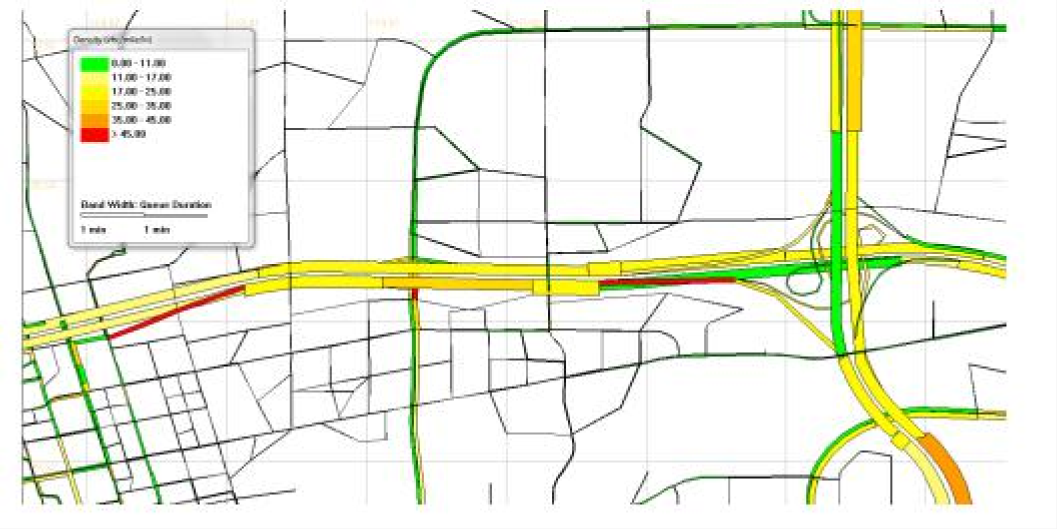
Open-sourece STALite/NeXTA library based on GMNS



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

Arizona State University

May 4th, 2020

1.Quick introduction of STAlite

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

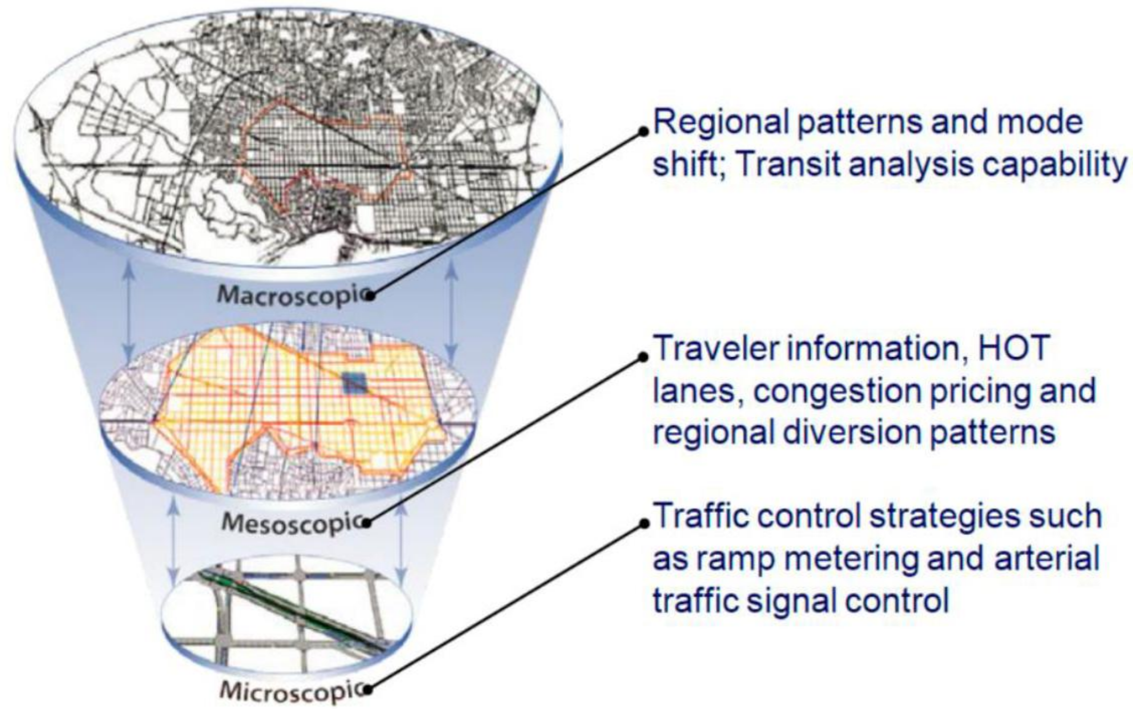
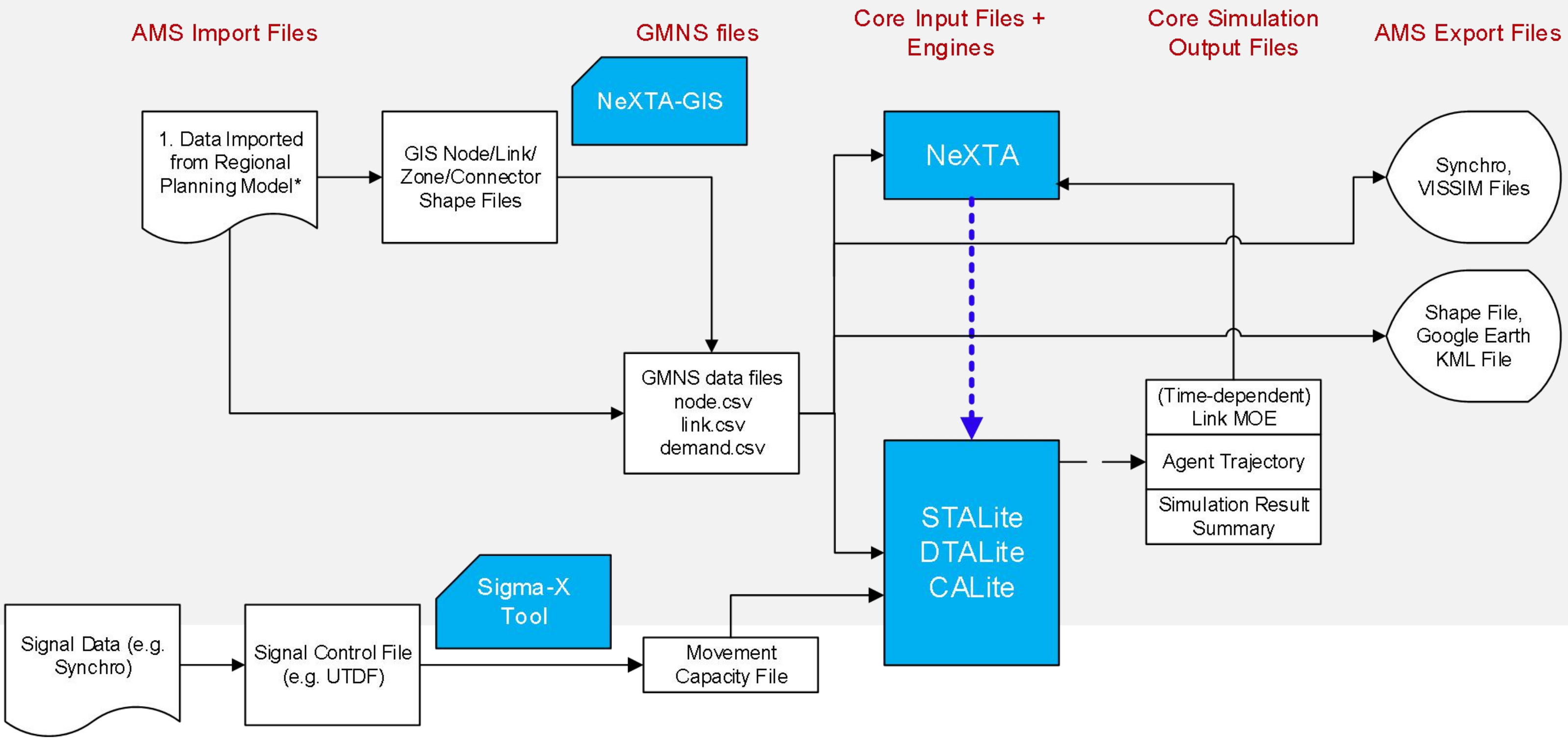

