

Modélisation informatique d'un réseau de transport urbain

Maxime BONCOUR

Travail en groupe avec Vadim HEMZELLEC-DAVIDSON

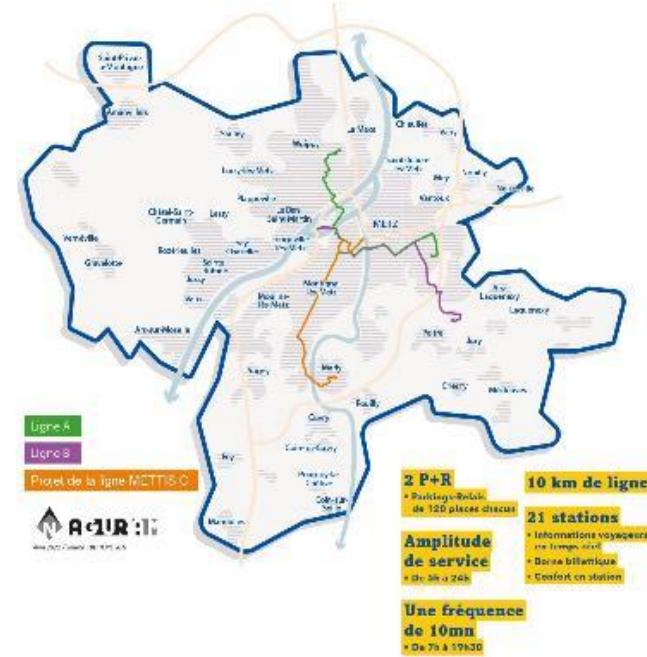
N° d'inscription : 36814

Objectif principal :

- ❖ Les bases de données permettent-elles la modélisation d'un réseau de transport pour trouver le chemin le plus court pour un déplacement et gérer les horaires afin d'éviter le phénomène des « trains de bus » ?

Sommaire :

- ❖ Approche à l'aide d'un **graphe**
- ❖ Approche à l'aide des **horaires**
- ❖ Gestion du **retard**



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

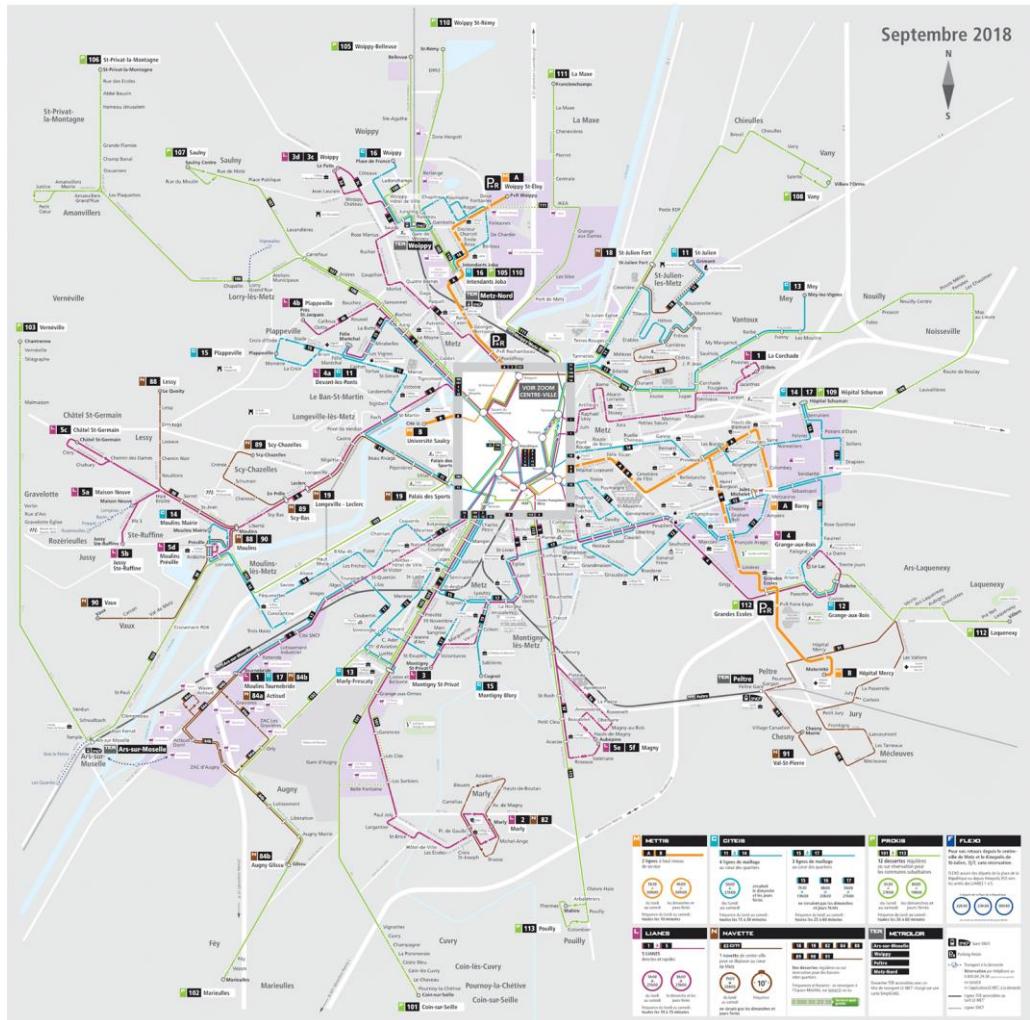
Présentation du réseau

26 lignes pour le réseau principal:

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

18 lignes secondaires

- Flexo
- Metz’O
- Navettes



Source : <https://www.lemetz.fr>

M A Woippy St-Éloy

P+R Woippy

16 1110

Deux Fontaines

Docteur Charcot

Émile Roux

Intendants Joba

René Cassin

Georges Bernanos

16 105 110

P+R Rochambeau

Pontiffroy

St-Vincent

Square du Luxembourg

Moyen Pont

L 3 4 | 5 E 15

E 106 107 111

N 18

M B

L 1 2 3 4 5

C 11 12 N 83 CITY

F 1 à 4

M B L 1 2 3 4 5

C 11 13 14 15

F 103 111

M B L 1 2 3 4 5

C 11 12 13 14

F 101 102 103 108 109

G 111 113 N 70 83 CITY

M B L 5 N 83 CITY

M B L 4 F 12

Seille

Gare Routière

Anima de Metz

Hôpital Legouest

Felix Alcan

Cimetière de l'Est

Auchan

Belletanche

Lycée R. Schuman

Provence

Hôpital Claude Bernard

Campus Irène

Les Bordes

Hauts de Blémont

Collège Hauts de Blémont

Cloutiers

Colombey

Metzamine

Jules Michelet

Cora

M A Borny

Arrêt accessible aux personnes à mobilité réduite

Espace Mobilité

Point de vente LE MET'

Parking Relais

Parking Vélo LE MET'

Les stations METTIS sont équipées de Distributeurs Automatiques de Titres. Pas de vente à bord.

Données retenues

Ressources GTFS

Ajour

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

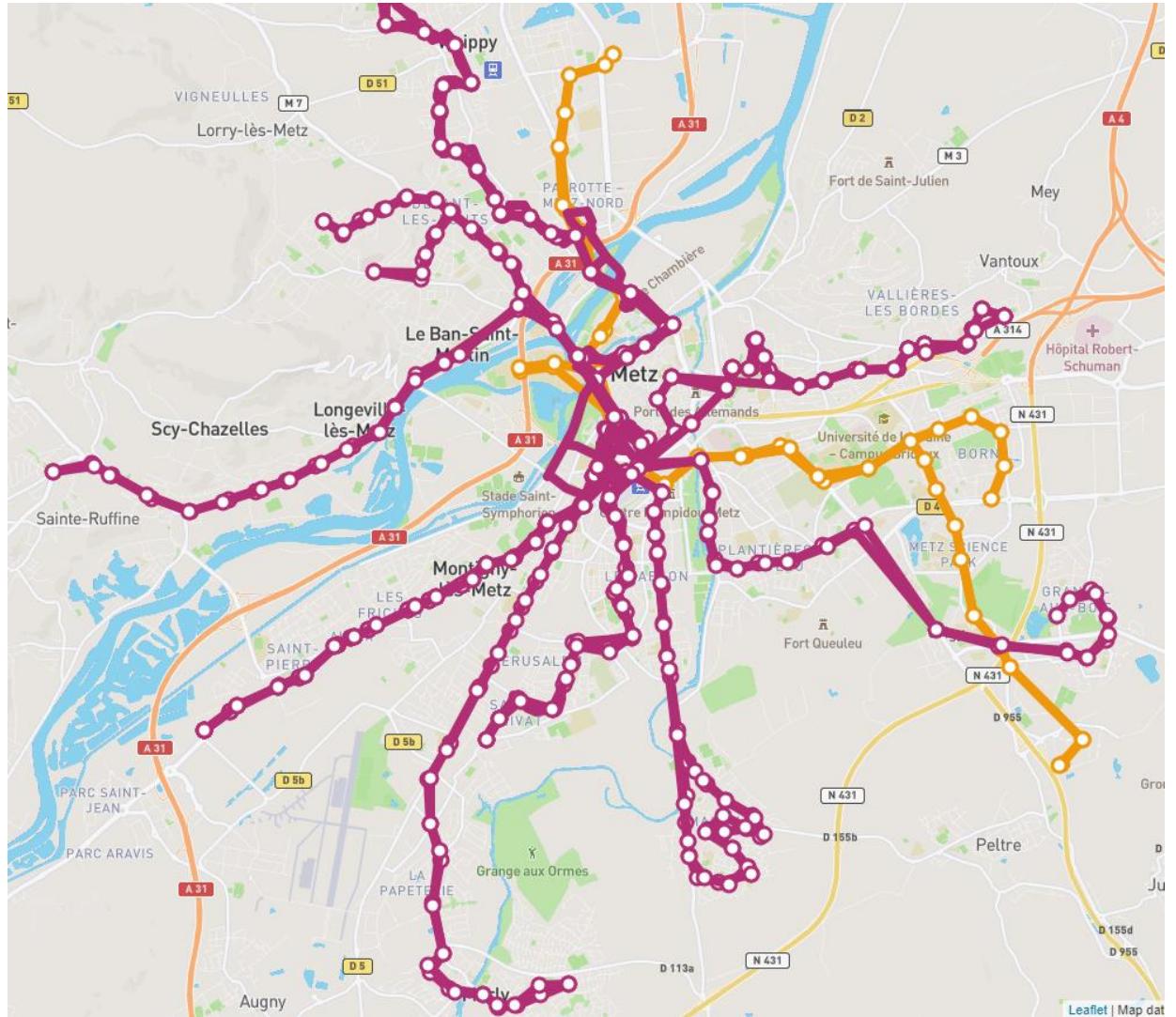
Télécharger

Fichier stops.txt

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_
57,, "8 MAI 45",,49.099805,6.138669,,https://services
46,, "8 MAI 45",,49.099927,6.138371,,https://services
450,, "11ème D'AVIATION",,49.085890,6.146239,,https://
852,, "11ème D'AVIATION",,49.086159,6.145297,,https://
323,, "18 AOUT",,49.191534,6.034971,,,0,,1
322,, "18 AOUT",,49.191388,6.035001,,,0,,1
928,, "19 NOVEMBRE",,49.091774,6.161323,,https://servi
929,, "19 NOVEMBRE",,49.092028,6.161481,,https://servi
519,, "DEUX FONTAINES",,49.148061,6.171578,,https://servi
451,, "DEUX FONTAINES",,49.148956,6.171576,,https://servi
530,, "TRENTE JOURS",,49.096271,6.245860,,https://servi
536,, "TRENTE JOURS",,49.096403,6.245779,,https://servi
```

Lignes retenues

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



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

Objectif de la première approche

- ❖ Adapter les données brutes
- ❖ Trouver le **plus court chemin** entre deux arrêts
- ❖ Faire une représentation visuelle par un **graphe du réseau messin**



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

Structure du code

Tri des sommets	Tri des arêtes
<ul style="list-style-type: none">Parcours de stops.csvSi l'arrêt n'a pas été découvert:<ul style="list-style-type: none">Ajout des arrêts et de ses coordonnées dans une listeCréation du nom d'affichageReconversion de la liste en csv	<ul style="list-style-type: none">Parcours des fichiers csv de chaque ligne (11)Identification du prédecesseur et du successeur de chaque arrêtPondérationReconversion en csv

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,st  
57,,,"8 MAI 45",,49.099805,6.138669,,https://servi  
46,,,"8 MAI 45",,49.099927,6.138371,,https://servi  
450,,,"11ème D'AVIATION",,49.085890,6.146239,,http  
852,,,"11ème D'AVIATION",,49.086159,6.145297,,http  
323,,,"18 AOUT",,49.191534,6.034971,,,0,,1  
322,,,"18 AOUT",,49.191388,6.035001,,,0,,1
```

stops.csv

Réorganisation des données (résultat du code)

