
Curso - Simulation of Urban MObility (SUMO)

Requisitos previos:

- Computador personal
- Sistema Operativo Linux (Ubuntu 22 -<https://ubuntu.com/download/desktop>). Puede ser nativo o máquina virtual.
- Acceso a Internet
- Conocimiento básicos de programación.
- Conocimiento de comandos básicos de Linux.

Resumen:

Este curso incluye una serie de prácticas destinadas a generar escenarios de simulación de tráfico vehicular. Cada práctica incluye pasos y recomendaciones necesarios para lograr los objetivos establecidos. Las implementaciones se basan en paquetes de código abierto (SUMO) e incluyen: (i) Generación de escenarios de micro movilidad vehicular, (ii) Generación de datos y estadísticas vehiculares, (iii) Interfaces de control de tráfico en línea, (iv) Sistemas inteligentes de transportación.

Autores

Erick Patricio Perez Peralta
erickpatriciopp9@gmail.com
<https://sites.google.com/view/erick-perez-p/>

Pablo Andrés Barbecho Bautista
pablo.barbecho@ucuenca.edu.ec
<https://www.pbarbecho.com/>

Tabla de Contenidos

1	Introducción al Simulador de movilidad Urbana (SUMO)	1
1.1	Objetivos	1
1.2	Introducción	1
1.3	Materiales	2
1.4	Instrucciones	2
1.4.1	Parte I: Instalación de SUMO	2
1.4.2	Parte II: Generación de archivos de simulación de SUMO	3
1.4.3	Parte III: Generación de una simulación usando las consolas CLI y GUI de SUMO.	9
2	Simulación de tráfico vehicular con SUMO WebWizard	11
2.1	Objetivos	11
2.2	Introducción	11
2.3	Materiales	11
2.4	Instrucciones	12
3	Estadísticas de movilidad vehicular en SUMO	15
3.1	Objetivos	15
3.2	Introducción	15
3.3	Materiales	16

3.4	Instrucciones	16
3.4.1	Parte I: Generar archivo de salida (outputs) en formato .csv	16
3.4.2	Parte II: Generar gráficas generales de visualización.	17
3.4.3	Parte III: Generar gráficas usando herramientas de visualización de SUMO.	18
3.4.4	Trayectorias:	18
3.4.5	Velocidad:	19
3.4.6	Densidad de vehículos:	20
3.4.7	Emisiones de gases:	21
4	Introducción a TraCI (Traffic Control Interface)	23
4.1	Objetivos	23
4.2	Introducción	23
4.3	Materiales	25
4.4	Instrucciones	25
4.4.1	Parte I: Requisitos iniciales de TraCI.	25
4.4.2	Parte II: Ejecutar una simulación usando la interfaz TraCI.	26
4.4.3	Parte III: Ejemplos de interacción TraCI - SUMO.	28
4.4.4	Ejemplo 1: Conteo de vehículos en línea.	29
4.4.5	Ejemplo 2: Cálculo de velocidad promedio de vehículos en línea.	29
4.4.6	Ejemplo 3: Cálculo de emisiones CO ₂ de vehículos en línea.	30
4.4.7	Ejemplo 4: Cambio de trayectoria de vehículos en línea.	32
5	Clasificación de movilidad con aprendizaje automático	34
5.1	Objetivos	34
5.2	Introducción	34
5.3	Materiales	35
5.4	Instrucciones	36
5.4.1	Aprendizaje Supervisado	36
5.4.2	Aprendizaje No Supervisado	38
6	Aprendizaje automático aplicado a ITS	40
6.1	Objetivos	40
6.2	Introducción	40
6.3	Materiales	41
6.4	Instrucciones	41
7	Aprendizaje por refuerzo (RL) aplicado a ITS.	44
7.1	Objetivos	44

7.2	Introducción	44
7.3	Materiales	46
7.4	Instrucciones	46
References		59



1. Introducción al Simulador de movilidad Urbana (SUMO)

*En esta práctica el estudiante instalará la plataforma de simulación de movilidad urbana (SUMO) y ejecutará una simulación sencilla. **Modalidad:** Trabajo Individual. **Recursos:** Computador personal con sistema operativo Linux o máquina virtual (VM). Recomendado Ubuntu 22 <https://ubuntu.com/download/desktop>.*

1.1 Objetivos

1. Utilizar los archivos binarios para instalar el simulador sobre el sistema operativo Linux (Guía realizada sobre - Ubuntu 22.04.3 LTS).
2. Conocer los archivos necesarios para correr una simulación sobre SUMO.
3. Lanzar una simulación desde la línea de comandos (CLI) y desde la interfaz gráfica (GUI).

1.2 Introducción

SUMO (Simulación de Movilidad Urbana), es un paquete de simulación de tráfico multimodal (i.e., vehículos livianos, transporte público, peatones, bicicletas, etc) continuo, microscópico (i.e., configuración por nodo). SUMO es una suite de simulación de tráfico gratuita y de código abierto que incluye diferentes herramientas para la generación de tráfico, estadísticas de movilidad y visualización. SUMO incluye modelos de movilidad realista, cambio de línea, emisiones de contaminantes, entre otros. Además, incluye la interface TraCI para control en línea de la simulación.

SUMO es ampliamente utilizado en diferentes temáticas de investigación:

- Evaluación de los ciclos semáforicos.
- Optimización de rutas.
- Reducción de emisiones contaminantes.
- Conducción autónoma.
- Predicción y mejora de tráfico vehicular.
- Evaluación de redes vehiculares (VANETs).
- Manejo cooperativo (Platooning).
- Seguridad vial y análisis de riesgos.
- Planificación de rutas de transporte público.

1.3 Materiales

- **Ordenador** con sistema operativo Linux (Ubuntu 22.04.3 LTS).

1.4 Instrucciones

Esta práctica se compone de 3 partes:

1. **Parte I:** Instalación de SUMO,
2. **Parte II:** Generación de archivos de simulación de SUMO,
3. **Parte III:** Generación de una simulación usando las consolas CLI y GUI de SUMO.

1.4.1 Parte I: Instalación de SUMO

En esta primera parte vamos a instalar la suite de SUMO. Existen varias formas para instalar SUMO, incluso sobre diferentes sistemas operativos <https://sumo.dlr.de/docs/Downloads.php>; sin embargo, en esta sección presentamos un resumen de la instalación manual de la suite de SUMO sobre Ubuntu a partir de los binarios. Una guía completa se puede encontrar en https://sumo.dlr.de/docs/Installing/Linux_Build.html.

