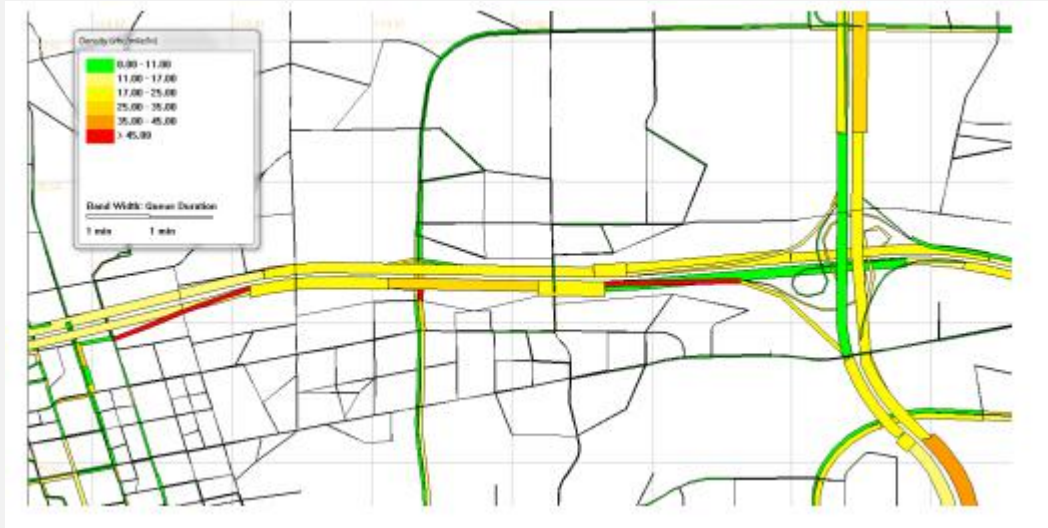


# Open-source STALite/NeXTA library based on GMNS



Dr. Xuesong (Simon) Zhou  
xzhou74@asu.edu

Arizona State University  
May 4<sup>th</sup>, 2020

# 1.Quick introduction of STAlite

- **Macroscopic assignment: --STAlite (Semi-dynamic assignment)**
- **Mesoscopic assignment: --DTAlite (Dynamic assignment)**
- **Microscopic assignment: --CAlite (Cellular Automatonassignment )**

