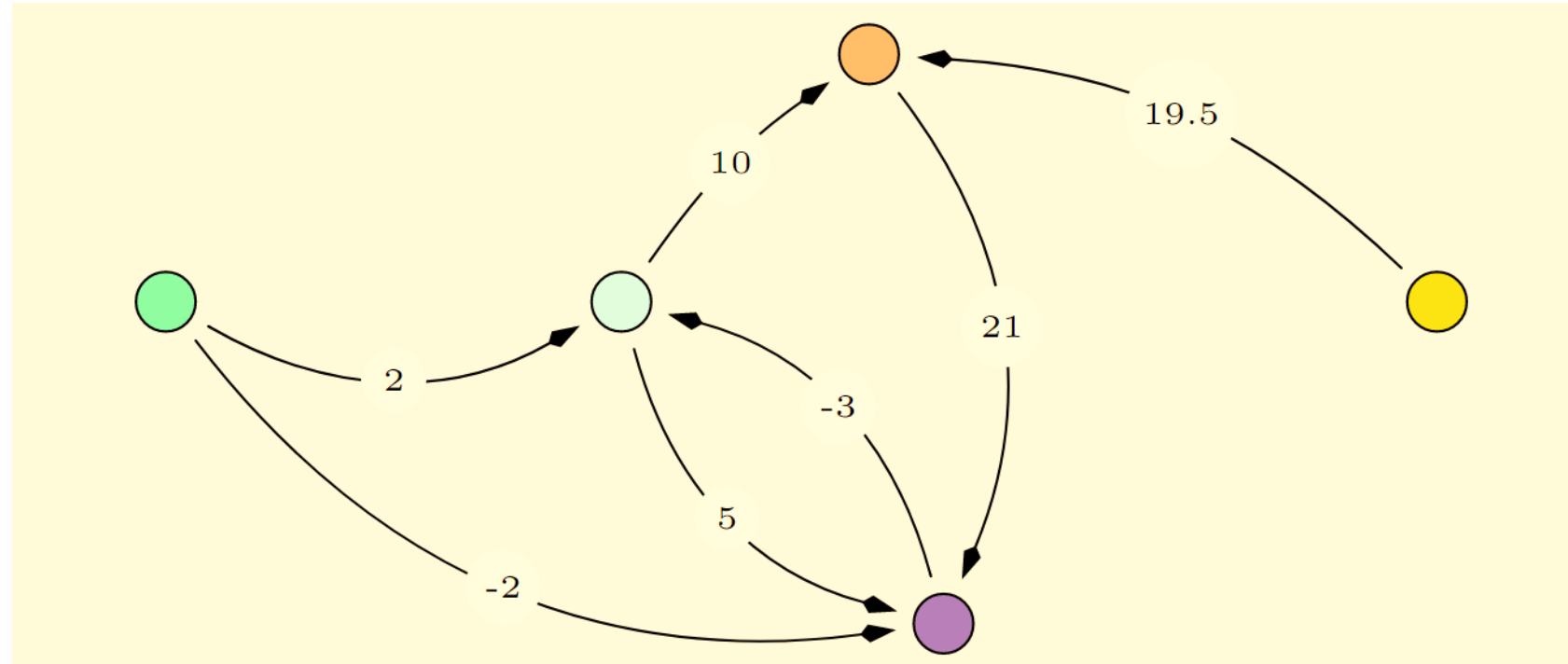

Fichier trips.txt

trips.csv

```
1 route_id,service_id,trip_id,trip_headsign,direction_id,block_id,shape_id
2 B-194,HIV2223-HDim2223-Dimanche-40,835666-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
3 B-194,HIV2223-HDim2223-Dimanche-40,835667-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
4 B-194,HIV2223-HDim2223-Dimanche-40,835668-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
5 B-194,HIV2223-HDim2223-Dimanche-40,835669-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
6 B-194,HIV2223-HDim2223-Dimanche-40,835670-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
7 B-194,HIV2223-HDim2223-Dimanche-40,835671-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116698,B0018
8 B-194,HIV2223-HDim2223-Dimanche-40,835672-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
9 B-194,HIV2223-HDim2223-Dimanche-40,835673-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
10 B-194,HIV2223-HDim2223-Dimanche-40,835674-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116698,B0018
11 B-194,HIV2223-HDim2223-Dimanche-40,835675-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
12 B-194,HIV2223-HDim2223-Dimanche-40,835676-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
13 B-194,HIV2223-HDim2223-Dimanche-40,835677-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116698,B0018
14 B-194,HIV2223-HDim2223-Dimanche-40,835678-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
15 B-194,HIV2223-HDim2223-Dimanche-40,835679-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
16 B-194,HIV2223-HDim2223-Dimanche-40,835680-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116698,B0018
17 B-194,HIV2223-HDim2223-Dimanche-40,835681-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116697,B0018
18 B-194,HIV2223-HDim2223-Dimanche-40,835682-HIV2223-HDim2223-Dimanche-40,"MB - HOPITAL MERCY",0,116699,B0018
```


Modélisation par un graphe orienté

- Graphe $G = (S, A)$
- Pondérations:
 - Temps de trajet
 - **Distances entre arrêts**
 - Nombre de passagers entrants
 - « Coût de franchissement »



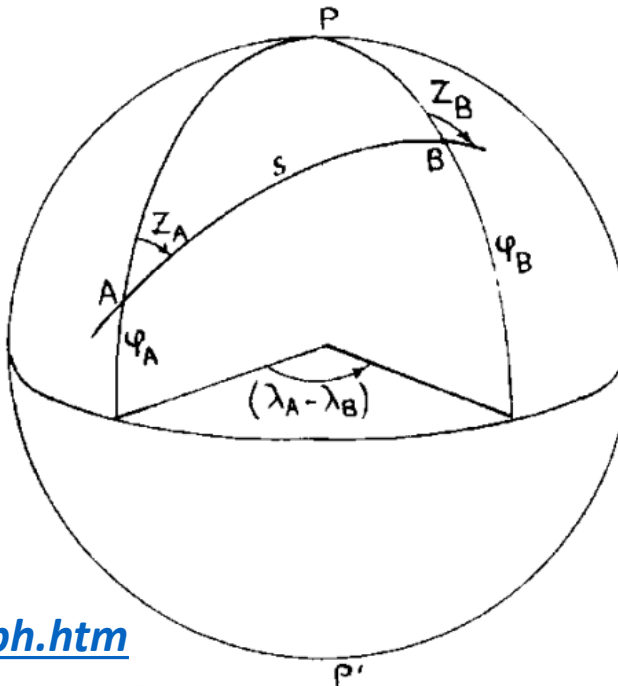
Source: <https://iagoleal.com/posts/algebraic-path/>

Pondération (calcul des distances)

- On considère deux points A et B de **latitudes** φ_A et φ_B et de **longitudes** λ_A et λ_B
- **Distance angulaire** en radians:
$$S_{A-B} = \arccos(\sin \varphi_A \sin \varphi_B + \cos \varphi_A \cos \varphi_B \cos(|\lambda_B - \lambda_A|))$$

Conversion en mètres:

$$d = S_{A-B} \times R_T$$



Source:

<https://www.zpag.net/math/GeoAEDistSph.htm>

Structure du code

Tri des sommets	Tri des arêtes
<ul style="list-style-type: none">• Parcours de stops.csv• Séparation des données de chaque ligne• Si l'arrêt n'a pas été découvert:<ul style="list-style-type: none">• Ajout des arrêts et de ses coordonnées dans une liste• Création du nom d'affichage• Reconversion de la liste en csv	<ul style="list-style-type: none">• Parcours des fichiers csv de chaque ligne (11)• Identification du prédécesseur et du successeur de chaque arrêt• Pondération• Reconversion en csv

```
stop_id,stop_code,stop_name,stop_desc,stop_la  
57,, "8 MAI 45",,49.099805,6.138669,,https://s  
46,, "8 MAI 45",,49.099927,6.138371,,https://s  
450,, "11ème D'AVIATION",,49.085890,6.146239,,  
852,, "11ème D'AVIATION",,49.086159,6.145297,,
```

