SUMO Various topics

Bibek Poudel

Nov, 5, 2024

SUMO Various topics

Bibek Poudel

GitHub Repository

github.com/poudel-bibek/SUMO-class

Nov, 5, 2024

Contents

- Libsumo
- Route Generation Tools
    - DuaRouter
    - JTRRouter
- Traffic Assignment Zones (TAZ)
- Pedestrian Simulation Demo

Libsumo

- A C++ library that provides TraCI API functionality
- Alternative to TraCI for simulation control
- Optimized for performance
- Supported Languages:
    - C++: Native support
    - Python, Java, Matlab

Libsumo vs Traci

Traci	**Libsumo**
Client-server architecture	Direct function calls
Moderate Performance	More efficient coupling with SUMO (no communication overhead) ➡️ Better performance
GUI	No GUI
Multiple simulation instances (parallelization via multiprocessing)	Single simulation instance (parallelization via multiprocessing)

Libsumo Demo

- Installation

```
pip install libsumo
```

Libsumo Demo

- Installation

```
pip install libsumo
```

- Run

```
python Libsumo/traffic_control.py
```

Libsumo Demo

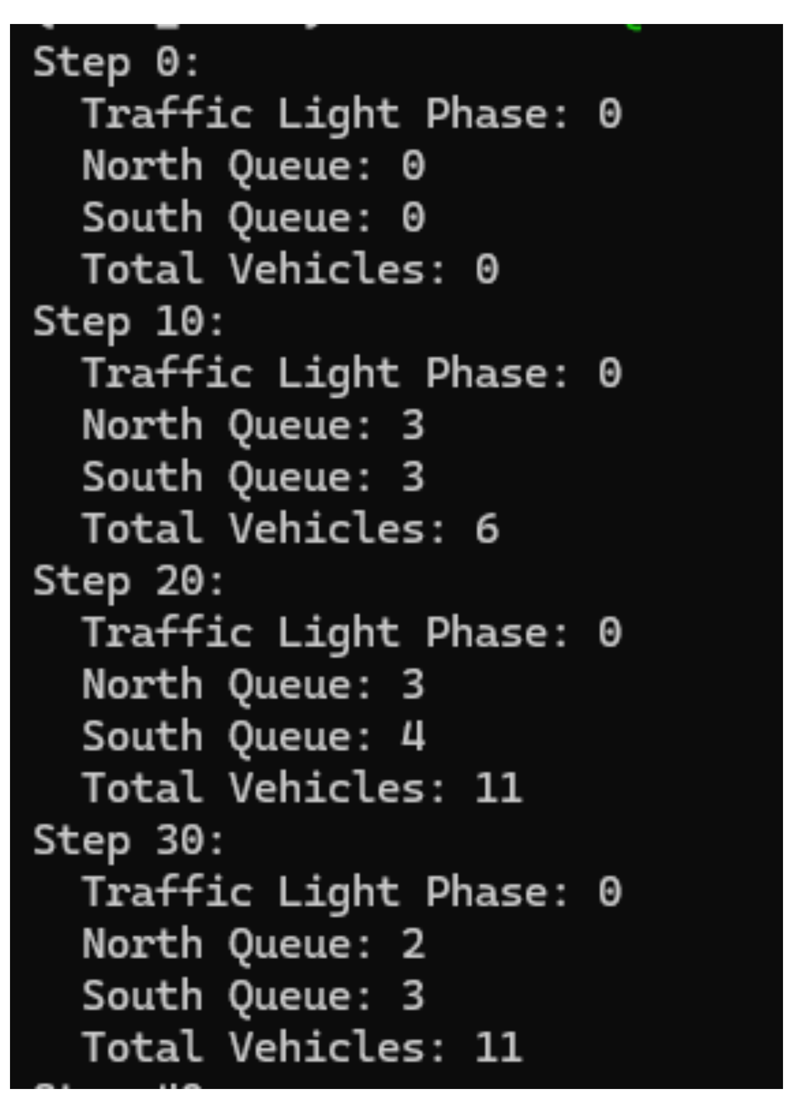
- Installation

```
pip install libsumo
```

