

Documentazione Progetto SUMO

Next Generation Networks

Docente del corso:

Fabrizio Granelli

Yousef Soliman Mekheimer - Gabriele Ughetto

Indice

- Introduzione
- Tutorial Progetto SUMO - Come Avviare la Simulazione
 - Introduzione
 - Requisiti
 - Procedura
 - Google Sheet
- Tutorial Progetto SUMO - Come Ricreare la Simulazione
- Descrizione Codice
 - Introduzione
 - File runner.py
 - File support.py
 - File gSeetsManager.py
 - File osm.sumocfg
- Problematiche
 - Mappa con connessioni imprecise o interrotte
 - Simulazione pesante
 - Modifica del file *gtfs_publictransport.rou.xml*
 - Aggiornamento API

Introduzione

SumoGui - Naso Generi - Progetto S... - (669 S... - (669 Java - SumoNGUI - Sumo Data

File Modifica Visualizza Inserisci Formato Dati Strumenti Estensioni Guida Appena modificato Condividi

629

6:33:38

	A	B	C	D
1	Bus Principale	Bus Incontrato	Tempo Connesso	Orario d'incontro
2	I4_d05:03	I4_d05:06	35	5:19:22
3	I4_d05:06	I4_d05:03	35	5:19:22
4	I4_d05:06	I15_d05:29	25	5:32:05
5	I15_d05:29	I4_d05:06	25	5:32:05
6	I15_d05:29	I11_d05:28	48	5:34:26
7	I11_d05:28	I15_d05:29	48	5:34:26
8	I3_d05:17	I8_d05:21	25	5:43:05
9	I3_d05:17	I3_d05:37	2	5:44:40
10	I8_d05:21	I3_d05:17	25	5:43:05
11	I8_d05:21	I3_d05:37	9	5:44:30
12	I3_d05:37	I8_d05:21	9	5:44:30
13	I3_d05:37	I3_d05:17	2	5:44:40
14	I6_d05:31	I6_d05:53	1	5:54:26
15	I6_d05:53	I6_d05:31	1	5:54:26
16	I6_d05:53	I1_d06:14	61	6:16:56
17	I6_d05:53	I1_d06:17	4	6:18:44
18	I6_d05:53	I6_d06:17	3	6:20:19
19	I6_d05:53	I2_d06:20	1	6:24:33
20	I2_d05:54	I8_d05:49	56	6:00:28
21	I2_d05:54	I4_d05:58	78	6:08:26
22	I2_d05:54	I13_d06:23	15	6:25:44
23	I2_d05:54	INP_d06:20	20	6:26:02
24	I8_d05:49	I2_d05:54	56	6:00:28
25	I8_d05:49	I4_d06:07	2	6:13:38
26	I1_d06:03	I6_d05:57	?	6:04:46

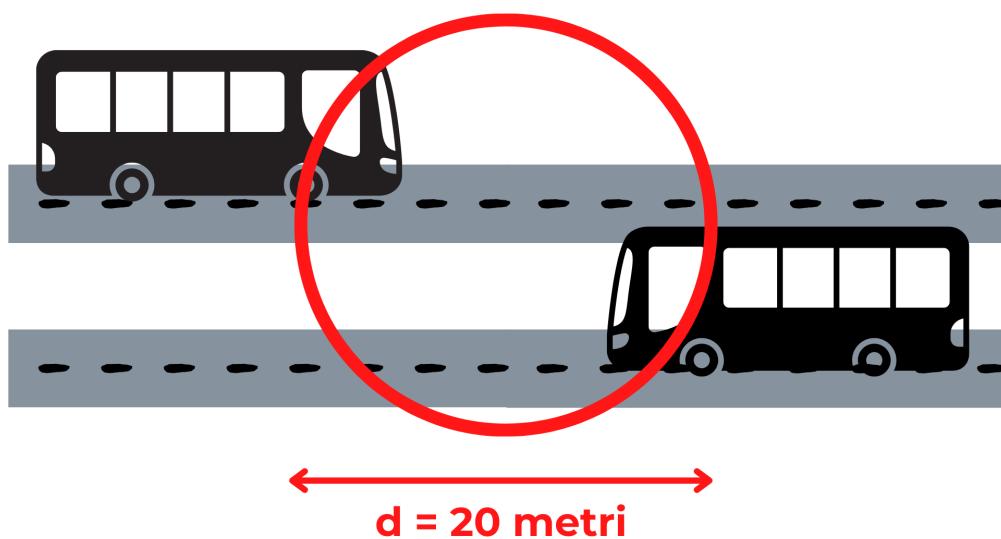
6:33:46 N bus: 20 e sono connessi: 0 bus
6:33:47 N bus: 20 e sono connessi: 0 bus
6:33:48 N bus: 20 e sono connessi: 0 bus
6:33:49 N bus: 20 e sono connessi: 0 bus
6:33:50 N bus: 20 e sono connessi: 0 bus
6:33:51 N bus: 21 e sono connessi: 0 bus
6:33:52 N bus: 21 e sono connessi: 0 bus
6:33:53 N bus: 21 e sono connessi: 0 bus
6:33:54 N bus: 21 e sono connessi: 0 bus
6:33:55 N bus: 21 e sono connessi: 0 bus
6:33:56 N bus: 21 e sono connessi: 0 bus
+ 6:33:57 N bus: 21 e sono connessi: 0 bus
6:33:58 N bus: 21 e sono connessi: 0 bus
6:33:59 N bus: 21 e sono connessi: 0 bus
6:34:00 N bus: 21 e sono connessi: 0 bus
6:34:01 N bus: 21 e sono connessi: 0 bus

