DTALite Tutorial, Version 1.0

DTALite Team

June 2, 2024

1 Introduction: Why Use DTALite?

[label=1.]Bridging the Gap Between Static and Dynamic Traffic Assignment Traditional traffic modeling tools primarily focus on static traffic assignment (STA). While these tools are powerful, they often lack the ability to accurately model the temporal dynamics of traffic congestion. DTALite bridges this gap by integrating both static and dynamic traffic assignment (DTA) models. This dual capability allows you to:

1.
   - Capture Traffic Dynamics: Understand how congestion forms, propagates, and dissipates over time.
   - Analyze Time-Dependent Demands: Assess operational performance across different times of the day, not just peak periods.

2. Accessible and Transparent Modeling

   Many commercial tools present barriers such as high costs, complex setup processes, and opaque modeling approaches. DTALite addresses these issues by offering:

   - Open-Source Accessibility: Free to use and modify, promoting transparency and collaboration.
   - Ease of Setup: Designed to be user-friendly, reducing the time and resources needed for model setup.

3. Integrated with the General Modeling Network Specification (GMNS)

   DTALite adopts the GMNS, a flexible and efficient open network standard developed by the Zephyr Foundation. This integration allows for:

   - Standardized Data Handling: Easier data exchange and model interoperability.
   - Broad Support: Enhanced compatibility with other transportation modeling tools.

4. Comprehensive Graphical User Interface (GUI)

   DTALite is paired with NeXTA (Network eXplorer for Traffic Analysis), an open-source GUI that offers:

   - Intuitive Visualization: Visualize traffic flow dynamics and route choice behavior in an integrated 2D environment.
   - Educational Value: An excellent tool for teaching transportation planning and optimization concepts.

5. Leveraging Parallel Computing

   Modern laptops and desktops come equipped with multi-core processors. DTALite is designed to take full advantage of these capabilities, providing:

   - Faster Processing: Significantly reduced computation times through parallel processing.
   - Efficient Resource Use: Optimal use of available computing power, making high-fidelity simulations feasible on standard hardware.

2 GMNS format

This tutorial provides a step-by-step guide on how to configure the `node.csv` and `link.csv` files for the General Modeling Network Specification (GMNS). These files are essential for setting up a traffic assignment model.

2.1 `node.csv` File

The `node.csv` file defines the nodes in the network, which represent intersections or points where links (roads) connect. Each row in the file corresponds to a node with specific attributes.

2.1.1 Fields in `node.csv`

- **node_id**: Unique identifier for each node.
- **x_coord**: X-coordinate of the node (e.g., longitude or distance).
- **y_coord**: Y-coordinate of the node (e.g., latitude or distance).
- **zone_id**: Zone identifier where the node is located. Zones are often used to group nodes into larger geographic areas.

2.2 Example of `node.csv`

```
node_id,x_coord,y_coord,zone_id
1,0,-0.1,1
2,40.3,0.1,2
3,19.8,14.8,0
4,19.7,-9.7,0
```

2.3 Explanation of the Example

- **node_id**: Unique identifiers 1, 2, 3, and 4. - **x_coord** and **y_coord**: Coordinates of each node. - **zone_id**: Zone identifiers where each node is located. Node 1 is in zone 1, node 2 is in zone 2, and nodes 3 and 4 are not assigned to any specific zone (zone 0).

2.4 `link.csv` File

The `link.csv` file defines the links in the network, which represent the roads connecting nodes. Each row in the file corresponds to a link with specific attributes.

2.4.1 Fields in `link.csv`

- **from_node_id**: The starting node of the link.
- **to_node_id**: The ending node of the link.
- **length**: The length of the link (in miles or kilometers, as specified in `settings.yml`).
- **lanes**: The number of lanes on the link.
- **free_speed**: The free-flow speed on the link (in mph or kph, as specified in `settings.yml`).
- **capacity**: The capacity of the link per lane (vehicles per hour).
- **link_type**: The type of link (e.g., freeway, arterial).
- **VDF_alpha**: The alpha parameter for the Volume-Delay Function (VDF).
- **VDF_beta**: The beta parameter for the VDF.
- **VDF_fftt**: The free-flow travel time in minutes.
- **VDF_plf**: Peak Load Factor (PLF) for the VDF, set to 1.0 by default.