Macroscopic
Mesoscopic
Microscopic
Regional patterns and mode shift; Transit analysis capability
Traveler information, HOT lanes, congestion pricing and regional diversion patterns
Traffic control strategies such as ramp metering and arterial traffic signal control

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 Automatonassignment)


AMS Import Files
GMNS files
Core Input Files + Engines
Core Simulation Output Files
AMS Export Files
NeXTA-GIS
1. Data Imported from Regional Planning Model*
GIS Node/Link/ Zone/Connector Shape Files
NeXTA
Synchro, VISSIM Files
Shape File, Google Earth KML File
GMNS data files node.csv link.csv demand.csv
(Time-dependent) Link MOE
Agent Trajectory
Simulation Result Summary
STALite DTALite CALite
Sigma-X Tool
Signal Data (e.g. Synchro)
Signal Control File (e.g. UTDF)
Movement Capacity File

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. Efficient **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

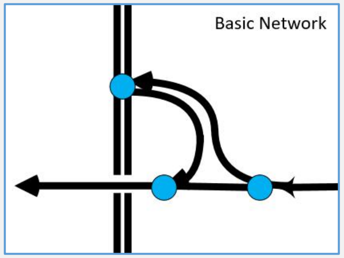
   ✓ BPR_X function that can obtain dynamic travel time efficiently

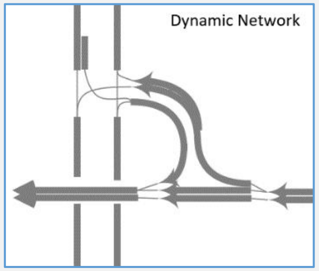
2. Network representation based on the General Modeling Network Specification (GMNS)

Network representation based on GMNS

github.com/zephyr-data-specs/GMNS

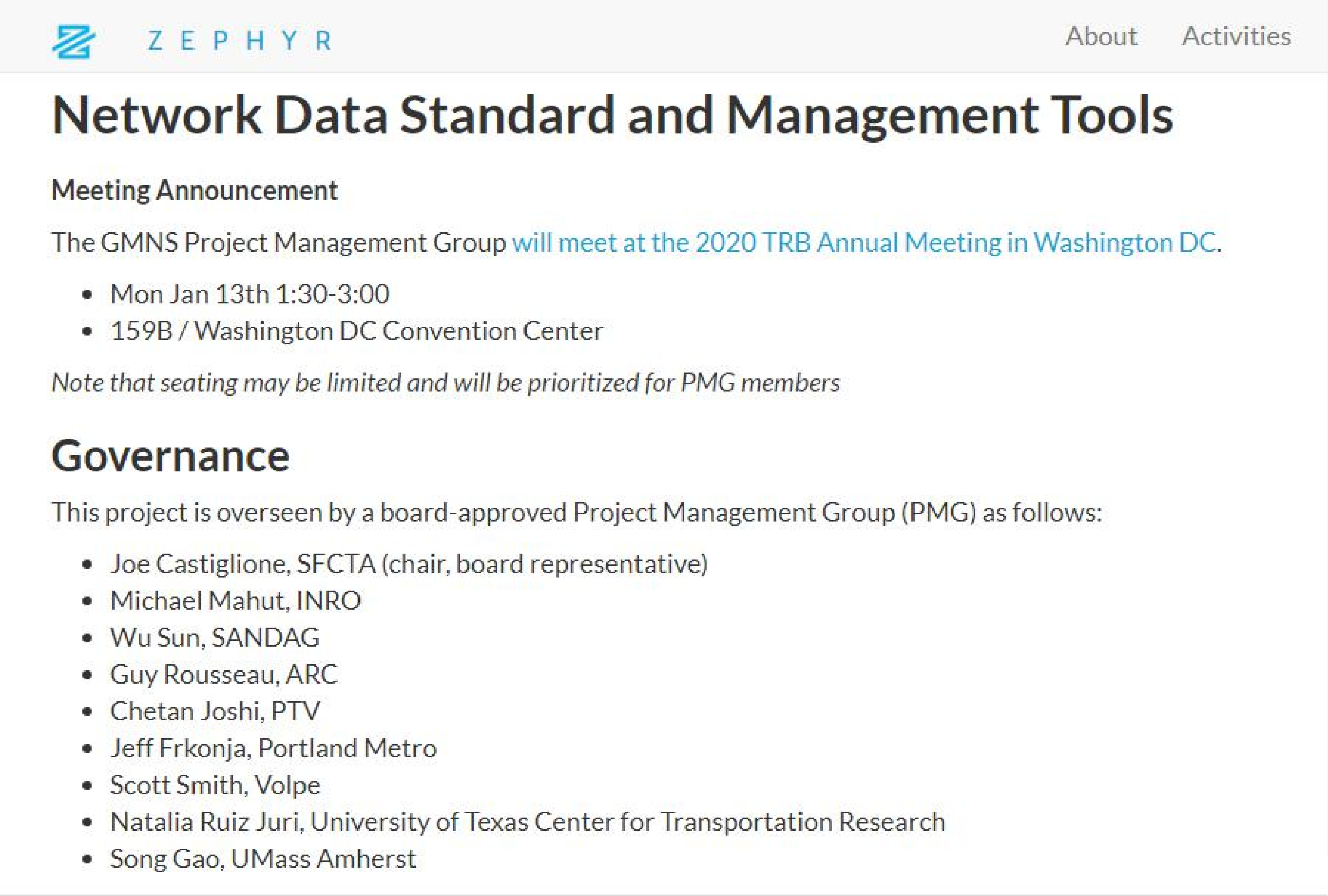
- The objective of the General Modeling Network Specification (GMNS) is to provide **a common human and machine-readable format** for sharing routable road network files
  - It is designed to be used in **multi-resolution and multi-modal** static and dynamic transportation planning and operations models
  - 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