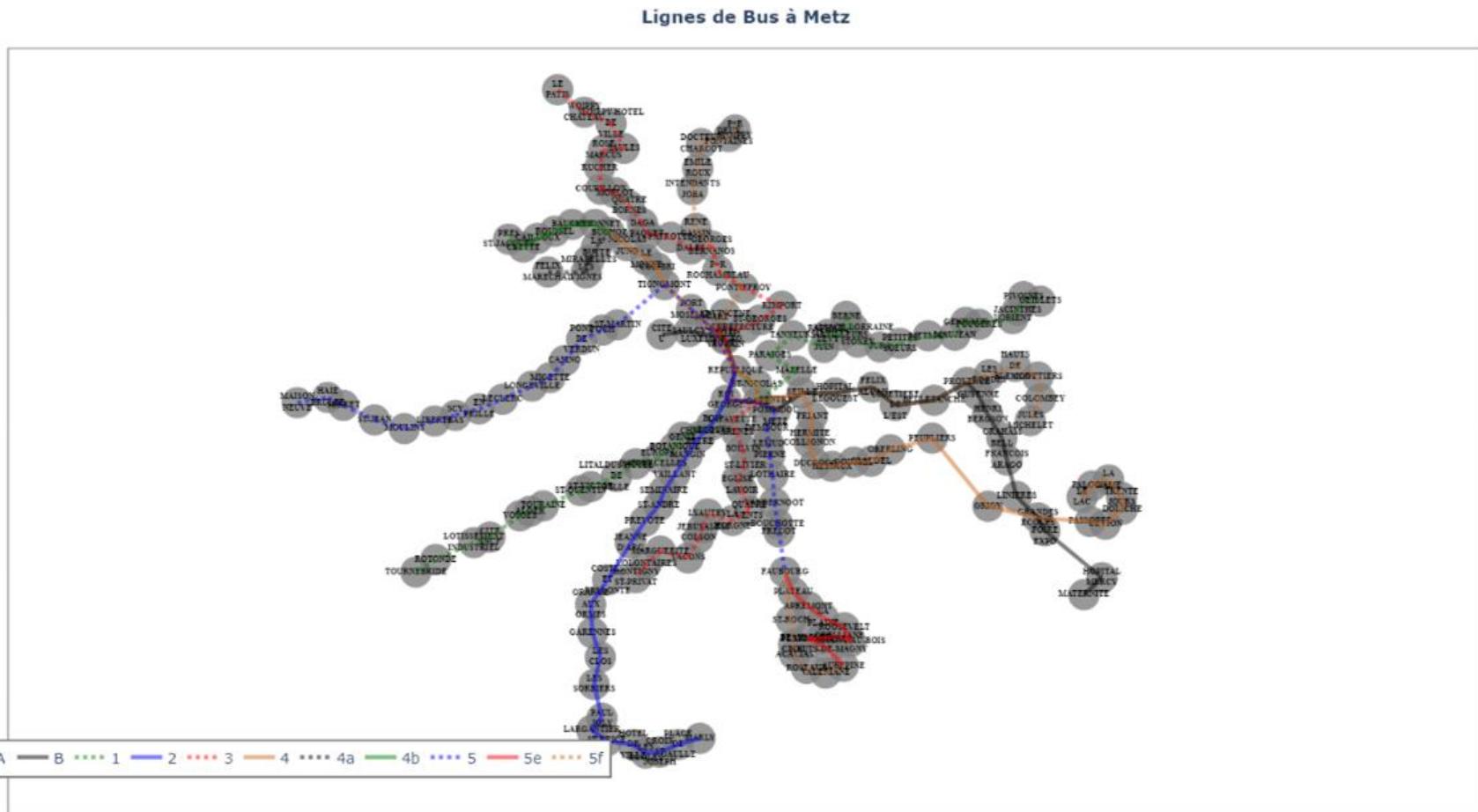
◆ Arêtes

line,station_name1,station_name2,distance
A,P+R WOIPPY,DEUX FONTAINES,139
A,DEUX FONTAINES,DOCTEUR CHARCOT,392
A,DOCTEUR CHARCOT,EMILE ROUX,404
A,EMILE ROUX,INTENDANTS JOBA,358
A,INTENDANTS JOBA,RENE CASSIN,632
A,RENE CASSIN,GEORGES BERNANOS,359
A,GEORGES BERNANOS,P+R ROCHAMBEAU,415
A,P+R ROCHAMBEAU,PONTIFFROY,448
A,PONTIFFROY,ST-VINCENT,499
A,ST-VINCENT,SQUARE DU LUXEMBOURG,288
A,SQUARE DU LUXEMBOURG,MOYEN PONT,234

◆ Sommets

name,lat,lon,display_name
P+R WOIPPY,49.150349,6.173323,P+R WOIPPY
DEUX FONTAINES,49.149369,6.172123,DEUX FONTAINES
DOCTEUR CHARCOT,49.148334,6.166976,DOCTEUR CHARCOT
EMILE ROUX,49.144727,6.166327,EMILE ROUX
INTENDANTS JOBA,49.141569,6.165333,INTENDANTS JOBA
RENE CASSIN,49.135897,6.165947,RENE CASSIN
GEORGES BERNANOS,49.133318,6.168920,GEORGES BERNANOS
P+R ROCHAMBEAU,49.129663,6.170075,P+R ROCHAMBEAU
PONTIFFROY,49.127140,6.174876,PONTIFFROY
ST-VINCENT,49.123270,6.171396,ST-VINCENT

Représentation visuelle finale



Utilisation

- Calcul de l'itinéraire optimal à prendre

Exemple sur le trajet précédent:



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

○ Arrêt GARE, Metz

Bus LE MET' - Metz **4**

Vers L4 - GRANGE-AUX-BOIS

Départs : 19:38 19:53 20:12

SEILLE

FRIANT

HERMITE

COLLIGNON

DUCROCQ

HESTAUTX

GOUSSEL

CLAUDEL

OBERLING

Voir les horaires

Eviter la ligne

! Plusieurs perturbations impactent cette ligne ▾

○ PEUPLIERS, Metz

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

Trajet proposé
par Le Met

Les trajets **coïncident** :

```
8 Distance la plus courte: 3513
9 Chemin le plus court: ['GARE', 'SEILLE', 'FRIANT', 'HERMITE', 'COLLIGNON', 'DUCROCQ', 'HESTEAUX', 'GOUSSEL', 'CLAUDEL', 'OBERLING', 'PEUPLIERS']
10
11 Process finished with exit code 1
```

Trajet proposé par Python
en utilisant l'algorithme de Bellman-Ford

Limitations de ce modèle

- ❖ Prend en compte **un nombre limité** de lignes
- ❖ Représentation **non réaliste**
- ❖ Graphe **non orienté**
- ❖ Adaptation de la **représentation visuelle**

2^{ème} approche : plus réaliste

Recherche du plus court chemin à l'aide des horaires



Utilisation du fichier stop_times.csv

trip_id	arrival_time	departure_time	stop_id	stop_sequence	pickup_type	drop_off_type
835666-HIV2223-HDim2223-Dimanche-40	05:13:00	05:13:00	6270	1	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:14:10	05:14:10	6280	2	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:16:01	05:16:01	998	3	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:18:00	05:18:00	6120	4	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:19:39	05:19:39	6130	5	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:21:00	05:21:00	6140	6	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:22:11	05:22:11	6150	7	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:23:36	05:23:36	6160	8	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:25:00	05:25:00	6170	9	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:26:29	05:26:29	6180	10	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:27:55	05:27:55	6190	11	0,0	
835666-HIV2223-HDim2223-Dimanche-40	05:29:31	05:29:31	6200	12	0,0	

- ❖ 600 000 lignes : informations sur tous les horaires du réseau

Objectif de cette approche

- ◊ Adapter les données brutes
- ◊ Trouver le **plus court chemin en temps** entre deux arrêts
- ◊ Déterminer l'**horaire d'arrivée**



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

Algorithmes du traitement des données

Tri des lignes	Fiches horaires
<ul style="list-style-type: none">• Un fichier par ligne• Regroupe tous les « stop_id » d'une ligne	<ul style="list-style-type: none">• Pour chaque arrêt du réseau• Regroupe toutes les données relatives à l'arrêt

route_id	service_id	trip_id	trip_headsign	dir	block_id	shape_id
B-194	HIV2223-HDim2223-Dimanche-40	835666-HIV2223-HDim2223-Dimanche-40	MB - HOPITAL MERCY	0	116697	B0018
B-194	HIV2223-HDim2223-Dimanche-40	835667-HIV2223-HDim2223-Dimanche-40	MB - HOPITAL MERCY	0	116697	B0018
B-194	HIV2223-HDim2223-Dimanche-40	835668-HIV2223-HDim2223-Dimanche-40	MB - HOPITAL MERCY	0	116699	B0018
B-194	HIV2223-HDim2223-Dimanche-40	835669-HIV2223-HDim2223-Dimanche-40	MB - HOPITAL MERCY	0	116697	B0018
B-194	HIV2223-HDim2223-Dimanche-40	835670-HIV2223-HDim2223-Dimanche-40	MB - HOPITAL MERCY	0	116699	B0018

trips.csv

trip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,drop_off_type
835666-HIV2223-HDim2223-Dimanche-40,05:13:00,05:13:00,6270,1,0,0
835666-HIV2223-HDim2223-Dimanche-40,05:14:10,05:14:10,6280,2,0,0
835666-HIV2223-HDim2223-Dimanche-40,05:16:01,05:16:01,998,3,0,0
835666-HIV2223-HDim2223-Dimanche-40,05:18:00,05:18:00,6120,4,0,0
835666-HIV2223-HDim2223-Dimanche-40,05:19:39,05:19:39,6130,5,0,0

stop_times.csv

Résultats

 C11 - DEVANT-LES-PONTS.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C11 - REPUBLIQUE.csv	30/12/2022 17:08	Fichier CSV Micro...	1 Ko
 C11 - ST-JULIEN.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C12 - GRANGE-AUX-BOIS.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C12 - REPUBLIQUE.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C13 - MARLY FRESCATY.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C13 - NOISSEVILLE.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C14 - GARE.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko
 C14 - HOPITAL SCHUMAN.csv	30/12/2022 17:07	Fichier CSV Micro...	1 Ko

Dossier Lignes2

stop_id
5057
17
5053
603
604
605
606
5055
5054
5039
5048
5049
5050
497
499
697
556
393
888
30
31
88

Un fichier csv
(C11- REPUBLIQUE)

Résultats

 1.csv	24/12/2022 04:35
 2.csv	24/12/2022 04:00
 3.csv	24/12/2022 04:42
 4.csv	24/12/2022 04:14
 5.csv	24/12/2022 04:39
 6.csv	24/12/2022 04:19
 7.csv	24/12/2022 04:25
...	
 1022.csv	24/12/2022 04:07
 1023.csv	24/12/2022 04:07
 1024.csv	24/12/2022 04:40
 2330.csv	24/12/2022 04:21
 2378.csv	24/12/2022 04:34
 2385.csv	24/12/2022 04:34
 2580.csv	24/12/2022 04:21
 3000.csv	24/12/2022 04:22