- Run

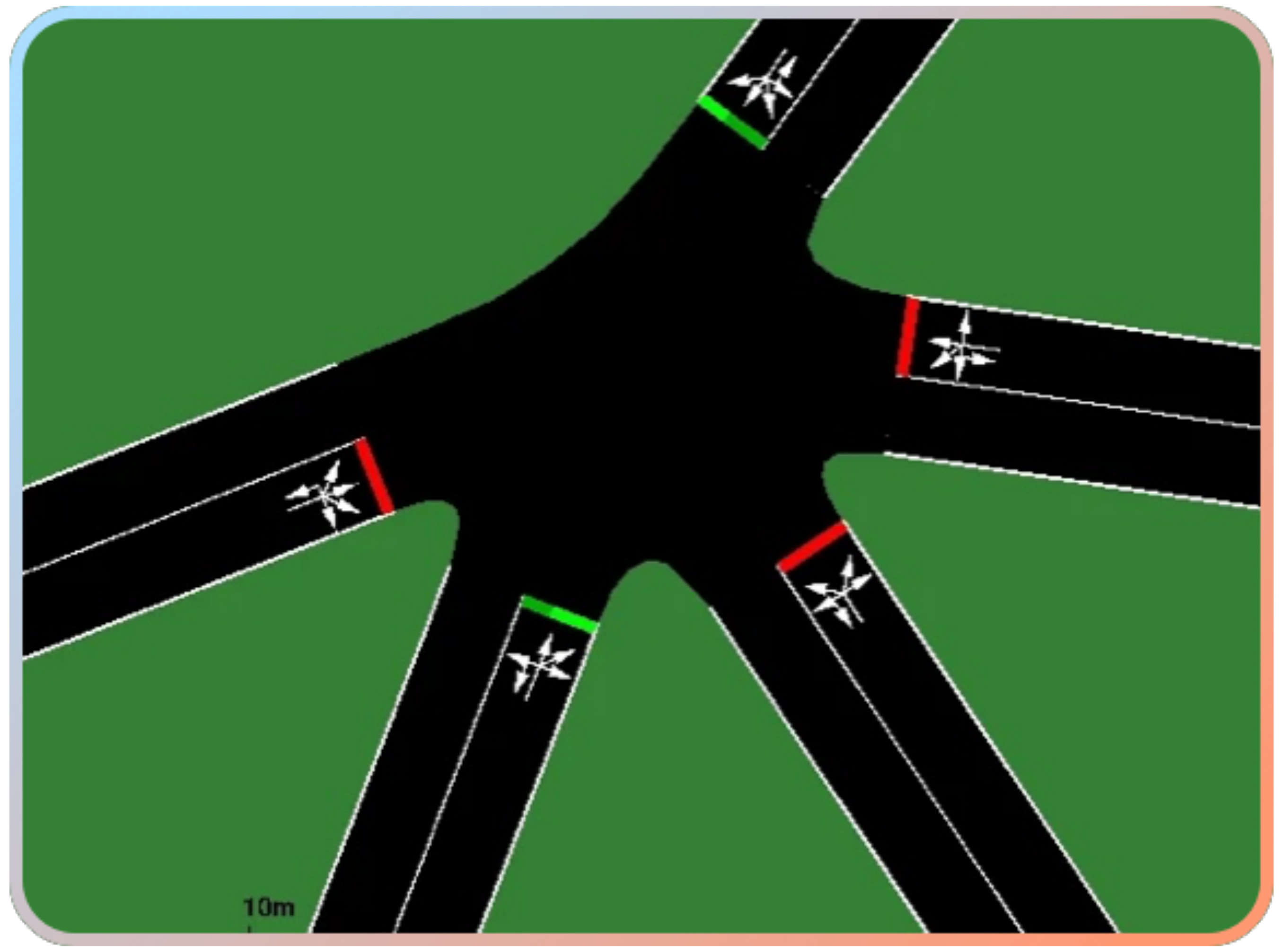
```
python Libsumo/traffic_control.py
```

- Results



```
Step 0:
  Traffic Light Phase: 0
  North Queue: 0
  South Queue: 0
  Total Vehicles: 0
Step 10:
  Traffic Light Phase: 0
  North Queue: 3
  South Queue: 3
  Total Vehicles: 6
Step 20:
  Traffic Light Phase: 0
  North Queue: 3
  South Queue: 4
  Total Vehicles: 11
Step 30:
  Traffic Light Phase: 0
  North Queue: 2
  South Queue: 3
  Total Vehicles: 11
```