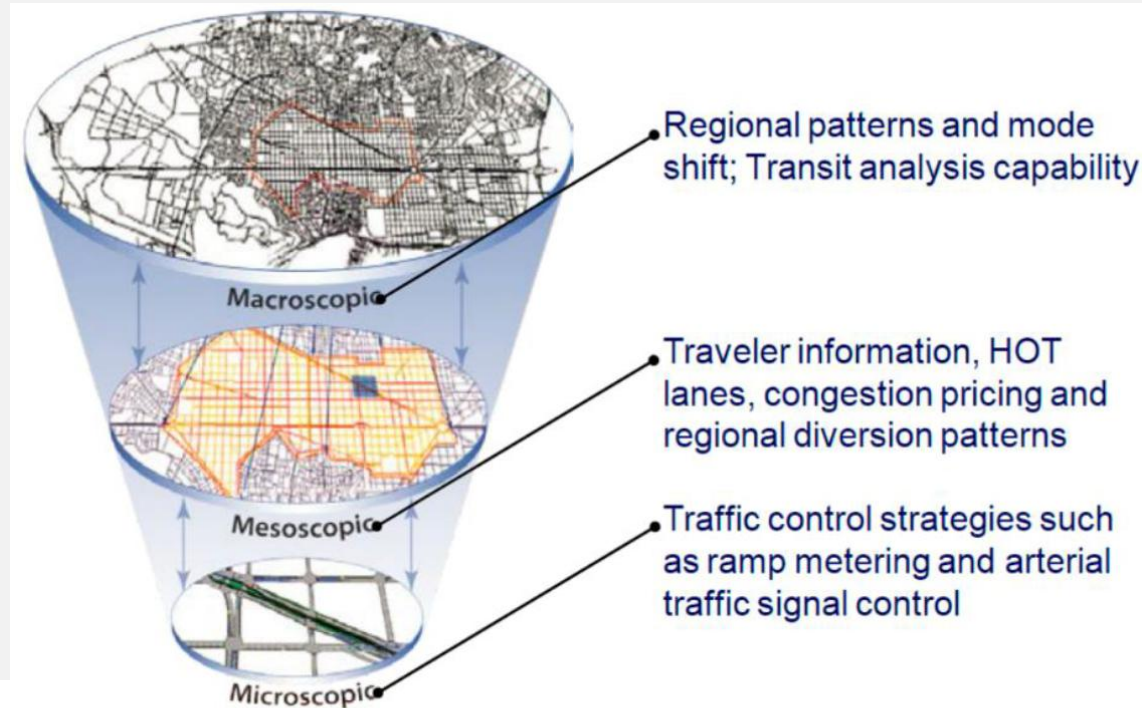
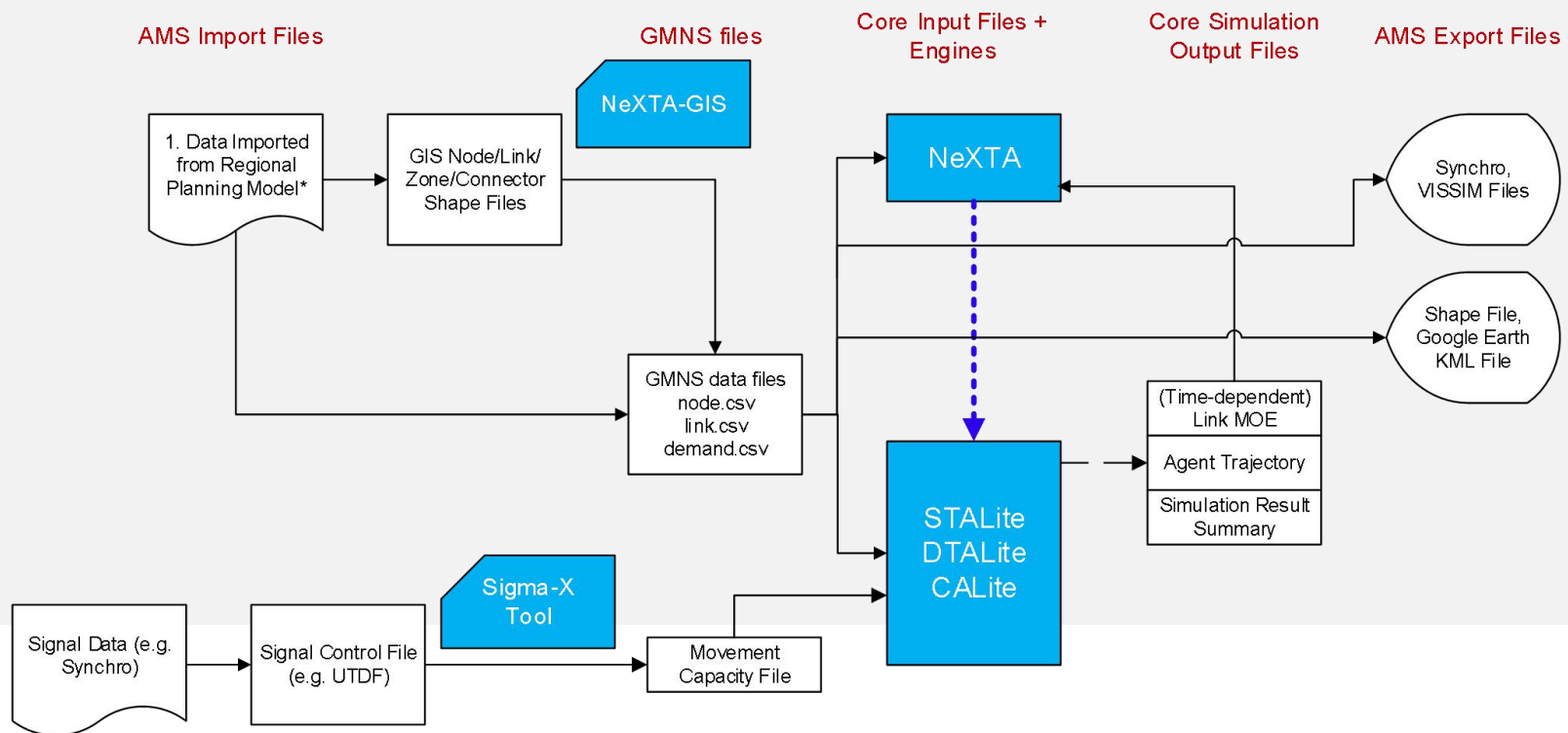


Figure 1: AMS tool resolutions (Source: FHWA Traffic Analysis Toolbox)

# Quick introduction of STAlite

- **Macroscopic assignment: --STAlite (Semi-dynamic assignment)**
- **Mesoscopic assignment: --DTAlite (Dynamic assignment)**
- **Microscopic assignment: --CALite (Cellular Automaton assignment)**



# Quick introduction of STAlite

Welcome to STAlite (Light-weight computational engine of Static Traffic Assignment and Semi-dynamic assignment)

STAlite is an open-source AMS library for efficiently macroscopic traffic assignment based on General Modeling Network Specification (GMNS) format

1. Network representation based on **GMNS format**
2. Easy to include demand from different **multiple time periods( AM, MD, PM, NT or Hourly)**
3. Provide API for both **C++ and Python interface**
4. Efficiently **multi-threading parallel computation and memory management**, implemented in **C++**

Utilize up to 40 CPU cores, 200 GB of Memory for networks with more than 50K nodes