sumo-configuration - SUMO 1.16.0

File Edit Settings Location Simulation Window Help

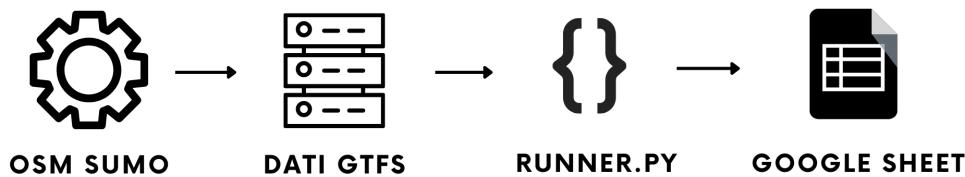
Time: 7:42:52 Delay (m/s): 0 Scale Traffic: 100% |

Il progetto realizzato permette di ottenere un riscontro sulle interconnessioni degli autobus della città di Trento, fornisce il numero di incroci, l'orario di incrocio e il tempo di connessione tra due bus che si incrociano in un area di diametro 20 metri.



La struttura del progetto si può suddividere in quattro fasi:

- Setting e requisiti di Network iniziali
 - Ricerca e mappatura dati GTFS sul Network
 - Elaborazione dei dati con Python e TraCI
 - Esportazione dei risultati su Google Sheet



Tutorial Progetto SUMO - Come Avviare la Simulazione

Introduzione:

Una volta clonata la repository nel proprio computer è possibile lanciare la simulazione presente nel progetto, realizzata in python, avviando il file runner.py presente nella cartella, una volta che sono soddisfatti i requisiti necessari.

Requisiti:

Per avviare la simulazione sopracitata è necessario rispettare i seguenti requisiti:

- [Sumo](#)
- [Python 3.X](#)
- Tool di python necessari: Pandas, gspread
- Account [Developer su Google](#)

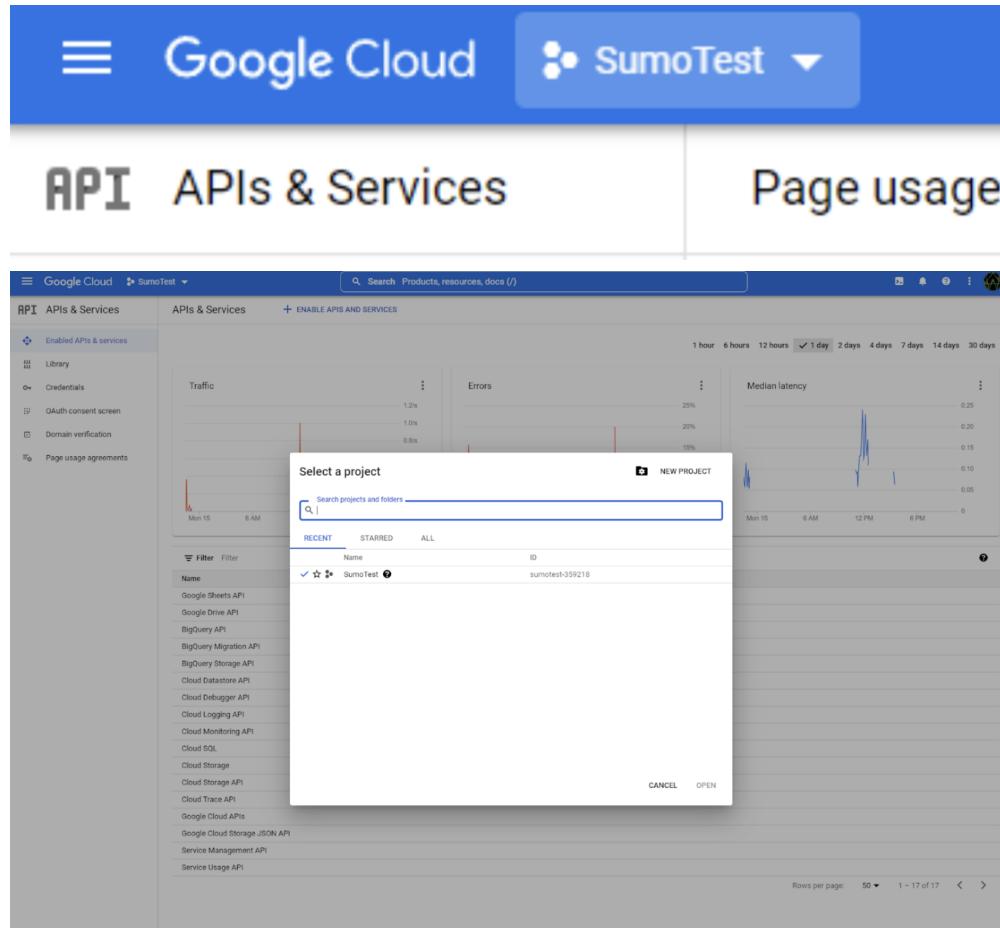
Procedura:

Installare SUMO seguendo dettagliatamente la documentazione, facendo attenzione ad impostare correttamente i path nelle variabili di sistema.