Route Generation Tools



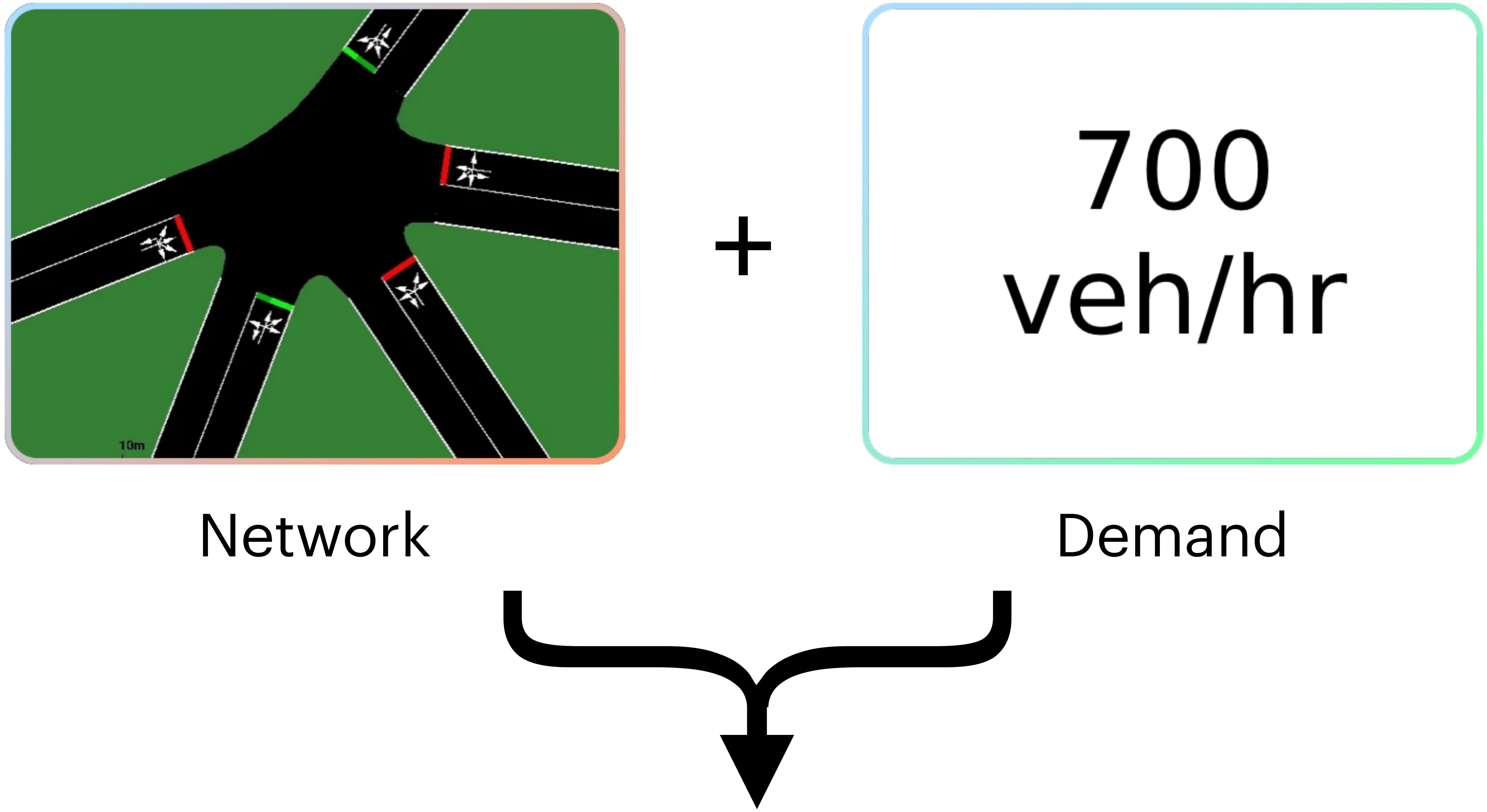
Network

+

700
veh/hr

Demand

Route Generation Tools



+

700
veh/hr

Network

Demand

Route

Route Generation Tools

Route	Trip	Flow
A plan for the journey	Actual journey of a vehicle, a route is taken during a trip	A stream of vehicles that follow a route
A route defines a sequence of edges	Info on a trip could include 1. What route to take 2. What is arrival time 3. Intermediate stops	E.g., rate at which vehicles are injected into the sim in a route

Route Generation Tools

- Route

```
<route id="route0" edges="edge_start middle1 middle2 edge_end"/>
```

- Trip

```
<trip id="0" depart="0.00" from="edge_start" to="edge_end"/>
```

- Flow

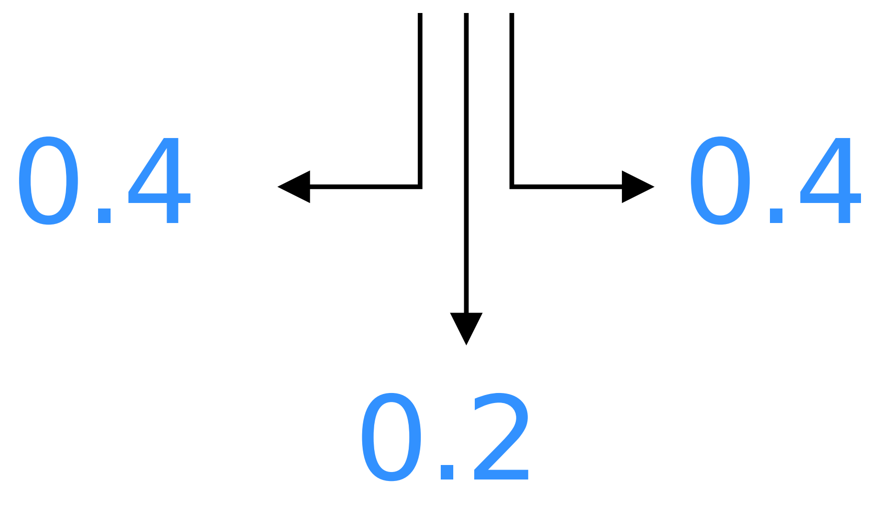
```
<flow id="f0" begin="0" end="3600" from="edge_start" to="edge_end" vehsPerHour="300"/>
```