Fichier bus_metz.csv (sommets)

bus_metz_backup.csv

```
1  name,lat,lon,display_name
2  "P+R WOIPPY",49.150349,6.173323,P+R<br>WOIPPY
3  "DEUX FONTAINES",49.149369,6.172123,DEUX<br>FONTAINES
4  "DOCTEUR CHARCOT",49.148334,6.166976,DOCTEUR<br>CHARCOT
5  "EMILE ROUX",49.144727,6.166327,EMILE<br>ROUX
6  "INTENDANTS JOBA",49.141569,6.165333,INTENDANTS<br>JOBA
7  "RENE CASSIN",49.135897,6.165947,RENE<br>CASSIN
8  "GEORGES BERNANOS",49.133318,6.168920,GEORGES<br>BERNANOS
9  "P+R ROCHAMBEAU",49.129663,6.170075,P+R<br>ROCHAMBEAU
10 "PONTIFFROY",49.127140,6.174876,PONTIFFROY
```

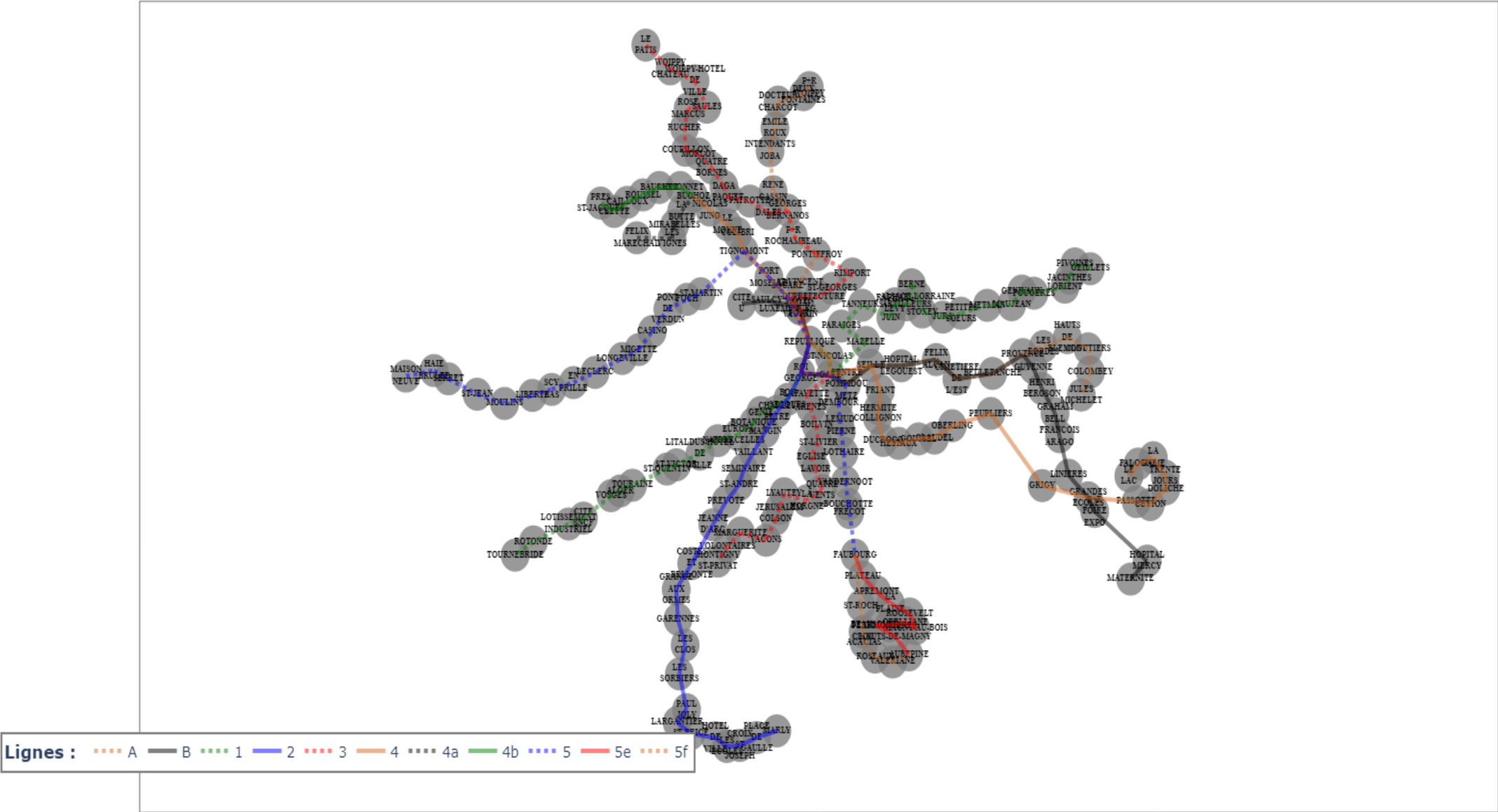

Fichier bus_metz_lignes2.csv (arêtes)

 bus_metz_lignes2.csv

```
1 line,station_name1,station_name2,distance
2 MB - HOPITAL MERCY,CITE U,SAULCY,375
3 MB - HOPITAL MERCY,SAULCY,MOYEN PONT,492
4 MB - HOPITAL MERCY,MOYEN PONT,REPUBLIQUE,557
5 MB - HOPITAL MERCY,REPUBLIQUE,ROI GEORGE,500
6 MB - HOPITAL MERCY,ROI GEORGE,GARE,437
7 MB - HOPITAL MERCY,GARE,CENTRE POMPIDOU METZ,277
8 MB - HOPITAL MERCY,CENTRE POMPIDOU METZ,SEILLE,469
9 MB - HOPITAL MERCY,SEILLE,HOPITAL LEGUEST,448
10 MB - HOPITAL MERCY,HOPITAL LEGUEST,FELIX ALCAN,526
```

Représentation visuelle finale

Lignes de Bus à Metz



Algèbre tropicale

- **Définition 1 (semi-anneau):** Un **semi-anneau** est un ensemble E muni d'une loi additive \oplus associative et commutative de neutre 0 et d'une loi multiplicative \otimes associative de neutre 1 vérifiant:
 - $\forall (a, b, c) \in E^3, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ et $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$
 - $\forall a \in E, a \oplus 0 = 0 \oplus a = 0$
- **Définition 2 (dioïde):** Un semi-anneau (E, \oplus, \otimes) est un **dioïde** si:
 - $\forall x \in E, x \oplus x = x$ (idempotence)

Dioïde min-plus

$$a \oplus b = \min\{a, b\}$$

$$a \otimes b = a + b$$

$$0 = \infty$$

$$1 = 0$$

- **Produit matriciel:** $(A \otimes B)_{i,j} = \bigoplus_{k=1}^n a_{i,k} \otimes b_{k,j}$

- **Réécriture de l'équation précédente:** $V = I \oplus (A \otimes V)$

Equation de Bellman

- Soit S l'ensemble fini des sommets du graphe.
 - On définit $V: S \times S \rightarrow (-\infty, +\infty]$ tel que:
 - $\forall s \in S, V(s, s) = 0$
 - $\forall s \neq t, V(s, t) = \min\{A(s, q) + V(q, t), q \in S\}$
 - On définit la matrice de terme général $I(s, t) = \begin{cases} 0, s = t \\ +\infty, s \neq t \end{cases}$
- Ainsi $V(s, t) = \min\{I(s, t), \min\{A(s, q) + V(q, t), q \in S\}\}$