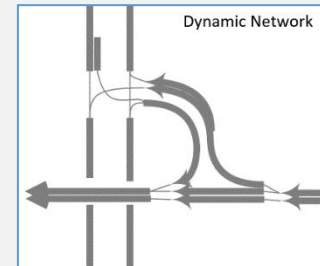
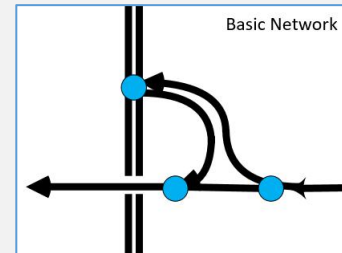
5. Extendable Volume Delay Function (VDF) functions:
  - ✓ Standard BPR function
  - ✓ BPR\_X function that can obtain dynamic travel time efficiently

## 2. Network representation based on the General Modeling Network Specification (GMNS)

# Network representation based on GMNS


[github.com/zephyr-data-specs/GMNS](https://github.com/zephyr-data-specs/GMNS)

- The objective of the General Modeling Network Specification (GMNS) is to provide **a common human and machine-readable format** for sharing routable road network files
  - It is designed to be used in **multi-resolution and multi-modal** static and dynamic transportation planning and operations models
  - It will facilitate the **sharing of tools and data sources** by modelers
- The project is overseen by a project management group, with MPO, city, industry, academic and US DOT participation. In 2019, with support from the **Federal Highway Administration**, the team developed requirements and an initial release of the specification



# Network representation based on GMNS

[github.com/zephyr-data-specs/GMNS](https://github.com/zephyr-data-specs/GMNS)

 Z E P H Y R

AboutActivities

## Network Data Standard and Management Tools

### Meeting Announcement

The GMNS Project Management Group [will meet at the 2020 TRB Annual Meeting in Washington DC](#).

- Mon Jan 13th 1:30-3:00
- 159B / Washington DC Convention Center

*Note that seating may be limited and will be prioritized for PMG members*

### Governance

This project is overseen by a board-approved Project Management Group (PMG) as follows:

- Joe Castiglione, SFCTA (chair, board representative)
- Michael Mahut, INRO
- Wu Sun, SANDAG
- Guy Rousseau, ARC
- Chetan Joshi, PTV
- Jeff Frkonja, Portland Metro
- Scott Smith, Volpe
- Natalia Ruiz Juri, University of Texas Center for Transportation Research
- Song Gao, UMass Amherst

# Network representation based on GMNS

[github.com/zephyr-data-specs/GMNS](https://github.com/zephyr-data-specs/GMNS)

Component of the data specification	Macro Models	Meso and Micro Models
Physical network elements on the map	Nodes, link_geometry	Nodes, link_geometry
Connecting the elements	Nodes and road_links	Movements and lanes
Link capacity	Link capacity	Emergent property of lanes and the model used
Intersection capacity	Not considered	Emergent property of lanes, movements and traffic controls
Speed	Link speed	Link speed and movement delay
Pedestrian network	Road_link pedestrian facility information	Road_link pedestrian facility information or offroad_links
Traffic controls	Node, Road_Link, Movement	Movement and signal tables
Elements that vary by time of day	Not used	Link_TOD, Movement_TOD