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

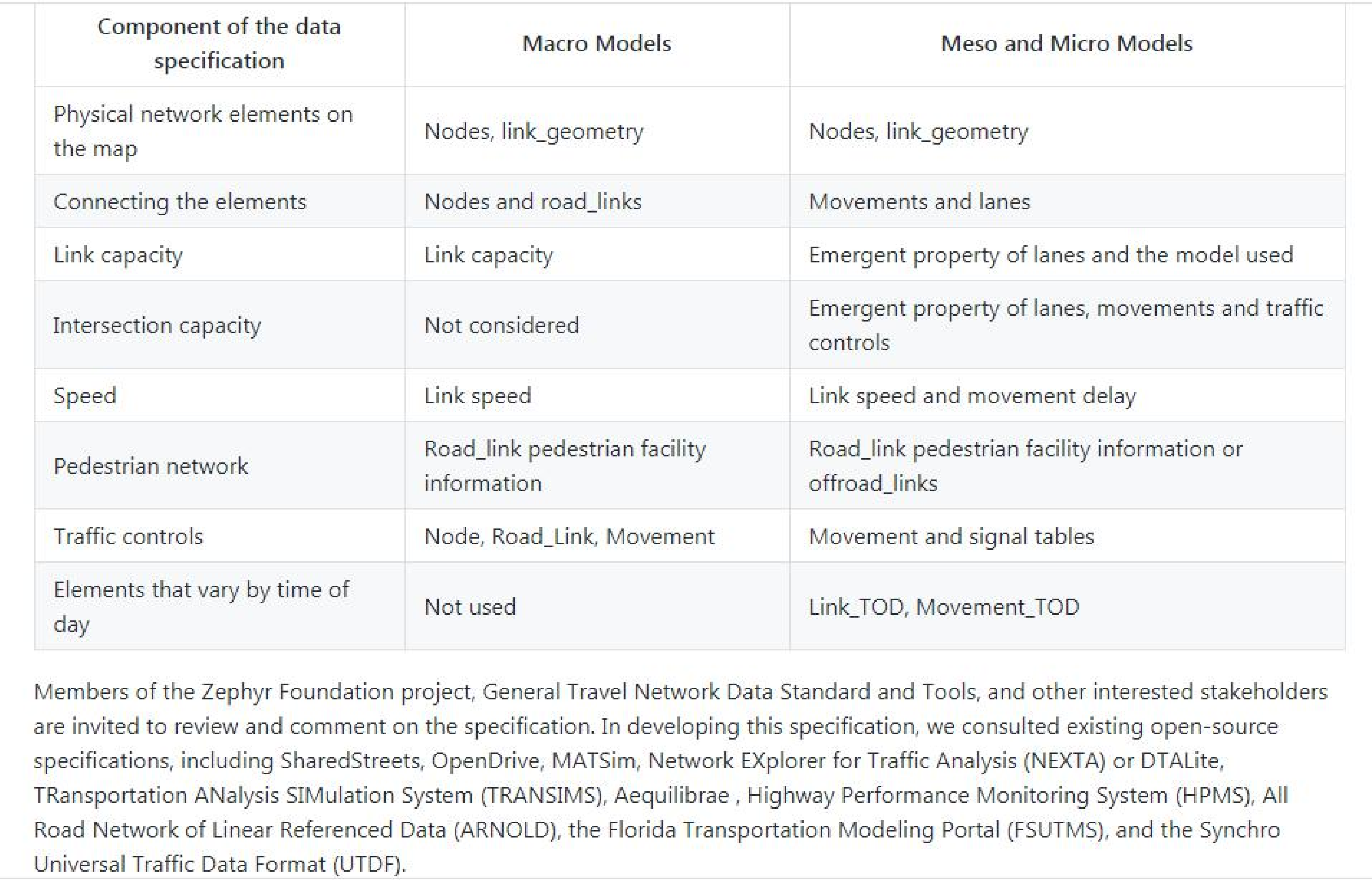
Governance

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

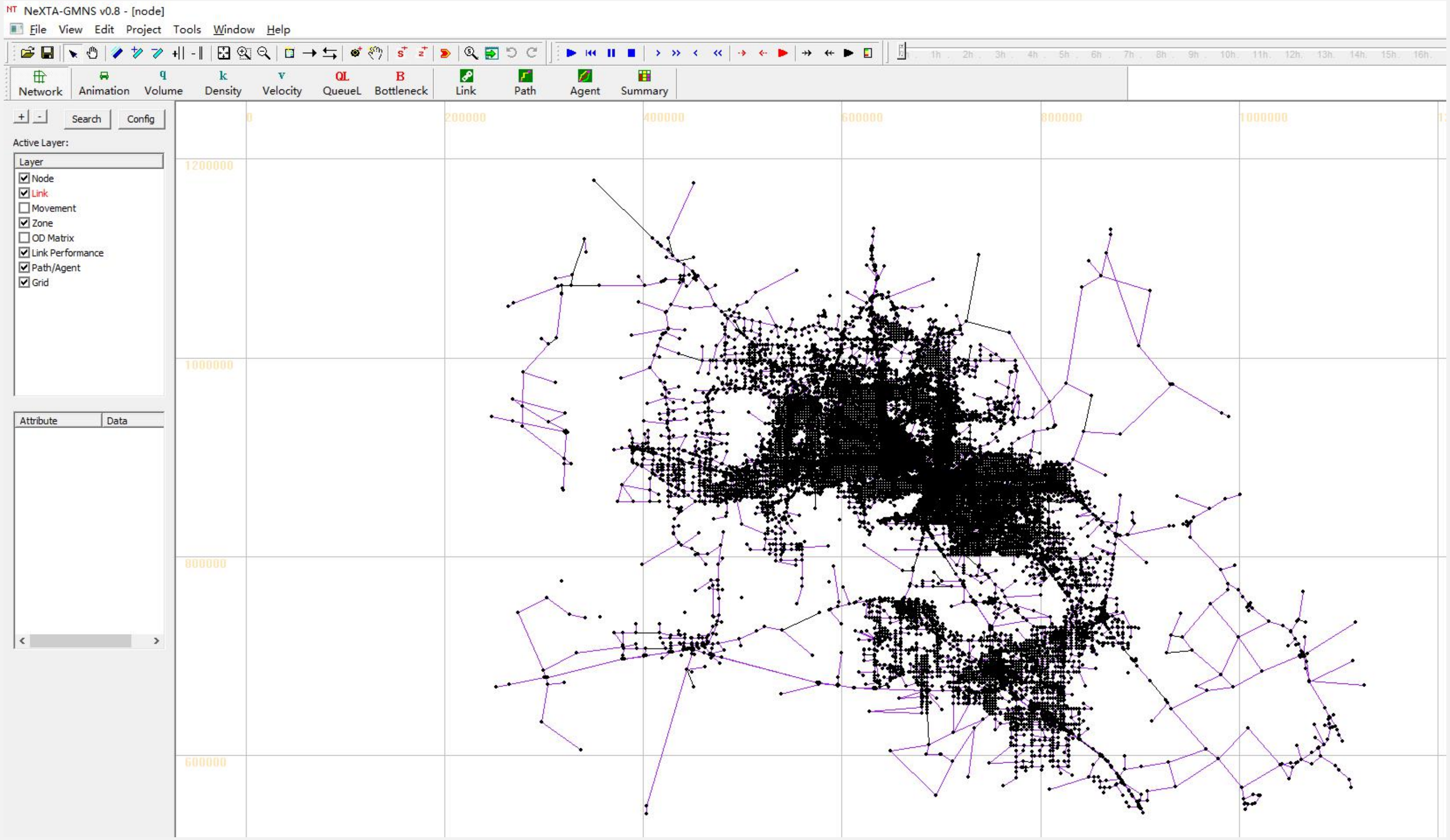


Component of the data specification	Macro Models	Meso and Micro Models
Physical network elements on the map	Nodes, link_geometry	Nodes, link_geometry
Connecting the elements	Nodes and road_links	Movements and lanes
Link capacity	Link capacity	Emergent property of lanes and the model used
Intersection 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), Aequilibrae , 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

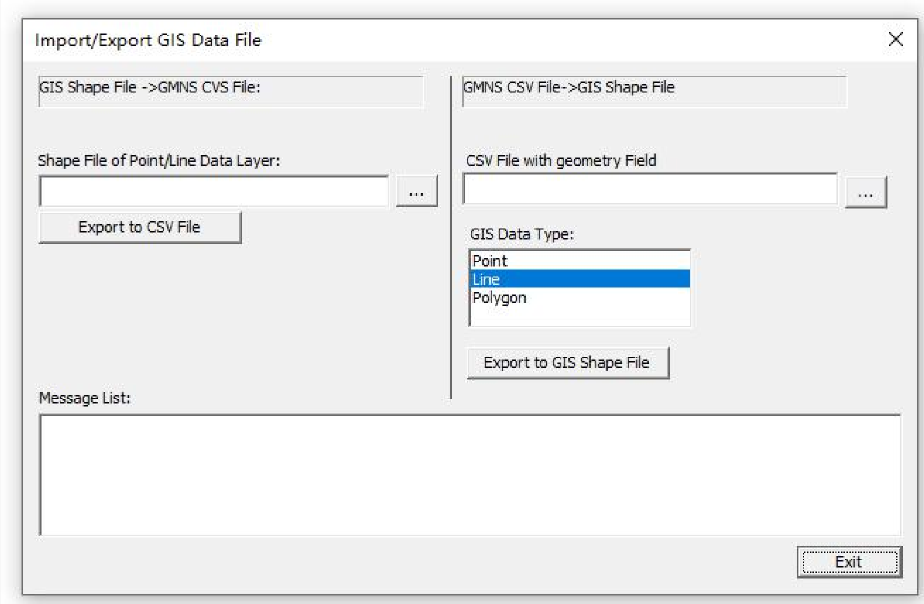
Open-source Nexta AMS data hub for GMNS format