836059-HIV2223-HDim2223-Dimanche-40,13:01:46,13:01:46,3,17,C11 - ST-JULIEN	
836060-HIV2223-HDim2223-Dimanche-40,12:06:46,12:06:46,3,17,C11 - ST-JULIEN	
836061-HIV2223-HDim2223-Dimanche-40,16:40:46,16:40:46,3,17,C11 - ST-JULIEN	
836106-HIV2223-HDim2223-Dimanche-40,09:06:41,09:06:41,3,13,L4 - GRANGE-AUX-BOIS	
836107-HIV2223-HDim2223-Dimanche-40,12:58:41,12:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836108-HIV2223-HDim2223-Dimanche-40,13:58:41,13:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836109-HIV2223-HDim2223-Dimanche-40,14:58:41,14:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836110-HIV2223-HDim2223-Dimanche-40,15:58:41,15:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836111-HIV2223-HDim2223-Dimanche-40,16:58:41,16:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836112-HIV2223-HDim2223-Dimanche-40,17:58:41,17:58:41,3,13,L4 - GRANGE-AUX-BOIS	
836113-HIV2223-HDim2223-Dimanche-40,18:59:00,18:59:00,3,13,L4 - GRANGE-AUX-BOIS	
836114-HIV2223-HDim2223-Dimanche-40,20:01:00,20:01:00,3,13,L4 - GRANGE-AUX-BOIS	
836115-HIV2223-HDim2223-Dimanche-40,21:30:00,21:30:00,3,15,L4 - GRANGE-AUX-BOIS	
836116-HIV2223-HDim2223-Dimanche-40,13:28:41,13:28:41,3,15,L4 - GRANGE-AUX-BOIS	

Extrait du fichier 3.csv

Dossier fichehoraire

Algorithme du plus court chemin

Plus court chemin

- Prendre l'arrêt et l'heure de départ.
- Enregistrer dans un dictionnaire les lignes découvertes par l'arrêt et leur « trip_id » .
- Parcourir tous les arrêts voisins et calculer le temps pour y parvenir.
- Ajouter les informations de parcours dans des listes.
- => Réappliquer le même algorithme avec l'arrêt suivant et l'heure d'arrivée.
- Arrêt final
- Programme de fin:
 - Parcourir les informations enregistrées pendant le parcours.
 - Les traités pour obtenir le parcours, et les bus à emprunter.

15:25 ○ Arrêt CIMETIERE, Saint-Julien-lès-Metz

Bus LE MET' - Metz 11 Vers C11 - DEVANT-LES-PONTS

ST-JULIEN EGLISE
TERRES ROUGES
TANNERIE
ARSENAL
FOURNIER
TANNEURS
PARAIGES
MAZELLE

Voir les horaires Eviter la ligne

15:43 ○ GARE, Metz

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

Quel est le lieu de départ: CIMETIERE

Quel est le lieu d'arrivée: GARE

Quel jour: Lundi

Quelle heure (Format HH:MM:SS): 15:15:15

Heure d'arrivée : 15:43:00

Chemin final : ['CIMETIERE', 'ST-JULIEN EGLISE', 'TERRES ROUGES', 'TANNERIE', 'ARSENAL', 'FOURNIER', 'TANNEURS', 'PARAIGES', 'MAZELLE', 'GARE']

Correspondances: [['C11 - DEVANT-LES-PONTS', 'CIMETIERE', '15:25:20']]

Autre exemple avec correspondance : CIMETIERE -> MEY LES VIGNES

Heure d'arrivée : 15:44:00

Chemin final : ['CIMETIERE', 'ST-JULIEN EGLISE', 'TERRES ROUGES', 'TANNERIE', 'TANNERIE', 'BILLOTTE', 'DUNANT', 'JOUSSE', 'JUGAN', 'CORCHADE', 'J.P. JEAN', 'SAULNOIS', 'MY MANGENOT', 'BARBE', 'FONNY', 'LES MOULINS', 'MEY LES VIGNES']

Correspondances: ['C11 - DEVANT-LES-PONTS', 'CIMETIERE', '15:25:20', 'C13 - NOISSEVILLE', 'TANNERIE', '15:31:00']'

Résultats

Départ **CIMETIERE, Saint-Julien-lès-Metz** ↑ ↓

Arrivée **Gare / Pôle multimodal, Metz**

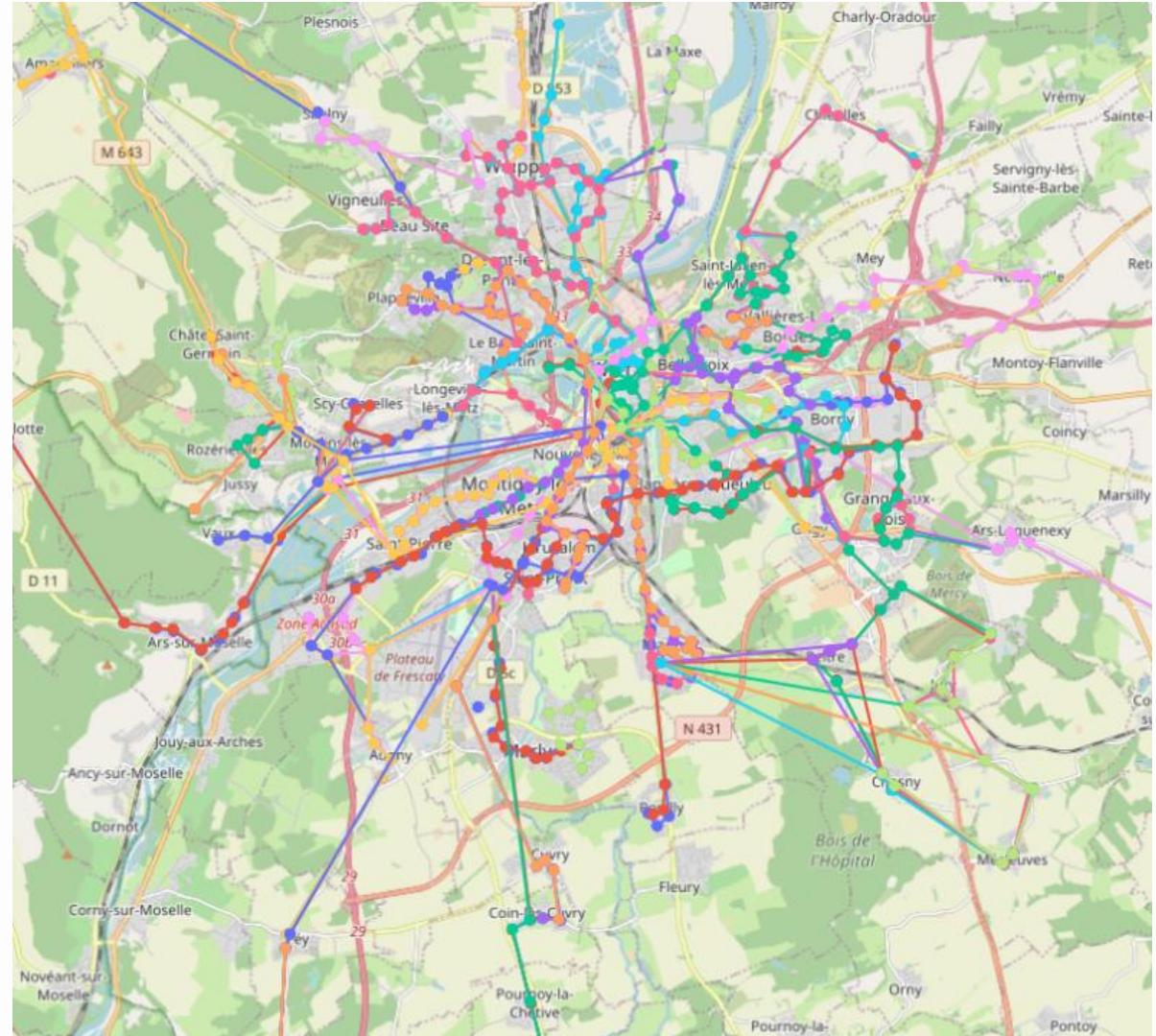
Partir le
lun 5 juin < >
à 15:15 < >
Options

RECHERCHER

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

Résultat visuel du réseau entier

- ❖ Toutes les lignes sont représentées
- ❖ Représentation interactive



Gestion du retard

- ◆ Eviter le phénomène des « train-de-bus »

Objectifs d'un programme informatique

- ◆ Rendre compte des retards sur les fiches horaires entre une situation qui est impactée par les trains de bus et une qui l'évite



Structure du code :

Requêtes à l'utilisateur :

- Heure de la simulation
- Jour de la simulation
- Nom de la ligne étudiée

Quelle heure (Format HH:MM:SS): 15:14:29

Quel jour: Lundi

Veuillez entrer le nom de la ligne à représenter: METTIS A

Récupération de données importantes

- Toutes les directions de la ligne voulue
- Récupération des bus en circulation à l'heure demandée
- Positionnement de chaque bus sur une carte

['MA - WOIPPY ST-ELOY', 'MA - BORNY']

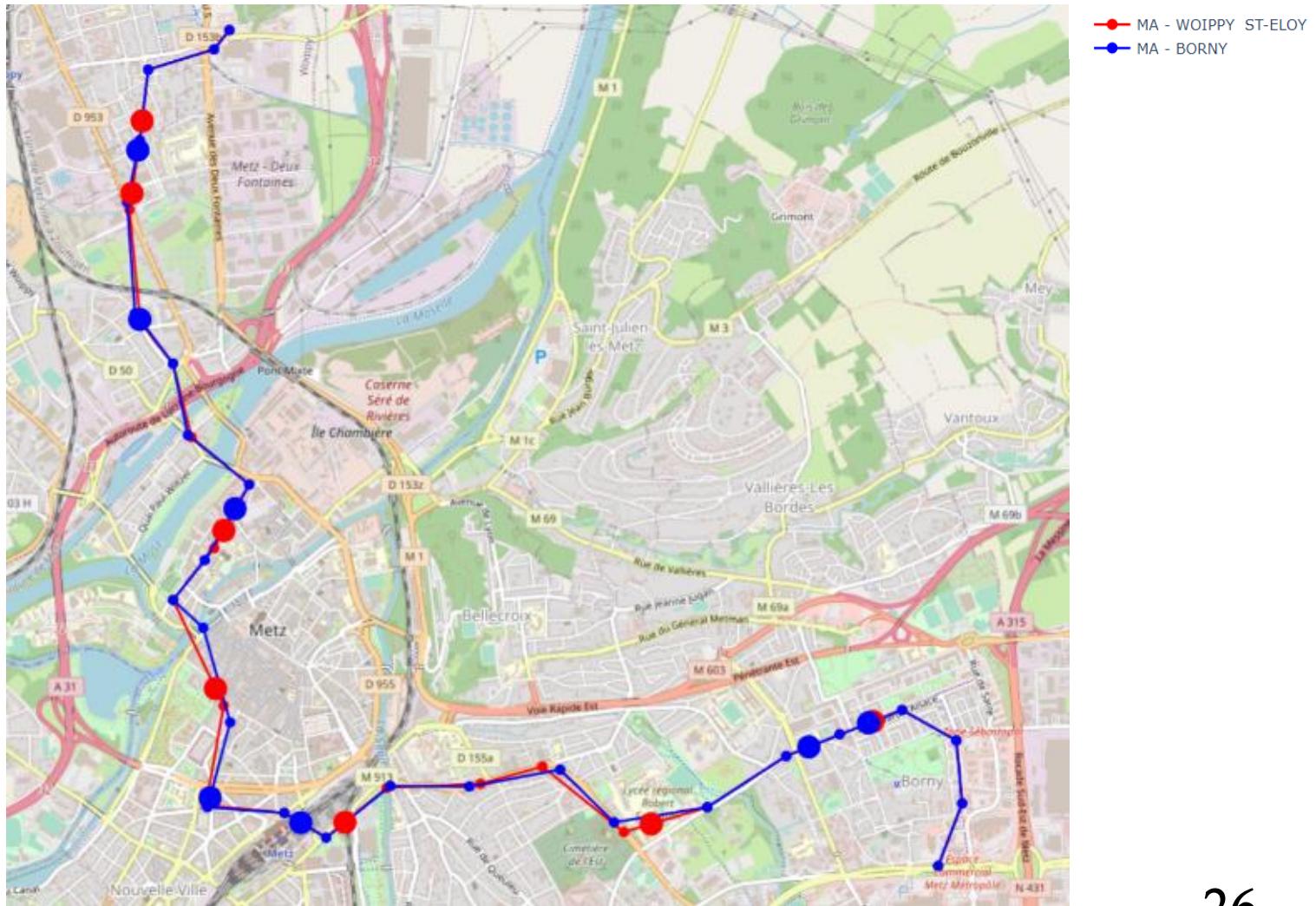
Ligne vers **WOIPPY**

Ligne vers **BORNY**

trip_id: ['830458-HIV2223-HSem2223-L-Ma-J-V-01', '823975-HV2223_PVS-PVS_2223-Semaine-01', '830457-HIV2223-HSem2223-L-Ma-J-V-01'
trip_id: ['830458-HIV2223-HSem2223-L-Ma-J-V-01', '823975-HV2223_PVS-PVS_2223-Semaine-01', '830457-HIV2223-HSem2223-L-Ma-J-V-01']