2.5 Example of `link.csv`

```
from_node_id,to_node_id,length,lanes,free_speed,capacity,link_type,VDF_alpha,VDF_beta,VDF_f
1,4,15,1,60,3000,2,0.15,4,,1
4,2,15,1,60,3000,2,0.15,4,,1
1,3,10,1,60,4000,1,0.15,4,,1
3,2,10,1,60,4000,1,0.15,4,,1
```

2.6 Explanation of the Example

- **from_node_id, to_node_id**: Defines the direction of the link.
- **length**: Length of the link in miles (as specified in `settings.yml`).
- **lanes**: Number of lanes on the link. Each link in this example has 1 lane.
- **free_speed**: Free-flow speed on the link in mph (as specified in `settings.yml`).
- **capacity**: Capacity of the link per lane in vehicles per hour.
- **link_type**: Type of link (e.g., 1 for motorway, 2 for highway/expressway).
- **VDF_alpha, VDF_beta**: Parameters for the Volume-Delay Function (VDF).
- **VDF_fftt**: Free-flow travel time in minutes.
- **VDF_plf**: Peak load factor, set to 1.0 by default.

This tutorial provides a comprehensive guide to configuring and understanding the `node.csv` and `link.csv` files for the GMNS. By following the instructions and examples provided, you can effectively set up and customize your traffic simulation network.

2.7 `demand.csv` File

The `demand.csv` file defines the demand between origin and destination zones in the network. Each row in the file corresponds to a specific demand between an origin zone and a destination zone.

2.8 Fields in `demand.csv`

- **o_zone_id**: Origin zone identifier. This specifies the starting zone for the demand.
- **d_zone_id**: Destination zone identifier. This specifies the ending zone for the demand.
- **volume**: Volume of demand between the origin and destination zones. This is typically measured in vehicles or passengers.

2.9 Example of `demand.csv`

```
o_zone_id,d_zone_id,volume
1,1,0
1,2,800
```

```
\section{YAML Configuration File}

The \texttt{settings.yml} file contains various settings and parameters required for running

\subsection{Assignment Settings}
\begin{verbatim}
assignment:
  number_of_iterations: 2
  route_output: 1
  simulation_output: 0
  number_of_cpu_processors: 4
  length_unit: mile
  speed_unit: mph
  UE_convergence_percentage: 0.001
  odme_activate: 0
```

- **number_of_iterations**: Number of iterations for the traffic assignment process.
- **route_output**: Enable (1) or disable (0) route information output.
- **simulation_output**: Enable (1) or disable (0) simulation results output.
- **number_of_cpu_processors**: Number of CPU processors for parallel processing.
- **length_unit**: Unit of length used in the simulation (mile/km).
- **speed_unit**: Unit of speed used in the simulation (mph/kph).
- **UE_convergence_percentage**: Convergence threshold for User Equilibrium (UE).
- **odme_activate**: Activate (1) or deactivate (0) Origin-Destination Matrix Estimation (ODME).

2.10 Mode Types

```
mode_types:
  - mode_type: auto
    vot: 10
    person_occupancy: 1
```

```
    pce: 1
  - mode_type: hov
    vot: 20
    person_occupancy: 2
    pce: 1
  - mode_type: truck
    vot: 10
    person_occupancy: 1
    pce: 2
```

- **mode_type**: Type of vehicle (auto, hov, truck).
- **vot**: Value of time in monetary units.
- **person_occupancy**: Average number of occupants per vehicle.
- **pce**: Passenger car equivalent.

2.11 Demand Periods

```
demand_periods:
  - period: AM
    time_period: 0700_0800
```

- **period**: Label for the time period.
- **time_period**: Specific time range for the period.

2.12 Demand Files

```
demand_files:
  - file_sequence_no: 1
    file_name: demand.csv
    demand_period: am
    mode_type: auto
    format_type: column
    scale_factor: 1
    departure_time_profile_no: 1
```

- **file_sequence_no**: Sequence number of the demand file.
- **file_name**: Name of the demand file.
- **demand_period**: Corresponding demand period.
- **mode_type**: Mode type for the demand.
- **format_type**: Format of the demand file.
- **scale_factor**: Scaling factor for the demand data.
- **departure_time_profile_no**: Reference to the departure time profile.

2.13 Subarea

```
subarea:
  - activate: 0
    subarea_geometry: POLYGON ((-180 -90, 180 -90, 180 90, -180 90, -180 -90))
```

- **activate**: Activate (1) or deactivate (0) subarea analysis.
- **subarea_geometry**: Geometry defining the subarea.