Quasi-inverse d'un élément, d'une matrice

- Pour un élément a :

$$a^* = \lim_{n \rightarrow +\infty} \bigoplus_{k=0}^n a^k$$

- Pour une matrice A :

$$A^* = \lim_{n \rightarrow +\infty} \bigoplus_{k=0}^n A^k$$

Système linéaire associé au plus court chemin à origine unique

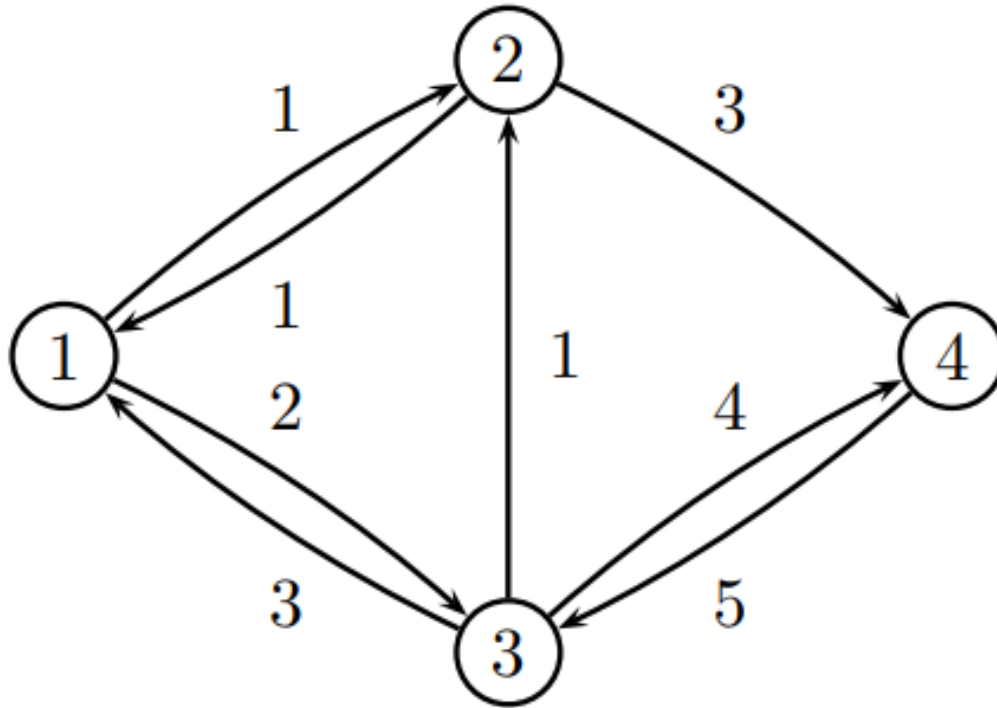
- $\delta(i, j) = \text{distance entre } i \text{ et } j$ (0 si $i = j$, $+\infty$ si i non relié à j)
- Plus court chemin reliant i_0 aux autres arrêts du réseau:

$$\begin{cases} X_0 = (\delta(i_0, i))_{i \leq n} \\ \forall n \in \mathbb{N}, X_{n+1} = X_n \otimes A \oplus X_0 \end{cases}$$

- Unicité de A^* , qui vérifie:

$$A^* = I \oplus (A \otimes A^*)$$

Matrice d'adjacence d'un graphe dans l'algèbre min-plus



$$\begin{pmatrix} 0 & 1 & 2 & \infty \\ 1 & 0 & \infty & 3 \\ 3 & 1 & 0 & 4 \\ \infty & \infty & 5 & 0 \end{pmatrix}$$

Source: <https://antoine.delignat-lavaud.fr/doc/tropical.pdf>

Applications en optimisation

Problème	E	\oplus	\otimes	0	1
Chemin de capacité maximum	$\mathbb{R}^+ \cup \{+\infty\}$	max	min	0	$+\infty$
Plus court chemin	$\mathbb{R} \cup \{+\infty\}$	min	+	$+\infty$	0
Plus long chemin	$\mathbb{R} \cup \{+\infty\}$	max	+	$-\infty$	0
Fiabilité d'un réseau	Polynômes idempotents	Différence symétrique	\times	0	1
Chemin de nombre d'arcs minimum	$\mathbb{N} \cup \{+\infty\}$	min	+	$+\infty$	0

Source: *Graphes, dioïdes et semi-anneaux, nouveaux modèles et algorithmes, Michel Gondran, Michel Minoux*

Matrice d'adjacence liée au réseau

matrice_adjacence.csv

```
1  0,139,inf,inf,inf,inf,inf,inf,inf,inf,
2  139,0,392,inf,inf,inf,inf,inf,inf,inf,
3  inf,392,0,404,inf,inf,inf,inf,inf,inf,
4  inf,inf,404,0,358,inf,inf,inf,inf,inf,
5  inf,inf,inf,358,0,632,inf,inf,inf,inf,
6  inf,inf,inf,inf,632,0,359,inf,inf,inf,
7  inf,inf,inf,inf,inf,359,0,415,inf,inf,
8  inf,inf,inf,inf,inf,inf,415,0,448,inf,
9  inf,inf,inf,inf,inf,inf,inf,448,0,499,
10 inf,inf,inf,inf,inf,inf,inf,inf,499,0,
```

Calcul de A^* : algorithme de Floyd-Warshall

- Algorithme en $O(n^3)$

Floyd-Warshall(A)

```
1   $D^0 \leftarrow A$ 
2  for  $k \leftarrow 1$  to  $n$ 
3      do for  $i \leftarrow 1$  to  $n$ 
4          do for  $j \leftarrow 1$  to  $n$ 
5              do  $D_{ij}^k \leftarrow \min\{D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}\}$ 
```

- D_{ij}^k contient le poids du plus court chemin du sommet i au sommet j parmi les chemins qui utilisent uniquement les nœuds $1, \dots, k$ comme nœuds intermédiaires.

Source: <https://terpconnect.umd.edu/~baras/publications/Books/S00245ED1V01Y201001CNT003.pdf>

Résultat: A^* pour le graphe du réseau

	A	B	C	D
1	0,4894,17450,4816,7983,9667,4789,1766,7326,4435,			
2	4807,0,17778,2086,8311,9775,5069,6573,7654,2950,			
3	17003,17409,0,16475,16510,20333,15455,18769,158			
4	4741,2166,16847,0,7380,8140,4138,6507,6723,1315,			
5	7904,8310,16878,7376,0,11234,6356,9670,3415,6895,			
6	9420,9607,20463,7969,10996,0,6145,11186,10339,67			
7	5391,5729,16434,4795,6967,4878,0,7157,6310,3742,			
8	1806,6700,19066,6622,9789,11473,6595,0,9132,6241			
9	7290,7696,16264,6762,3458,10620,5742,9056,0,6285			
10	4398,2852,16274,1214,6807,6825,3278,6164,6150,0,			

Interprétation

15	4000,3220,13331
16	3594,1702,17626
17	752,5646,18202,!
18	6002,6408,16038
19	15246,15652,163

Entrez le nom de votre arrêt de départ : TIGNOMONT

Entrez le nom de votre arrêt d'arrivée : ST-MARTIN

La distance du chemin le plus court est : 752

Un trajet direct est possible avec la ligne : L5 - MAISON NEUVE

Lignes prioritaires et vitesse

- Ordre de priorité:
 - Mettis
 - Lianes
 - Citeis
 - Proxis
 - Navettes

Vitesse moyenne	18 km/h ²
Réseaux connexes	LE MET'