1. Actualizar los paquetes del sistema operativo.

```
#sudo apt update
#sudo apt upgrade
```

2. Instalar prerequisitos de SUMO:

```
#sudo apt install git cmake python3 g++ libxerces-c-dev libfox-1.6-dev
libgdal-dev libproj-dev libgl2ps-dev python3-dev swig default-jdk
maven libeigen3-dev

#sudo apt install python3-pandas python3-rtree python3-pyproj
```

Descripción general de prerequisitos:

- git: permite clonar repositorios de Github
- cmake: g++ generadores y compiladores,
- libxerces: manejo de archivos xml
- libfox-1.6-dev: manejo de GUI de SUMO
- libgdal-dev: para manejo de formatos geoespaciales,
- libproj-dev: soporte para la geoconversión y referencias
- libgl2ps-dev: usado por OpenGL,
- ccache: acelerar las construcciones,
- ffmpeg-devel: para manejo de vídeo,
- libOpenSceneGraph-devel: para la GUI en 3D,
- gtest: para pruebas de unidad, no utilice 1.13 o posterior,
- gettext: manejo de idiomas (interno),
- texttest, xvfb y tkdiff: para los ensayos de aceptación,
- copos, un estilo y autopep para la comprobación de estilo,
- swig: conector de programas de bajo a alto nivel,
- python3-dev: entorno para python3,

- jdk: entorno de java,
 - maven: gestor de proyectos de software,
 - libeigen3-dev: para manejo de matrices en C.
3. Una vez instalados los prerrequisitos, es momento de obtener los binarios de SUMO. Para esto, nos dirigimos al repositorio de Github del proyecto <https://github.com/eclipse-sumo/sumo>. Se recomienda descargar SUMO dentro del directorio /opt/ de Linux, propio para instalar aplicaciones de terceros y opcionales.

```
# git clone --recursive https://github.com/eclipse-sumo/sumo
```

Note que este procedimiento descargará la versión más reciente (1.19.0).

4. Luego ha de instalar algunas librerías comunes de python (e.g., pandas). SUMO incluye un archivo de estos requerimientos y es necesario estar ubicados dentro de la carpeta **sumo** descargado en el paso anterior, con el fin de llamar al archivo **requirements.txt**:

```
pip install -r tools/requirements.txt
```

5. Antes de compilar el proyecto ha de definir la variable de entorno permanente **SUMO_HOME**. Este paso es muy importante ya que durante el curso se requiere que el sistema operativo conozca donde se encuentran los binarios del programa. Puede usar el comando de Linux **pwd** para ver el path completo donde se encuentra SUMO.

Es necesario modificar los archivos **/etc/bash.bashrc** y **/home/<user>/.bashrc**. Ha de agregar el path al SUMO al final de los archivos mencionados:

```
export SUMO_HOME="/home/user/sumo-version/"
```

Puede verificar que la variable de entorno se encuentre correctamente configurada con:

```
echo $SUMO_HOME
```

6. Finalmente, ha de crear una carpeta **build** dentro la carpeta raíz SUMO. Se ubica dentro de la carpeta **build** y ejecuta uno a uno los siguientes comando que construyen los binarios de SUMO:

```
cmake -B build .  
cmake --build build -j $(nproc)
```

El comando **nproc** es el número de procesadores de su PC. Este proceso puede tomar varios minutos.

7. Puede verificar la instalación abriendo una terminal de Linux y ejecutando **sumo** que mostrará la versión instalada.

1.4.2 Parte II: Generación de archivos de simulación de SUMO

Una vez instalado SUMO, el primer paso es generar el escenario de simulación. Para construir el escenario, usamos las herramientas pre-instaladas en SUMO. Podemos ver una clasificación éstas herramientas en azul en la Fig. 1.1. Además, la Fig. 1.1 muestra los archivos de entrada y salida de cada herramienta, en verde.

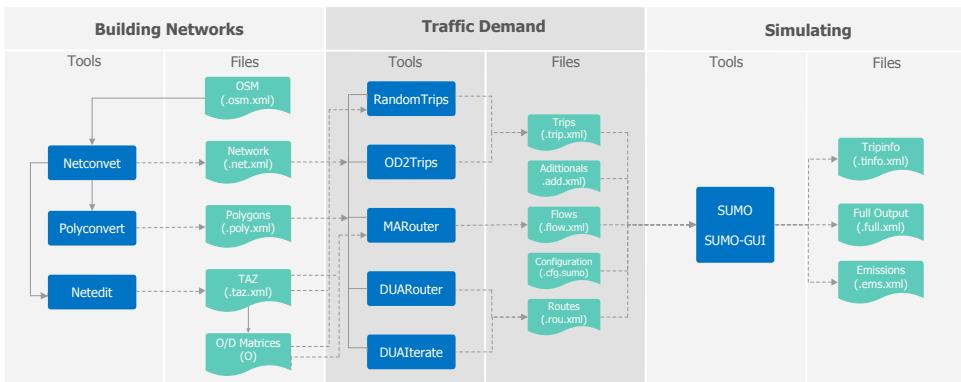


Figure 1.1: Flujo de trabajo de simulación de movilidad de tráfico. Herramientas integradas SUMO y archivos relacionados para realizar simulaciones. Tomado de [1].

Tenemos 3 grupos principales de herramientas, según sus funciones (ver Building Networks, Traffic Demand y, Simulating en la Fig. 1.1):

1. **El primer grupo de herramientas incluye:** red de carreteras, elementos de movilidad (ej., semáforos) y polígonos como edificios, parques etc. Los polígonos son de especial interés cuando evaluamos redes inalámbricas (modelos de obstrucción). En la Fig. 1.1, a la izquierda tenemos las diferentes herramientas de SUMO para generar la red de carreteras (**Netconvert**), los polígonos (**Polyconvert**). Si deseamos modificar gráficamente el mapa, lógica de semáforos, o crear zonas de interés, podemos usar un editor gráfico llamado **Netedit**.
2. **El segundo grupo de herramientas incluye:** generación de la demanda de tráfico. Aquí se definen las rutas de los vehículos, cantidad de vehículos, tipo de vehículos (transporte público, pesados, livianos, bicicletas, peatones). En la Fig. 1.1, en el centro tenemos las herramientas para generación de tráfico.
3. **El tercer grupo de herramientas incluye:** interfaces de simulación. Tenemos dos interfaces: línea de comandos (CLI), gráfica (GUI).

A continuación introducimos algunas de las herramientas que utilizaremos durante la práctica, según la clasificación de la Fig. 1.1:

- **Building Networks:**

- **Netconvert:** Esta herramienta convierte un archivo de mapa de entrada OpenStreetMaps (.osm) en un archivo de red de carreteras legible por SUMO. Este archivo se guarda en ASCII con un formato XML simple conocido como archivo de red (.net.xml). La herramienta netconvert proporciona un conjunto de opciones de procesamiento (por ejemplo, vías cortadas, pasos a desnivel, rotundas, etc.). Este archivo es obligatorio.
- **Polyconvert:** Esta herramienta genera todos los polígonos (por ejemplo, edificios, terrenos, etc.) a partir de la fuente del mapa ingresada (.osm). El archivo de salida (.poly.xml) contiene todas las formas geométricas del mapa. Este archivo

es opcional.

- *Netedit*: Esta herramienta permite a los usuarios editar/crear mapas personalizados. Viene con una interfaz gráfica de usuario donde los usuarios pueden editar las propiedades de los elementos de un mapa, como carreteras, semáforos, etc.