2.14 Link Types

```
link_types:
  - link_type: 1
    link_type_name: motorway
    type_code: f
    traffic_flow_model: kw
    allowed_uses_p1: null
    allowed_uses_p2: null
    allowed_uses_p3: null
    k_jam_km: 300
    emissions_auto_co2: 20785.99541;0.0002;0.0042;0.3412
    emissions_auto_nox: 5.53516;0.0003;0.0043;0.0959
    emissions_hov_co2: 10392.99771;0.0002;0.0042;0.3412
    emissions_hov_nox: 2;0.0003;0.0043;0.0959
    emissions_truck_co2: 23816.14583;0.0002;0.0042;0.3412
    emissions_truck_nox: 6.342370833;0.0003;0.0043;0.0959
    emissions_bus_co2: 25115.20833;0.0002;0.0042;0.3412
    emissions_bus_nox: 6.688318333;0.0003;0.0043;0.0959
    emissions_ev_co2: 0;0;0;0
    emissions_ev_nox: 0;0;0;0
  - link_type: 2
    link_type_name: Highway/Expressway
    type_code: f
  - link_type: 3
    link_type_name: Collector
    type_code: a
  - link_type: 6
    link_type_name: Major arterial
    type_code: a
  - link_type: 7
    link_type_name: Ramp
    type_code: f
  - link_type: 8
    link_type_name: Ramp
    type_code: f
  - link_type: 5
```

```
  link_type_name: Zonal connector
  type_code: c
```

- **link_type**: Unique identifier for the link type.
- **link_type_name**: Name of the link type.
- **type_code**: Code representing the type of link (f for freeway, a for arterial streets, c for zone connectors).
- **traffic_flow_model**: Traffic flow model used (kw for kinematic wave).
- **k_jam_km**: Jam density in vehicles per kilometer.
- **emissions_auto_co2, emissions_auto_nox, etc.**: Emissions factors for different vehicle types.

2.15 Departure Time Profile

```
departure_time_profile:
  - departure_time_profile_no: 1
    time_period: 0700_0800
    T0420: 0.005002
    T0425: 0.005002
    T0430: 0.005002
    T0435: 0.005207
    T0440: 0.005207
    T0445: 0.005207
    T0450: 0.005677
    T0455: 0.005677
    T0460: 0.005677
    T0465: 0.005994
    T0470: 0.005994
    T0475: 0.005994
    T0480: 0.006018
```

- **departure_time_profile_no**: Identifier for the departure time profile.
- **time_period**: Time period for the departure profile.
- **T0420, T0425, T0430, etc.**: Departure rates at specific time intervals.

3 Python Script for Generating settings.yml

The following Python script generates the `settings.yml` file based on various input structures.

```python
import yaml

def save_settings_yml(filename, assignment_settings, mode_types, demand_periods, demand_files, subarea, link_types, departure_time_profiles):
    settings = {
        'assignment': assignment_settings,
        'mode_types': mode_types,
        'demand_periods': demand_periods,
        'demand_files': demand_files,
        'subarea': subarea,
        'link_types': link_types,
        'departure_time_profile': departure_time_profiles
    }
    with open(filename, 'w') as file:
        yaml.dump(settings, file)

# Example usage
assignment_settings = {
    'number_of_iterations': 2,
    'route_output': 1,
    'simulation_output': 0,
    'number_of_cpu_processors': 4,
    'length_unit': 'mile',
    'speed_unit': 'mph',
    'UE_convergence_percentage': 0.001,
    'odme_activate': 0
}

mode_types = [
    {'mode_type': 'auto', 'vot': 10, 'person_occupancy': 1, 'pce': 1},
    {'mode_type': 'hov', 'vot': 20, 'person_occupancy': 2, 'pce': 1},
    {'mode_type': 'truck', 'vot': 10, 'person_occupancy': 1, 'pce': 2}
]

demand_periods = [
    {'period': 'AM', 'time_period': '0700_0800'}
]

demand_files = [
    {'file_sequence_no': 1, 'file_name': 'demand.csv', 'demand_period': 'am', 'mode_type': 'auto', 'format_type': 'column', 'scale_factor': 1, 'departure_time_profile_no': 1}
]

subarea = [
    {'activate': 0, 'subarea_geometry': 'POLYGON ((-180 -90, 180 -90, 180 90, -180 90, -180 -90))'}
]

link_types = [
    {'link_type': 1, 'link_type_name': 'motorway', 'type_code': 'f', 'traffic_flow_model': 'kw', 'allowed_uses_p1': None, 'allowed_uses_p2': None, 'allowed_uses_p3': None, 'k_jam_km': 300, 'emissions_auto_co2': '20785.99541;0.0002;0.0042;0.3412', 'emissions_auto_nox': '5.53516;0.0003;0.0043;0.0959', 'emissions_hov_co2': '10392.99771;0.0002;0.0042;0.3412', 'emissions_hov_nox': '2;0.0003;0.0043;0.0959', 'emissions_truck_co2': '
```

```
    23816.14583;0.0002;0.0042;0.3412', 'emissions_truck_nox': '
    6.342370833;0.0003;0.0043;0.0959', 'emissions_bus_co2': '
    25115.20833;0.0002;0.0042;0.3412', 'emissions_bus_nox': '
    6.688318333;0.0003;0.0043;0.0959', 'emissions_ev_co2': '0;0;0;0', '
    emissions_ev_nox': '0;0;0;0'},
    {'link_type': 2, 'link_type_name': 'Highway/Expressway', 'type_code': 'f'},
    {'link_type': 3, 'link_type_name': 'Collector', 'type_code': 'a'},
    {'link_type': 6, 'link_type_name': 'Major arterial', 'type_code': 'a'},
    {'link_type': 7, 'link_type_name': 'Ramp', 'type_code': 'f'},
    {'link_type': 8, 'link_type_name': 'Ramp', 'type_code': 'f'},
    {'link_type': 5, 'link_type_name': 'Zonal connector', 'type_code': 'c'}  #
    Note: Zone connectors should be marked as 'c' to prevent through traffic. 'f
    ' is for freeways, and 'a' is for arterial streets. Freeways use the
    kinematic wave model; arterial streets use simple point queue or spatial
    queue models based on k_jam_km.
]

departure_time_profiles = [
    {'departure_time_profile_no': 1, 'time_period': '0700_0800', 'T0420':
    0.005002, 'T0425': 0.005002, 'T0430': 0.005002, 'T0435': 0.005207, 'T0440':
    0.005207, 'T0445': 0.005207, 'T0450': 0.005677, 'T0455': 0.005677, 'T0460':
    0.005677, 'T0465': 0.005994, 'T0470': 0.005994, 'T0475': 0.005994, 'T0480':
    0.006018}
]

save_settings_yml('settings.yml', assignment_settings, mode_types, demand_periods
    , demand_files, subarea, link_types, departure_time_profiles)
```