Résultats

Simulation à 15:14:26



Création du retard:

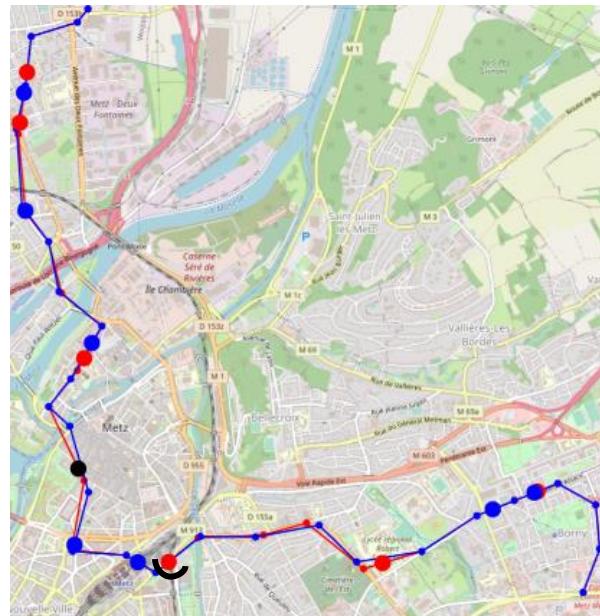
Requêtes à l'utilisateur :

- Quel bus va être affecté par un arrêt
- Nouvelle heure de simulation

2 situations envisageables

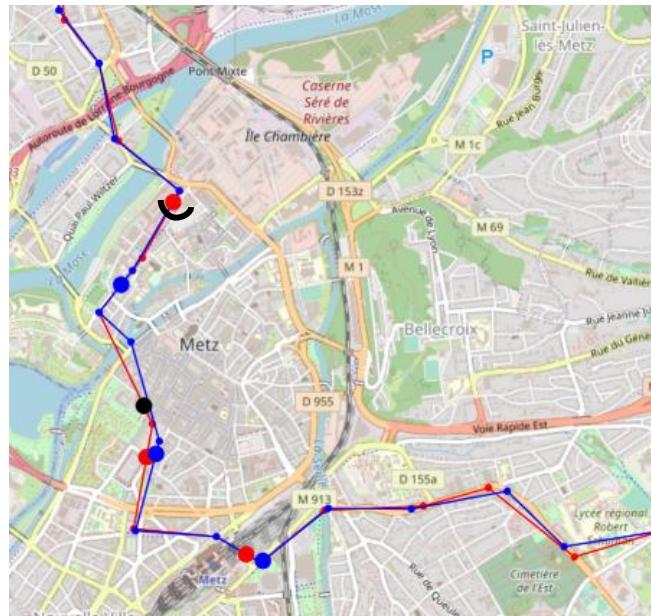
Avec phénomène du train de bus	Sans phénomène du train de bus
<ul style="list-style-type: none">• Refaire une simulation• Laisser le bus statique• Si un bus doit se trouver devant dans la nouvelle simulation, le laisser derrière le bus à l'arrêt.• Dictionnaire de « progression »• Ajouter le retard que prend chaque bus dans un dictionnaire de « retard »	<ul style="list-style-type: none">• Refaire une simulation des bus• Laisser le bus statique• Les bus dépassent le bus à l'arrêt• Dans le dictionnaire ajouter un retard au bus à l'arrêt uniquement.

Résultats visuels



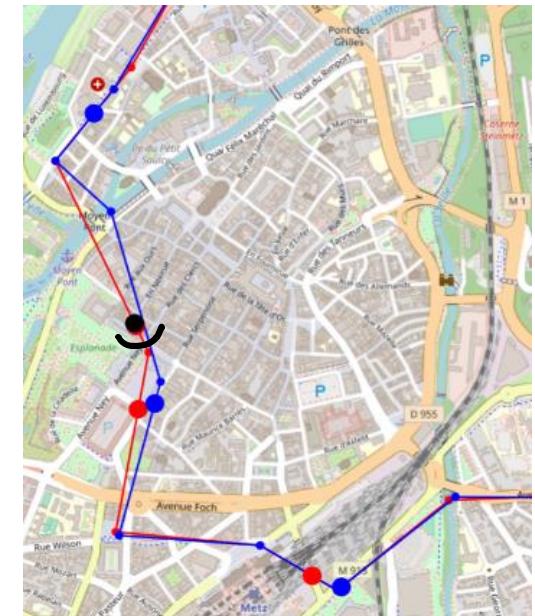
Avant simulation

Simulation à 15:14:26



Sans train de bus

Simulation à 15:26:14



Avec train de bus

Analyse des résultats

830457-HIV22	15:32:00	0	INTENDANTS MA - WOIPPY ST-ELOY
830458-HIV22	15:42:00	0	INTENDANTS MA - WOIPPY ST-ELOY
823974-HV22	15:26:00	00:10:48	INTENDANTS MA - WOIPPY ST-ELOY
823975-HV22	15:38:00	0	INTENDANTS MA - WOIPPY ST-ELOY
830457-HIV22	15:33:59	0	EMILE ROUX MA - WOIPPY ST-ELOY
830458-HIV22	15:43:59	0	EMILE ROUX MA - WOIPPY ST-ELOY
823974-HV22	15:27:59	00:10:48	EMILE ROUX MA - WOIPPY ST-ELOY
823975-HV22	15:39:59	0	EMILE ROUX MA - WOIPPY ST-ELOY
830456-HIV22	15:25:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
830457-HIV22	15:35:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
830458-HIV22	15:45:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
823974-HV22	15:29:45	00:10:48	DOCTEUR CH. MA - WOIPPY ST-ELOY
823975-HV22	15:41:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
830456-HIV22	15:27:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY
830457-HIV22	15:37:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY
830458-HIV22	15:47:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY
823974-HV22	15:31:25	00:10:48	DEUX FONTAI MA - WOIPPY ST-ELOY
823975-HV22	15:43:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY

Sans train de bus
Simulation à 15:25:14

830457-HIV22	15:32:00	00:04:48	INTENDANTS MA - WOIPPY ST-ELOY
830458-HIV22	15:42:00	0	INTENDANTS MA - WOIPPY ST-ELOY
823974-HV22	15:26:00	00:10:48	INTENDANTS MA - WOIPPY ST-ELOY
823975-HV22	15:38:00	0	INTENDANTS MA - WOIPPY ST-ELOY
830457-HIV22	15:33:59	00:04:48	EMILE ROUX MA - WOIPPY ST-ELOY
830458-HIV22	15:43:59	0	EMILE ROUX MA - WOIPPY ST-ELOY
823974-HV22	15:27:59	00:10:48	EMILE ROUX MA - WOIPPY ST-ELOY
823975-HV22	15:39:59	0	EMILE ROUX MA - WOIPPY ST-ELOY
830456-HIV22	15:25:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
830457-HIV22	15:35:45	00:04:48	DOCTEUR CH. MA - WOIPPY ST-ELOY
830458-HIV22	15:45:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
823974-HV22	15:29:45	00:10:48	DOCTEUR CH. MA - WOIPPY ST-ELOY
823975-HV22	15:41:45	0	DOCTEUR CH. MA - WOIPPY ST-ELOY
830456-HIV22	15:27:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY
830457-HIV22	15:37:25	00:04:48	DEUX FONTAI MA - WOIPPY ST-ELOY
830458-HIV22	15:47:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY
823974-HV22	15:31:25	00:10:48	DEUX FONTAI MA - WOIPPY ST-ELOY
823975-HV22	15:43:25	0	DEUX FONTAI MA - WOIPPY ST-ELOY

Avec train de bus
Simulation à 15:35:28

830457-HIV22	15:32:00	00:15:02	INTENDANTS JOB MA - WOIPPY ST-ELOY
830458-HIV22	15:42:00	00:05:02	INTENDANTS JOB MA - WOIPPY ST-ELOY
823974-HV22	15:26:00	00:21:02	INTENDANTS JOB MA - WOIPPY ST-ELOY
823975-HV22	15:38:00	00:09:02	INTENDANTS JOB MA - WOIPPY ST-ELOY
830457-HIV22	15:33:59	00:15:02	EMILE ROUX MA - WOIPPY ST-ELOY
830458-HIV22	15:43:59	00:05:02	EMILE ROUX MA - WOIPPY ST-ELOY
823974-HV22	15:27:59	00:21:02	EMILE ROUX MA - WOIPPY ST-ELOY
823975-HV22	15:39:59	00:09:02	EMILE ROUX MA - WOIPPY ST-ELOY
830457-HIV22	15:35:45	00:15:02	DOCTEUR CHARC MA - WOIPPY ST-ELOY
830458-HIV22	15:45:45	00:05:02	DOCTEUR CHARC MA - WOIPPY ST-ELOY
823974-HV22	15:29:45	00:21:02	DOCTEUR CHARC MA - WOIPPY ST-ELOY
823975-HV22	15:41:45	00:09:02	DOCTEUR CHARC MA - WOIPPY ST-ELOY
830457-HIV22	15:37:25	00:15:02	DEUX FONTAINES MA - WOIPPY ST-ELOY
830458-HIV22	15:47:25	00:05:02	DEUX FONTAINES MA - WOIPPY ST-ELOY
823974-HV22	15:31:25	00:21:02	DEUX FONTAINES MA - WOIPPY ST-ELOY