- **Herramientas de generación de movilidad de tráfico:**

- *RandomTrips*: Está destinado a implementaciones rápidas. Aquí, los puntos de origen/destino se seleccionan aleatoriamente en el mapa y los vehículos se distribuyen uniformemente dentro de un período de tiempo.
- *MARouter*: Generar una demanda de tráfico macroscópica. Durante el proceso de generación de tráfico se considera la distribución de la ruta, es decir, cada ruta incluye la probabilidad de ser seleccionada.
- *DUARouter*: Genera una lista de rutas que incluye la ruta completa entre los puntos de origen y destino. Utiliza el algoritmo de Dijkstra para calcular la ruta más corta. Además, el método de asignación de tráfico considera una red vacía cada vez que se genera un vehículo, lo cuál puede decantar en congestión durante la simulación (deadlock).
- *DUAIterate*: Esta herramienta utiliza un método de asignación llamado iterativo, que intenta calcular el equilibrio. Esto significa que las rutas de los vehículos se generan con el mínimo coste (por ejemplo, tiempo de viaje). Esto se hace llamando iterativamente a la herramienta *DUARouter*. Las situaciones de congestión son menos probables.
- *OD2Trips*: Genera una lista de definiciones de viaje según la cantidad de vehículos a insertar en la simulación dentro del intervalo de tiempo codificado en un archivo adicional llamado O/D matrices. Al igual que la herramienta *RandomTrips*, utiliza un método de asignación incremental.

En la Fig. 1.2, se presenta un resumen de los archivos de entrada y salida esperados en cada herramienta de generación de tráfico. En este punto, se recomienda revisar el artículo [2] que amplía los modelos y diferencias entre las diferentes herramientas de generación de tráfico como Duarouter, Randomtrips, MARouter, OD2Trips, etc.

Tool	Inputs	Outputs	Traffic assignment method
OD2Trips (OD2)	Network, O/D Matrices, Additionals	Trips	Incremental
DUARouter (DUAR)	Network, Trips, Additionals	Routes	Empty-network
MARouter (MAR)	Network, O/D Matrices, Additionals	Flows	Incremental
RandomTrips (RT)	Network, Additionals	Trips	Incremental
DUAIterate (DUA)	Network, Trips, Additionals	Routes	Iterative

Figure 1.2: Herramientas de generación de tráfico de SUMO. Tomada de [2]

Finalmente, un escenario de simulación en SUMO se compone por los menos de tres archivos (ver Fig. 1.3). La extensión de los archivo es .xml. Note que el archivo *.poly* es opcional.

Veamos una descripción de los archivos presentados en la Fig. 1.3:

- **Archivo de red (.net)**: Este archivo es generado con la herramienta *Netconvert* y

contiene toda la información acerca del mapa, carreteras y calles.

- **Archivo de polígonos (.poly):** Este archivo es generado con la herramienta *Polyconvert* contiene toda la información acerca de edificios y otros obstáculos como parques que existen en el mapa.
- **Archivo de viajes (.rou):** Este archivo contiene toda la información referente a las rutas de vehículos dentro de la simulación. La manera más sencilla para generar este archivo es usando la herramienta *RandomTrips*.
- **Archivo de Configuración (.cfg):** Los archivos antes mencionados están organizados en un archivo de configuración único donde se indica el tiempo de simulación, los directorios de ubicación de los archivos .net y .rou entre otros.

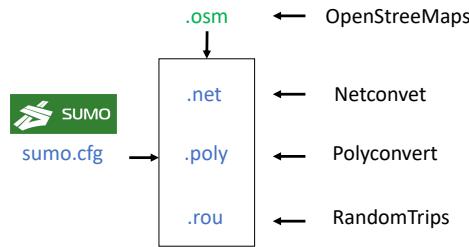


Figure 1.3: Archivos de simulación de SUMO.

A continuación se describen los pasos para generar los archivos necesarios (ver Fig. 1.3) para correr una simulación:

1. Antes de comenzar a generar los archivos, se sugiere crear una carpeta por proyecto, donde se almacenarán todos los archivos de simulación.
2. El primer archivo a generar es el mapa (ver .osm en Fig.1.3). SUMO soporta diferentes formatos, vamos a exportar un mapa descargado de OpenStreetMaps (OSM), <https://www.openstreetmap.org/>. Seleccione una sección pequeña de su ciudad y haga clic en "Exportar" y luego seleccione la opción "Seleccionar manualmente un área diferente", ver Fig. 1.4.



Figure 1.4: Seleccionar área y exportar mapa en formato .osm.

Nota: A través de este método (OSM), solo se permite exportar un número máximo de

50000 nodos, incluyendo carreteras, señales de tráfico, rutas de transporte público, etc.

3. Vamos a generar el siguiente archivo que corresponde a la red de carreteras (.net) (ver Fig. 1.3). Para esto, usamos la herramienta *netconvert*:

```
netconvert --ramps.guess --remove-edges.isolated --edges.join --  
geometry.remove --osm-files map.osm -o bcn.net.xml
```

Donde:

- `--ramps.guess` permite predecir rampas que no se lograron especificar en el mapa.
- `--remove-edges.isolated` elimina los bordes asilados del mapa.
- `--edges.join` fusiona nodos que están cercanos entre sí.
- `--geometry.remove` reemplaza los nodos que solo definen la geometría del borde por puntos de geometría.
- `--osm-files` indica la ruta y el nombre al archivo `.osm` (descargado desde OSM).
- `-o` indica la ruta y el nombre del archivo de salida.

La herramienta tiene varias opciones que se pueden agregar, por ejemplo, excluir rutas de tranvía. Para conocer las opciones nos referimos a la documentación [3]. La entrada de *netconvert* es el archivo `.osm` previamente descargado y la salida es el archivo de la red (.net), por ejemplo `bcn.net.xml`, donde `bcn` es un nombre cualquiera que le damos al mapa y se mantiene las extensiones `.net.xml`.

4. Ahora generamos el archivo de los polígonos (ver Fig. 1.3). Invocamos la herramienta *polyconvert*:

```
polyconvert --net-file bcn.net.xml --osm-files map.osm -o bcn.poly.xml
```

Donde:

- `--net-file` indica la ruta y el nombre del archivo de la red (net).
- `--osm-files` indica la ruta y el nombre al archivo `.osm` (descargado desde OSM).
- `-o` indica la ruta del archivo de polígonos (edificios, parques, etc.) de salida con extensión `.poly.xml`.

Otras opciones de *polyconvert* se encuentran en [4].

5. Ahora generamos las rutas (.rou.xml). Usamos la herramienta *RandomTrips* que hace referencia a un script en Python. ***RandomTrips*** está ubicado en el directorio: `/opt/sumo/tools/`. Corremos el script de Python `randomTrips.py`.

```
python3 randomTrips.py -n bcn.net.xml -r bcn.rou.xml
```

Donde:

- `-r` genera un archivo de rutas con DUAROUTER. Archivo de rutas de salida con extensión `.rou.xml`.
- `-n` indica la ruta y el nombre del archivo de la red (.net).