Su Windows sconsigliamo di installare SUMO nelle Cartelle Program Files o Program Files(x86) poiché possono esserci dei conflitti se vengono avviate simulazioni senza permessi di amministratore. Installando SUMO ad esempio nella cartella utente non si presentano queste problematiche. IMPORTANTE alle variabili di sistema è necessario aggiungere "/your/path/to/sumo/tools" PYTHONPATH Installare Python ed i moduli utilizzando PIP, il package installer di Python per la versione che si intende utilizzare per lanciare la simulazione.

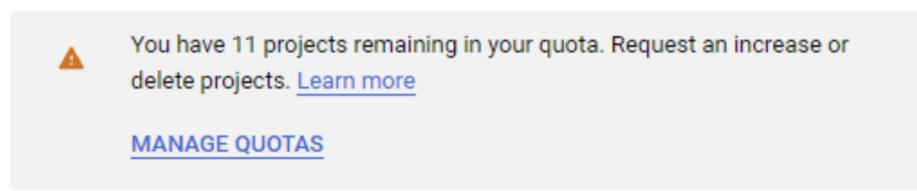
Google Sheet:

Dalla [console per sviluppatori di google](#) è necessario creare un nuovo progetto, cliccando nel menù a tendina a destra della scritta Google Cloud



Creare un progetto:

New Project



Una volta nella dashboard del progetto, cercare nella barra di ricerca: Google Sheets API

The screenshot shows the Google Cloud Platform's APIs & Services dashboard. On the left, under 'Enabled APIs & services', the 'Google Sheets API' is listed. In the center, there's a search bar with 'Search: google sheets api'. Below it, a chart titled 'Stan latency' shows data over a 24-hour period. At the bottom, there are filters for 'Name', 'Requests', 'Errors (%)', and 'Latency, median (ms)'.



Google Sheets API

[Google Enterprise API](#)

The Sheets API gives you full control over the content and appearance of your spreadsheet data.

[ENABLE](#)

[TRY THIS API](#)

Ed abilitare le API, fare la stessa cosa per le API di Google Drive



Google Drive API

[Google Enterprise API](#)

The Google Drive API allows clients to access resources from Google Drive

[ENABLE](#)

[TRY THIS API](#)

Recandosi nella dashboard selezionare Service Accounts

The screenshot shows the Google Cloud Platform dashboard for the project "My Project 87612". The left sidebar lists various services: Cloud overview, Recent, View all products (Pinned), IAM & Admin, Billing, APIs & Services, Marketplace, Compute Engine, Cloud Storage, VPC network, Kubernetes Engine, BigQuery, SQL, Security, Cloud Run, and Google Maps Plat... (More products). The "IAM & Admin" menu is currently open, displaying sub-options: IAM (selected), Identity & Organization, Policy Troubleshooter (THIS PROJECT), Policy Analyzer, Organization Policies, Service Accounts, Workload Identity Federation, Labels, Tags, Settings, Privacy & Security, Identity-Aware Proxy, Roles, Audit Logs, Manage Resources, Create a Project, Asset Inventory, Essential Contacts, Groups, Early Access Center, and Quotas.

E creare un nuovo service account

Service accounts + CREATE SERVICE ACCOUNT DELETE MANAGE ACCESS REFRESH HELP ASSISTANT

Service accounts for project "My Project 87612"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts.](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies.](#)

Filter Enter property name or value

Email	Status	Name	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
No rows to display							

1 Service account details

Service account name

Service Account

Display name for this service account

Service account ID *

service-account

X C

Email address: service-account@exemplary-fiber-359520.iam.gserviceaccount.com



Service account description

Describe what this service account will do

CREATE AND CONTINUE

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

DONE

CANCEL

premere create and continue e selezionare il ruolo Editor:

Service account details

Grant this service account access to project (optional)

Grant this service account access to My Project 87612 so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

The screenshot shows the 'Grant this service account access to project (optional)' step. A modal window is open, titled 'Select a role'. It lists several categories: 'Quick access' (Currently used: Basic), 'By product or service' (Access Approval, Access Context Manager, Actions), and 'Actions'. Under 'Actions', the 'Editor' role is selected, highlighted in grey. A tooltip for 'Editor' states: 'View, create, update, and delete most Google Cloud resources. See the list of included permissions.' A 'DONE' button is visible at the bottom left of the modal.

Una volta creato cliccare l'email del Service Account

	Email	Status	Name	Description
<input type="checkbox"/>	service-account@exemplary-fiber-359520.iam.gserviceaccount.com		Service Account	

Recandosi in Keys

The screenshot shows the 'Keys' tab for the service account. A warning message states: '⚠ Service account keys could pose a security risk if compromised. We recommend you avoid directly authenticating service accounts on Google Cloud [here](#).' Below this, instructions say: 'Add a new key pair or upload a public key certificate from an existing key pair.' and 'Block service account key creation using [organization policies](#). Learn more about setting organization policies for service accounts.' An 'ADD KEY' button is visible at the top left of the key list area. The key list table has columns: Type, Status, Key, Key creation date, and Key expiration date. A message at the bottom says 'No rows to display'.