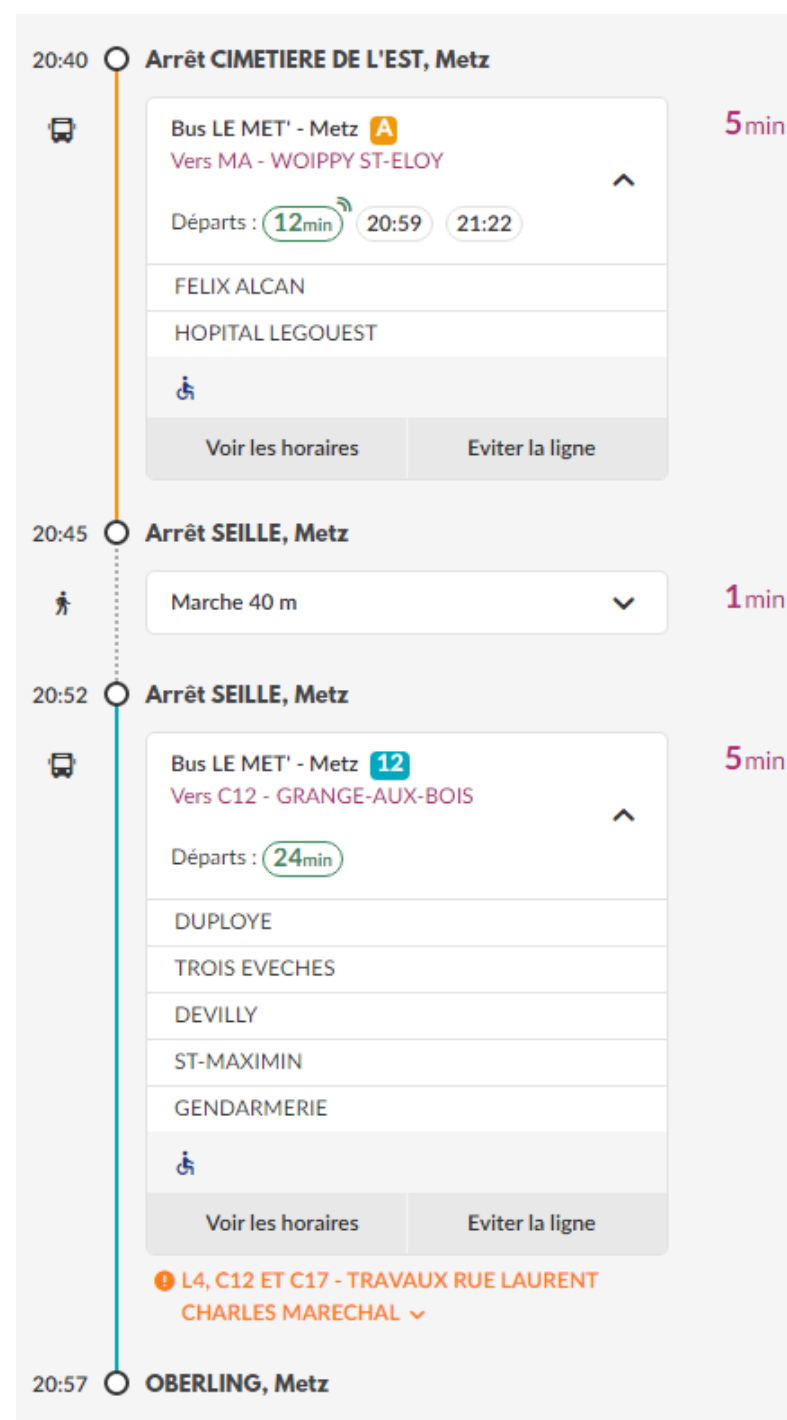
Source: [https://fr.wikipedia.org/wiki/Mettis_\(bus_à_haut_niveau_de_service\)](https://fr.wikipedia.org/wiki/Mettis_(bus_à_haut_niveau_de_service))

Exemple

```
Entrez le nom de votre arrêt de départ : CIMETIERE DE L'EST
Entrez le nom de votre arrêt d'arrivée : OBERLING
La distance du chemin le plus court est : 3114
Un trajet direct n'est pas possible. Calcul du trajet le plus court...
Le trajet est le suivant :
CIMETIERE DE L'EST -> FELIX ALCAN -> HOPITAL LEGUEST -> SEILLE -> DUPLOYE -> TROIS EVECHES -> DEVILLY -> ST-MAXIMIN -> GENDARMERIE -> OBERLING
Entre CIMETIERE DE L'EST et FELIX ALCAN : Ligne MB - HOPITAL MERCY
Entre FELIX ALCAN et HOPITAL LEGUEST : Ligne MB - HOPITAL MERCY
Entre HOPITAL LEGUEST et SEILLE : Ligne MB - HOPITAL MERCY
Entre SEILLE et DUPLOYE : Ligne C12 - GRANGE-AUX-BOIS
Entre DUPLOYE et TROIS EVECHES : Ligne C12 - GRANGE-AUX-BOIS
Entre TROIS EVECHES et DEVILLY : Ligne C12 - GRANGE-AUX-BOIS
Entre DEVILLY et ST-MAXIMIN : Ligne C12 - GRANGE-AUX-BOIS
Entre ST-MAXIMIN et GENDARMERIE : Ligne C12 - GRANGE-AUX-BOIS
Entre GENDARMERIE et OBERLING : Ligne C12 - GRANGE-AUX-BOIS
Le temps estimé du trajet est de : 12.379999999999999 minute(s)
```

Comparaison avec LeMet'

Source: <https://services.lemet.fr/fr/>



Cas d'un trajet direct




Source: <https://services.lemet.fr/fr/>

Trajet annoncé par le programme...

```
Entrez le nom de votre arrêt de départ : GARE
Entrez le nom de votre arrêt d'arrivée : PEUPLIERS
La distance du chemin le plus court est : 3513
Un trajet direct est possible avec la ligne : 4
Le trajet est le suivant :
GARE -> SEILLE -> FRIANT -> HERMITE -> COLLIGNON -> DUCROCQ -> HESTAUX -> GOUSSEL -> CLAUDEL -> OBERLING -> PEUPLIERS
Le temps estimé du trajet est de : 11.709999999999999 minute(s)
```

...et par LeMet'