Una opción de utilidad es `-p`. Esta opción permite generar tráfico en el que consten n vehículos distribuidos uniformemente en un período de tiempo. `-p` se obtiene de:

$$p = \frac{t_1 - t_0}{n} \quad (1.1)$$

Donde, t_1 y t_0 son el tiempo final e inicial de la simulación, respectivamente. Por ejemplo, 3600 segundos, comenzando en el segundo 0. Reemplazando en la ecuación 1.1 se obtiene el valor de $p=36$.

$$p = \frac{3600}{100} = 36 \quad (1.2)$$

Finalmente, el comando sería el siguiente:

```
python3 randomTrips.py -b 0 -e 3600 -p 36 -n bcn.net.xml -r bcn.rou.xml
```

Otras opciones de la herramienta *RandomTrips* están disponibles en [5].

6. Finalmente, necesitamos el archivo de configuración de sumo (ver Fig. 1.3). Este archivo contiene a todos los archivos necesarios para la simulación (.net, .poly y .rou).

El archivo `sumo.cfg` (ver Código 1.1) tiene un formato .xml e incluye directivas del escenario de simulación. La configuración de este archivo se divide en bloques. En el Código 1.1 identificamos el bloque `<input>`, `<processing>`, y `<time>`.

La sección de entrada `<input>` establece la ubicación de los archivos de la red, rutas de vehículos y polígonos. La sección de procesamiento `<processing>` incluye la directiva `<ignore-route-errors>`. Esto permite ignorar los errores que pueden generarse por rutas que no se pueden completar (no hay conexión entre el punto origen y destino). Finalmente, la sección de tiempo establece el tiempo de inicio, finalización y la duración de los pasos de tiempo de la simulación.

```
<?xml version="1.0" encoding="UTF-8"?>
<input>
    <net-file value="bcn.net.xml"/>
    <route-files value="bcn.rou.xml"/>
    <additional-files value="bcn.poly.xml"/>
</input>

<processing>
    <ignore-route-errors value="true"/>
</processing>

<time>
    <begin value="0"/>
    <end value="10000"/>
    <step-length value="1"/>
</time>
</configuration >
```

Listing 1.1: Archivo `sumo.cfg`.

1.4.3 Parte III: Generación de una simulación usando las consolas CLI y GUI de SUMO.

A continuación, se describen los pasos necesarios para correr una simulación usando la CLI y GUI de SUMO.

1. **Simulación usando la CLI:** necesitamos indicar el archivo de configuración con extensión sumo.cfg.

```
#sumo -c bcn.sumo.cfg
```

Donde, -c indica la ubicación y el nombre del archivo de configuración xx.sumo.cfg.

```
sumo@sumo-VirtualBox:~/sumo/bin$ ./sumo -c cuenca.sumo.cfg
Step #0.00 (12ms ~= 83.33*RT, -83.33UPS, vehicles TOT 1 ACT 1 BUF 0)
Warning: No connection between edge '118366336#2' and edge '45420539#0' found.
Warning: No route for vehicle '8' found.
Warning: No connection between edge '48202935' and edge '334948974#6' found.
Warning: No route for vehicle '23' found.
Warning: No connection between edge '389769675#0' and edge '256907298#0' found.
Warning: No route for vehicle '12' found.
Warning: No connection between edge '408449634#1' and edge '334948974#7' found.
Warning: No route for vehicle '95' found.
Step #100.00 (2ms ~= 500.00*RT, -49500.00UPS, vehicles TOT 100 ACT 99 BUF 1)
Warning: No connection between edge '495309213#0' and edge '48859314#3' found.
Warning: No route for vehicle '132' found.
Warning: No connection between edge '408463706#1' and edge '334948974#2' found.
Warning: No route for vehicle '16' found.
Warning: No connection between edge '408463706#1' and edge '311212627#3' found.
Warning: No route for vehicle '178' found.
Warning: No connection between edge '45420545#2' and edge '42201281#1' found.
Warning: No route for vehicle '181' found.
Warning: No connection between edge '48537834' and edge '408713354' found.
Warning: No route for vehicle '199' found.
Step #200.00 (3ms ~= 333.33*RT, -62333.33UPS, vehicles TOT 200 ACT 187 BUF 1)
Warning: No connection between edge '49577750' and edge '-97264414#4' found.
```

Figure 1.5: Simulacion en marcha desde la línea de comandos de Linux.

Como vemos en la Fig. 1.5, se presentan ciertas advertencia refiriéndose a rutas sin conexión. La **CLI** suele utilizarse para ejecutar simulaciones largas y pesadas.

2. **Simulación usando la GUI:** Para ejecutar la simulación con interfaz gráfica (GUI), abrimos una terminal de Linux y ejecutamos sumo-gui.
3. En el menú superior de la GUI de SUMO (ver Fig. 1.6), seleccionamos *File → Open Simulation* y buscamos el archivo de configuración del escenario xx.sumo.cfg.

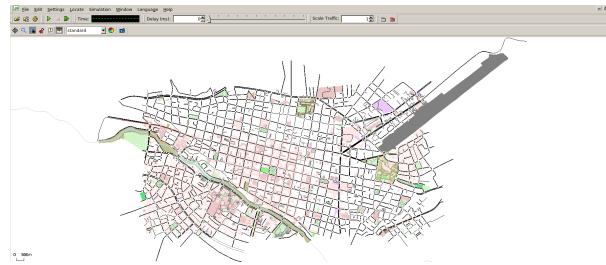


Figure 1.6: Archivo de configuración cargado.

En la parte inferior de la GUI se muestran posibles errores al intentar cargar los archivos .net y .poly definidos en el archivo de configuración .cfg.

4. Finalmente, para ejecutar la simulación consideraremos la velocidad de simulación en la parte superior (recuadro rojo de la Fig. 1.7). Luego, es necesario dar clic en la opción *Run* (recuadro azul de la Fig. 1.7) y la simulación estará en marcha (ver Fig. 1.8).

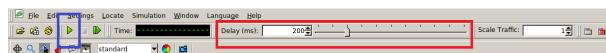


Figure 1.7: Controles de la velocidad de simulación.

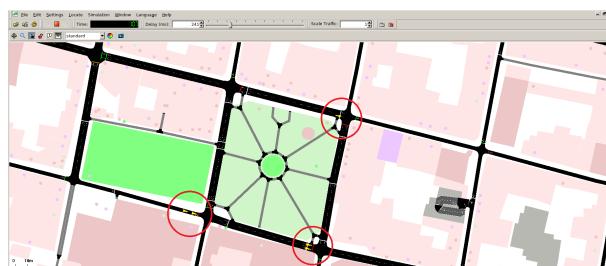


Figure 1.8: Simulación en marcha desde la interfaz gráfica de SUMO.



2. Simulación de tráfico vehicular con SUMO WebWizard

En esta práctica el estudiante generará los archivos de simulación de SUMO utilizando la herramienta OSMWebWizard. Esta herramienta, incluida en la suite de SUMO permite generar simulaciones de manera muy sencilla, automatizando la generación de archivos de simulación. **Modalidad:** Trabajo Individual. **Recursos:** Computador personal con sistema operativo Linux y SUMO.