Avec train de bus
Simulation à 15:25:14

Conclusion

- ◊ Approche avec les horaires assez longue, ne marche pas sur des exemples très complexes
 - Algorithme de YEN (Y-PT)
- ◊ Gestion du retard
 - Adaptation en temps réel

```

1 #Importations nécessaires à tout le TIPE#
2 from math import *
3 import csv
4 import networkx as nx
5 import pandas as pa
6 import plotly.graph_objects as go
7 import bellmanford as bf
8 import matplotlib.pyplot as plt
9 from datetime import timedelta, datetime
10 import random as rd
11 #####
12
13 ###Traitements des sommets du graphe###
14 f = open('stops.csv')
15
16 lignecompt = 0
17 arrrets = []
18 nomdejavu = []
19 for ligne in f:
20     tempo = []
21     if (lignecompt >= 1):
22         x = ligne.split(",")
23         if x[2] not in nomdejavu:
24             tempo.append(x[2])
25             nomdejavu.append(x[2])
26             tempo.append(x[4])
27             tempo.append(x[5])
28             arrrets.append(tempo)
29
30     lignecompt += 1
31
32
33 f.close()
34
35 fic = open("bus_metz.csv", "w", newline="")
36
37 writer = csv.writer(fic)
38 writer.writerow(("name","lat","lon","display_name"))
39 writer = csv.writer(fic, lineterminator='\n',
40                     quoting=csv.QUOTE_NONE, quotechar=None)
41
42 #Fonction pour créer le nom d'affichage
43 def transname(name):
44     newname = name.replace(" ","<br>")
45     displayname = []
46     listename = list(newname)
47     for j in range(len(listename)):
48         if listename[j] != ('"' or ''):
49
50             displayname.append(listename[j])
51
52     final = ''.join(displayname)
53     return final
54
55 for i in range(len(arrrets)):
56     name,lat,lon= arrrets[i]
57     displayname = transname(name)
58     writer.writerow((name, lat, lon, displayname))
59
60 fic.close()
61
62 ###Traitement des arêtes du graphe###
63
64 url = [
65     ['A', 'Lignes\mettis a.csv'],
66     ['B', 'Lignes\mettis b.csv'],
67     ['1', 'Lignes\l1.csv'],
68     ['2', 'Lignes\l2.csv'],
69     ['3', 'Lignes\l3.csv'],
70     ['4', 'Lignes\l4 normal.csv'],
71     ['4a', 'Lignes\l4a.csv'],
72     ['4b', 'Lignes\l4b.csv'],
73     ['5', 'Lignes\l5 normal.csv'],
74     ['5e', 'Lignes\l5e.csv'],
75     ['5f', 'Lignes\l5f.csv']
76 ]
77
78 arrrets = []
79 def ajoutligne(lignebus):
80     f = open(lignebus[1])
81     lignecompt = 0
82     for ligne in f:
83         tempo = []
84         if (lignecompt >= 1):
85             x = ligne.split(",")
86             tempo.append(lignebus[0])
87             tempo.append(x[2])
88             arrrets.append(tempo)
89
90         lignecompt += 1
91
92     f.close()
93
94
95 for k in range(len(url)):
96     ajoutligne(url[k])
97
98 fic = open('bus_metz_lignes.csv', 'w', newline='')
99
100 writer = csv.writer(fic)
101 writer.writerow(['line', 'station_name1', 'station_name2', 'distance'])
102 writer = csv.writer(fic, lineterminator='\n',
103                     quoting=csv.QUOTE_NONE, quotechar=None)
104
105 def arretsuivant(arrrets, i):
106     line, arret = arrrets[i]
107     if (i+1) < len(arrrets):
108         linesuiv, arretsuiv = arrrets[i + 1]
109         if line == linesuiv:
110             return True, arretsuiv
111         else:
112             return False, arretsuiv
113
114 def deg2rad(dd):
115
116     return float(dd)/180*pi
117
118 def distance(arret, arretsuiv):
119
120     fichier = open('bus_metz.csv')
121     lignecompteur = 0
122     for ligne in fichier:
123         x = ligne.split(",")
124         if lignecompteur >= 1:
125             if x[0] == arret:
126                 latA = deg2rad(x[1])
127                 longA = deg2rad(x[2])
128
129             if x[0] == arretsuiv:
130                 latB = deg2rad(x[1])
131                 longB = deg2rad(x[2])
132
133         lignecompteur += 1
134
135 RT = 6378137
136 S = acos(sin(latA) * sin(latB) + cos(latA) * cos(latB) * cos(abs(longB - longA)))
137 # distance entre les 2 points, comptée sur un arc de grand cercle
138 fichier.close()
139 return int(S * RT)
140
141 for i in range(len(arrrets)):
142     line, arret = arrrets[i]
143     statut, arretsuiv = arretsuivant(arrrets, i)
144
145 newarret = arret.replace('"', '')
146 newarretsuv = arretsuiv.replace('"', '')
147
148 if statut == True:
149     dist = distance(arret, arretsuiv)
150     writer.writerow((line , newarret , newarretsuv,dist))
151
152 fic.close()
153
154
155
156 ### Visualisation de la première approche ###
157
158 T = pa.read_csv('bus_metz_lignes_backup.csv')
159
160 G = nx.from_pandas_edgelist(T, source='station_name1', target='station_name2', edge_attr='distance')
161 M = pa.read_csv('bus_metz_backup.csv')
162
163 fig = go.Figure()
164
165 fig.add_trace(
166     go.Scattergeo(
167         mode='markers + text',
168         lon=M['lon'],
169         lat=M['lat'],
170         text=M['display_name'].apply(lambda x: \
171             '<b>' + str(x) + \
172             '</b>'),
173             textposition='middle center',
174             textfont=dict(color="black", size=8, family="bold"),
175             showLegend=False,
176             marker={'symbol': 'circle-dot',
177                     'size': 30,
178                     'opacity': 0.8,
179                     'color': 'gray'},
180         )
181     )
182 fig.update_layout(
183     showlegend=True,
184     legend={'x': 0,
185             'y': 0.1,
186             'title': '<b>Lignes : </b>',
187             'orientation': 'h',
188             'bordercolor': 'gray',
189             'font': {'family': 'Verdana', 'size': 14},
190             'borderwidth': 2,
191             },
192     title={'text': "<b>Lignes de Bus à Metz</b>",
193             'font': {'family': 'Verdana'},
194             'y': 0.93,
195             'x': 0.5},
196     )
197 fig.update_geos(
198     fitbounds='locations',
199     showland=True,
200     landcolor='white',
201     projection_type='natural earth'
202     )
203
204
205 N = M.groupby('name').first()
206
207 T[['x1']] = [N.loc[n, 'lon'] for n in T['station_name1']]
208 T[['y1']] = [N.loc[n, 'lat'] for n in T['station_name1']]
209 T[['x2']] = [N.loc[n, 'lon'] for n in T['station_name2']]
210 T[['y2']] = [N.loc[n, 'lat'] for n in T['station_name2']]
211
212 couleurs = ['black', 'green', 'blue', 'red', 'chocolate']
213 dot = ['dot', 'solid']
214 L = []
215 ListeLigne = ["A","B","1","2","3","4","4a","4b","5","5e","5f"]
216 compteur=0
217 for i in ListeLigne:
218     A = T.query('line == @i')
219     for k in A.index:
220         fig.add_trace(
221             go.Scattergeo(
222                 mode = 'lines',
223                 lon = [A.loc[k, 'x1'], A.loc[k, 'x2']],
224                 lat = [A.loc[k, 'y1'], A.loc[k, 'y2']],
225                 hovertext = A['distance'],
226                 opacity= 0.5,
227                 showlegend= True if i not in L else False,
228                 name = i,
229                 line = {'color' : couleurs[compteur%5 - 1],
230                         'dash' : dot[compteur%2],
231                         'width' : 4},
232                 )
233             )
234     L.append(i)
235     compteur += 1
236
237 fig.show()
238
239
240 ### Algorithme du plus court chemin en fonction des distances###
241
242 def transname(name):
243     displayname = []
244     listename = list(name)
245     for j in range(len(listename)):
246         if listename[j] != ('"' or ''):
247             displayname.append(listename[j])
248
249             displayname.append(listename[j])

```

```

258     final = ''.join(displayname)
259     return final
260
261
262 dico = {}
263 arrets = open('gtfs\stops.csv')
264 read = csv.reader(arrrets)
265 for ligne in read:
266     if ligne[2] == 'stop_name':
267         continue
268     if transname(ligne[2]) not in dico:
269         dico[transname(ligne[2])] = [ligne[0]]
270     else:
271         listetempo = dico[transname(ligne[2])]
272         dico[transname(ligne[2])] = listetempo + [ligne[0]]
273
274 arrrets.close()
275
276 T = pa.read_csv('bus_metz_lignes_backup.csv')
277 G = nx.from_pandas_edgelist(T, source='station_name1', target='station_name2', edge_attr='distance')
278
279 longueur, noeuds, cyclenegatif = bf.bellman_ford(G, source="GARE", target="PEUPLIERS", weight="distance")
280 print("Distance la plus courte: {}.".format(longueur))
281 print("Chemin le plus court: {}.".format(noeuds))
282
283 ##Traitement de données des lignes##
284 ##Tri de toutes les lignes##
285 fic = open('gtfs/trips.csv')
286 f = csv.reader(fic)
287 dejavu = []
288 for ligne in f:
289     if ligne[3] == 'trip_headsign':
290         continue
291     if ligne[3] not in dejavu:
292         dejavu.append(ligne[3])
293
294 lignecsv = open('Lignes2/'+str(ligne[3])+'.csv', 'w', newline='')
295 writer = csv.writer(lignecsv)
296 writer.writerow(['stop_id'])
297 writer = csv.writer(lignecsv, lineterminator='\n',
298                     quoting=csv.QUOTE_NONE, quotechar=None, escapechar='\\')
299
300 trip_id = ligne[2]
301 temps = open('gtfs/stop_times.csv')
302 read = csv.reader(temps)
303 for lignel1 in read:
304     if lignel1[0] == trip_id:
305         writer.writerow([lignel1[3], ])
306 lignecsv.close()
307
308 fic.close()
309
310 ##Création des fiches horaires pour tous les arrêts du réseau##
311 fic = open('bus_metz_lignes.csv', 'w', newline='')
312
313 writer = csv.writer(fic)
314 writer.writerow(['line', 'station_name1', 'station_name2', 'distance'])
315 writer = csv.writer(fic, lineterminator='\n',
316                     quoting=csv.QUOTE_NONE, quotechar=None)
317
318 def transname(name):
319     displayname = []
320     listename = list(name)
321     for j in range(len(listename)):
322         if listename[j] != ('"' or ' '):
323             displayname.append(listename[j])
324
325     final = ''.join(displayname)
326     return final
327
328 dico = {}
329 arrrets = open('gtfs\stops.csv')
330 read = csv.reader(arrrets)
331 for ligne in read:
332     if ligne[2] == 'stop_name':
333         continue
334     if transname(ligne[2]) not in dico:
335         dico[transname(ligne[2])] = [ligne[0]]
336     else:
337         listetempo = dico[transname(ligne[2])]
338         dico[transname(ligne[2])] = listetempo + [ligne[0]]
339
340 arrrets.close()
341
342 def fiche_horaire(stop_id):
343     fiche = open("fichehoraire/" + stop_id + '.csv', 'w', newline='')
344     writer = csv.writer(fiche)
345
346     writer.writerow(["trip_id", "arrival_time", "departure_time", "stop_id", "stop_sequence", "trip_headsign"])
347     writer = csv.writer(fiche, lineterminator='\n',
348                         quoting=csv.QUOTE_NONE, quotechar=None, escapechar='\\')
349
350
351 f = open('gtfs\stop_times.csv')
352 read = csv.reader(f)
353 for ligne in read:
354     if ligne[3] == stop_id:
355         fich = open('gtfs/trips.csv')
356         read2 = csv.reader(fich)
357         for ligne2 in read2:
358             if ligne2[2]==ligne[0]:
359                 writer.writerow([ligne[0],ligne[1],ligne[2],ligne[3],ligne[4],ligne2[3]])
360
361         fich.close()
362         f.close()
363
364
365
366 for liste_arrets in dico.values():
367     for elts in liste_arrets:
368         fiche_horaire(elts)
369
370
371 ##Algorithme de plus court chemin à l'aide des horaires##
372 f = open('gtfs/stops.csv')
373 depart = input("Quel est le lieu de départ: ")
374 verif = False
375 for ligne in f:
376     if depart in ligne:
377         verif = True
378
379 if verif == False:
380     raise ValueError("Mauvais nom d'arrêt")
381 f.close()
382
383 f = open('gtfs/stops.csv')
384 arrivee = input("Quel est le lieu d'arrivée: ")
385
386 verif2 = False
387 for ligne in f:
388     if arrivee in ligne:
389         verif2 = True
390
391 if verif2 == False:
392     raise ValueError("Mauvais nom d'arrêt")
393 f.close()
394
395 jour = input("Quel jour: ")
396 heure = input("Quelle heure (Format HH:MM:SS):  ")
397
398
399 def journee(day):
400     if day == "Dimanche":
401         return ["HIV2223-HDin2223-Dimanche-40"]
402     elif day == "Samedi":
403         return ["HIV2223-HSan2223-Samedi-28"]
404     elif day == "Lundi":
405         return ["HIV2223-HSen2223-L-Ma-J-V-01", "HV2223_PVS-PVS_2223-Semaine-01",
406                 "HIV2223-HSen2223-L-Ma-J-V-01-1101100"]
407     elif day == "Mardi":
408         return ["HIV2223-HSen2223-L-Ma-J-V-01", "HV2223_PVS-PVS_2223-Semaine-01",
409                 "HV2223_PVS-PVS_2223-Semaine-01-0100100", "HIV2223-HSem2223-L-Ma-J-V-01-0100100",
410                 "HIV2223-HSen2223-L-Ma-J-V-01-1101100"]
411     elif day == "Mercredi":
412         return ["HIV2223-HMer2223-Mercredi-10", "HV2223_PVS-PVS_2223-Semaine-01",
413                 "HV2223_PVS-PVS_2223-Semaine-01-0010000"]
414     elif day == "Jeudi":
415