NeXTA-GMNS v0.8 - [node]
File View Edit Project Tools Window Help
1h 2h 3h 4h 5h 6h 7h 8h 9h 10h 11h 12h 13h 14h 15h 16h
Network Animation Volume Density Velocity QueueL Bottleneck Link Path Agent Summary
Search Config
Active Layer:
Layer
Attribute Data
0
200000
400000
600000
800000
1000000
1200000
1000000
800000
600000

☑ Node
☑ Link
□ Movement
☑ Zone
□ OD Matrix
☑ Link Performance
☑ Path/Agent
☑ Grid

Open-source Nexta AMS data hub for GMNS format

A Convert GIS Shape File to GMNS CVS


Import/Export GIS Data File

GIS Shape File ->GMNS CVS File:

Shape File of Point/Line Data Layer:

GMNS CSV File->GIS Shape File

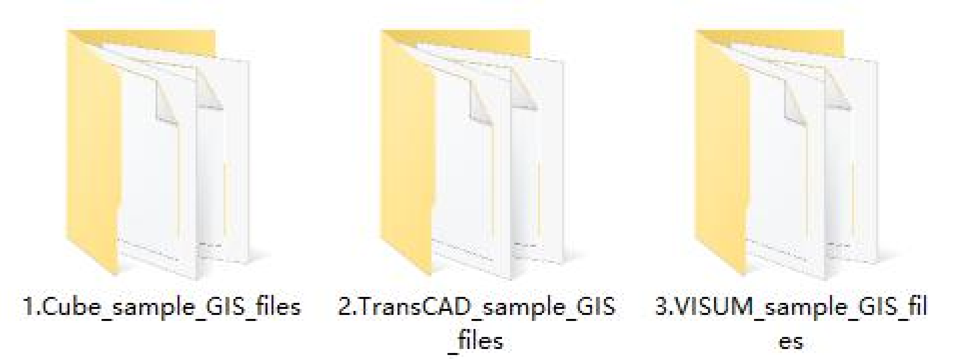
CSV File with geometry Field

GIS Data Type: Line

Message List:

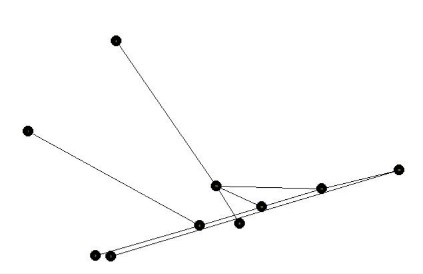
Shape files 2 GMNS CVS files

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

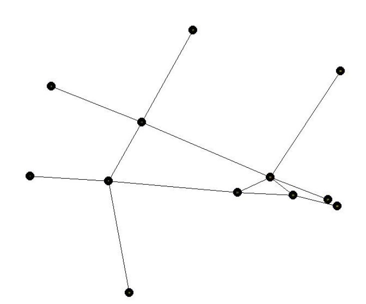


Open-source Nexta AMS data hub for GMNS format

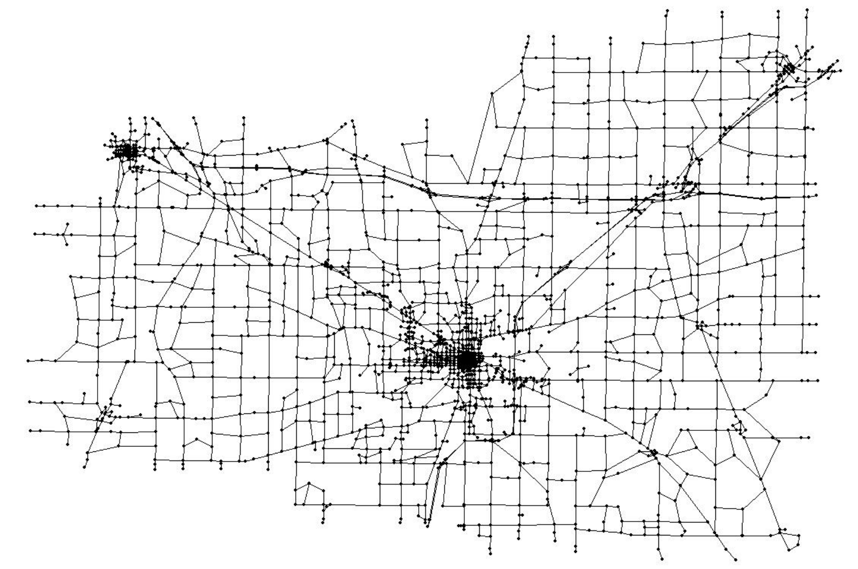
B. Display Macroscopic Network



Freeway Interchange



Bicycle multiple facilities



Macro-network of Lima

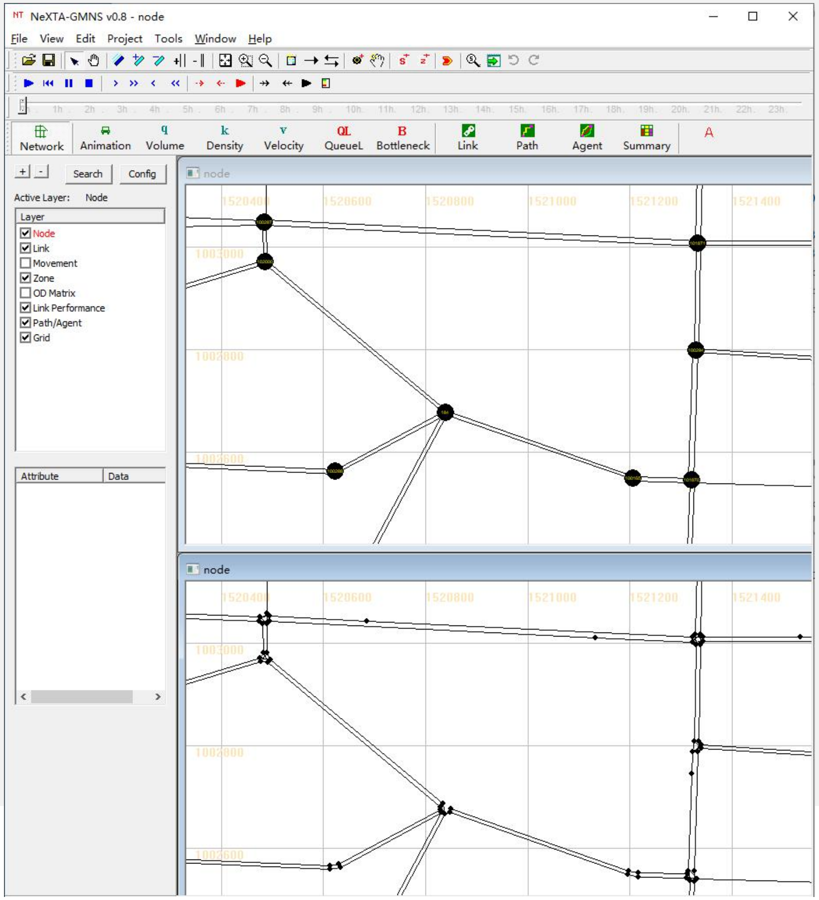
Three working examples in Github https://github.com/zephyr-data-specs/GMNS/

Open-source Nexta AMS data hub for GMNS format

C. Convert Macroscopic network to mesoscopic network

- **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

Tile horizontally



Active Layer: Node

Layer

☑ Node
☑ Link
☐ Movement
☑ Zone
☐ OD Matrix
☑ Link Performance
☑ Path/Agent
☑ Grid