2.1 Objetivos

- Generar una simulación de SUMO usando la herramienta WebWizard.
- Comprender las ventajas y desventajas de usar OSMWebWizard.

2.2 Introducción

Dentro de la suite de herramientas de SUMO, una herramienta particularmente útil es OSMWebWizard. Este componente permite a los usuarios aprovechar los datos geoespaciales disponibles en OpenStreetMap (OSM) para generar escenarios de simulación realistas y detallados y sin limitación del número de nodos.

OSMWebWizard simplifica significativamente el proceso de generación de archivos de simulación de SUMO. Se basa en una interfaz gráfica Web (se recomienda usar Firefox) que guía a los usuarios a través de los pasos necesarios para importar mapas, definir atributos de la red vial, y configurar parámetros de simulación.

Un inconveniente de la herramienta, es que no permite modificaciones directas en la generación de archivos. Por ejemplo, *netconvert* permite agregar opciones para mejorar la calidad de la red de carreteras (.net) que no se pueden configurar con OSMWebWizard.

En esta práctica, exploraremos cómo utilizar el OSMWebWizard para crear escenarios de simulación en SUMO.

2.3 Materiales

- Ordenador con sistema operativo Linux (Ubuntu 22.04.3 LTS).
- SUMO instalado.

2.4 Instrucciones

A continuación se describen los pasos para ejecutar la herramienta OSMWebWizard de SUMO.

1. La herramienta se localiza dentro del directorio **tools** de SUMO en la ruta: `/opt/sumo/tools`. OSMWebWizard es un script en Python `osmWebWizard.py` y para ejecutarlo escribimos en la terminal:

```
# ./ osmWebWizard . py
```

Se abrirá un navegador web con el mapa de OSM (ver Fig. 2.1). Se puede modificar la ubicación (e.j., Quito). Podemos seleccionar la opción *Seleccionar Área* o simplemente el mapa en pantalla será el mapa que se exporte. De igual manera, podemos marcar la opción *Add Polygons* si necesitamos generar el archivo de polígonos (opcional). En el icono del carro, nos aparece un segundo menú que nos permite configurar el número de vehículos en la simulación, así como otro tipo de vehículos (ver Fig. 2.2).

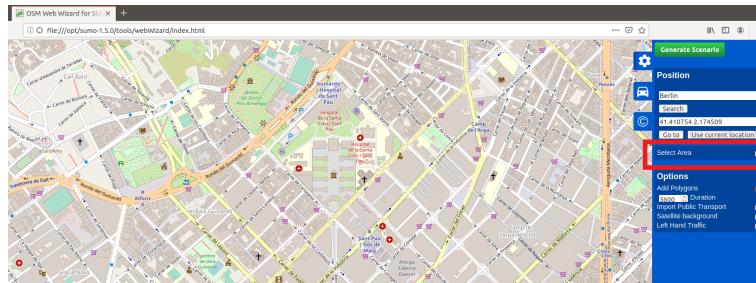


Figure 2.1: osmWebWizard tool.

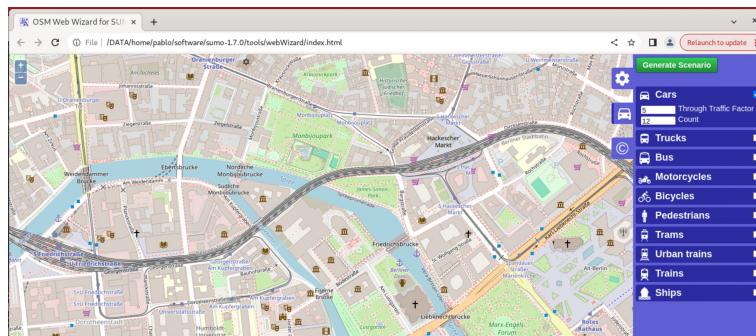


Figure 2.2: osmWebWizard tool.

2. Una vez presionemos en *Generate Scenario* (botón verde en Fig. 2.1), podemos mirar en consola los comandos que se ejecutan automáticamente. Finalizado el proceso, se nos presenta un resumen en consola y se agrega una carpeta dentro del directorio `/opt/sumo/tools/` cuyo nombre corresponde a la fecha en la que se generaron los

archivos. Además, se ejecutará automáticamente la simulación en la GUI de SUMO.

En este punto es importante identificar los archivos generados dentro de la carpeta. En la Fig. 2.3 se listan los archivos generados de manera automática.

```
-rw-rw-r-- 1 pablo pablo    354 Mar  5 03:35 build.bat
-rw-rw-r-- 1 pablo pablo 7354804 Mar  5 03:34 osm.net.xml
-rw-rw-r-- 1 pablo pablo   1212 Mar  5 03:34 osm.netccfg
-rw-rw-r-- 1 pablo pablo  59092 Mar  5 03:35 osm.passenger.trips.xml
-rw-rw-r-- 1 pablo pablo 1416755 Mar  5 03:34 osm.poly.xml
-rw-rw-r-- 1 pablo pablo    691 Mar  5 03:34 osm.polycfg
-rw-rw-r-- 1 pablo pablo   851 Mar  5 03:35 osm.sumocfg
-rw-rw-r-- 1 pablo pablo     88 Mar  5 03:35 osm.view.xml
-rw-rw-r-- 1 pablo pablo 13234119 Mar  5 03:34 osm_bbox.osm.xml
-rwxr-xr-x 1 pablo pablo     23 Mar  5 03:35 run.bat
```

Figure 2.3: Archivos SUMO generados.

A continuación se presenta una descripción de cada archivo:

- `build.bat` es un bash script que contiene el comando para correr la herramienta `randomTrips.py` indicando cada una de las opciones agregadas.
- `osm.netccfg` es el archivo de configuración usado para el netconvert. Construye el archivo final `.net`.
- `osm.net.xml` es el archivo de carreteras final `.net`.
- `osm.poly.xml` es el archivo de configuración final de polígonos `.poly`.
- `osm_bbox.osm.xml` es el archivo del mapa en formato `.osm`.
- `osm.passenger.trips.xml` contiene las rutas de los vehículos (pasajeros).
- `osm.polycfg` contiene la información del archivo usado para generar los polígonos.
- `osm.sumocfg` es el archivo de configuración de la simulación de sumo.
- `osm.view.xml` contiene la configuración de la ejecución en el GUI de SUMO.
- `run.bat` es un script que contiene el comando para correr la simulación de sumo desde la interfaz gráfica.

Los archivos marcados en azul son los que nos interesa para reproducir la simulación. Podemos copiarlos a otro directorio y modificarlos o usarlos para ejecutar una nueva simulación.

Recordando de la práctica anterior, en el archivo `sumo.cfg`, teníamos como entrada `<input>` los archivos: `.net`, `.rou` y `.poly`. En este caso tenemos los archivos `.net`, `trips` y `.poly`. Se observa que ahora se utiliza el archivo `.trips` es cual reemplaza al archivo `.rou`, más adelante explicaremos la diferencia entre `.trips` y `.rou`.