Route Generation Tools

- DuaRouter
    - Dynamically assigns routes during the simulation, allowing for real-time adjustments
    - Uses Dijkstra's shortest path algorithm for route computation
    - Considers edge weights (like travel time or distance)
- JTRouter

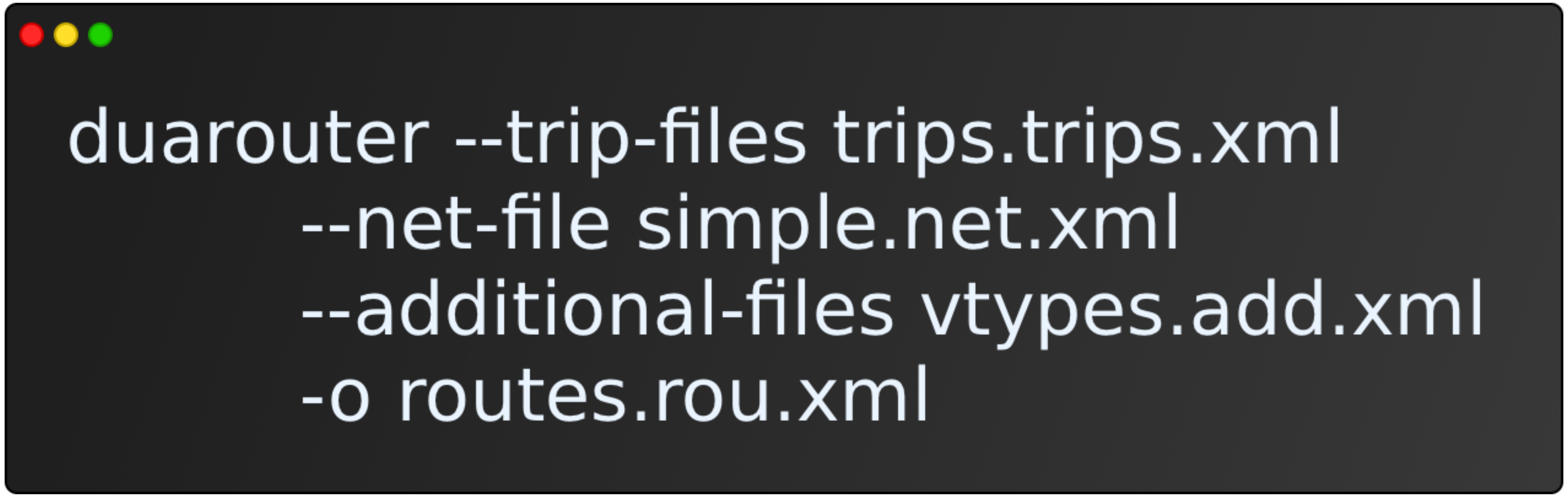
Route Generation Tools

- DuaRouter
  - Dynamically assigns routes during the simulation, allowing for real-time adjustments
  - Uses Dijkstra's shortest path algorithm for route computation
  - Considers edge weights (like travel time or distance)
- JTRouter
  - Precomputes routes based on static input data (flows and turning ratios) before the simulation
  - Based on junction turning ratios or traffic volumes (input)
  - Ideal for well-defined traffic demand such as replicating specific traffic studies



Route Generation Tools Example

- DuaRouter call
  - Trips file specifies the demand (who and when)
  - Additional files with vehicle types