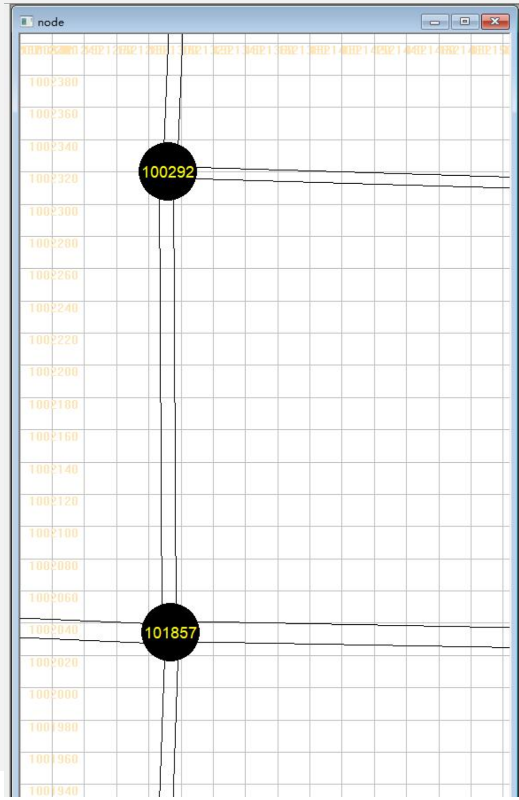
Macro-Net

Meso-Net

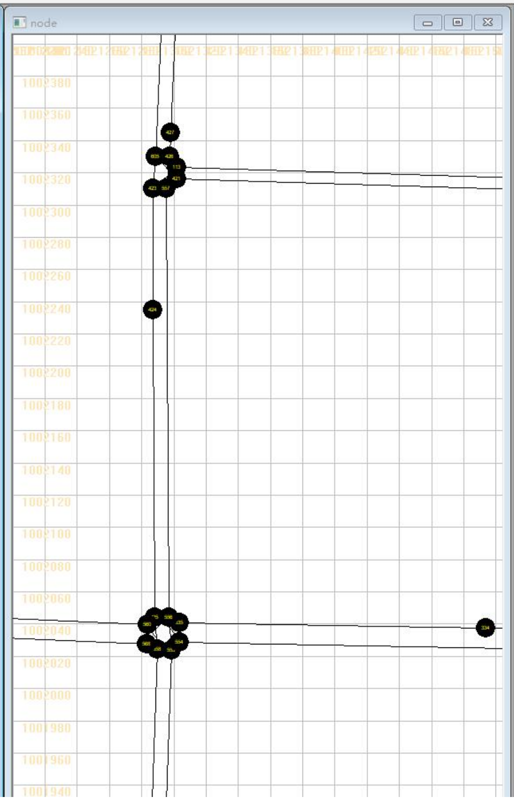
Open-source Nexta AMS data hub for GMNS format

D. Macroscopic network → Mesoscopic network →Microscopic network

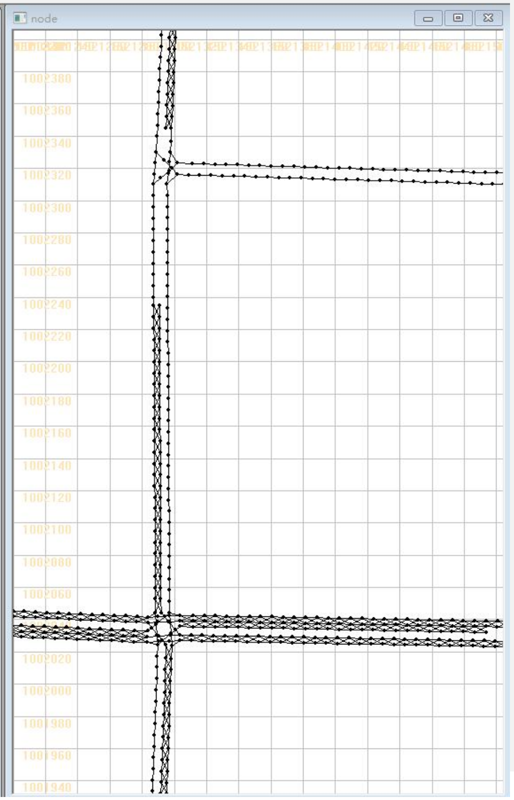
Macroscopic Network



Mesoscopic Network



Microscopic Network



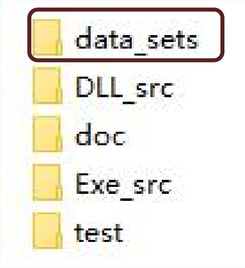
Microscopic network: cell-based network (5 meters or 15 feet per cell)

4. Source codes of STAlite

Dataset

STAlite foleder include: data_sets, DLL_src,

Data_sets is the folder with input data (GMNS format)

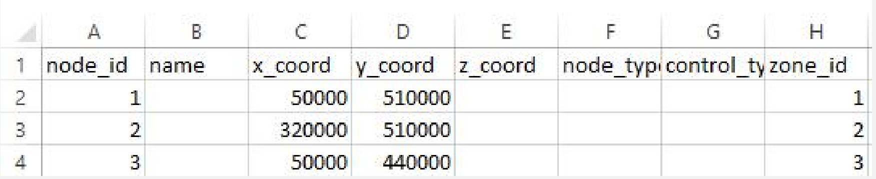


data_sets
DLL_src
doc
Exe_src
test

Network information

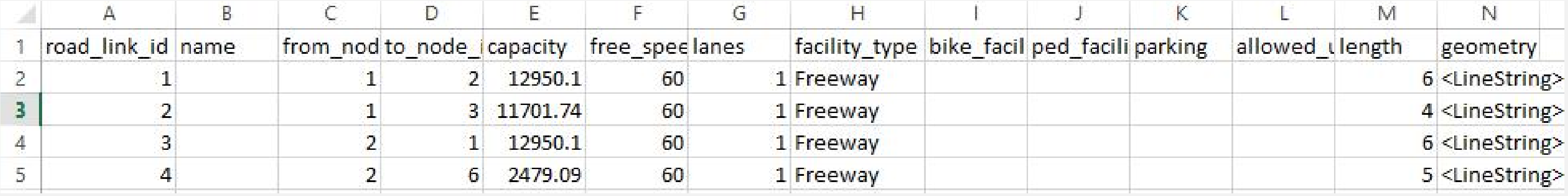
node.csv
road_link.csv

node.csv



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

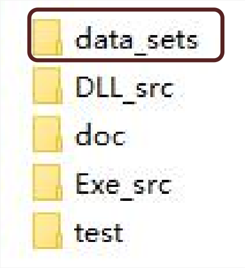
road_link.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	road_link_id	name	from_nod	to_node_i	capacity	free_spee	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>

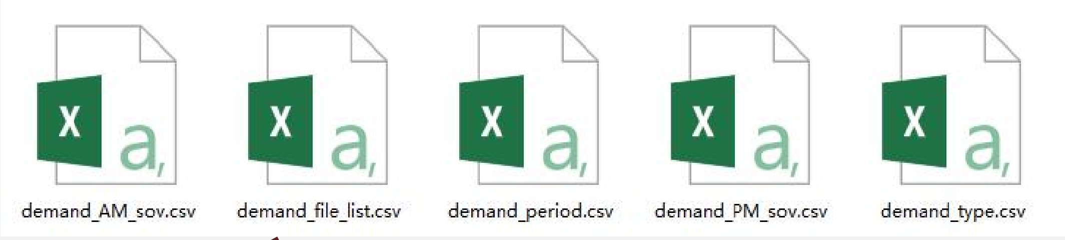
Dataset (cont'd)

Data_sets is the folder with input data (GMNS format)

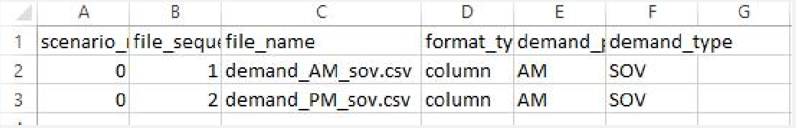


data_sets
DLL_src
doc
Exe_src
test

Demand files

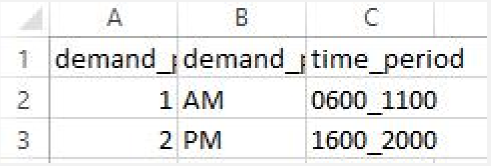


demand_AM_sov.csv
demand_file_list.csv
demand_period.csv
demand_PM_sov.csv
demand_type.csv