3. Genere nuevamente la simulación usando los archivos generados con `osmWebWizard`.
-

NOTA: Si al correr el script `osmWebWizard.py` salta el siguiente error:

```
osmWebWizard.py: error: typemap file "\${SUMO\_HOME}/data/typemap/
osmPolyconvert.typ.xml" not found
```

Es necesario verificar que la variable de entorno se encuentre configurada correctamente `echo $SUMO_HOME`, y verificar la existencia del archivo `osmPolyconvert.typ.xml` en el path especificado en el error.



3. Estadísticas de movilidad vehicular en SUMO

En esta práctica el estudiante explorará como extraer estadísticas de las simulaciones en SUMO

Modalidad: Trabajo Individual. **Recursos:** Computador personal con sistema operativo Linux y la SUMO pre-instalado.

3.1 Objetivos

- Generar archivos de salida (outputs) de la simulación en SUMO.
- Entender el contenido de los archivos de salida de las simulaciones.
- Analizar los datos de las simulaciones con scripts en Python.

3.2 Introducción

SUMO permite generar una gran cantidad de estadísticas, basada en los datos de vehículos, simulación, carreteras, intersecciones, semáforos, entre otros. Los archivos de salida escritos por SUMO están en formato XML. SUMO incluye la herramienta `xml2csv.py` que permite convertir los outputs a un formato de archivo plano (csv) que se puede procesar con la mayoría de programas para análisis de datos (R, Python, Excel, etc).

En esta práctica nos enfocamos en 3 ejemplos (outputs) que capturan datos de la red vehicular:

1. ***tripinfo***: Esta salida contiene la información sobre el tiempo de viaje del vehículo, distancia recorrida, numero de cambios de rutas entre otros datos. Nota: La información se genera para cada vehículo tan pronto como el vehículo llega a su destino y se retira de la red. En el caso que el vehículo no llegue a su destino durante el tiempo de simulación, este vehículo no se considera en la estadística <https://sumo.dlr.de/docs/Simulation/Output/TripInfo.html>.
2. ***fcd***: La exportación FCD (datos de automóviles flotantes) contiene la ubicación, la velocidad, y tipo de vehículo para cada vehículo en la red. Esta información se captura para paso de la simulación (cada 1s por defecto). Esta captura se comporta como un dispositivo GPS de alta frecuencia súper preciso para cada vehículo. Los resultados se pueden procesar aún más utilizando la herramienta TraceExporter para adaptar la frecuencia, las velocidades del equipo, la precisión y el formato de los datos <https://sumo.dlr.de/docs/Simulation/Output/FCDOutput.html>.

3. **trajectories**: Los datos de trayectoria incluyen el nombre, la posición, la velocidad y la aceleración de cada vehículo <https://sumo.dlr.de/docs/Simulation/Output/AmitranOutput.html>.

3.3 Materiales

- **Ordenador** con sistema operativo Linux (Ubuntu 22.04.3 LTS).
- **SUMO** instalado.

3.4 Instrucciones

Dividimos la práctica en tres partes:

1. **Parte I**: Generar archivo de salida (outputs) en formato .csv
2. **Parte II**: Generar gráficas generales de visualización.
3. **Parte III**: Generar gráficas usando herramientas de visualización de SUMO.

3.4.1 Parte I: Generar archivo de salida (outputs) en formato .csv

A continuación se describen los pasos para configurar los archivos de salida en una simulación.

1. El primer paso es generar los archivos necesarios para la simulación. Esto se logra de manera manual con las herramientas de SUMO (Práctica 3), o utilizando la herramienta *osmWebWizard* (Práctica 4).
2. Luego, nos enfocamos en modificar el archivo de configuración del escenario *sumo.cfg* o *.sumocfg*. En este archivo XML, es necesario agregar un nuevo bloque con la etiqueta *<outputs>*. En este bloque agregamos las salidas que se requieren capturar, como sigue:

```
<output>
    <tripinfo-output value="tripinfo.xml"/>
    <fcd-output value="fcd.xml">
        <amitran-output value="trajectories.xml">
    </output>
```

Dentro del bloque *<outputs>* se indica tres salidas: *tripinfo*, *fcd* y *amitran* outputs. Además, se especifica el nombre del archivo de salida y la ruta donde se guardara el archivo de salida. Cabe anotar que la documentación no presenta la sintaxis de los tópicos de salida. Para conocer la sintaxis, podemos verificar las posibles opciones dentro del tópico *<outputTopicType>* localizado en el archivo */opt/sumo/data/xsd/sumoConfiguration.xsd*.

3. Una vez agregados los outputs, podemos ejecutar la simulación mediante la CLI o GUI de SUMO. Por ejemplo mediante la CLI, abrimos una terminal y ejecutamos:

```
sumo -c osm.sumocfg
```

4. Una vez finalizada la simulación, de acuerdo a las rutas especificadas para los archivos de salida, podemos encontrarlos en dicha ubicación. Por ejemplo, la Fig. 3.1 se muestra el archivo *tripinfo.xml* generado una vez finaliza la simulación. Dado que este

archivo mantiene un formato xml, en el siguiente paso se convierte a .csv para facilitar el manejo de los datos.

5. Para convertir archivos xml en archivos csv, localizamos la herramienta `xml2csv.py`. Esta se ubica en la ruta `/opt/sumo/tools/xml/`. Para ejecutar el script `xml2csv.py`, hemos de indicar el archivo de entrada con la opción `-s`, por ejemplo el archivo `tripinfo.csv`, y la opción `-o` nos permite indicar el archivo de salida (`xxxx.csv`).

```
python3 /opt/sumo/tools/xml/xm1csv.py -s tripinfo.xml, -o tripinfo.csv
```

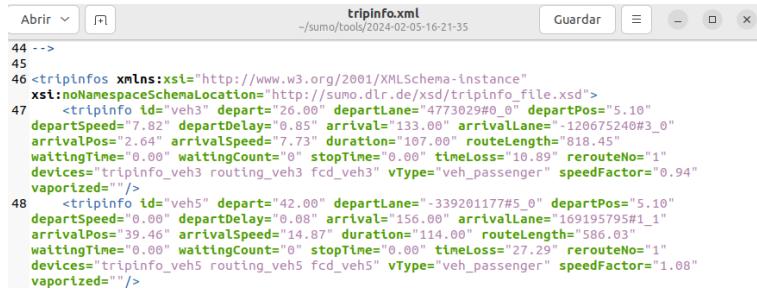


Figure 3.1: Archivos `tripinfo.xml`.

3.4.2 Parte II: Generar gráficas generales de visualización.

En este punto, con el archivo .csv se pueden hacer un análisis de los datos o generar gráficas de las estadísticas que se requiera.

Por último, presentamos un script de ejemplo en Python que permite graficar los datos presentes en el archivo tripinfo.csv. Mediante línea de comandos el usuario deba ingresar el path al archivo .csv más las columnas que desea graficar. El nombre de la columna seleccionada debe coincidir con el nombre de la columna en el archivo .csv.

El código ejemplo está disponible de manera pública en el Github <https://github.com/erickperez9/ManualSUMO.git> y fue realizado por el estudiante. Erick Perez. El código se encuentra dentro en ManualSUMO/Scripts plots /plots-v1-4.py.