```

```

436     return ["HIV2223-HSen2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
437             "HIV2223-HSen2223-L-Ma-J-V-01-1101108"]
438 elif day == "Vendredi":
439     return ["HIV2223-HSen2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
440             "HIV2223_PVS-PVS_2223-Semaine-01-0100100", "HIV2223-HSen2223-L-Ma-J-V-01-0100100"]
441     "HIV2223-HSen2223-L-Ma-J-V-01-1101108"]
442 raise ValueError("Meuvain format jour")
423
424 day = journee(jour)
425
426 def verifday(day, trip):
427     statuts = False
428     for elt in day:
429         if elt in trip:
430             statuts = True
431     return statuts
432
433 def transformeheure(heure):
434     h = heure.split(":")
435     return int(h[0]) + 3600 + int(h[1]) * 60 + int(h[2])
436
437 def transformeheureinverse(tempo):
438     heures = tempo // 3600
439     minutes = (tempo % 3600) // 60
440     secondes = tempo % 60
441
442 temps_formate = "{:02d}::{:02d}::{:02d}".format(heures, minutes, secondes)
443 return temps_formate
444
445 def transname(name):
446     displayname = []
447     listename = list(name)
448     for j in range(len(listename)):
449         if listename[j] != ("'" or ' '):
450             displayname.append(listename[j])
451
452 final = ''.join(displayname)
453 return final
454
455 dico = {}
456 arrets = open('gtfs\stops.csv')
457 read = csv.reader(arrets)
458 for ligne in read:
459     if ligne[2] == 'stop_name':
460         continue
461     if transname(ligne[2]) not in dico:
462         dico[transname(ligne[2])] = [ligne[0]]
463     else:
464         listetempo = dico[transname(ligne[2])]
465         dico[transname(ligne[2])] = listetempo + [ligne[0]]
466
467 arrets.close()
468
469 def diffheure(h1, h2):
470     return transformeheure(h1) - transformeheure(h2)
471
472 def min_dif(liste, heure_min_dif):
473     if liste == []:
474         return None
475     minimum = liste[0]
476     for elt in liste:
477         if diffheure(elt[1], heure_min_dif) < diffheure(minimum[1], heure_min_dif):
478             minimum = elt
479     return minimum
480
481 def recherche_heure(trip_headsign, stop_id, day, heure):
482     f = open('fichehoraire/' + str(stop_id) + '.csv')
483     fic = csv.reader(f)
484     listeheure = []
485     for ligne in fic:
486         if verifday(day, ligne[0]):
487             if ligne[5] == trip_headsign:
488                 if transformeheure(ligne[1]) >= transformeheure(heure):
489                     listeheure.append(ligne)
490
491     def evo(dico):
492         compttot = len(dico)
493         compt = 0
494         for elts in dico.items():
495             if elts[1] == True:
496                 compt += 1
497         print(str(compt) + "/" + str(compttot))
498
499     return min_dif(listeheure, heure)
500
501 dicodejavu = {}
502 arrets = open('gtfs/stops.csv')
503 arretsR = csv.reader(arrets)
504 for ligne in arretsR:
505     if ligne[2] in dicodejavu:
506         continue
507     dicodejavu[ligne[2]] = False
508
509 fic2 = open('gtfs/trips.csv')
510 f2 = csv.reader(fic2)
511 liste_trip_id = []
512 dico_trip_id = {}
513
514 for ligne in f2:
515     if ligne[3] == 'trip_headsign':
516         continue
517     if ligne[3] not in liste_trip_id:
518         liste_trip_id.append(ligne[3])
519         dico_trip_id[ligne[3]] = [None, None]
520
521 fic2.close()
522
523 def ponderation(trip, stop_id_arrivee, heure_depart, day):
524     result = recherche_heure(trip, stop_id_arrivee, day, heure_depart)
525     return diffheure(result[1], heure_depart)
526
527 def diffheure(result[1], heure_depart):
528
529 def ponderation2(trip_id, stop_id_suiv):
530     fic = open('fichehoraire/' + str(stop_id_suiv) + '.csv')
531     f = csv.reader(fic)
532     for ligne in f:
533         if ligne[0] == trip_id:
534             return ligne
535
536
537 def arret_suivant(trip_id, stop_id): # Renvoie l'arrêt suivant d'une ligne donnée
538     fic = open('Lignes2/' + str(trip_id) + '.csv')
539     f = csv.reader(fic)
540     compteur = 0
541     for ligne in f:
542         if ligne[0] == trip_id:
543             if compteur == 1:
544                 return ligne[0]
545             if ligne[0] == stop_id:
546                 compteur += 1
547
548     return None
549     fic.close()
550
551
552 fic = open('bus_metz_lignes.csv', 'w', newline='')
553 writer = csv.writer(fic)
554 writer.writerow(['line', 'station_name1', 'station_name2', 'time', 'name_depart'])
555 writer.writerow(['\n', quoting=csv.QUOTE_NONE, quotechar=None])
556 writer = csv.writer(fic, lineterminator='\n',
557                         quoting=csv.QUOTE_NONE, quotechar=None)
558
559 def min_ponderation(Liste):
560     mini = min(Liste, key=lambda x: x[1])
561     return mini
562
563
564
565 def min_ponderation2(Liste):
566     mini = min(Liste, key=lambda x: transformeheure(x[1]))
567     return mini
568
569
570 def evo(dico):
571     compttot = len(dico)
572     compt = 0
573     for elts in dico.items():
574         if elts[1] == True:
575             compt += 1
576     print(str(compt) + "/" + str(compttot))
577
578
579 #
580
581 def verif_en_arriere(ligne, dico):
582     stop_id = ligne[3]
583     trip_headsign = ligne[5]
584     if dico[ligne[5]] == [None, None]:
585         return False
586     stop_id_decouverte = dico[ligne[5]][1]
587     fic = open('Lignes2/' + str(trip_headsign) + '.csv')
588     f = csv.reader(fic)
589     compteur = 0
590     for row in f:
591         if row[0] == stop_id_decouverte:
592             if compteur == 1:
593                 return False
594             else:
595                 return True
596
597     if row[0] == stop_id:
598         compteur += 1
599     fic.close()
600
601 def verif_en_arriere2(ligne, dico):
602     stop_id = ligne[3]
603     trip_headsign = ligne[5]
604     if dico[ligne[5]] == [None, None]:
605         return False
606     stop_id_decouverte = dico[ligne[5]][1]
607     if stop_id == stop_id_decouverte:
608         return False
609     fic = open('Lignes2/' + str(trip_headsign) + '.csv')
610     f = csv.reader(fic)
611     compteur = 0
612     for row in f:
613         if row[0] == stop_id_decouverte:
614             if compteur == 1:
615                 return True
616             else:
617                 return False
618
619     if row[0] == stop_id:
620         compteur += 1
621     fic.close()
622
623 statut = False
624 liste_mini = []
625 dicotripietemp = {}
626 def latestmilleur(info,elt):
627     trip_id_avant = info[0]
628     stop_id_decouverte_avant = info[1]
629     fich2 = open('fichehoraire/' + str(stop_id_decouverte_avant) + '.csv')
630     f = csv.reader(fich2)
631     next(f)
632     h1 = None
633     h2 = None
634     for row in f:
635         if row[0] == trip_id_avant:
636             h1 = transformeheure(row[1])
637         if row[0] == elt[0]:
638             h2 = transformeheure(row[1])
639     if h1 == None or h2 == None:
640         return False
641     if h2 < h1:
642         return True
643     else:
644         return False
645
646 fich2.close()
647
648 def creation_arretes(stop_name_D, stop_name_A, heure, day):
649     print("Départ: ", stop_name_D, "heure: ", heure)
650     global statut
651
652     if statut:
653         return
654     liste_stop_id_D = dico[stop_name_D]
655     liste_trip_headsign_depart = [] # Toutes les lignes qui passent par l'arrêt de départ
656
657     for elt in liste_trip_id: # Remplissage de la liste trip_headsign
658         for stop_id in liste_stop_id_D:

```

```

665 fic = open('Lignes2/' + str(elt) + '.csv')
666 f = csv.reader(fic)
667 for ligne in f:
668     if ligne[0] == stop_id:
669         liste_trip_headsign_depart.append(elt)
670     break
671 fic.close()
672
673 #Liste des candidats possibles, cela donne l'heure la plus proche
674 #pour chaque ligne
675 liste_tempo = []
676 for trip in liste_trip_headsign_depart:
677     for stop_id in liste_stop_id_0:
678
679         result = recherche_heure(trip, stop_id, day, heure)
680         if result == None:
681             continue
682         liste_tempo.append(result)
683
684 # Si une ligne n'a jamais été vue, ajoute le trip_id de découverte, et le stop_id qui a fait la découverte
685 for elt in liste_tempo:
686     if dico_trip_id[elt[5]] == [None, None]:
687         dico_trip_id[elt[5]] = [elt[0], elt[3]]
688     elif elt[1] == dico_trip_id[elt[5]][0] and dico_trip_id[elt[5]] != [None, None]:
689         if latestelleur(dico_trip_id[elt[5]], elt):
690             dico_trip_id[elt[5]] = [elt[0], elt[3]]
691
692 liste_recherche_heure = []
693 for stop_id in liste_stop_id_0:
694     for elt in liste_trip_headsign_depart:
695         trip_id = dico_trip_id[elt]
696         ligne = ponderation2(trip_id[0], stop_id)
697         if ligne == None:
698             continue
699         if verif_en_arriere(ligne, dico_trip_id):
700             liste_recherche_heure.append(ligne)
701
702 if liste_recherche_heure == []:
703     return
704
705 minimum = min_ponderation2(liste_recherche_heure)
706 liste_mini.append(minimum)
707
708 newheure = minheure[1]
709 if stop_name_D == stop_name_A:
710     statut = True
711     fin(stop.name_D, stop.name_A, liste_mini, newheure)
712
713 #Remplir la liste des arrêts suivants à l'arrêt considéré
714 liste_stop_id_suiv = []
715 for result in liste_tempo:
716     for stop_id in liste_stop_id_0:
717
718         stop_id_suiv = arret_suivant(result[5], stop_id)
719         if stop_id_suiv == None:
720             continue
721         if stop_id_suiv in liste_stop_id_suiv:
722             continue
723
724         liste_stop_id_suiv.append(stop_id_suiv)
725
726
727
728
729 liste_tempo_suiv = []
730 for stop_id_suiv in liste_stop_id_suiv:
731     liste_trip_headsign_suiv = [] # Liste des trip_headsign qui sont à l'arrêt suivant
732     for elt in liste_trip_id:
733         fic = open('Lignes2/' + str(elt) + '.csv')
734         f = csv.reader(fic)
735         for ligne in f:
736             if ligne[0] == stop_id_suiv:
737                 liste_trip_headsign_suiv.append(elt)
738                 break
739         fic.close()
740
741     for elt in liste_trip_headsign_suiv:
742         if dico_trip_id == [None, None]:
743             continue
744         trip_id = dico_trip_id[elt][0]
745         result = ponderation2(trip_id, stop_id_suiv)
746         if result == None:
747             continue
748
749         temps_total = diffheure(result[1], newheure)
750
751         if dicodejavu[stopid_to_name(stop_id_suiv)] == True:
752             continue
753
754         liste_tempo_suiv.append([stop_id_suiv, temps_total, stopid_to_name(stop_id_suiv)])
755
756 dicodejavu[stop_name_D] = True
757 evide(dicodejavu)
758
759 liste_tempo_suiv_triee = sorted(liste_tempo_suiv, key=lambda x: x[1])
760
761 for elt in liste_tempo_suiv_triee:
762     stop_id_suiv2 = elt[0]
763     temps = elt[1]
764     stop_name_suiv_liste = [k for k, v in dico.items() if stop_id_suiv2 in v]
765     if dicodejavu[stop_name_suiv_liste[0]] == True:
766         continue
767     creation_arretes(stop_name_suiv, stop_name_A, transformeheureinverse(transformeheure(newheure)) + temps
768     ), day)
769
770 def stopid_to_name(stop_id):
771     fic = open('gtfs/stops.csv')
772     f = csv.reader(fic)
773     for row in f:
774         if row[0] == str(stop_id):
775             return row[2]
776
777 # Faire un dico avec tous les trip_id parcourus avec le temps qui correspond
778
779 chemin_trip_id = []
780 def fin(stop_name_D, stop_name_A, liste_mini, newheure):
781     global pathtot
782     trip_id_ini, heure, _, stop_id, _, trip_headsign = liste_mini[len(liste_mini) - 1]
783     trip_id_debut = liste_mini[0][0]
784     stopiddebut = liste_mini[0][1]
785
786     path = []
787     def auto(trip_id, stop_id, trip_headsign):
788         path = []
789         global chemin_trip_id
790         global pathtot
791         info_avant = [v for k, v in dico_trip_id.items() if trip_id in v]
792         stop_id_avant = info_avant[0][1]
793
794         fich = open('Lignes2/' + str(trip_headsign) + '.csv')
795         f = csv.reader(fich)
796         next(f)
797         statusboucle = False
798         for ligne in f:
799
800             if int(ligne[0]) == int(stop_id_avant):
801                 statusboucle = True
802             if statusboucle:
803                 path.append(int(ligne[0]))
804             if int(ligne[0]) == int(stop_id):
805                 statusboucle = False
806
807         fich.close()
808
809         # Comparer les noms
810         for k in liste_mini:
811             if stopid_to_name(k[3]) == stopid_to_name(stop_id_avant):
812                 next_info = k
813
814         nexttrip_id = next_info[0]
815         nexttripheadsign = next_info[1]
816         nextstop_id = next_info[3]
817
818         # Récupérer les correspondances
819         trip_id_corr = info_avant[0][0]
820         stop_id_corr = info_avant[0][1]
821         fic = open('ficheshoraires/' + str(stop_id_corr) + '.csv')
822         f = csv.reader(fic)
823         for ligne in f:
824             if ligne[0] == trip_id_corr:
825                 chemin_trip_id = [ligne[5], stopid_to_name(ligne[3]), ligne[1]] + chemin_trip_id
826
827             if ligne[0] == trip_id_corr:
828                 chemin_trip_id = [ligne[5], stopid_to_name(ligne[3]), ligne[1]] + chemin_trip_id
829
830
831         pathtot = path + pathtot
832         if nextstop_id == stopiddebut:
833             chemin_stop_name = [stopid_to_name(i) for i in pathtot]
834             print("Heure d'arrivée : ", newheure)
835             print("Chemin final : ", chemin_stop_name)
836             print("Correspondances : ", chemin_trip_id)
837
838         else:
839             aux(nexttrip_id, nextstop_id, nexttripheadsign)
840
841 creation_arretes(depart, arrivee, heure, day)
842
843 #Création d'un fichier qui contient tous les arrêts de tout le réseau#
844 fich = open("gtfs/stops.csv")
845 f = csv.reader(fich)
846
847 lignecompt = 0
848
849 arrrets = []
850 nomdejavu = []
851 for ligne in f:
852     tempo = []
853     if (lignecompt >= 1):
854         tempo.append(ligne[2])
855         tempo.append(ligne[4])
856         tempo.append(ligne[5])
857         tempo.append(ligne[0])
858         arrrets.append(tempo)
859
860 lignecompt += 1
861
862
863 fich.close()
864
865 fic = open("bus_metz.csv", "w", newline="")
866
867 writer = csv.writer(fic)
868 writer.writerow(["stop_id", "lat", "lon", "display_name"])
869 writer = csv.writer(fic, lineterminator='\n',
870 quoting=csv.QUOTE_NONE, quotechar=None)
871
872
873 def transname(name):
874     newname = name.replace(" ", "<br>")
875     displayname = []
876     listename = list(newname)
877     for j in range(len(listename)):
878         if listename[j] != ('"' or '''):
879             displayname.append(listename[j])
880
881     displayname.append(listename[j])
882
883 final = ''.join(displayname)
884
885 for i in range(len(arrrets)):
886     name,lat,lon, stop_id = arrrets[i]
887     displayname = transname(name)
888     writer.writerow((stop_id, lat, lon, displayname))
889
890 fic.close()
891
892 #Création du nouveau visuel#
893
894 M = pd.read_csv('bus_metz.csv')
895
896 fig = go.Figure()
897
898 fig = go.Figure(go.Scattermapbox(
899     mode = "markers+text",
900     lon = M['lon'],
901     lat = M['lat'],
902     showlegends=True,
903     marker = go.scattermapbox.Marker(size=9),
904     text=M['display_name'].apply(lambda x: \
905         '<b>' + str(x) + \
906         '</b>'),
907         textfont=dict(color="black", size=8, family="bold"),
908         textposition="middle center"
909 ))
910
911 fic2 = open('gtfs/trips.csv')

```

```

912 f2 = csv.reader(fic2)
913 liste_trip_id = []
914 for ligne in f2:
915     if ligne[3] == 'trip_headsign':
916         continue
917     if ligne[3] not in liste_trip_id:
918         liste_trip_id.append(ligne[3])
919 fic2.close()
920
921
922 for elt in liste_trip_id:
923     fic = open('Lignes2/' + str(elt) + '.csv')
924     f = csv.reader(fic)
925     Listetempo_stop_id_of_one_trip = []
926     for ligne in f:
927         if ligne[0] == "stop_id":
928             continue
929         else:
930             Listetempo_stop_id_of_one_trip.append(int(ligne[0]))
931     Mtemp = M[M['stop_id'].isin(Listetempo_stop_id_of_one_trip)]
932     def sortir(col):
933         correspondence = {Listetempo_stop_id_of_one_trip: order for order, Listetempo_stop_id_of_one_trip in
934         enumerate(Listetempo_stop_id_of_one_trip)}
935     return col.map(correspondence)
936
937
938 Mtemp.sort_values(by="stop_id", key=lambda column: column.map(lambda e: Listetempo_stop_id_of_one_trip.
939 index(e)), inplace=True)
940
941 fig.add_trace(go.Scattermapbox(
942     mode="markers+lines+text",
943     lon=Mtemp['lon'],
944     lat=Mtemp['lat'],
945     showlegend=True,
946     marker=go.scattermapbox.Marker(size=9),
947     text=Mtemp['display_name'].apply(lambda x: \
948         '<b>' + str(x) + \
949         '</b>'),
950     textfont=dict(color="black", size=8, family='bold'),
951     textposition='middle center'
952 ))
953
954 fig.update_layout(mapbox_style="open-street-map")
955 fig.show()
956
957 ##Algorithme de gestion du retard##
958
959 def is_valid_time(time_str):
960     try:
961         time = datetime.strptime(time_str, "%H:%M:%S")
962         return True
963     except ValueError:
964         return False
965
966 schedule = ""
967
968 color = ["red", "blue", "green", "lime", "seagreen", "violet"]
969 while not is_valid_time(schedule):
970     schedule = input("Quelle heure (Format HH:MM:SS): ")
971
972 jour = input("Quel jour: ")
973
974
975 def journee(day):
976     if day == "Dimanche":
977         return ["HIV2223-Hdim2223-Dimanche-40"]
978     elif day == "Samedi":
979         return ["HIV2223-HSam2223-Samedi-20"]
980     elif day == "Lundi":
981         return ["HIV2223-HSam2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
982                 "HIV2223-HSam2223-L-Ma-J-V-01-1101100"]
983     elif day == "Mardi":
984         return ["HIV2223-HSam2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
985                 "HIV2223_PVS-PVS_2223-Semaine-01-0100100", "HIV2223-HSam2223-L-Ma-J-V-01-0100100",
986                 "HIV2223-HSam2223-L-Ma-J-V-01-1101100"]
987     elif day == "Mercredi":
988         return ["HIV2223-HMer2223-Mercredi-10", "HIV2223_PVS-PVS_2223-Semaine-01",
989                 "HIV2223_PVS-PVS_2223-Semaine-01-0010000"]
990     elif day == "Jeudi":
991         return ["HIV2223-HSam2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
992                 "HIV2223-HSam2223-L-Ma-J-V-01-1101100"]
993
994     elif day == "Vendredi":
995         return ["HIV2223-HSem2223-L-Ma-J-V-01", "HIV2223_PVS-PVS_2223-Semaine-01",
996                 "HIV2223_PVS-PVS_2223-Semaine-01-0100100", "HIV2223-HSem2223-L-Ma-J-V-01-0100100",
997                 "HIV2223-HSem2223-L-Ma-J-V-01-1101100"]
998     raise ValueError("Mauvais format jour")
999
1000
1001 day = journee(jour)
1002
1003
1004 def verifday(day, trip):
1005     statuts = False
1006     for elt in day:
1007         if elt in trip:
1008             statuts = True
1009     return statuts
1010
1011 #Initialisation de listes pour la visualisation
1012 pt_lat = []
1013 pt_lon = []
1014 colorlist = []
1015 textlist = []
1016 statutdepassement = 0
1017
1018 def find_stops(trip_headsign):
1019     fich = open("Lignes2/" + str(trip_headsign) + ".csv")
1020     f = csv.reader(fich)
1021     next(f)
1022     stops = []
1023     for row in f:
1024         stops.append(row[0])
1025     return stops
1026
1027 def transformeheure(heure):
1028     h = heure.split(":")
1029     return int(h[0]) * 3600 + int(h[1]) * 60 + int(h[2])
1030
1031 def transformeheureinverse(temps):
1032     heures = temps // 3600
1033     minutes = (temps % 3600) // 60
1034     secondes = temps % 60
1035
1036     temps_formatte = "{:02d}::{:02d}::{:02d}".format(heures, minutes, secondes)
1037     return temps_formatte
1038
1039
1040 M = pd.read_csv('bus_netz.csv')
1041
1042 # Demander à l'utilisateur de saisir le trip_headsign de la ligne à représenter
1043 trip = input("Veuillez entrer le nom de la ligne à représenter: ")
1044
1045 with open('gtfs/routes.txt') as f:
1046     reader = csv.reader(f)
1047     next(reader)
1048     for row in reader:
1049         if row[3] == trip:
1050             route_id = row[1]
1051
1052 # Récupérer tous les trip_headsign en fonction d'un numéro de route
1053 trip_headsign_liste = []
1054 with open('gtfs/trips.txt') as f:
1055     reader = csv.DictReader(f)
1056     for row in reader:
1057         if row['route_id'] == route_id:
1058             if row['trip_headsign'] not in trip_headsign_liste:
1059                 trip_headsign_liste.append(row['trip_headsign'])
1060
1061 # Récupérer les stop_id associés au trip_headsign dans le dossier Lignes2
1062
1063 #Création de la figure vide
1064 fig = go.Figure(go.Scattermapbox(
1065
1066 ))
1067
1068 #On ajoute les lignes sur la figure
1069 for i in range(len(trip_headsign_liste)):
1070     trip_headsign = trip_headsign_liste[i]
1071     with open('Lignes2/{}.csv'.format(trip_headsign)) as f:
1072         reader = csv.reader(f)
1073         next(reader) # Skip header row
1074         stop_ids = [int(row[0]) for row in reader]
1075
1076