19:00 ○ Arrêt GARE, Metz

 Bus LE MET' - Metz **4**
Vers L4 - GRANGE-AUX-BOIS

Départs : 19:00 19:30 20:02

SEILLE

FRIANT

HERMITE

COLLIGNON

DUCROCQ

HESTAUX

GOUSSEL

CLAUDEL

OBERLING

Voir les horaires Eviter la ligne

13min

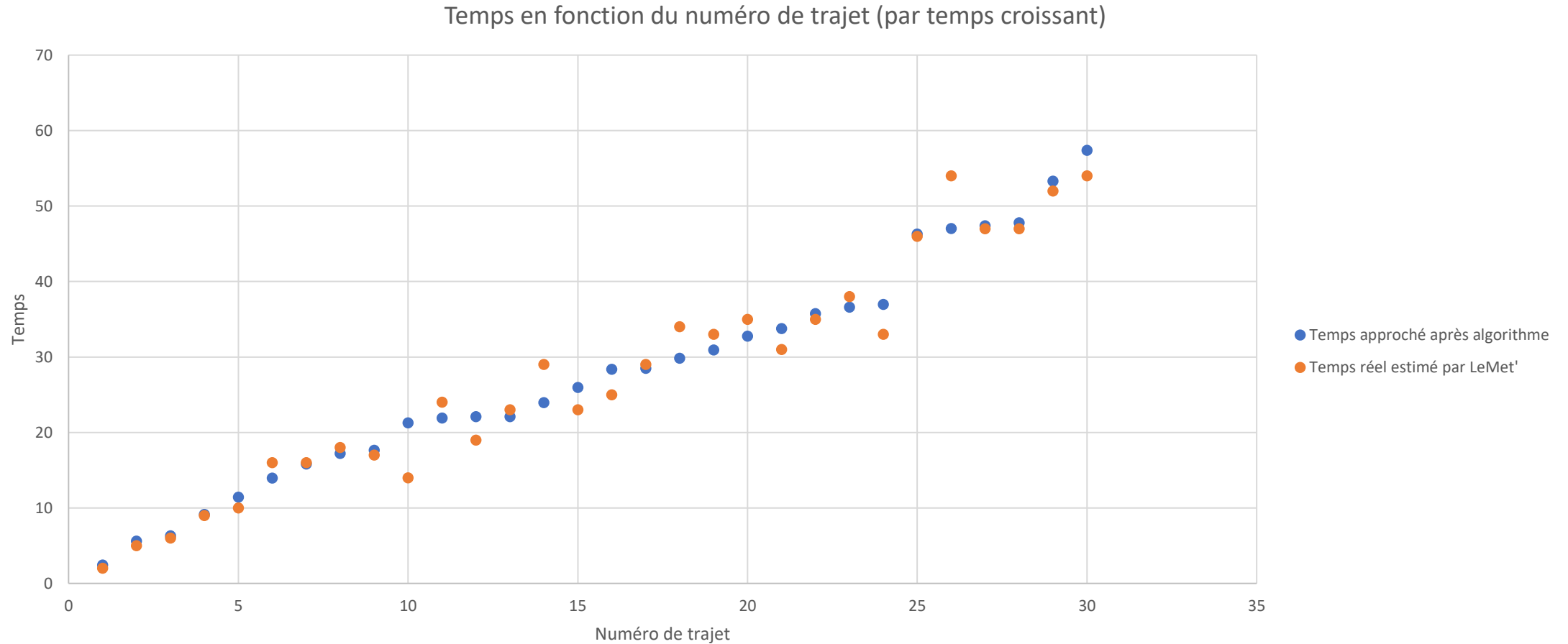
19:13 ○ PEUPLIERS, Metz

! L4, C12 ET C17 - TRAVAUX RUE LAURENT CHARLES MARECHAL

Simulation de trajets aléatoires

```
travel_times.csv
1 Depart,Arrivee,Temps
2 GRANGE AUX ORMES,PAUL JOLY,6.303333333333334
3 HOTEL DE VILLE,BOUCHOTTE,36.61333333333333
4 GRANDES ECOLES,MARLY,53.29666666666667
5 MORLOT,LES ECOLES,46.293333333333334
6 PROVENCE,JUIN,21.28
7 ST-VICTOR,LONGEVILLE,29.82
8 COLSON,ST-VINCENT,21.913333333333334
9 LES CLOS,SANSONNET,35.753333333333334
10 SEMINAIRE,QUATRE BORNES,28.360000000000003
11 FAUBOURG,OBERLING,22.090000000000003
12 BOILVIN,CIMETIERE DE L'EST,13.956666666666667
13 HOPITAL MERCY,HAIE BRULEE,57.39
14 MOYEN PONT,JERUSALEM,17.66
15 CROIX ST JOSEPH,GRIGY,47.01
16 MAZELLE,ROTONDE,22.106666666666667
17 PLATEAU,EMILE ROUX,33.760000000000005
18 HAUTS DE BLEMONT,BUCHOZ,28.500000000000004
19 ROOSEVELT,NICOLAS JUNG,32.776666666666667
20 ST-GEORGES,INTENDANTS JOBA,11.44
21 TRENTE JOURS,LORIENT,47.79
22 ROUSSEL,PIERNE,25.966666666666665
23 COLOMBEY,EGLISE,23.96
24 ROSEAUX,LOTISSEMENT INDUSTRIEL,36.95666666666667
25 REPUBLIQUE,MANGIN,9.156666666666666
26 ST-ANDRE,JEANNE D'ARC,2.4233333333333333
27 CLAUDEL,VANDERNOOT,17.216666666666667
28 GARENNES,SERRET,47.39
29 HERMITE,MOULINS,30.943333333333333
30 LAFAYETTE,SEILLE,5.5933333333333334
31 BOTANIQUE,QUATRE VENTS,15.803333333333333
```

Comparatif avec les résultats du Met



Causes d'erreurs

- Travaux
- Trajets prix à une heure précise sur LeMet'
- Distances à vol d'oiseau
- Trajets piétons
- Correspondances
- Vitesse uniforme

Source: <https://services.lemet.fr/fr/>

19:21 ○ BOTANIQUE, Metz

Marche 1,2 km 13min


19:36 ○ Arrêt EGLISE, Metz

Bus LE MET' - Metz 3 2min
Vers L3 - MONTIGNY ST-PRIVAT

Départs : 20:14 20:32 20:50

! L3 - Travaux rue de la folie

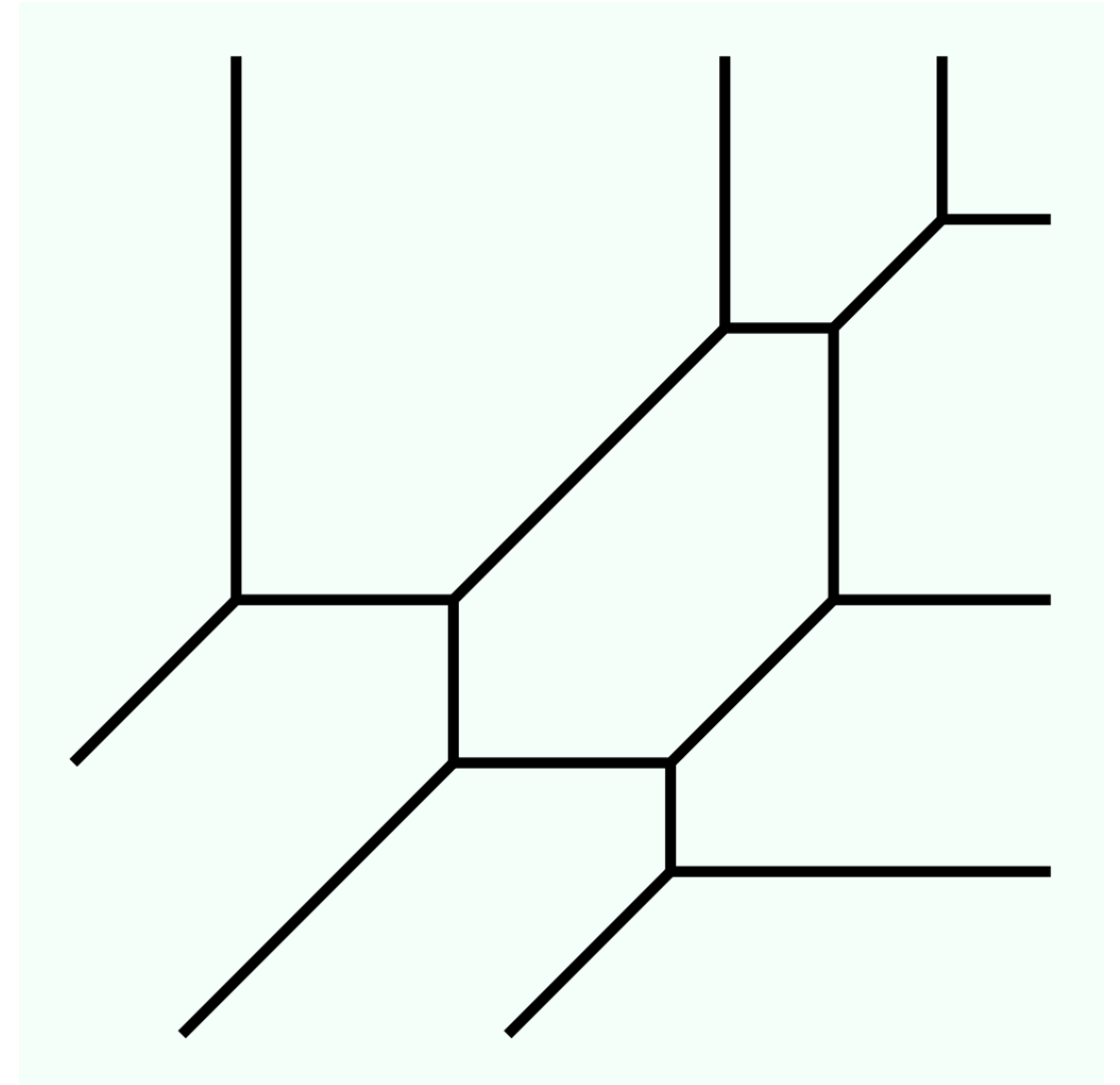
En raison de travaux, du MARDI 9 MAI AU VENDREDI 7 JUILLET 2023, les arrêts "Daga" de la ligne L3 seront reportés sur les arrêts "Paquet" dans les deux sens.