Por ejemplo, para graficar la columna tripinfo_routeLength, que representan la distancia recorrida, ejecutamos:

```
python3 plots-v1-4.py -i tripinfo.csv --columns tripinfo_routeLength
```

O si deseamos incluir más columnas en la gráfica podemos agregar `emissions_CO2_abs` y `tripinfo_duration`, siendo las emisiones de CO2 y la duración de cada viaje respectivamente.

```
python3 plots-v1-4.py -i tripinfo.csv --columns tripinfo_routeLength ,  
emissions CO2 abs,tripinfo duration
```

La Fig. 3.2 presenta un ejemplo del plot generado por el script plots-v1-4.py. Las Fig. 3.2 ilustra los resultados del último comando.

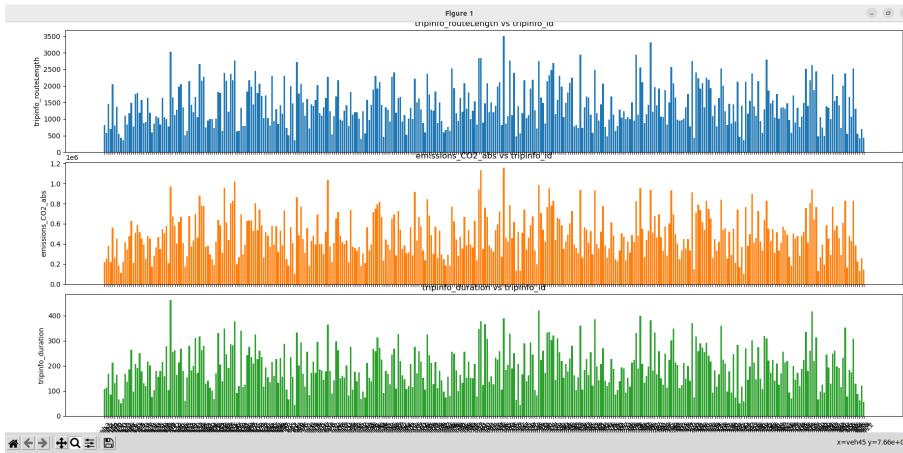


Figure 3.2: Gráficas de: tripinfo_routeLength, emissions_CO2_abs y tripinfo_duration vs tripinfo_id.

3.4.3 Parte III: Generar gráficas usando herramientas de visualización de SUMO.

1. Adicional a los ejemplos de gráficas personalizadas de la Parte II, SUMO incluye diferentes herramientas de visualización de los datos <https://sumo.dlr.de/docs/Tools/Visualization.html>. En esta parte final de la práctica usamos las herramientas de visualización para generar plots para una inspección inicial de la simulación:

- Trayectorias de vehículos.
- Velocidad de vehículos.
- Densidad de vehículos.
- Emisiones de vehículos.

3.4.4 Trayectorias:

Para graficar las trayectorias de los vehículos de la simulación en SUMO, se puede utilizar dos scripts. El primero es `plotXMLAttributes.py` que está ubicado en la ruta `tools/visualization/`. Para correr este script se requiere como entrada el archivo de salida `fcd_output` generado en la Parte I de la práctica. Ejecutamos:

```
python3 tools/visualization/plotXMLAttributes.py -x x -y y -s 1 -o
allXY_output.png fcd.xml --scatterplot
```

Donde `-x` es el atributo del eje x; `-y` es el atributo del eje y; `-s` es el tamaño; `-o` es el nombre del archivo de salida; `--scatterplot` genera un diagrama de dispersión en lugar de un diagrama de líneas.

La segunda forma para graficar trayectorias es utilizar el script `plot_trajectories.py` que está ubicado en la ruta `tools/`. Para correr este script se requiere como entrada el archivo `fcd_output`. Ejecutamos:

3.4 Instrucciones

```
python3 tools/plot_trajectories.py -t xy -o allLocations_output.png  
fcd.xml
```

Donde `-t` es el tipo de trayectoria antes mencionado; `-o` es el nombre del archivo de salida. Además, es posible seleccionar y graficar la trayectoria de uno o más vehículos. Para esto se requiere de aplicar un filtro al script `plot_trajectories.py`, con el parámetro `-filter-ids` el cual indica el ID del vehículo seleccionado. Si se requiere filtrar un vehículo adicional, basta con colocar el ID del otro vehículo seguido de una coma.

La Fig. 3.3 muestra un ejemplo de la gráfica de trayectorias de vehículos. Cada trayectoria se muestra en un color diferente. Vemos que las trayectorias describen la red de carreteras urbanas.

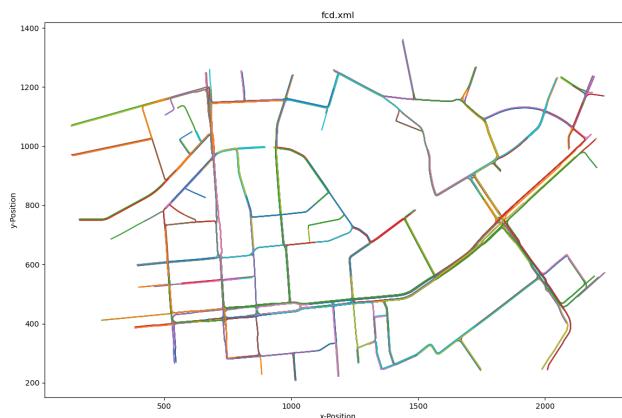


Figure 3.3: Trayectoria de vehículos en la simulación. Gráfica generada con herramientas de SUMO.

3.4.5 Velocidad:

La gráfica de velocidades de vehículos nos permite comparar la velocidad de los vehículos durante la simulación. Usamos el script `plot_trajectories.py` ubicado en `/tools`. El script requiere como entrada el archivo `fcd.xml`. A continuación se muestra un ejemplo de uso:

```
python tools/plot_trajectories.py -t ts -o timeSpeed_output.png fcd.  
xml --filter-ids veh1, veh40
```

En la Figura 3.4 se muestra las velocidades de los vehículo con IDs *1* y *40*. Se observa que el vehículo *1* comienza su viaje en el segundo 0 y termina en el segundo 220, con sus variaciones respectivas de velocidad en *m/s*. Mientras que el vehículo *40*, comienza su recorrido en el segundo 320 y termina en el segundo 500.

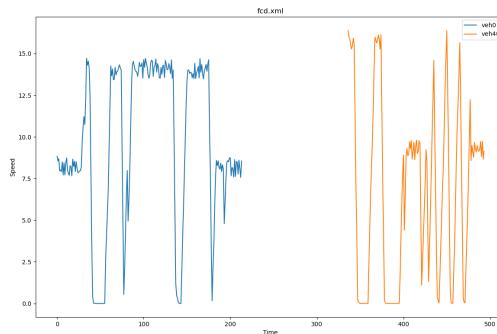


Figure 3.4: Comparación de velocidades de los vehículos *veh1* y *veh40*.