è necessario creare una nuova chiave privata in json:

Create private key for "Service Account"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

JSON

Recommended

P12

For backward compatibility with code using the P12 format

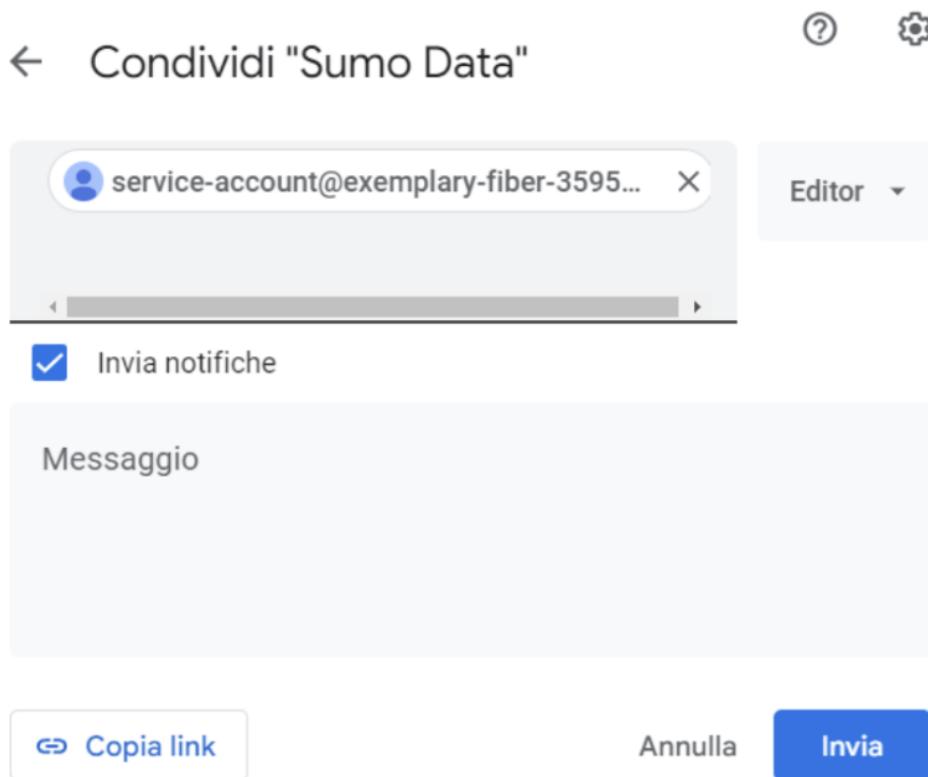
CANCEL **CREATE**

Questo file andrà rinominato in token.json ed inserito nella cartella del programma.

 stopinfos.xml	16/07/2022 19:00	Documento XML	3 KB
 support.py	14/08/2022 22:45	Python File	1 KB
 test.py	02/08/2022 20:55	Python File	3 KB
 test.txt	31/07/2022 18:07	Documento di testo	1 KB
 token.json	14/08/2022 23:01	Adobe After Effect...	3 KB
 trips.trips.xml	16/07/2022 19:00	Documento XML	2 KB
 trips.trips.xml.errorlog	16/07/2022 19:00	File ERRORLOG	1 KB

Ora sarà necessario creare il foglio di calcolo su Google docs, il quale è importante che venga rinominato Sumo Data (Con nomi differenti la simulazione non funzionerà), noi consigliamo semplicemente di creare una copia del file da noi creato: [Sumo Data](#)

Nella copia del file Sumo Data è importante condividere l'accesso con la mail del Service Account creato in precedenza:



Se sono stati eseguiti tutti i passaggi correttamente è ora possibile lanciare la simulazione utilizzando il file runner.py

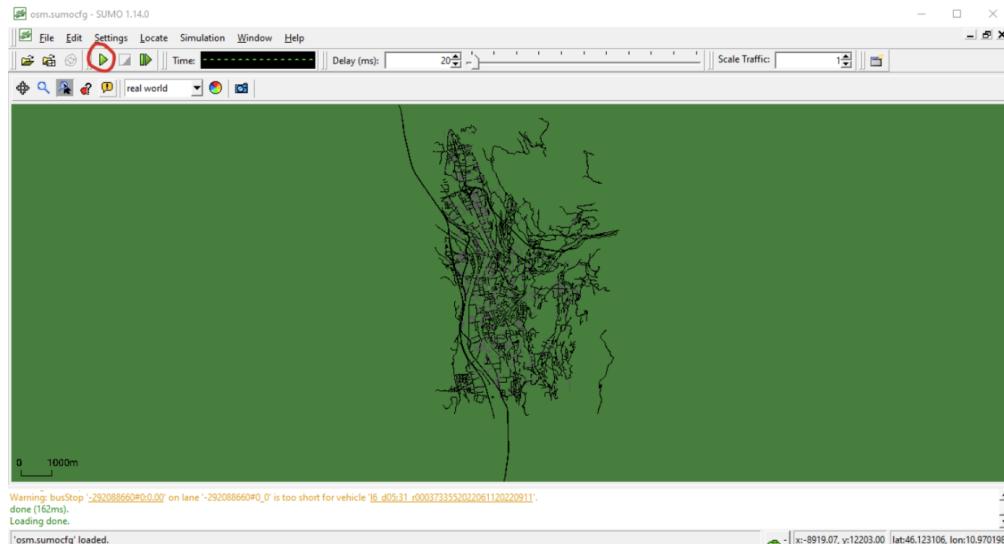
pt_vtypes.xml	11/08/2022 17:12	File di origine XML	1 KB
run.bat	11/08/2022 17:12	File batch Windows	1 KB
runner.py	16/08/2022 10:09	Python File	5 KB
stopinfos.xml	11/08/2022 17:12	File di origine XML	3 KB
support.py	16/08/2022 10:09	Python File	1 KB