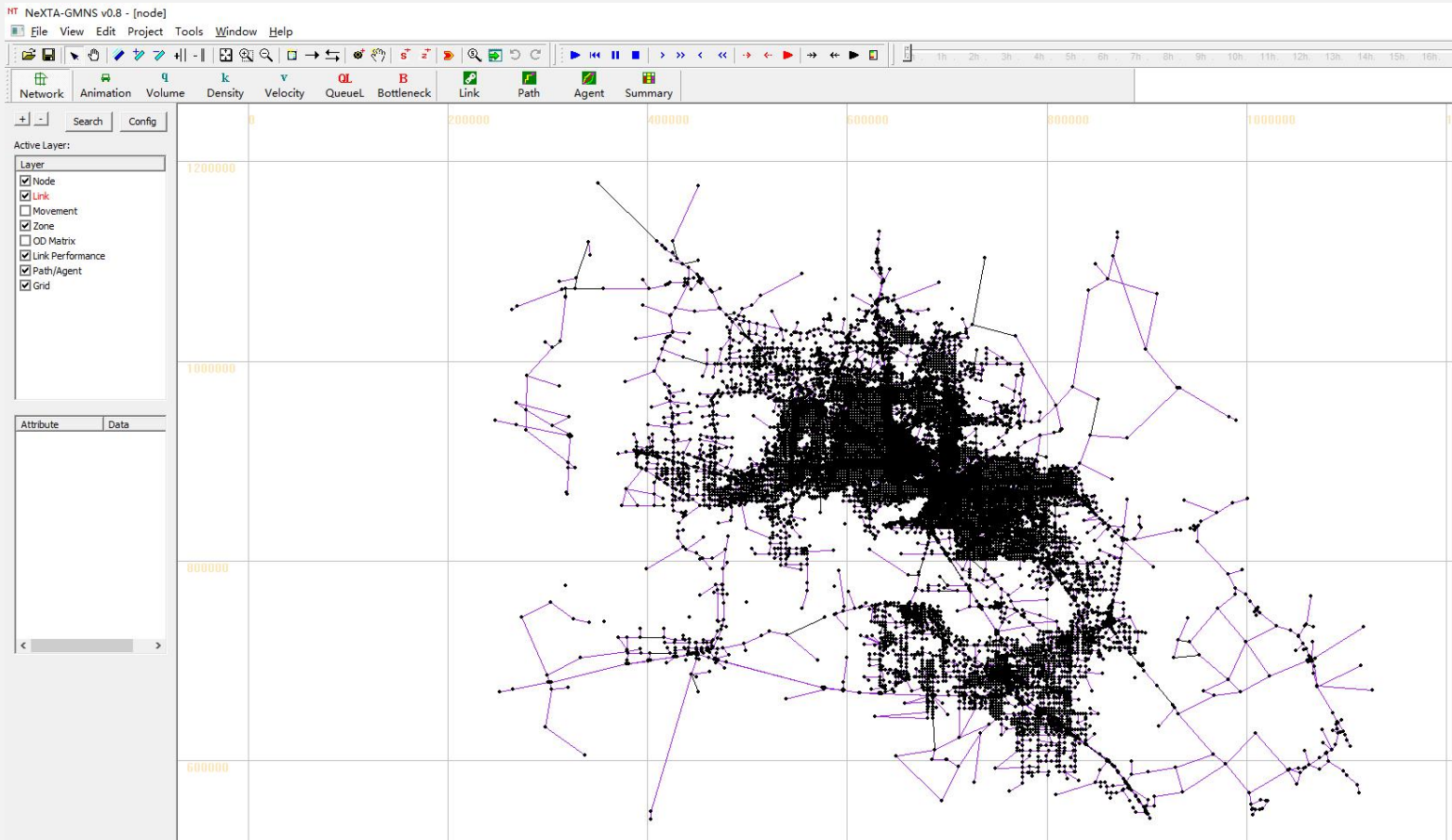
Members of the Zephyr Foundation project, General Travel Network Data Standard and Tools, and other interested stakeholders are invited to review and comment on the specification. In developing this specification, we consulted existing open-source specifications, including SharedStreets, OpenDrive, MATSim, Network EXplorer for Traffic Analysis (NEXTA) or DTALite, TRANSPORTATION ANALYSIS SIMULATION SYSTEM (TRANSIMS), Aequilibræ, Highway Performance Monitoring System (HPMS), All Road Network of Linear Referenced Data (ARNOLD), the Florida Transportation Modeling Portal (FSUTMS), and the Synchro Universal Traffic Data Format (UTDF).