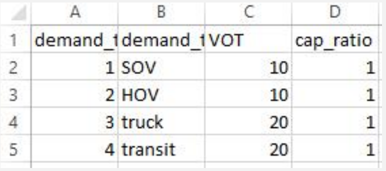


scenario_i	file_seque	file_name	format_ty	demand_p	demand_type
0	1	demand_AM_sov.csv	column	AM	SOV
0	2	demand_PM_sov.csv	column	AM	SOV

Managing demand files for different time periods



demand_p	demand_p	time_period
1	AM	0600_1100
2	PM	1600_2000



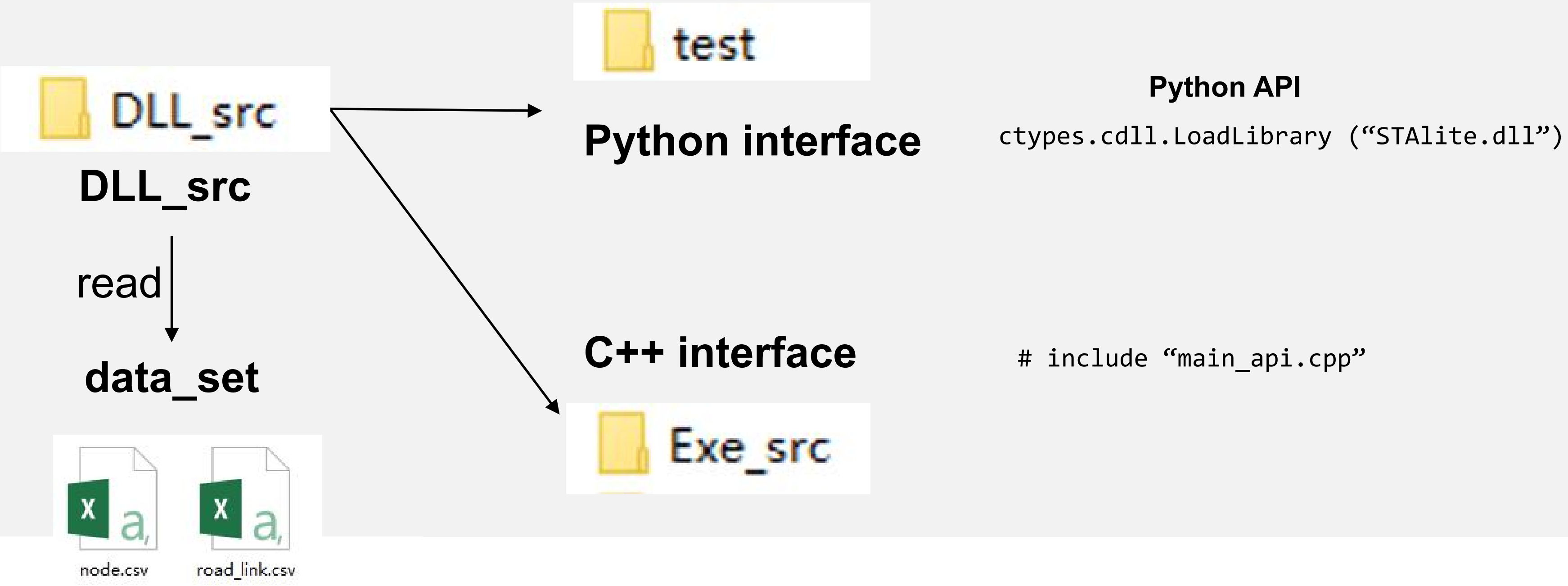
demand_t	demand_t	VOT	cap_ratio
1	SOV	10	1
2	HOV	10	1
3	truck	20	1
4	transit	20	1

- 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



DLL_src

read

data_set

Python interface

Python API

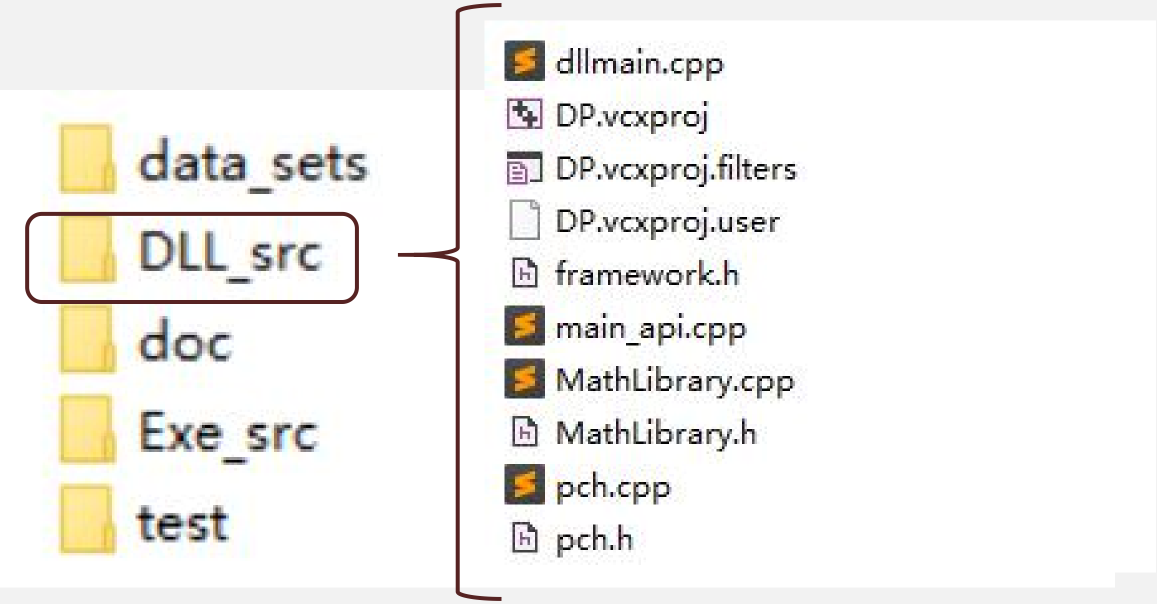
```
ctypes.cdll.LoadLibrary (“STAlite.dll”)
```