Si aprirà in automatico il file Google Sheet Sumo Data

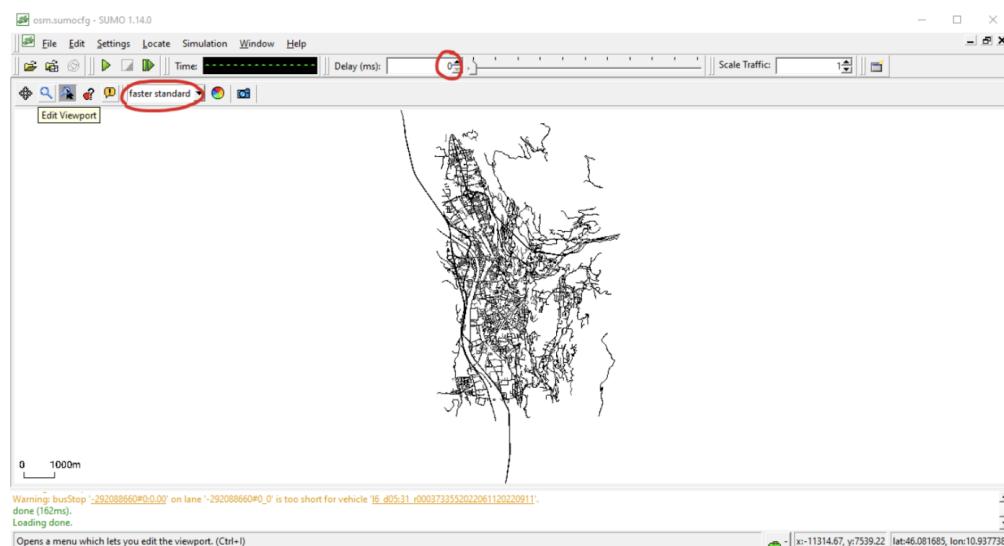
A screenshot of a Google Sheets document titled "Sumo Data". The menu bar includes "File", "Modifica", "Visualizza", "Inserisci", "Formato", "Dati", "Strumenti", "Estensioni", and "Guida". The toolbar includes "Stampa", "Zoom", "100%", and a "Solo visualizzazione" (View only) button. The sheet has a header row with columns labeled "A", "B", "C", "D", and "E". Row 1 contains the titles: "Bus Principale", "Bus Incontro", "Tempo Connesso", and "Orario d'incontro". Rows 2 through 16 are empty.

	A	B	C	D	E
1	Bus Principale	Bus Incontro	Tempo Connesso	Orario d'incontro	
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

e per far partire la simulazione sarà necessario premere sulla freccia verde Run



per velocizzare la simulazione suggeriamo di ridurre il Delay a 0 e impostare la visualizzazione della mappa in faster standard



si apre anche la finestra dello script python che contiene:

- Orario
- Numero bus presenti sulla mappa
- Numero di bus connessi tra loro

```

C:\Windows\py.exe
6:06:13 N bus: 13 e sono connessi: 0 bus
6:06:14 N bus: 13 e sono connessi: 0 bus
6:06:15 N bus: 13 e sono connessi: 0 bus
6:06:16 N bus: 13 e sono connessi: 0 bus
6:06:17 N bus: 13 e sono connessi: 0 bus
6:06:18 N bus: 13 e sono connessi: 0 bus
6:06:19 N bus: 13 e sono connessi: 0 bus
6:06:20 N bus: 13 e sono connessi: 0 bus
6:06:21 N bus: 13 e sono connessi: 0 bus
6:06:22 N bus: 13 e sono connessi: 0 bus
6:06:23 N bus: 13 e sono connessi: 0 bus
6:06:24 N bus: 13 e sono connessi: 0 bus
6:06:25 N bus: 13 e sono connessi: 0 bus
6:06:26 N bus: 13 e sono connessi: 0 bus
6:06:27 N bus: 13 e sono connessi: 0 bus
6:06:28 N bus: 13 e sono connessi: 0 bus
6:06:29 N bus: 13 e sono connessi: 0 bus
6:06:30 N bus: 13 e sono connessi: 0 bus
6:06:31 N bus: 13 e sono connessi: 0 bus
6:06:32 N bus: 13 e sono connessi: 0 bus
6:06:33 N bus: 13 e sono connessi: 0 bus
6:06:34 N bus: 13 e sono connessi: 0 bus
6:06:35 N bus: 13 e sono connessi: 0 bus
6:06:36 N bus: 13 e sono connessi: 0 bus
6:06:37 N bus: 13 e sono connessi: 0 bus
6:06:38 N bus: 13 e sono connessi: 0 bus
6:06:39 N bus: 13 e sono connessi: 0 bus
6:06:40 N bus: 13 e sono connessi: 0 bus
6:06:41 N bus: 13 e sono connessi: 0 bus
6:06:42 N bus: 13 e sono connessi: 0 bus
6:06:43 N bus: 13 e sono connessi: 0 bus
6:06:44 N bus: 13 e sono connessi: 0 bus
6:06:45 N bus: 13 e sono connessi: 0 bus
6:06:46 N bus: 13 e sono connessi: 0 bus
6:06:47 N bus: 13 e sono connessi: 0 bus
6:06:48 N bus: 13 e sono connessi: 0 bus
6:06:49 N bus: 13 e sono connessi: 0 bus
6:06:50 N bus: 13 e sono connessi: 0 bus
6:06:51 N bus: 13 e sono connessi: 0 bus
6:06:52 N bus: 13 e sono connessi: 0 bus