### 3. Open-source Nexta AMS data hub for GMNS format

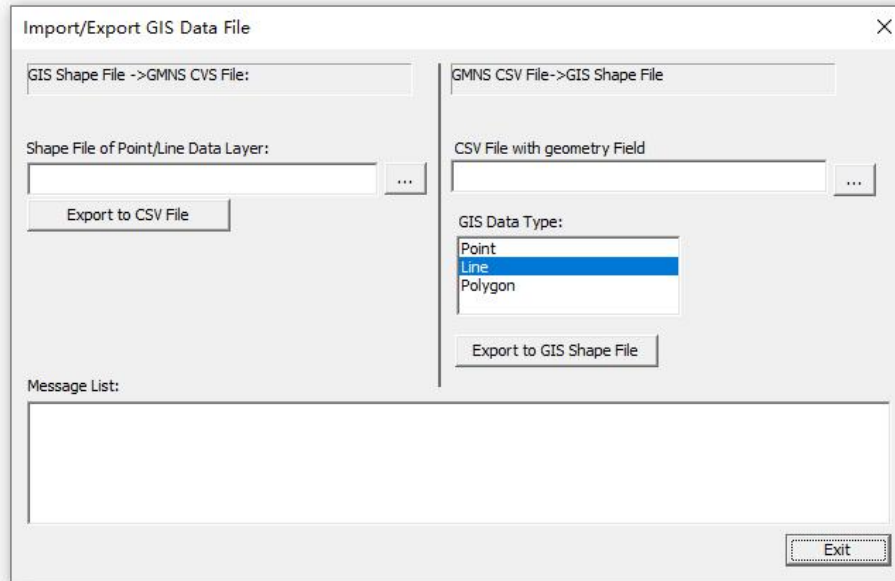


# Open-source Nexta AMS data hub for GMNS format



# Open-source Nexta AMS data hub for GMNS format

## A Convert GIS Shape File to GMNS CVS



Shape files 2 GMNS CVS files

- **Cube GIS files**
- **TransCAD GIS files**
- **VISUM GIS files**



1.Cube\_sample\_GIS\_files



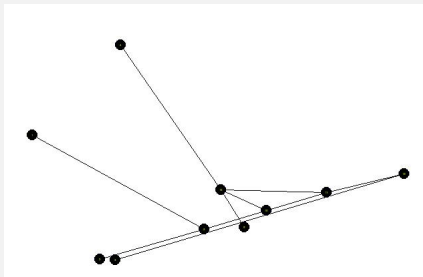
2.TransCAD\_sample\_GIS\_files



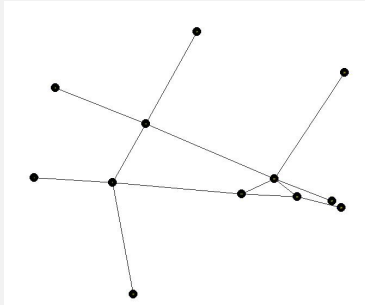
3.VISUM\_sample\_GIS\_files

# Open-source Nexta AMS data hub for GMNS format

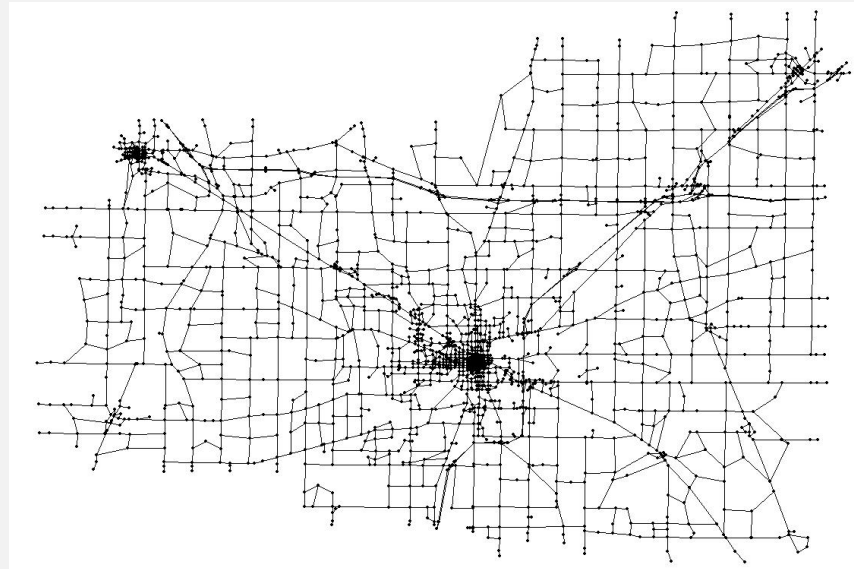
## B. Display Macroscopic Network



**Freeway Interchange**



**Bicycle multiple facilities**



**Macro-network of Lima**

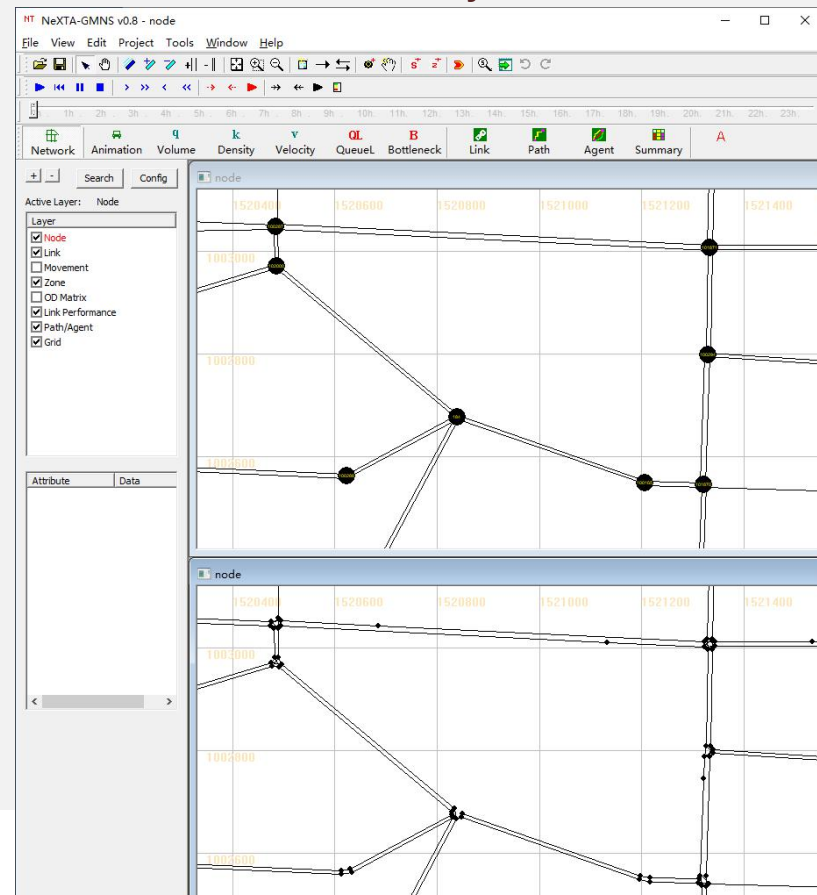
[Three working examples in Github https://github.com/zephyr-data-specs/GMNS/](https://github.com/zephyr-data-specs/GMNS/)