19:38 ○ QUATRE VENTS, Metz

Conclusion

- Résolution du problème du plus court chemin de manière linéaire par changement d'algèbre
- Calcul de la semi-inverse à l'aide de la série géométrique de la matrice du graphe: algorithmes de Floyd-Warshall, Dijkstra, Dantzig...
- Polymorphisme des algorithmes
- Mathématiques tropicales



Source: https://en.wikipedia.org/wiki/Tropical_geometry

Lignes.py

```
1 import csv
2 from math import *
3
4
5 url = [
6     ['A', 'Lignes\mettis a.csv'],
7     ['B', 'Lignes\mettis b.csv'],
8     ['1', 'Lignes\l1.csv'],
9     ['2', 'Lignes\l2.csv'],
10    ['3', 'Lignes\l3.csv'],
11    ['4', 'Lignes\l4 normal.csv'],
12    ['4a', 'Lignes\l4a.csv'],
13    ['4b', 'Lignes\l4b.csv'],
14    ['5', 'Lignes\l5 normal.csv'],
15    ['5e', 'Lignes\l5e.csv'],
16    ['5f', 'Lignes\l5f.csv']
17 ]
18
19 arrets = []
20
21 # Séparation selon la ligne empruntée
22 def ajoutligne(lignebus):
23     f = open(lignebus[1])
24     lignecompt = 0
25     for ligne in f:
26         tempo = []
27         if (lignecompt >= 1):
28             x = ligne.split(",")
29             tempo.append(lignebus[0])
30             tempo.append(x[2])
31             arrets.append(tempo)
32
33         lignecompt += 1
34
35     f.close()
36
37
38 for k in range(len(url)):
39     ajoutligne(url[k])
40
41
42 fic = open('bus_metz_lignes.csv', 'w', newline='')
43
44 writer = csv.writer(fic)
45 writer.writerow(('line', 'station_name1', 'station_name2', 'distance'))
46 writer = csv.writer(fic, lineterminator='\n',
47                     quoting=csv.QUOTE_NONE, quotechar=None)
48
49 # Détermination de l'arrêt suivant
50 def arretsuivant(arrets, i):
51     line, arret = arrets[i]
52     if (i+1) < len(arrets):
53
54         linesuiv, arretsuiv = arrets[i + 1]
55         if line == linesuiv:
56             return True, arretsuiv
```

```
57         else:
58             return False, arretsuiv
59
60 # Conversion degré / radian
61 def deg2rad(dd):
62
63     return float(dd)/180*pi
64
65 # Calcul de la distance entre 2 arrêts
66 def distance(arret,arretsuiv):
67
68     fichier = open('bus_metz.csv')
69     lignecompteur = 0
70     for ligne in fichier:
71         x = ligne.split(",")
72         if lignecompteur >= 1:
73
74             if x[0] == arret:
75                 latA = deg2rad(x[1])
76                 longA = deg2rad(x[2])
77
78                 if x[0] == arretsuiv:
79                     latB = deg2rad(x[1])
80                     longB = deg2rad(x[2])
81
82                 lignecompteur += 1
83             RT = 6378137
84             S = acos(sin(latA) * sin(latB) + cos(latA) * cos(latB) * cos(abs(longB - longA)))
85             # distance entre les 2 points, comptée sur un arc de grand cercle
86             fichier.close()
87             return int(S * RT)
88
89 # Créer la première liste des arrêtes
90 for i in range(len(arrets)):
91
92     line, arret = arrets[i]
93     statut, arretsuiv = arretsuivant(arrets, i)
94
95     newarret = arret.replace("'", '')
96     newarretsuiv = arretsuiv.replace("'", '')
97
98     if statut == True:
99         dist = distance(arret, arretsuiv)
100         writer.writerow((line , newarret , newarretsuiv,dist))
101
102 fic.close()
103
```

Visualisation.py

```

1  import networkx as nx
2  import pandas as pa
3  import plotly.graph_objects as go
4
5  T = pa.read_csv('bus_metz_lignes.csv')
6
7  G = nx.from_pandas_edgelist(T, source='station_name1', target='station_name2',
8  edge_attr='distance')
9
10 # Lecture des données
11 M = pa.read_csv('bus_metz_backup.csv')
12 print(M)
13 fig = go.Figure()
14
15 # Tracé du graphe
16 fig.add_trace(
17     go.Scattergeo(
18         mode='markers + text',
19         lon=M['lon'],
20         lat=M['lat'],
21         text=M['display_name'].apply(lambda x: \
22                                     '<b>' + str(x) \
23                                     + '</b>'),
24         textposition='middle center',
25         textfont=dict(color="black", size=8, family='bold'),
26         showlegend=False,
27         marker={
28             'symbol': 'circle-dot',
29             'size': 30,
30             'opacity': 0.8,
31             'color': 'gray'
32         }
33     )
34 )
35 fig.update_layout(
36     showlegend=True,
37     legend={
38         'x': 0,
39         'y': 0.1,
40         'title': '<b>Lignes : </b>',
41         'orientation': 'h',
42         'bordercolor': 'gray',
43         'font': {'family': 'Verdana', 'size': 14},
44         'borderwidth': 2,
45     },
46     title={
47         'text': "<b>Lignes de Bus Ã Metz</b>",
48         'font': {'family': 'Verdana'},
49         'y': 0.93,
50         'x': 0.5
51     }
52 )
53 fig.update_geos(
54     fitbounds='locations',
55     showland=True,
56     landcolor='white',
57     projection_type='natural earth'
58 )
59
60 N = M.groupby('name').first()