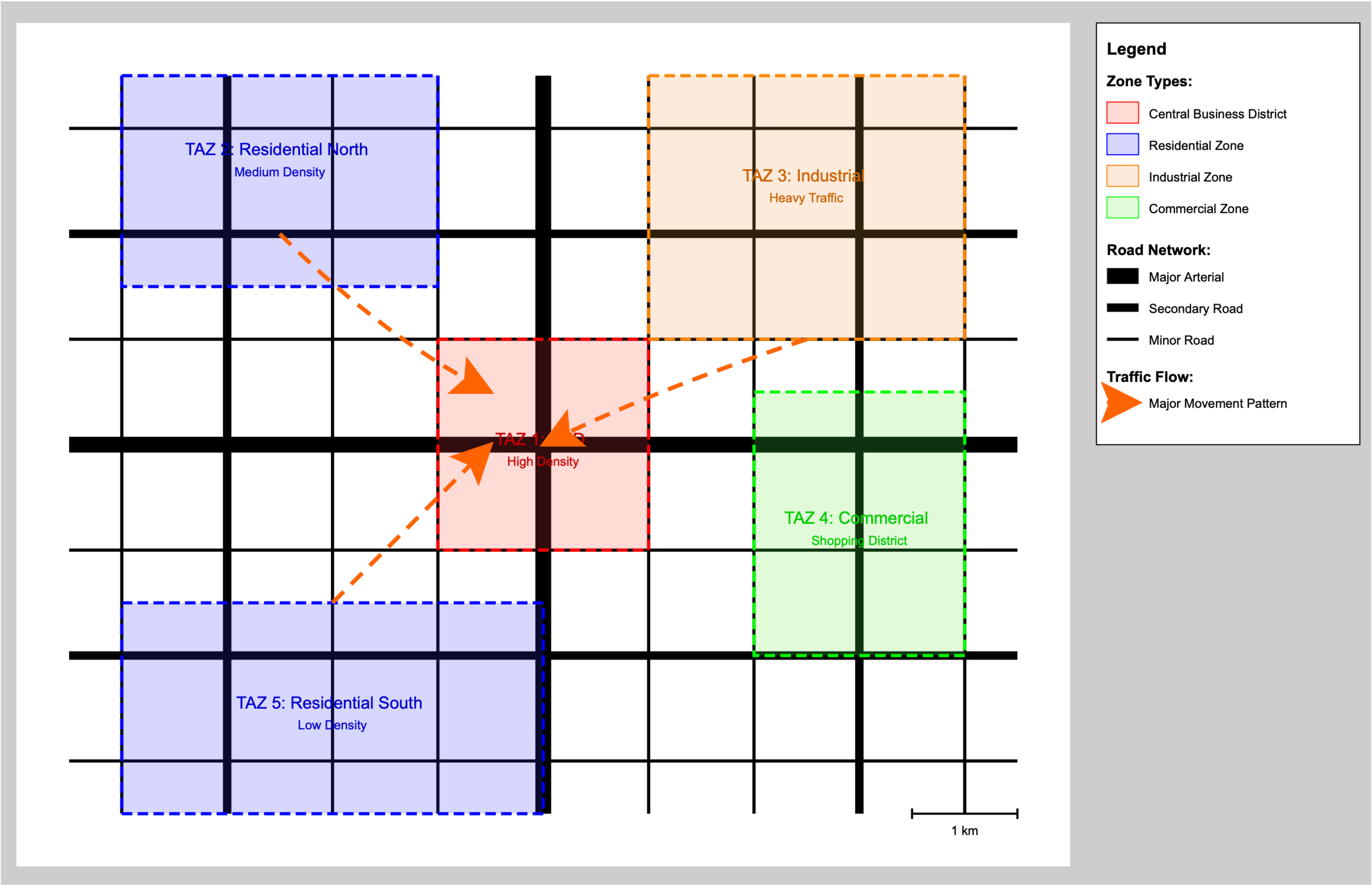
```
duarouter --trip-files trips.trips.xml
          --net-file simple.net.xml
          --additional-files vtypes.add.xml
          -o routes.rou.xml
```

Traffic Assignment Zones (TAZ)

- High fidelity vs Low fidelity

Traffic Assignment Zones (TAZ)