# Open-source Nexta AMS data hub for GMNS format

## C. Convert Macroscopic network to mesoscopic network

Tile horizontally

- **Network generation procedure:**
  - **Step 1:** Obtain the exact geometry string for each link by performing lane-based offset with respect to the original geometry coordinate.
  - **Step 2:** Split links into segments based on provided segment data so that the number of lanes on each segment keeps the same
  - **Step 3:** Generate mesoscopic links for each segment
  - **Step 4:** Create a connector (mesolink) to connect corresponding inbound mesolink and outbound mesolink for each movement at intersections
  - **Step 5:** Build microscopic network by discretizing each mesolink into cells with constant length, where two consecutive cells are connected by a micronode.
  - **Step 6:** Output networks

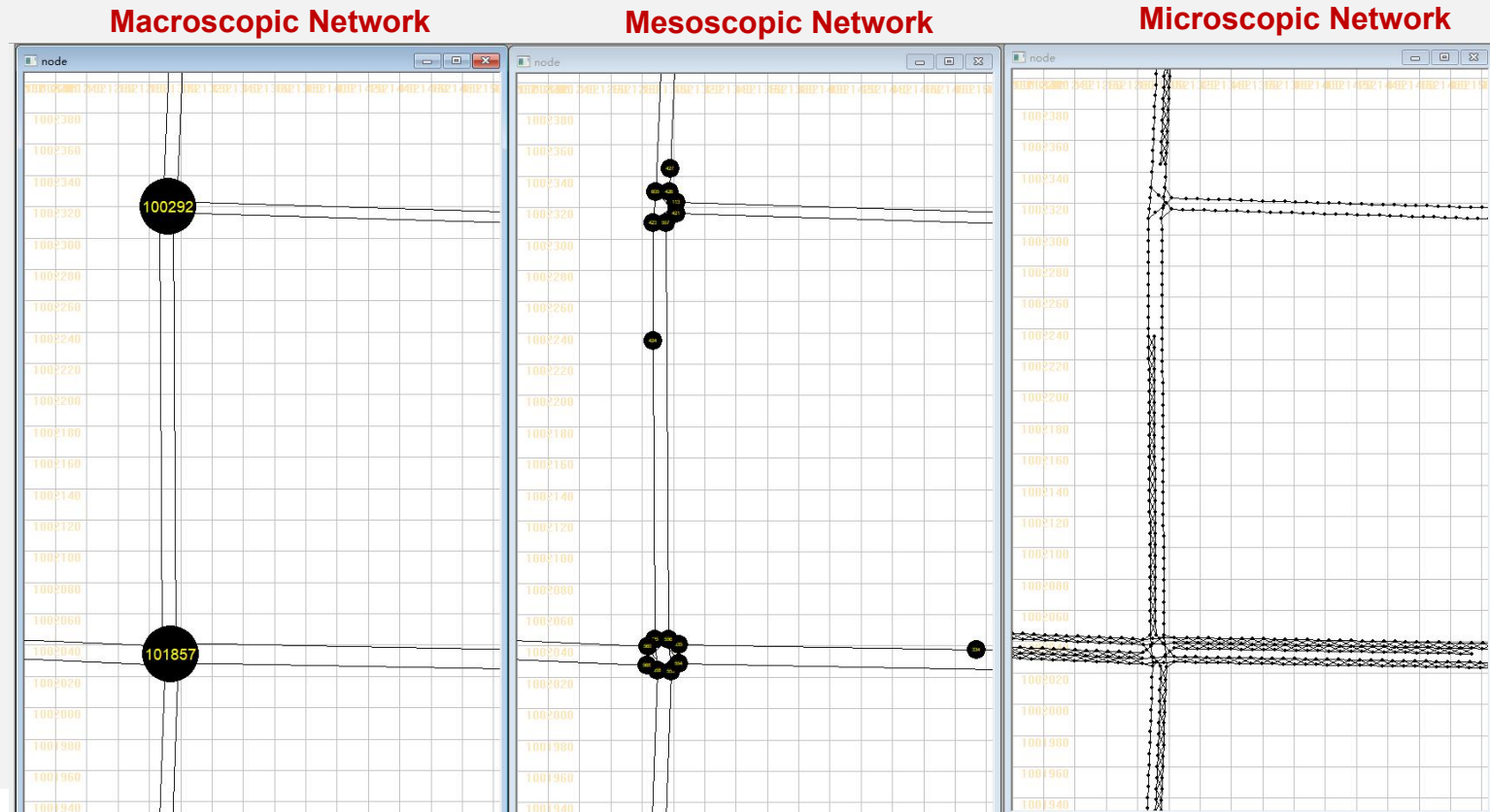


Macro-Net

Meso-Net

# Open-source Nexta AMS data hub for GMNS format

D. Macroscopic network → Mesoscopic network → Microscopic network



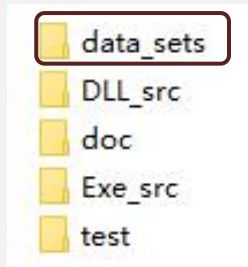
Microscopic network: cell-based network (5 meters or 15 feet per cell)