```



```
56 #print(N.head())
57
58 T['x1'] = [N.loc[n, 'lon'] for n in T['station_name1']]
59 T['y1'] = [N.loc[n, 'lat'] for n in T['station_name1']]
60 T['x2'] = [N.loc[n, 'lon'] for n in T['station_name2']]
61 T['y2'] = [N.loc[n, 'lat'] for n in T['station_name2']]
62
63 #print(T.head())
64
65 couleurs = ['black', 'green', 'blue', 'red', 'chocolate']
66 dot = ['dot', 'solid']
67 L= []
68 ListeLigne = ["A","B","1","2","3","4","4a","4b","5","5e","5f"]
69 compteur=0
70 for i in ListeLigne:
71     A = T.query('line == @i')
72     for k in A.index:
73         fig.add_trace(
74             go.Scattergeo(
75                 mode = 'lines',
76                 lon = [A.loc[k, 'x1'], A.loc[k, 'x2']],
77                 lat = [A.loc[k, 'y1'], A.loc[k, 'y2']],
78                 hovertext = A['distance'],
79                 opacity= 0.5,
80                 showlegend= True if i not in L else False,
81                 name = i,
82                 line = {'color' : couleurs[compteur%5 - 1],
83                        'dash' : dot[compteur%2],
84                        'width' : 4}
85             )
86         )
87     L.append(i)
88     compteur += 1
89
90 fig.show()
91
92
```

nouvelles lignes.py

```
1 import csv
2 from math import *
3
4 # Utilisation de trips.csv
5 fic2 = open('trips.csv')
6 f2 = csv.reader(fic2)
7 liste_trip_id = []
8
9 for ligne in f2:
10     if ligne[3] == 'trip_headsign':
11         continue
12     if ligne[3] not in liste_trip_id:
13         liste_trip_id.append(ligne[3])
14
15 fic2.close()
16
17 # Conversion degré / radian
18 def deg2rad(dd):
19     return float(dd)/180*pi
20
21 # Calcul de la distance entre 2 arrêts
22 def distance(arret,arretsuiv):
23
24     fichier = open('stops.csv')
25     f = csv.reader(fichier)
26     next(f)
27     for x in f:
28         if int(x[0]) == arret:
29             print(x)
30             latA = deg2rad(x[4])
31             longA = deg2rad(x[5])
32             nomA = x[2]
33         if int(x[0]) == arretsuiv:
34             latB = deg2rad(x[4])
35             longB = deg2rad(x[5])
36             nomB = x[2]
37
38     RT = 6378137
39     S = acos(sin(latA) * sin(latB) + cos(latA) * cos(latB) * cos(abs(longB - longA)))
40     # distance entre les 2 points, comptée sur un arc de grand cercle
41     fichier.close()
42     return (int(S * RT),nomA,nomB)
43
44 fic = open('bus_metz_lignes2.csv', 'w', newline='')
45
46 writer = csv.writer(fic)
47 writer.writerow(('line', 'station_name1', 'station_name2', 'distance'))
48 writer = csv.writer(fic, lineterminator='\n',
49                     quoting=csv.QUOTE_NONE, quotechar=None)
50
51 # Association de la distance au trajet correspondant et création des nouveaux csv
52 for elt in liste_trip_id:
53     fic = open('Lignes2/' + str(elt) + '.csv')
54     f = csv.reader(fic)
55     next(f) #Ignorer l'en-tête
56     liste = list(f)
```

```
57     print(liste)
58     for i in range(0,len(liste)-1):
59         stop_id = int(liste[i][0])
60         stop_id_suivant = int(liste[i+1][0])
61         d, nomA, nomB = distance(stop_id,stop_id_suivant)
62         writer.writerow((str(elt) , nomA , nomB, d))
63     fic.close()
64
65     print(liste_trip_id)
66
67
68
69
70
71
72
73
74
75
76
77
```

Plus court chemin.py

```

1  import csv
2  import math
3  from math import inf
4  import numpy as np
5  import random
6
7  # Chargement du fichier bus_metz_lignes.csv
8  with open('bus_metz_lignes2.csv', 'r') as file:
9      csv_reader = csv.reader(file)
10     lignes = list(csv_reader)[1:] # Ignorer les en-têtes
11
12  # Chargement du fichier bus_metz.csv
13  with open('bus_metz.csv', 'r') as file:
14      csv_reader = csv.reader(file)
15      bus = list(csv_reader)[1:] # Ignorer les en-têtes
16
17  # Création d'une liste contenant toutes les stations
18  stations = []
19  for ligne in lignes:
20      station1 = ligne[1]
21      station2 = ligne[2]
22      if station1 not in stations:
23          stations.append(station1)
24      if station2 not in stations:
25          stations.append(station2)
26
27  # Création d'un dictionnaire pour stocker les distances entre les stations
28  distances = {}
29
30  # Parcours des lignes et mise à jour du dictionnaire des distances
31  for ligne in lignes:
32      station1 = ligne[1]
33      station2 = ligne[2]
34      distance = int(ligne[3])
35
36      if station1 not in distances:
37          distances[station1] = {}
38      if station2 not in distances:
39          distances[station2] = {}
40
41      distances[station1][station2] = distance
42      distances[station2][station1] = distance
43
44  # Création de la matrice d'adjacence initialisée
45  num_stations = len(stations)
46  A = [[inf] * num_stations for _ in range(num_stations)]
47  for i in range(len(A)):
48      A[i][i] = 0
49
50  # Remplissage de la matrice d'adjacence à partir du dictionnaire des distances
51  for i, station1 in enumerate(stations):
52      for j, station2 in enumerate(stations):
53          if station2 in distances[station1]:
54              A[i][j] = distances[station1][station2]
55
56  # Fonction pour compter le nombre de coefficients non nuls dans une liste

```

```
57 def non_nul(L):
58     return sum(elem is not inf for elem in L)
59
60 # Recherche du nombre maximal de coefficients non nuls
61 max_count = 0
62 max_indices = []
63
64 for i, row in enumerate(A):
65     count = non_nul(row)
66     if count > max_count:
67         max_count = count
68         max_indices = [i]
69     elif count == max_count:
70         max_indices.append(i)
71
72 # Récupération des noms des stations correspondantes
73 station_names = [stations[i] for i in max_indices]
74
75 # Affichage des résultats
76 if max_count > 0:
77     print("Stations les plus fréquentées:")
78     for name in station_names:
79         print(name)
80 else:
81     print("La matrice est vide.")
82
83 # Sauvegarde de la matrice d'adjacence dans un fichier CSV
84 with open('matrice_adjacence.csv', 'w', newline='') as file:
85     csv_writer = csv.writer(file)
86     for row in A:
87         csv_writer.writerow(row)
88
89 # Créer des dictionnaires pour mémoriser les indices et les lignes de chaque arrêt
90 stop_indices = {}
91 stop_lines = {}
92
93 with open('bus_metz.csv', 'r') as f:
94     reader = csv.reader(f)
95     next(reader) # ignore header
96     for i, row in enumerate(reader):
97         name = row[0]
98         stop_indices[name] = i
99
100 with open('bus_metz_lignes2.csv', 'r') as f:
101     reader = csv.reader(f)
102     next(reader) # ignore header
103     for row in reader:
104         line, stop1, stop2, _ = row
105         if stop1 not in stop_lines:
106             stop_lines[stop1] = set()
107         if stop2 not in stop_lines:
108             stop_lines[stop2] = set()
109         stop_lines[stop1].add(line)
110         stop_lines[stop2].add(line)
111
112 # Créer une matrice d'adjacence
113 num_stops = len(stop_indices)
114 A = [[math.inf]*num_stops for _ in range(num_stops)]
115 for i in range(num_stops):
116     A[i][i] = 0
```

```

117
118 with open('bus_metz_lignes2.csv', 'r') as f:
119     reader = csv.reader(f)
120     next(reader) # ignore header
121     for row in reader:
122         _, stop1, stop2, distance = row
123         i = stop_indices[stop1]
124         j = stop_indices[stop2]
125         A[i][j] = int(distance)
126
127 # Algorithme de Floyd-Warshall adapté conservant le trajet
128 def floyd_warshall(matrix):
129     n = len(matrix)
130     dist = matrix.copy()
131     pred = [[None]*n for _ in range(n)]
132
133     for i in range(n):
134         for j in range(n):
135             if matrix[i][j] != math.inf:
136                 pred[i][j] = i
137
138     for k in range(n):
139         for i in range(n):
140             for j in range(n):
141                 if dist[i][j] > dist[i][k] + dist[k][j]:
142                     dist[i][j] = dist[i][k] + dist[k][j]
143                     pred[i][j] = pred[k][j]
144
145     return dist, pred
146
147 # Calcul de la quasi-inverse de la matrice
148 a_etoile, predecessors = floyd_warshall(A)
149
150 # Écrire la matrice dans un fichier CSV
151 with open('A_etoile.csv', 'w', newline='') as f:
152     writer = csv.writer(f)
153     writer.writerows(a_etoile)
154
155 # Reconstituer le chemin effectivement suivi
156 def reconstituer_chemin(predecessors, start_name, end_name):
157     # Convertir les noms d'arrêt en indices
158     start = stop_indices[start_name]
159     end = stop_indices[end_name]
160
161     # Initialiser le chemin avec l'arrêt de fin
162     path = [end]
163
164     # Remonter le chemin depuis la fin jusqu'au début
165     while path[-1] != start:
166         path.append(predecessors[start][path[-1]])
167
168     # Convertir les indices du chemin en noms d'arrêt
169     path = [list(stop_indices.keys())[i] for i in path[::-1]]
170
171     return path
172
173 # Demander à l'utilisateur son arrêt de départ et d'arrivée
174 start_name = input("Entrez le nom de votre arrêt de départ : ")
175 end_name = input("Entrez le nom de votre arrêt d'arrivée : ")
176