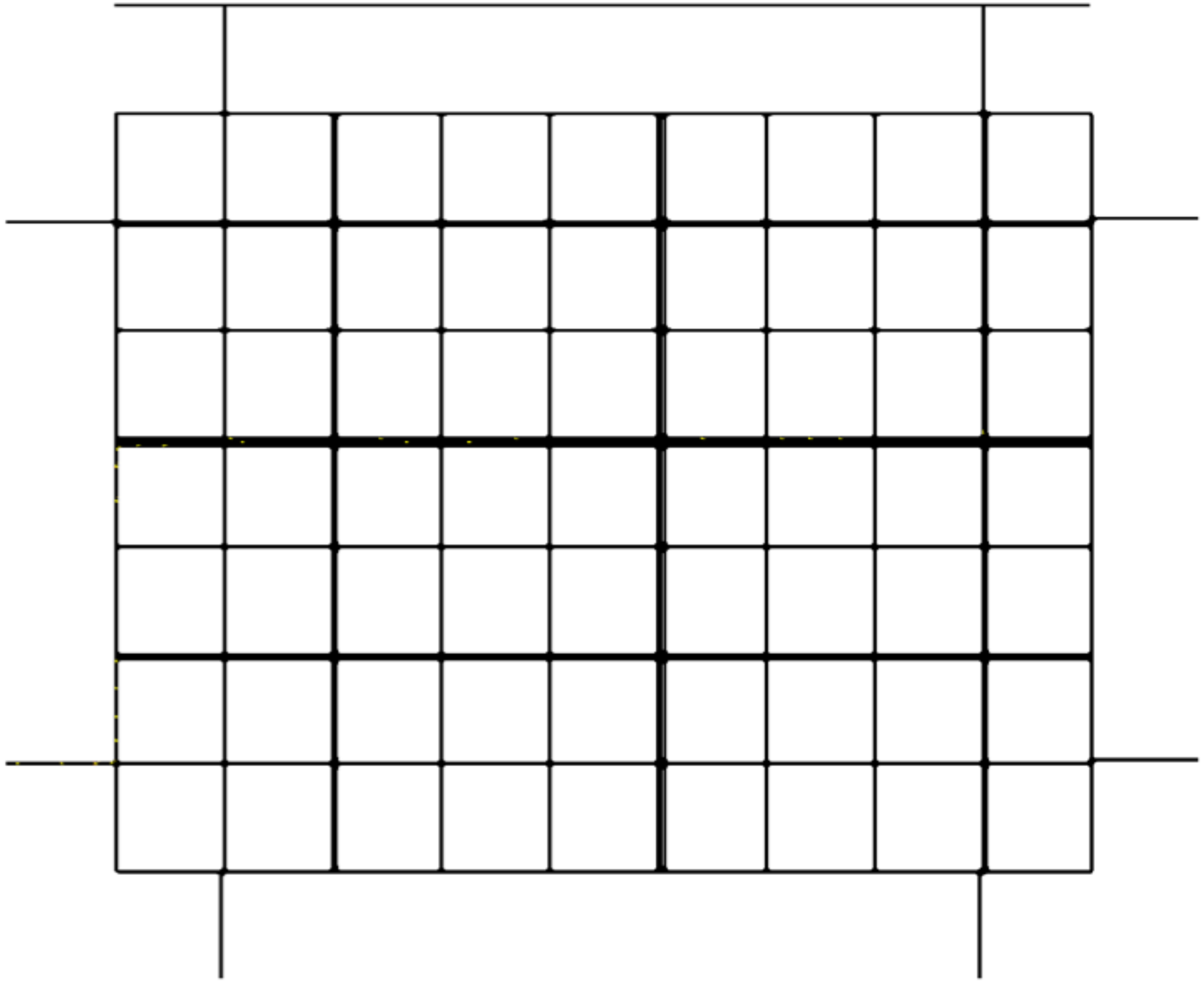
- High fidelity vs Low fidelity
- TAZ: Areas that represent origins and destinations for traffic