C++ interface

```
# include “main_api.cpp”
```

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

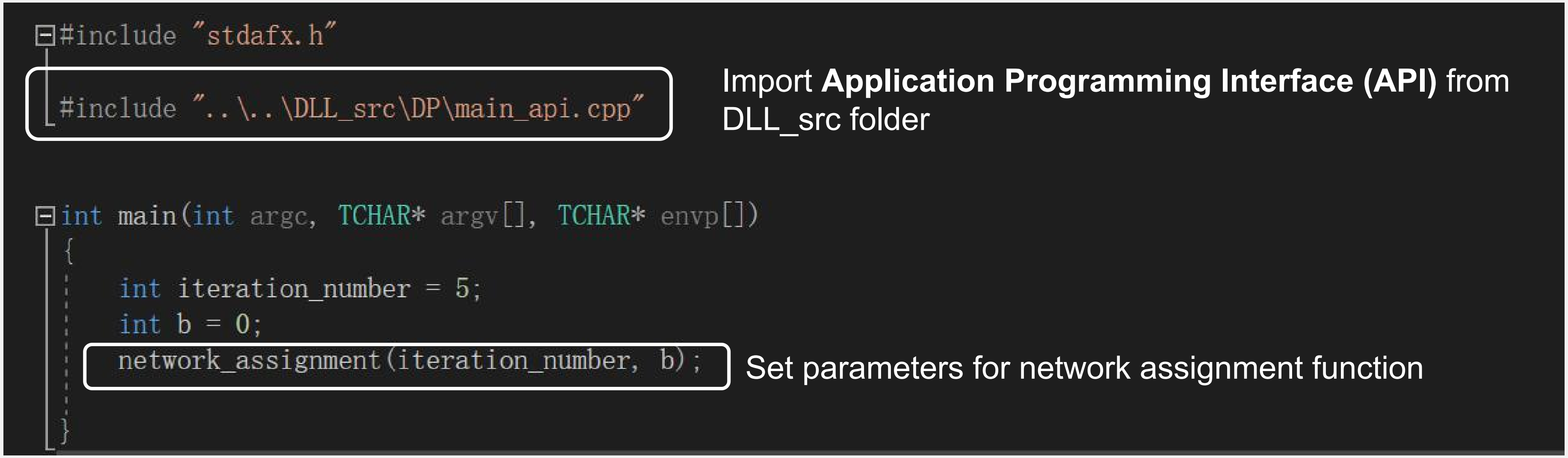


- data_sets
- DLL_src
- doc
- Exe_src
- test

- dllmain.cpp
- DP.vcxproj
- DP.vcxproj.filters
- DP.vcxproj.user
- framework.h
- main_api.cpp
- MathLibrary.cpp
- MathLibrary.h
- pch.cpp
- pch.h

Source code for building STALite Executable

Exe_src includes executable console of C++ to run STAlite



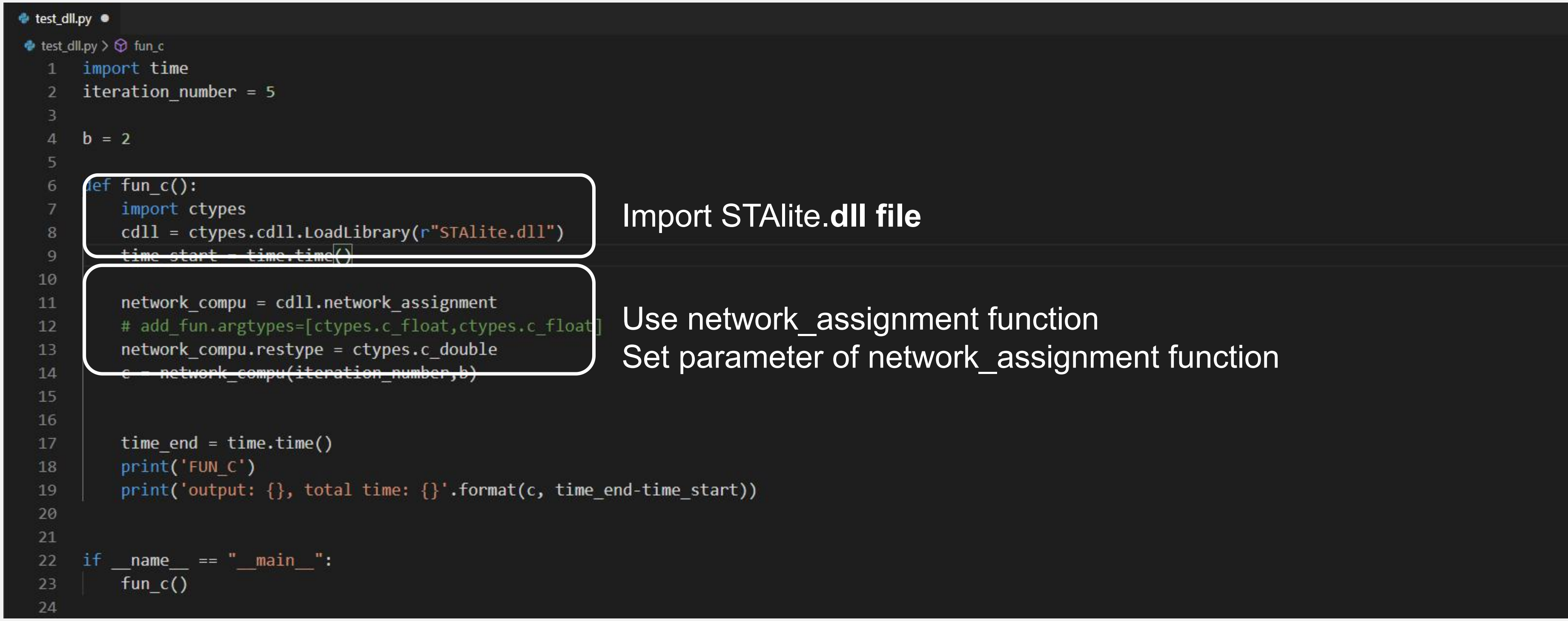
```
#include "stdafx.h"

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


int main(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int iteration_number = 5;
    int b = 0;
    network_assignment(iteration_number, b);

}
```

Import **Application Programming Interface (API)** from DLL_src folder

Set parameters for network assignment function

Python API test environment

test includes executable console Python to run STAlite



```python
import time
iteration_number = 5

b = 2

def fun_c():
    import ctypes
    cdll = ctypes.cdll.LoadLibrary(r"STAlite.dll")
    time_start = time.time()

    network_compu = cdll.network_assignment
    # add_fun.argtypes=[ctypes.c_float,ctypes.c_float]
    network_compu.restype = ctypes.c_double
    c = network_compu(iteration_number,b)


    time_end = time.time()
    print('FUN_C')
    print('output: {}, total time: {}'.format(c, time_end-time_start))


if __name__ == "__main__":
    fun_c()
```

Import STAlite.**dll file**

Use network_assignment function

Set parameter of network_assignment function

THANKS