```

```
177 # Appeler la fonction pour récupérer le chemin le plus court
178 path = reconstituer_chemin(predecessors, start_name, end_name)
179
180 # Afficher la distance du plus court chemin
181 distance = a_etoile[stop_indices[start_name]][stop_indices[end_name]]
182 print('La distance du chemin le plus court est : {}'.format(distance))
183
184 def assign_weight(line):
185     if line.startswith('M'):
186         return 1
187     elif line.startswith('L'):
188         return 2
189     elif line.startswith('C'):
190         return 3
191     elif line.startswith('P'):
192         return 4
193     elif line.startswith('N'):
194         return 5
195     else:
196         return 6
197
198 def get_lines_taken(path):
199     # Initialiser une liste pour stocker les lignes prises
200     lines_taken = []
201
202     # Parcourir chaque arrêt du chemin
203     for i in range(len(path) - 1):
204         # Lire le fichier CSV ligne par ligne
205         with open('bus_metz_lignes2.csv', 'r') as f:
206             reader = csv.reader(f)
207             next(reader) # Skip header
208
209             # Initialiser le poids le plus petit à infini
210             smallest_weight = inf
211             ligne_a_prendre = None
212
213             for line, station_name1, station_name2, _ in reader:
214                 # Si une ligne a une liaison directe entre les deux arrêts
215                 if (station_name1 == path[i] and station_name2 == path[i+1]) or \
216                     (station_name1 == path[i+1] and station_name2 == path[i]):
217                     # Calculer le poids de la ligne
218                     weight = assign_weight(line)
219
220                     # Si le poids est plus petit que le poids actuel le plus petit
221                     if weight < smallest_weight:
222                         smallest_weight = weight
223                         ligne_a_prendre = line
224
225             # Ajouter la ligne avec le poids le plus petit à la liste des lignes prises
226             lines_taken.append(ligne_a_prendre)
227
228     return lines_taken
229
230
231 # Appeler la fonction pour récupérer les lignes empruntées
232 lines_taken = get_lines_taken(path)
233
234 # Définir le temps de trajet comme le temps direct + 2 minutes par correspondance
235 def temps_theorique_minute(start_name, end_name, vitesse):
```



```

236     t = (((a_etoile[stop_indices[start_name]][stop_indices[end_name]])*(10**
(-3))/vitesse)*60)+2*
((len(set(get_lines_taken(reconstituer_chemin(predecessors,start_name,end_name)))))-1)
237     return t
238
239 lines_stops = {}
240 with open('bus_metz_lignes2.csv', 'r') as f:
241     reader = csv.reader(f)
242     next(reader) # Skip header
243
244     for line, station_name1, station_name2, _ in reader:
245         if line not in lines_stops:
246             lines_stops[line] = [station_name1, station_name2]
247         else:
248             if lines_stops[line][-1] == station_name1:
249                 lines_stops[line].append(station_name2)
250             else:
251                 lines_stops[line].append(station_name1)
252
253
254 def trajet_direct(start_name, end_name):
255     # Vérifier si un trajet direct est possible
256     direct_line = stop_lines[start_name].intersection(stop_lines[end_name])
257
258     if direct_line:
259         direct_line = direct_line.pop()
260         line_stops = lines_stops[direct_line]
261
262         # Trouver les indices des arrêts de départ et d'arrivée
263         start_index = line_stops.index(start_name)
264         end_index = line_stops.index(end_name)
265
266         # Obtenir l'itinéraire direct
267         if start_index < end_index:
268             trajet = line_stops[start_index:end_index+1]
269         else:
270             trajet = line_stops[start_index:end_index-1:-1]
271
272         return True, direct_line, trajet
273     else:
274         return False, None, None
275
276 # Appeler la fonction pour vérifier si un trajet direct est possible
277 trajet_direct_possible, direct_line, trajet = trajet_direct(start_name, end_name)
278
279 if trajet_direct_possible:
280     # Si un trajet direct est possible, l'utiliser
281     print(f'Un trajet direct est possible avec la ligne : {direct_line}')
282     print('Le trajet est le suivant :')
283     print(' -> '.join(trajet))
284 else:
285     # Sinon, exécuter l'algorithme de Dijkstra
286     print('Un trajet direct n\'est pas possible. Calcul du trajet le plus court...')
287     path = reconstituer_chemin(predecessors, start_name, end_name)
288     print('Le trajet est le suivant :')
289     print(' -> '.join(path))
290     for i, line in enumerate(lines_taken):
291         print('Entre {} et {} : Ligne {}'.format(path[i], path[i+1], line))
292
293 print('Le temps estimé du trajet est de :',
temps_theorique_minute(start_name,end_name,18), 'minute(s)')

```

```
294
295 # Simulation de trajets aléatoire
296 def selection_arrets(file_name, num_stops):
297     with open(file_name, 'r') as f:
298         reader = csv.reader(f)
299         next(reader) # Skip header
300         all_stops = [row[0] for row in reader] # Obtenir tous les arrêts
301
302     arrets_aleatoires = random.sample(all_stops, num_stops) # En sélectionner une partie
aléatoirement
303     return arrets_aleatoires
304
305 def temps_de_trajet(stops):
306     travel_times = []
307     for i in range(0, len(stops)-1, 2): # Faire des paires d'arrêts pour créer des
trajets
308         start_name = stops[i]
309         end_name = stops[i+1]
310         time = temps_theorique_minute(start_name, end_name, 18)
311         travel_times.append((start_name, end_name, time))
312     return travel_times
313
314 def write_to_csv(file_name, data):
315     with open(file_name, 'w', newline='') as f:
316         writer = csv.writer(f)
317         writer.writerow(["Depart", "Arrivee", "Temps"])
318         writer.writerows(data)
319
320 arrets_aleatoires = selection_arrets('bus_metz.csv', 60)
321 travel_times = temps_de_trajet(arrets_aleatoires)
322 write_to_csv('travel_times.csv', travel_times)
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
```

352
353
354
355
356