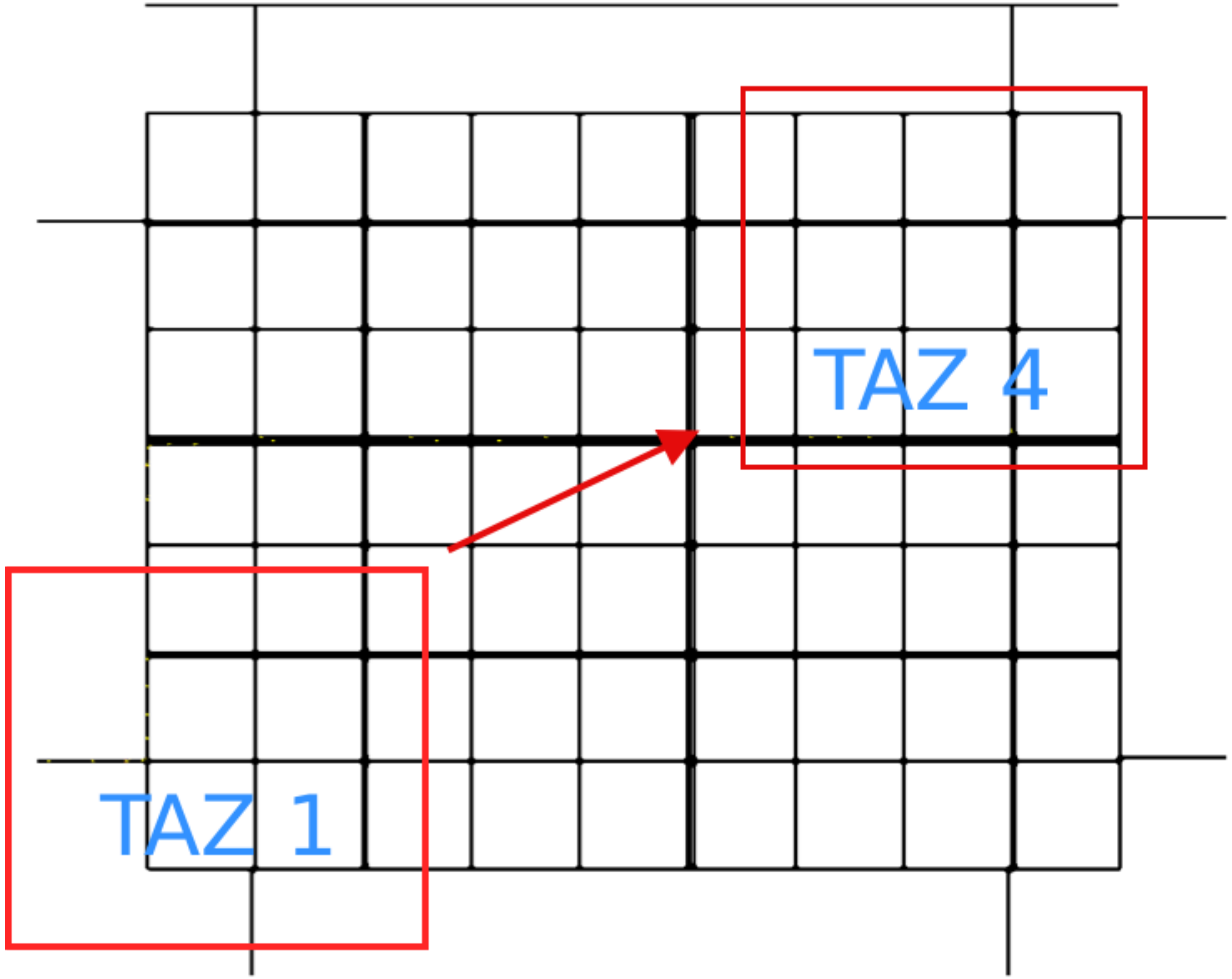
Traffic Assignment Zones (TAZ)

- High fidelity vs Low fidelity
- TAZ: Areas that represent origins and destinations for traffic
- Key Components:
    - Source (where trips begin)
    - Sink (where trips end)
    - Weight (probability of trips between zones)
- Allows macro-level traffic modeling