```

e man mano che il programma rileva le connessioni tra i bus le trascriverà sul file Google Sheet indicando

- Nome del bus principale
- Nome del bus che viene incontrato
- Tempo per il quale rimangono connessi
- Orario in cui avviene l'incrocio

	A	B	C	D
1	Bus Principale	Bus Incontrato	Tempo Connesso	Orario d'incontro
2	I4_d05:03	I4_d05:06	35	5:19:22
3	I4_d05:06	I4_d05:03	35	5:19:22
4	I4_d05:06	I15_d05:29	25	5:32:05
5	I15_d05:29	I4_d05:06	25	5:32:05
6	I15_d05:29	I11_d05:28	48	5:34:26
7	I11_d05:28	I15_d05:29	48	5:34:26
8	I3_d05:17	I8_d05:21	25	5:43:05
9	I3_d05:17	I3_d05:37	2	5:44:40
10	I8_d05:21	I3_d05:17	25	5:43:05
11	I8_d05:21	I3_d05:37	9	5:44:30
12	I3_d05:37	I8_d05:21	9	5:44:30
13	I3_d05:37	I3_d05:17	2	5:44:40
14	I6_d05:31	I6_d05:53	1	5:54:26
15	I6_d05:53	I6_d05:31	1	5:54:26

L'aggiornamento sul foglio di calcolo avviene ogni 1800 secondi (30 minuti) trascorsi all'interno della simulazione.

Tutorial Progetto SUMO - Come Ricreare la Simulazione

Descrizione Codice

Introduzione

Il codice descritto in questa sezione funziona esclusivamente per il file *gtfs_publictransport.rou.xml* presente in questa simulazione, l'id dei veicoli è stato infatti modificato come in seguente esempio: `vehicle id="lNumeroLinea_dOrarioPartenza_rNumeroRoute"` Il file di partenza ottenuto dai file GTFS l'id del veicolo era esclusivamente *NumeroRoute*

File runner.py

Il fine *runner.py* è il file che permette di lanciare la simulazione e consente la comunicazione attraverso TraCI, lo script è composto nel seguente modo: Inizialmente vengono importati i seguenti moduli

```
import os
import sys
import optparse
import datetime
import support
import traci
import gSheetsManager

from numpy import double
from pandas import options

from sumolib import checkBinary
```

In accordo con la [documentazione ufficiale](#), è consigliato introdurre una struttura condizionale che restituisce un errore qualora non fosse impostata correttamente la variabile d'ambiente *SUMO_HOME*.

```
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("please declare environment variable 'SUMO_HOME'")
```

La funzione *get_options()* ritorna l'opzione che permette di lanciare la simulazione con o senza *gui*.

```
def get_options():
    opt_parser = optparse.OptionParser()
    opt_parser.add_option("--nogui", action="store_true",
```

```
        default=False, help="run the commandLine version of sumo")
options, args = opt_parser.parse_args()
return options
```

La funzione *run()* è responsabile della comunicazione tra lo script e SUMO, per fare ciò viene utilizzato TraCI.

```
def run():
    connectedBusNow = 0
    step = 0
    connectedBus = []
    while traci.simulation.getMinExpectedNumber() > 0:
        traci.simulationStep()
```

Il while proseguirà fino a che *getMinExpectedNumber()* ritornerà 0, ciò significherà che tutti gli elementi della simulazione sono stati analizzati e non sarà più necessario proseguire. Ogni ciclo while corrisponde ad 1 secondo all'interno della simulazione. Mentre attraverso la funzione *simulationStep()* traci acquisirà tutti gli elementi del ciclo corrente, come veicoli e tempo. La presenza della funzione permette allo script di acquisire le informazioni della simulazione.

```
print(str(datetime.timedelta(seconds = double(traci.simulation.getTime())))+" N
bus: ", end="")
print(traci.vehicle.getIDCount(), end="")
print(" e sono connessi: " + str(connectedBusNow) + " bus")
connectedBusNow = 0

vehicle = traci.vehicle.getIDList()
```

Nella console dello script per ogni ciclo verranno stampati: l'orario, il numero di bus presenti nella simulazione ed il numero di bus connessi nel ciclo corrente. Inoltre in una lista vengono salvati tutti i nomi dei veicoli presenti nel ciclo corrente.

```
for i in vehicle:
    bus = i.split('_')
    line = bus[0]
    depTime = bus[1]
    currentBus = str(line) + "_" +str(depTime)
```

Il primo *for* scorre la lista dei veicoli e divide l'id, utilizzando come divisore l'underscore, in una lista *bus*.

```
for j in vehicle:
    distance = support.getDistance(traci.vehicle getPosition(i),
traci.vehicle.getPosition(j))
    if i == j:
        continue
```

Successivamente è presente un altro for nidificato al primo, in questo caso viene nuovamente scorsa la lista dell'*id* dei veicoli, viene salvata la distanza tra il veicolo *i* ed il veicolo *j* in una variabile *distance* utilizzando una funzione del file *support*, la quale riceve in ingresso la posizione dei due veicoli attraverso la funzione *getPosition()*. L'*if* presente salterà al ciclo successivo se i due veicoli risultassero uguali.

```
if distance <= 20:
    otherBus = j.split('_')
    lineOB = otherBus[0]
    depTimeOB = otherBus[1]
    connectedBusNow = connectedBusNow + 1
    currentConnectedBus = str(lineOB) + "_" + str(depTimeOB)
```

Il secondo *if* presente verificherà che la distanza tra i due veicoli sia minore uguale a 20 metri, se così fosse, viene suddiviso anche il bus presente in *j* con lo stesso criterio del veicolo *i*.