## 4. Source codes of STAlite

# Dataset

STAlite folder include: data\_sets, DLL\_src,

**Data\_sets** is the folder with input data (GMNS format)



## Network information



node.csv

	A	B	C	D	E	F	G	H
1	node_id	name	x_coord	y_coord	z_coord	node_type	control_ty	zone_id
2	1		50000	510000				1
3	2		320000	510000				2
4	3		50000	440000				3

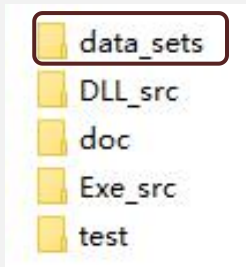
road\_link.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	road_link_id	name	from_nod	to_node	capacity	free_speed	lanes	facility_type	bike_facil	ped_facili	parking	allowed_u	length	geometry
2	1		1	2	12950.1	60	1 Freeway						6	<LineString>
3	2		1	3	11701.74	60	1 Freeway						4	<LineString>
4	3		2	1	12950.1	60	1 Freeway						6	<LineString>
5	4		2	6	2479.09	60	1 Freeway						5	<LineString>

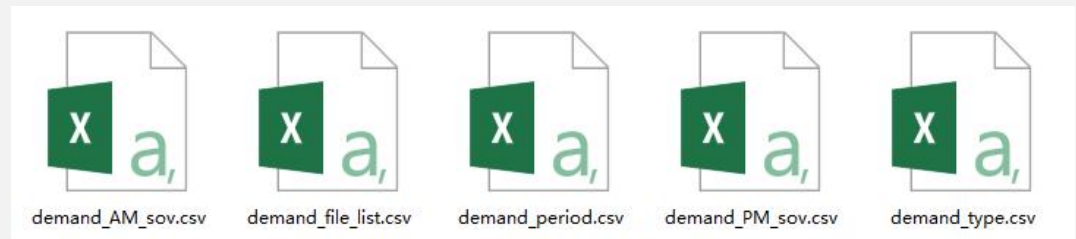


# Dataset (cont'd )

**Data\_sets** is the folder with input data (GMNS format)



## Demand files



	A	B	C	D	E	F	G
1	scenario	file_seq	file_name	format_ty	demand	demand_type	
2		0	1 demand_AM_sov.csv	column	AM	SOV	
3		0	2 demand_PM_sov.csv	column	AM	SOV	

Managing demand files for different time periods

	A	B	C
1	demand	demand	time_period
2		1 AM	0600_1100
3		2 PM	1600_2000

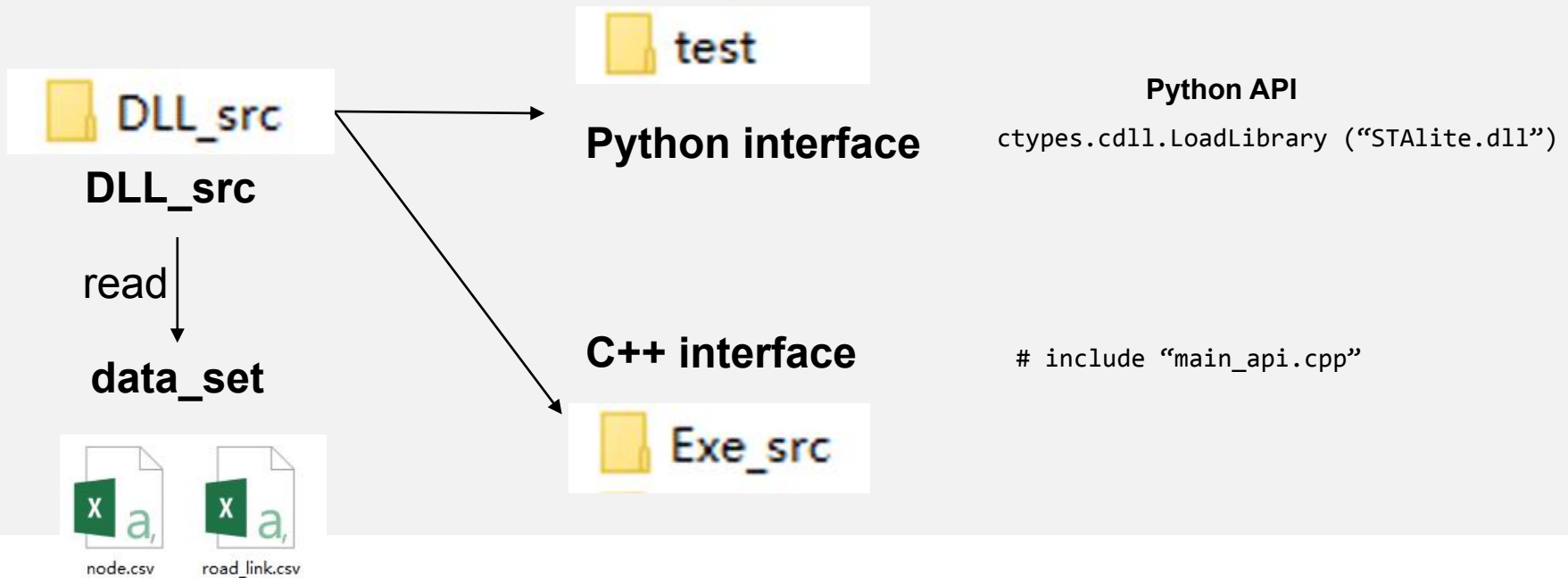
	A	B	C	D
1	demand_t	demand_t	VOT	cap_ratio
2		1 SOV	10	1
3		2 HOV	10	1
4		3 truck	20	1
5		4 transit	20	1