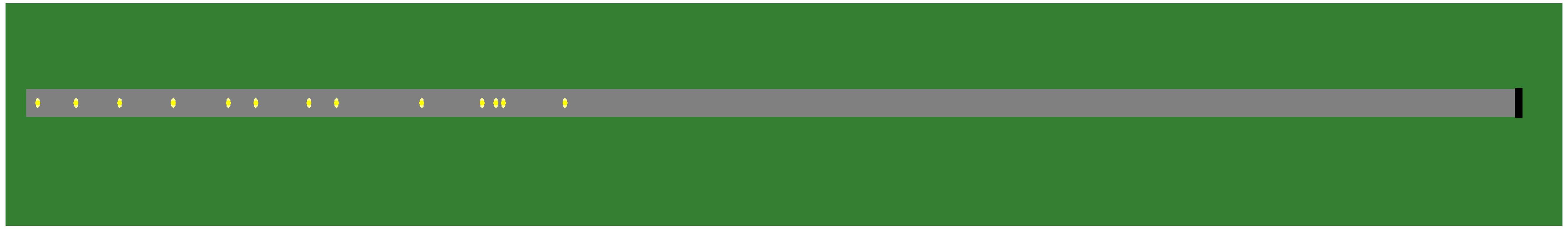
TAZ Demo



TAZ Demo

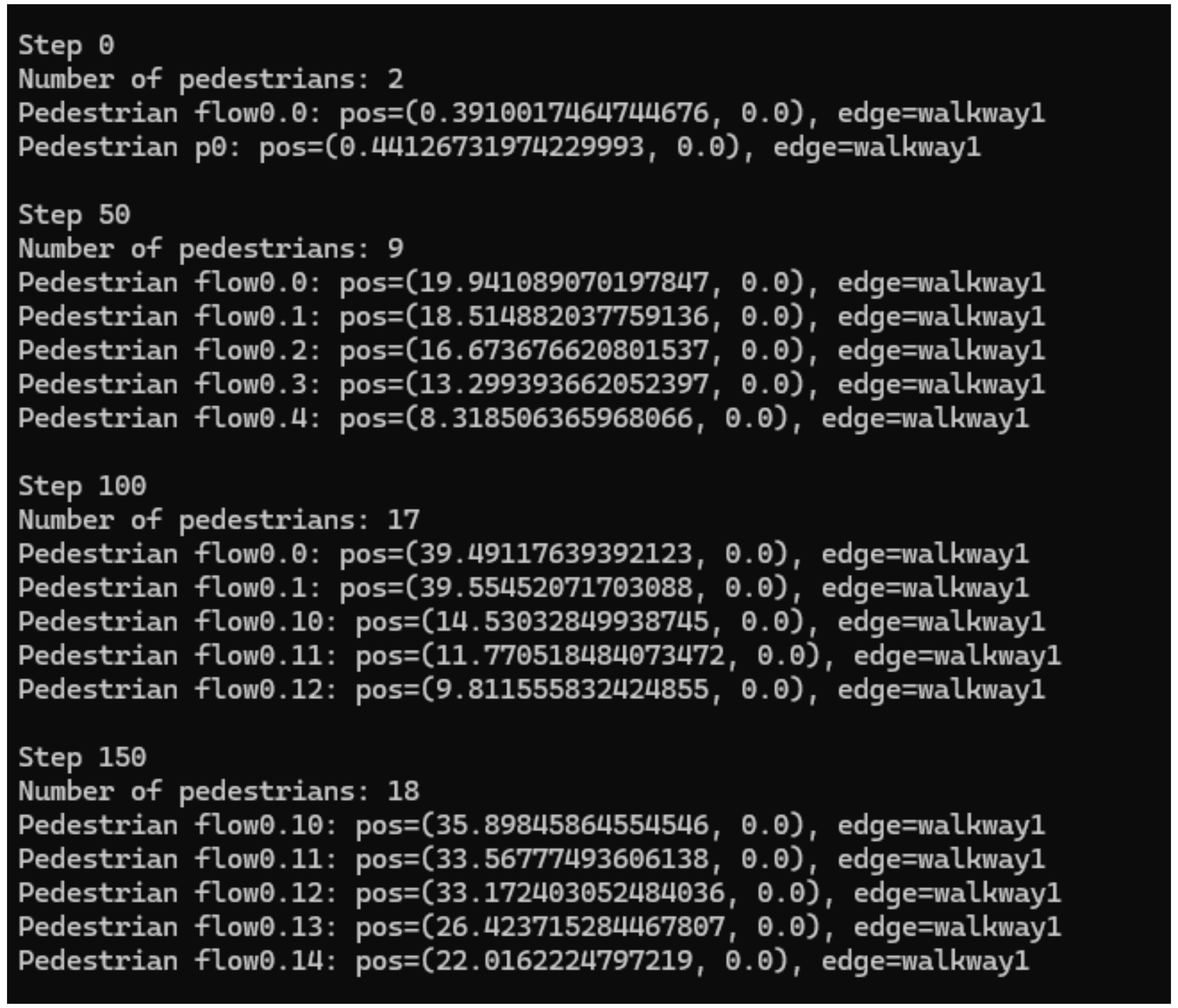


Pedestrian Simulation Demo



Pedestrian walkway

Results



```
Step 0
Number of pedestrians: 2
Pedestrian flow0.0: pos=(0.3910017464744676, 0.0), edge=walkway1
Pedestrian p0: pos=(0.44126731974229993, 0.0), edge=walkway1

Step 50
Number of pedestrians: 9
Pedestrian flow0.0: pos=(19.941089070197847, 0.0), edge=walkway1
Pedestrian flow0.1: pos=(18.514882037759136, 0.0), edge=walkway1
Pedestrian flow0.2: pos=(16.673676620801537, 0.0), edge=walkway1
Pedestrian flow0.3: pos=(13.299393662052397, 0.0), edge=walkway1
Pedestrian flow0.4: pos=(8.318506365968066, 0.0), edge=walkway1

Step 100
Number of pedestrians: 17
Pedestrian flow0.0: pos=(39.49117639392123, 0.0), edge=walkway1
Pedestrian flow0.1: pos=(39.55452071703088, 0.0), edge=walkway1
Pedestrian flow0.10: pos=(14.53032849938745, 0.0), edge=walkway1
Pedestrian flow0.11: pos=(11.770518484073472, 0.0), edge=walkway1
Pedestrian flow0.12: pos=(9.811555832424855, 0.0), edge=walkway1

Step 150
Number of pedestrians: 18
Pedestrian flow0.10: pos=(35.89845864554546, 0.0), edge=walkway1
Pedestrian flow0.11: pos=(33.56777493606138, 0.0), edge=walkway1
Pedestrian flow0.12: pos=(33.172403052484036, 0.0), edge=walkway1
Pedestrian flow0.13: pos=(26.423715284467807, 0.0), edge=walkway1
Pedestrian flow0.14: pos=(22.0162224797219, 0.0), edge=walkway1
```

Please start playing around with the code