Struttura della lista connectedBus

Nella lista *connectedBus* verranno salvati tutti i veicoli ed è organizzata nel seguente modo:

```
connctedBus=[[BusPrincipale, {BusConnesso:[TempoDiConnessione,
OrarioInizioConnessione]}]]
```

è una lista contenente un'altra lista la quale è strutturata come:

- Posizione 0: Bus principale, nel ciclo *for* nidificato corrisponde ad *i*;
- Posizione 1: è un dizionario che contiene i dettagli delle connessioni

Il dizionario è a sua volta organizzato come:

- Chiave: corrisponde al Bus secondario, connesso al primo, corrisponde a *j* nel ciclo *for* nidificato;
- Valore: Il valore della chiave è una lista contenente; alla posizione 0 il tempo totale della connessione tra i due autobus, mentre nella posizione 1 è presente l'orario della prima connessione tra i due mezzi. Al dizionario possono aggiungersi più chiavi nel caso l'autobus principale incontrasse più veicoli durante la sua corsa.

Proseguendo con la descrizione del ciclo *for* nidificato:

```
if support.inList(currentBus, connectedBus) == None:
    addBus = [currentBus, {currentConnectedBus:[1,
str(datetime.timedelta(seconds = double(traci.simulation.getTime())))]}]
    connectedBus.append(addBus)
```

Il primo *if* successivo alla verifica della distanza inferiore a 20[m] verifica se all'interno di *connectedBus*, sia presente una lista con l'autobus principale, utilizzando una funzione del file *support*, nel caso non fosse

presente viene creata.

```
elif support.inDictionary(currentConnectedBus,
connectedBus[support.inList(currentBus, connectedBus)]) == False:
    addDictionary = {currentConnectedBus:[1, str(datetime.timedelta(seconds =
double(traci.simulation.getTime())))]}
    connectedBus[support.inList(currentBus, connectedBus)]
[1].update(addDictionary)
```

La seconda struttura condizionale è un *elif*, per cui se esiste in *connectedBus* una lista con l'autobus *i*, viene verificata la presenza del dizionario con l'autobus secondario, nel caso mancasse viene creato il dizionario ed il suo valore, con orario corrispondente al ciclo while in cui si trova la simulazione.

```
elif support.inDictionary(currentConnectedBus,
connectedBus[support.inList(currentBus, connectedBus)]):
    time = connectedBus[support.inList(currentBus, connectedBus)]
[1].get(currentConnectedBus)[0]+1
    meatingTime = connectedBus[support.inList(currentBus, connectedBus)]
[1].get(currentConnectedBus)[1]
    addDictionary = {currentConnectedBus:[time, meatingTime]}
    connectedBus[support.inList(currentBus, connectedBus)]
[1].update(addDictionary)
```

La terza ed ultima struttura condizionale conferma la presenza della lista con l'autobus principale e il dizionario con l'autobus secondario, per cui viene incrementato il tempo di connessione tra i due autobus.

```
if connectedBus != [] and step >= 1800:
    gSheetsManager.updateSheet(connectedBus)
    step = 1
elif step >= 1800:
    step = 1
else:
    step +=1
```

Al di fuori dei due cicli *for* nidificati si trova la struttura condizionale responsabile dell'aggiornamento del foglio di calcolo, che avviene ogni 1800 secondi (cicli while) all'interno della simulazione. L'aggiornamento del foglio di calcolo avviene tramite una funzione dello script *gSheetsManager*.

```
traci.close()
sys.stdout.flush()
```

Al di fuori del ciclo while si trovano i comandi per terminare la connessione TraCI e la simulazione.

```

if __name__ == "__main__":
    options = get_options()
    if options.nogui:
        sumoBinary = checkBinary('sumo')
    else:
        sumoBinary = checkBinary('sumo-gui')

    traci.start([sumoBinary, "-c", "osm.sumocfg"])

run()

```

Al di fuori della funzione `run()` si trova il main, che avvierà o meno la gui a seconda delle opzioni ed assocerà alla simulazione il file `osm.sumocfg` il quale conterrà i file da importare nella simulazione. Dopo il lancio della simulazione verrà chiamata la funzione `run()`.