3.4.6 Densidad de vehículos:

Otra de las herramientas que incluye SUMO es `plot_net_dump.py` ubicado en `tools/visualization/`. Este script muestra una red, donde los colores y el ancho de los bordes de la red se establecen en función de los datos en las carreteras.

Las medidas de densidad se localizan en el archivo `edge.data.xml` (https://sumo.dlr.de/docs/Simulation/Output/Lane-_or_Edge-based_Traffic_Measures.html). Este archivo contiene datos de cada carretera como la ocupación o densidad (veh/km) en cada instante de la simulación.

Note que no hemos generado este output anteriormente. Bastaría con agregar el tópico respectivo (`edgedata-output`) en los outputs del archivo `sumo.cfg`

```
<output>
    <emission-output value="co2.xml"/>
    <tripinfo-output value="tripinfo.xml"/>
    <fcd-output value="fcd.xml"/>
    <edgedata-output value="edgedata.xml"/>
</output>
```

El archivo resultante `edgedata.xml` almacena entre otros datos la densidad de vehículos en cada carretera *veh/km*. A continuación se muestra un ejemplo de uso de esta herramienta para graficar la densidad de los vehículos en cada carretera:

```
python3 /home/sumo/sumo/tools/visualization/plot_net_dump.py -v -n osm
.net.xml --measures density,density --xlabel [m] --ylabel [m] --
default-width 1 -i edgedata.xml,edgedata.xml --xlim 000,2600 --
ylim 000,1500 --default-width .5 --min-color-value 0 --max-color-
value 5 --max-width-value 5 --min-width-value 0 --max-width 3 --
min-width .5 --colormap winter -o density-color-map.png
```

Donde:

- `-v` (verbose) imprime el progreso en pantalla.
- `-n` indica el archivo de entrada de red, es decir, el archivo `.net`.
- `-xlabel [m]` indica la etiqueta del eje x.
- `-ylabel [m]` indica la etiqueta del eje y.

- `-default-width` indica el valor del ancho de las líneas de las carreteras.
- `-i` indica los archivos de entrada donde se encuentra la medida a graficar.
- `-xlim` indica el límite del mapa en metros del eje x.
- `-ylim` indica el límite del mapa en metros del eje y.
- `-min-color-value` define el valor mínimo de color del borde
- `-max-color-value` define el valor máximo de color del borde
- `-min-width-value` define el valor del ancho mínimo del borde.
- `-max-width` define el ancho máximo del borde.
- `-min-width` define el ancho mínimo del borde.
- `-colormap` indica el color de la barra de valores.
- `-o` indica el archivo de salida.

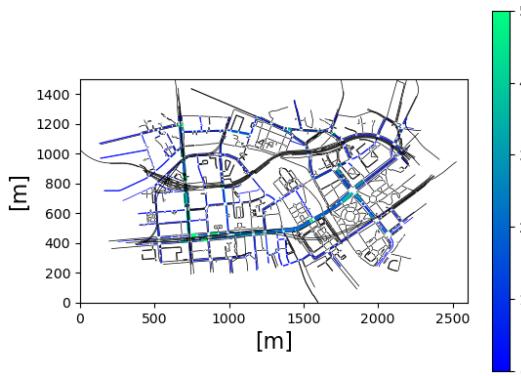


Figure 3.5: Mapa de color de densidad de flujo de vehículos.

La Fig. 3.5, muestra la densidad por carreta. Por medio de la barra de calor a la derecha, vemos sobre el mapa las carreteras con mayor (verde) y menor (azul) cantidad de vehículos durante la simulación. En este ejemplo graficamos la densidad en las carreteras, por tal motivo se indica el valor `-measures` como `density,density;` sin embargo podemos usar otras capturas como la ocupación (https://sumo.dlr.de/docs/Simulation/Output/Lane-_or_Edge-based_Traffic_Measures.html).

3.4.7 Emisiones de gases:

Para lograr esto, se requiere generar los datos de emisiones de los vehículos en la simulación. A diferencia de los ejemplos anteriores donde se agregaba el tópico en el archivo de configuración del escenario sumo.cfg, aquí se requiere (i) agregar la configuración en un archivo adicional (additional-file) y (ii) importar el archivo adicional dentro del bloque inputs del sumocfg.

(i) El nuevo archivo es el edgeData https://sumo.dlr.de/docs/Simulation/Output/Lane-_or_Edge-based_Emissions_Measures.html. Donde, se debe especificar el id, el periodo (tiempo total de simulación) de toma de datos, el tipo de datos a recibir (emisiones) y el nombre del archivo de salida. A continuación se muestra el código de este nuevo archivo.

```

<additional xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/additional_file.
    xsd">

    <edgeData id="1" period="3600" type="emissions" file="edgeData
        .xml" excludeEmpty="true"/>

</additional>

```

(ii) Para importar el nuevo archivo creado al archivo de configuración de sumo se requiere indicar la ruta de ubicación de este archivo en el apartado de archivos adicionales. Por ejemplo:

```

<input>
    <net-file value="osm.net.xml.gz"/>
    <route-files value="osm.passenger.trips.xml"/>
    <additional-files value="osm.poly.xml.gz,edgeData.xml"/>
</input>

```

La Fig. 3.6, muestra las emisiones absoultas (total) de CO₂ en cada calle del mapa. Usamos la herramienta `plot_net_dump.py` para graficar el mapa de color en base a las emisiones CO₂ medidas en las calles del mapa. Note que, adicional a los nombres del archivo de entrada y salida, el valor a graficar también cambia (`-measures CO2_abs,CO2_abs`), así como los límites respectivos.

```

python3 /home/sumo/sumo/tools/visualization/plot_net_dump.py -v -n osm
    .net.xml --measures CO2_abs,CO2_abs --xlabel [m] --ylabel [m] --
    default-width 1 -i edgeData-output-v2.xml,edgeData-output-v2.xml
    --xlim 000,2600 --ylim 000,1500 --default-width .2 --min-color-
    value 0 --max-color-value 2000000 --max-width-value 2000000 --min-
    width-value 0 --max-width 1 --min-width 1 --colormap turbo -o
    density-color-map.png

```

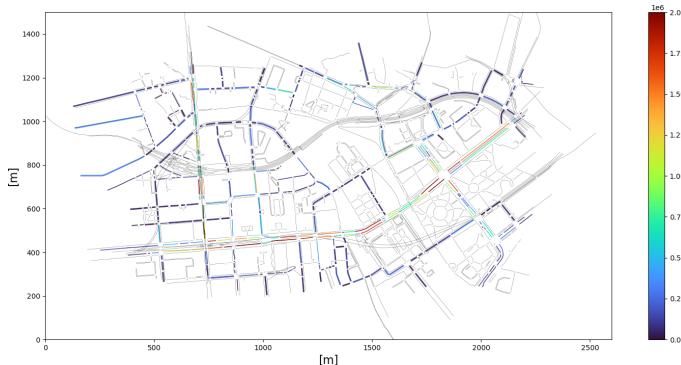


Figure 3.6: Mapa de color de CO₂ absoluto de vehículos.