- How many time periods are considered?
- HHMM format

# Source code structure

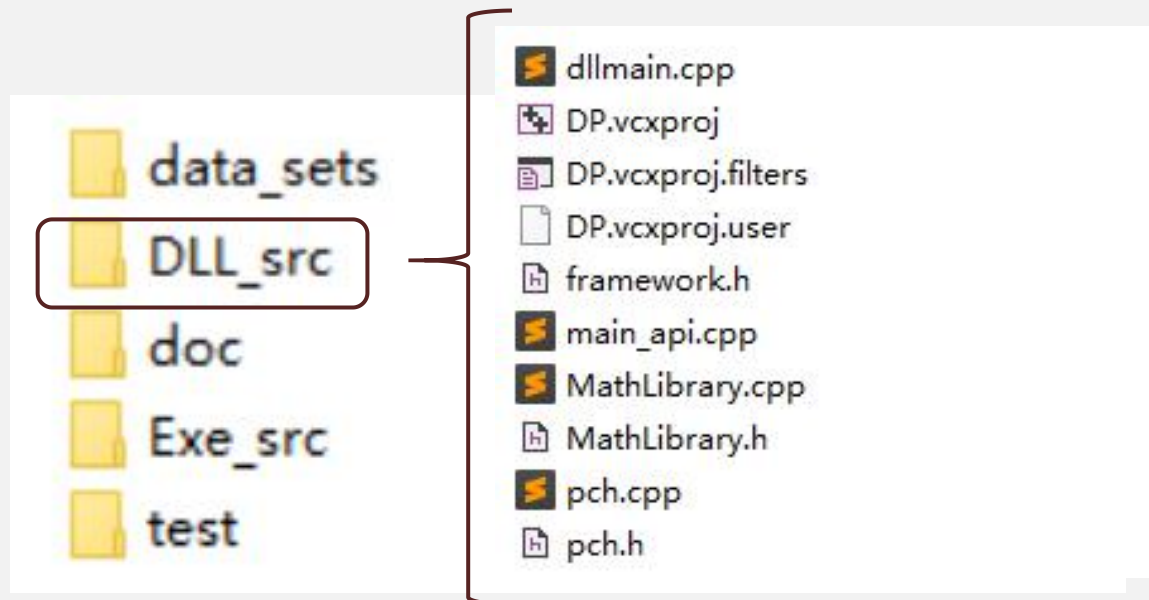
You probably want to choose a programming environment from which to use STAlite. We have two available interfaces to use STAlite:

- Execute STAlite using C++ interface
- Execute STAlite using Python interface



# Source code for building STALite DLL as Python API

**DLL\_src** is a dynamic link library folder used for holding various key STALite functions and procedures for traffic assignments



# Source code for building STALite Executable

**Exe\_src** includes executable console of C++ to run STAlite

```
#include "stdafx.h"
```

```
#include "..\..\DLL_src\DP\main_api.cpp"
```

Import **Application Programming Interface (API)** from DLL\_src folder

```
int main(int argc, TCHAR* argv[], TCHAR* envp[])
```

```
{
```

```
    int iteration_number = 5;
```

```
    int b = 0;
```

```
    network_assignment(iteration_number, b);
```

Set parameters for network assignment function

```
}
```

# Python API test environment

**test** includes executable console Python to run STAlite

```
test_dll.py •
test_dll.py > fun_c
1  import time
2  iteration_number = 5
3
4  b = 2
5
6  def fun_c():
7      import ctypes
8      cdll = ctypes.cdll.LoadLibrary(r"STAlite.dll")
9      time_start = time.time()
10
11      network_compu = cdll.network_assignment
12      # add_fun.argtypes=[ctypes.c_float,ctypes.c_float]
13      network_compu.restype = ctypes.c_double
14      c = network_compu(iteration_number,b)
15
16
17      time_end = time.time()
18      print('FUN_C')
19      print('output: {}, total time: {}'.format(c, time_end-time_start))
20
21
22  if __name__ == "__main__":
23      fun_c()
24
```

Import STAlite.dll file

Use network\_assignment function  
Set parameter of network\_assignment function

THANKS