File support.py

```

import math
from operator import index
from turtle import distance
from itertools import chain
from numpy import mat

def getDistance(mainBusPos, otherBusPos):
    Xcoord = mainBusPos[0] - otherBusPos[0]
    Ycoord = mainBusPos[1] - otherBusPos[1]
    distance = round(math.sqrt(math.pow(Xcoord, 2)+math.pow(Ycoord, 2)),2)
    return distance

def inList(bus, connectedBusList):
    pos = next(((i, l.index(bus)) for i, l in enumerate(connectedBusList) if bus in l), None)
    if pos == None:
        return None
    return pos[0]

def inDictionary(otherBus, connectedBusList):
    if otherBus in connectedBusList[1]:
        return True
    return False

```

All'interno del file di supporto saranno presenti 3 funzioni:

- `getDistance(mainBusPos, otherBusPos)`: la funzione calcolerà la distanza tra i due autobus usando la formula della distanza tra 2 punti;
- `inList(bus, connectedBusList)`: verifica la presenza o meno dell'autobus principale all'interno di una lista;
- `inDictionary(otherBus, connectedBusList)`: controlla se nel dizionario è presente l'elemento passato, ovvero l'autobus secondario.

File gSeetsManager.py

Il file è responsabile della connessione con le API di google sheet e l'aggiornamento del foglio di calcolo collegato.

```
import gspread
import webbrowser

def next_available_row(worksheet):
    str_list = list(filter(None, worksheet.col_values(1)))
    return str(len(str_list)+1)

def updateSheet(busList):
    lastLine = 1
    for element in busList:
        #print(element[0])
        for key, value in element[1].items():
            lastLine = lastLine + 1

    updateRange = "A2:D" + str(lastLine)
    cell_list = wks.range(updateRange)

    i = 0
    for element in busList:
        for key, value in element[1].items():
            cell_list[i].value = element[0]
            i = i + 1
            cell_list[i].value = key
            i = i + 1
            cell_list[i].value = value[0]
            i = i + 1
            cell_list[i].value = value[1]
            i = i + 1
            range = "C" + str(2) + ":D" + str(i)
            wks.format(range, {"horizontalAlignment": "CENTER"})
            wks.update_cells(cell_list)

sa = gspread.service_account("token.json")
print("Apertura Foglio di Calcolo in Corso!")
#print(sa)

sh = sa.open("Sumo Data")

wks = sh.sheet1
webbrowser.open(sh.url)
range = "A2:D" + str(next_available_row(wks))
wks.batch_clear([range])
```

Lo script inizialmente instaura una connessione con il foglio di calcolo e cancellerà le celle riempite da dati di simulazioni precedenti. Nello script sono presenti 2 funzioni:

- *next_aviable_row(worksheet)*: la funzione restituirà un intero con il numero della prima riga vuota;
- *updateSheet(busList)*: la funzione raggrupperà il range di valori da aggiornare e con un'unica chiamata alle API aggiornerà il foglio di calcolo nel range di valori corrispondente a *busList*.

File osm.sumocfg

Attraverso questo file è possibile configurare la simulazione impostando file addizionali in accordo con la [documentazione](#)

```
<input>
    <net-file value="osm.net.xml"/>
    <additional-files
value="pt_vtypes.xml,gtfs_publictransport.add.xml,gtfs_publictransport.rou.xml"/>
</input>
```

I file che abbiamo impiegato in input nella simulazione sono, *osm.net.xml* che contiene la rete stradale della città di trento. Come file addizionali invece: *pt_vtypes.xml* che contiene la tipologia di veicoli impiegati nella simulazione, *gtfs_publictransport.add.xml* contenente la lista di fermate ed i percorsi che gli autobus effettueranno durante la simulazione ed infine *gtfs_publictransport.rou.xml* il quale è responsabile del percorso che gli autobus devono seguire.

Problematiche

Durante il progetto abbiamo riscontrato alcune problematiche.

Mappa con connessioni imprecise o interrotte

Il tool fornito da SUMO osmWebWizard, che consente di selezionare una regione geografica su Open Street Map e convertirla in una rete pronta per essere simulata, presenta delle problematiche per aree molto grandi. Abbiamo tentato gli altri metodi consigliati nella sezione [OSM](#) della documentazione, ma la problematica persisteva. La mappa dunque rappresentata non è totalmente fedele a quella della città di Trento e alcuni percorsi degli autobus potrebbero averne risentito con la conseguenza che il percorso simulato non rappresenterà quello reale in alcune zone.

Simulazione pesante

La simulazione deve coprire una zona molto vasta con numerose intersezioni, infatti nelle ore in cui sono presenti più autobus, dalle 7:00 alle 16:00 circa, il tempo di computazione che richiede ogni ciclo aumenta in maniera significativa. Al fine di non rendere ulteriormente pesante la simulazione abbiamo deciso di non implementare un traffico generato randomicamente.

Modifica del file *gtfs_publictransport.rou.xml*

Il file `gtfs_publictransport.rou.xml` ottenuto dallo script `gtfs2pt.py` utilizzando il GTFS fornito da Trentino Trasporti presentava un id degli autobus poco riconoscibile, pertanto abbiamo modificato manualmente l'id di ogni bus nel seguente modo:

```
vehicle id="lNumeroLinea_dOrarioPartenza_rNumeroRoute"
```

Aggiornamento API

Durante la simulazione l'aggiornamento del foglio di calcolo non avviene ad ogni ciclo, le motivazioni di questa scelta sono due, la prima è il limite di richieste al minuto delle api google, le quali non permettono di stare al passo con l'esecuzione della simulazione. La seconda motivazione riguarda la velocità con cui viene eseguita la richiesta, TraCI infatti prima di proseguire al ciclo successivo attenderà l'avvenuta scrittura nel foglio di calcolo e il tempo di attesa per ogni chiamata è significativo. La chiamata alle API sarà eseguita una sola volta raggruppando tutte le celle da aggiornare.