```

```

177 # Trier les arrêts en fonction de leur ordre dans la ligne
178 M_temp = M[M['stop_id'].isin(stop_ids)]
179
180
181 def sortir(col):
182     correspondance = {stop_ids: order for order, stop_ids in
183         enumerate(stop_ids)}
184     return col.map(correspondance)
185
186
187 M_temp.sort_values(by="stop_id", key=lambda column: column.map(lambda e: stop_ids.index(e)),
188                     inplace=True)
189
190
191 # Créer le visuel de la ligne
192 fig.add_trace(go.Scattermapbox(
193     mode="markers+lines+text",
194     lon=M_temp['lon'],
195     lat=M_temp['lat'],
196     name=trip_headsign,
197     line=dict(color=color[i]),
198     showlegend=True,
199     marker=go.scattermapbox.Marker(size=9),
200     text=M_temp['display_name'].apply(lambda x: '<b>' + str(x) + '</b>'),
201     textfont=dict(color="black", size=8, family="bold"),
202     textposition='middle center'
203 ))
204
205 # Dictionnaire pour vérifier si un trip_id appartient au bon trip_headsign
206 dicoverif = {}
207
208 #Dictionnaire qui sert à obtenir le pourcentage de progression de l'arrêt retard dans pointinter
209 dicolatlonretard = {}
210
211
212 #Fonction qui permet de positionner les bus sur la ligne
213 def pointinter(stop_id, stop_id_suiv, arrival_time, next_arrival_time, schedule_time, colornb, trip_id,
214     statut):
215     global statutdepassement
216     h2 = transformeheure(next_arrival_time)
217     h1 = transformeheure(arrival_time)
218     h = transformeheure(schedule_time)
219     dureetot = h2 - h1
220     temps = h - h1
221     pourcentageprogression = (100 * temps) / dureetot
222     f = open("gtfs/stops.csv")
223     reader = csv.reader(f)
224     for row in reader:
225         if row[8] == stop_id:
226             lat1 = row[4]
227             lon1 = row[5]
228         if row[8] == stop_id_suiv:
229             lat2 = row[4]
230             lon2 = row[5]
231
232     difflatlat = float(lat2) - float(lat1)
233     difflat = (difflatlat * pourcentageprogression) / 100
234     newlat = float(lat1) + difflat
235     difflon = float(lon2) - float(lon1)
236     difflon = (difflon * pourcentageprogression) / 100
237     newlon = float(lon1) + difflon
238
239     if statut == 0:
240         dicolatlonretard[trip_id] = [pourcentageprogression, lat1, lat2, lon1, lon2, temps, stop_id]
241
242     dicopprogress[trip_id].append(pourcentageprogression)
243
244     #Si on veut un train de bus
245     #Vérification si un arrêt était avant le décalage dans le temps et si il est après le bus en retard après
246     #le décalage
247     if statut == 1 and dicopprogress[trip_id][0] < dicopprogress[idbusretard][0] <= dicopprogress[trip_id][2] \
248         and dicoverif[trip_id] == dicoverif[idbusretard] and statutdepassement == 1:
249         pourcentageprogressionr, lat1r, lat2r, lon1r, lon2r, tempsr, stop_idr = dicolatlonretard[idbusretard]
250         difflatlatr = float(lat2r) - float(lat1r)
251         rand = rd.uniform(0,4)
252         difflatr = (difflatlatr * (pourcentageprogressionr - rand)) / 100
253         newlatr = float(lat1r) + difflatr
254         difflonr = float(lon2r) - float(lon1r)
255         difflonr = (difflonr * (pourcentageprogressionr - rand)) / 100
256         newlonr = float(lon1r) + difflonr
257
258         #avoir le temps de retard qui le bus prends à l'aide d'un fonction
259         recherchetimeprétard(schedule_time, tempsr, stop_idr, trip_id)
260
261         pt_lat.append(newlatr)
262         pt_lon.append(newlonr)
263         colorlist.append(color[colornb])
264         textlist.append(trip_id)
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
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
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

1239
1240 #On ajoute la couche et on affiche
1241 fig.add_trace(trace2)
1242 fig.update_layout.mapbox_style="open-street-map")
1243 fig.show()
1244
1245 #Dictionnaire pour suivre le retard de chaque bus
1246 dicretard = {}
1247 for tripid in trip_id_total:
1248     dicretard[tripid] = 0
1249
1250 ### Gestion du retard NBH
1251
1252 print("Parmi les différents bus suivants, lequel voulez vous qu'il subisse un retard?")
1253 DicoTrip_id = {}
1254 compt = 0
1255 for k in textlist:
1256     DicoTrip_id[compt] = k
1257     compt += 1
1258
1259 indicebusretard = int(input(""))
1260
1261 idbusretard = DicoTrip_id[indicebusretard]
1262 newcolorlist = colorlist.copy()
1263 # Garder en mémoire les lon et lat de l'arrêt qui doit subir un arrêt
1264 longitudes = trace2.lon
1265 latitudes = trace2.lat
1266
1267 latretard = latitudes[indicebusretard]
1268 lonretard = longitudes[indicebusretard]
1269
1270 #On change uniquement la couleur du bus qui subit un arrêt.
1271 newcolorlist[indicebusretard] = "black"
1272
1273
1274 # Convertir fig.data en liste
1275 fig_data_list = list(fig.data)
1276
1277 # Supprimer la trace souhaitée
1278 fig_data_list.remove(trace2)
1279
1280 # Reconvertisr fig_data_list en tuple
1281 fig.data = tuple(fig_data_list)
1282
1283 trace2 = go.Scattermapbox(
1284     mode="markers",
1285     lon=pt_lon,
1286     lat=pt_lat,
1287     marker=go.scattermapbox.Marker(size=19, color=newcolorlist),
1288     text=textlist
1289 )
1290
1291 fig.add_trace(trace2)
1292 fig.update_layout.mapbox_style="open-street-map")
1293
1294 fig.show()
1295
1296 statutdepassement = int(input("Voulez-vous simuler un dépassement (0) ou un train de bus (1): "))
1297
1298 newhoraire = ""
1299
1300 while not is_valid_time(newhoraire):
1301     newhoraire = input("Quelle heure (Format HH:MM:SS): ")
1302
1303 difftime = datetime.strptime(newhoraire, "%H:%M:%S") - datetime.strptime(schedule, "%H:%M:%S")
1304
1305 diff_timedelta = timedelta(seconds=difftime.seconds)
1306
1307 # Extraction des heures, minutes et secondes de la différence
1308 diff_heure = diff_timedelta.seconds // 3600
1309 diff_minute = (diff_timedelta.seconds % 3600) // 60
1310 diff_seconde = diff_timedelta.seconds % 60
1311 diff_format = "{:02d}:{:02d}:{:02d}".format(diff_heure, diff_minute, diff_seconde)
1312
1313 dicretard[idbusretard] = diff_format
1314
1315
1316 def nouveaux_horaires(schedule_time, stops, trip_headsign_liste, day, trip_id_total):
1317     for k in range(len(stops)-1):
1318         stop_id, stop_id_suiv = stops[k], stops[k+1]
1319         fich = open("fichehoraire/" + str(stop_id) + ".csv")
1320         reader1 = csv.reader(fich)
1321         for row in reader1:
1322
1323             trip_id, arrival_time, departure_time, stop_id, stop_sequence, trip_headsign = row
1324             if trip_id in trip_id_total:
1325
1326                 nbcolor = trip_headsign_liste.index(row[5])
1327                 fich2 = open("fichehoraire/" + str(stop_id_suiv) + ".csv")
1328                 reader2 = csv.reader(fich2)
1329                 for row2 in reader2:
1330                     if row2[8] == trip_id:
1331                         next_arrival_time = row2[1]
1332                         if transformeheure(arrival_time) >= transformeheure("23:59:59"):
1333                             continue
1334                         if transformeheure(next_arrival_time) >= transformeheure("23:59:59"):
1335                             continue
1336                         arrival_time1 = datetime.strptime(arrival_time, "%H:%M:%S").time()
1337                         next_arrival_time1 = datetime.strptime(next_arrival_time, "%H:%M:%S").time()
1338                         if arrival_time1 <= datetime.strptime(schedule_time, "%H:%M:%S").time() <
1339                             next_arrival_time1:
1340                             if verifday(day, trip_id):
1341                                 dicprogress[trip_id].append(k)
1342                                 pointinter(stop_id, stop_id_suiv, arrival_time, next_arrival_time,
1343                                         schedule_time,
1344                                         nbcolor, trip_id, 1)
1345
1346                         pt_lat = []
1347                         pt_lon = []
1348                         colorlist = []
1349                         textlist = []
1350                         for elt in trip_headsign_liste:
1351                             stops = find_stops(elt)
1352                             trip_id_list = nouveaux_horaires(newhoraire, stops, trip_headsign_liste, day, trip_id_total)
1353
1354                         # Convertir fig.data en liste
1355                         fig_data_list = list(fig.data)
1356
1357                         # Supprimer la trace souhaitée
1358                         fig_data_list.remove(trace2)
1359
1360                         # Reconvertisr fig_data_list en tuple
1361                         fig.data = tuple(fig_data_list)
1362
1363
1364                         # Obtenir le nouvel indice du bus à l'arrêt
1365                         textliststop = trace2.text
1366                         for i, text in enumerate(textliststop):
1367                             if text == idbusretard:
1368                                 special_point_index = i
1369                                 break
1370
1371                         # Changer les attributs du bus à l'arrêt
1372                         colorlist[special_point_index] = "black"
1373                         pt_lat[special_point_index] = latretard
1374                         pt_lon[special_point_index] = lonretard
1375
1376                         trace2 = go.Scattermapbox(
1377                             mode="markers",
1378                             lon=pt_lon,
1379                             lat=pt_lat,
1380                             marker=go.scattermapbox.Marker(size=19, color=colorlist),
1381                             text=textlist
1382 )
1383
1384                         fig.add_trace(trace2)
1385
1386                         fig.show()
1387
1388
1389 def stopid_to_name(stop_id):
1390     fic = open('gtfs/stops.csv')
1391     f = csv.reader(fic)
1392     for row in f:
1393         if row[0] == str(stop_id):
1394             return row[2]
1395
1396 #Création des fiches horaires avec le retard
1397 def fichehorairepostretard(schedule_time, stops, trip_id_total):
1398     fich = open('resultretard/' + "retard" + '.csv', 'w', newline='')
1399     writer = csv.writer(fiche)
1400
1401     writer.writerow(("trip_id", "arrival_time", "late_time", "stop_id", "stop_name", "trip_headsign"))
1402     writer = csv.writer(fiche, lineterminator='\n',
1403                         quoting=csv.QUOTE_NONE, quotechar=None, escapechar='\\')
1404
1405     for stop_id in stops:
1406         fich = open("fichehoraire/" + str(stop_id) + ".csv")
1407         reader1 = csv.reader(fich)
1408         for row in reader1:
1409             if row[0] in trip_id_total:
1410                 horaireretard = dicretard[row[0]]
1411                 if horaireretard == 8:
1412                     horaireretard = "00:00:00"
1413                     resultat = transformeheureinverse(transformeheure(str(horaireretard)) +
1414                                         transformeheure(str(row[1])))
1415                     if datetime.strptime(resultat, "%H:%M:%S").time() >= datetime.strptime(
1416                         (schedule_time, "%H:%M:%S").time()):
1417                         name = stopid_to_name(stop_id)
1418                         writer.writerow((row[0], row[1], dicretard[row[0]], row[3], name, row[5]))
1419
1420     stops_total = []
1421     for elt in trip_headsign_liste:
1422         stops = find_stops(elt)
1423         stops_total += stops
1424
1425 fichehorairepostretard(newhoraire, stops_total, trip_id_total)

```