This tutorial provides a comprehensive guide to configuring and generating the `settings.yml` file for the DTALite package. By following the instructions and using the provided Python script, you can easily set up and customize your traffic simulation settings.

4 Learning Outcomes of User Equilibrium

By the end of this chapter, students will be able to:

- Understand the basic concept of User Equilibrium in network modeling.
- Prepare and manipulate input files for traffic assignment using DTALite.
- Visualize and validate network connectivity using NeXTA.
- Perform traffic assignment and analyze the results.
- Apply basic Intelligent Transportation Systems (ITS) concepts in network modeling.

4.1 Introduction

This chapter uses the 01_two_corridor as an example to introduce the basic content and usage of DTALite. This example employs a simple case with a single origin-to-destination pair and two paths: $p = 1$ for the primary path

(link_type=Freeway), and $p = 2$ for the alternative path (link_type=Arterial). Each path consists of two links. Path 1 has a free-flow travel time of 20 minutes, while Path 2 has a free-flow travel time of 30 minutes.

4.2 Real-world Scenario: Desert City

Consider a city called 'Desert City', which is divided into different districts. There is a traffic flow demand between two districts: 'Downtown' and 'Marketplace'. These districts are connected by an arterial road and a freeway. The arterial road passes through a district called 'Suburbs', and the freeway passes through a district called 'Steel Plant'. Note that there is no production or attraction in the 'Suburbs' and 'Steel Plant' districts; the links merely pass through them in this scenario.

4.3 Step-by-Step Guide

4.3.1 Step 1: Download and Locate the Project Folder

First, download and unzip the released packages from GitHub. Locate the project folder of "01_two_corridor". Typically, copy DTALite.exe and NeXTA.exe into the same folder for easy access. The list of data files is as follows:

4.3.2 Step 2: Prepare Input Files

Users can prepare their own input files according to the data structure described in Section 3.1, which mainly consists of network files, demand files, and configuration files. For example, the core input files for 01_two_corridor are as follows:

node.csv

```
node_id,x_coord,y_coord,zone_id
1,0,-0.1,1
2,40.3,0.1,2
3,19.8,14.8,0
4,19.7,-9.7,0
```

link.csv

```
from_node_id,to_node_id,length,lanes,free_speed,capacity,link_type,VDF_alpha,VDF_beta,VDF_f
1,4,15,1,60,3000,2,0.15,4,,1
4,2,15,1,60,3000,2,0.15,4,,1
1,3,10,1,60,4000,1,0.15,4,,1
3,2,10,1,60,4000,1,0.15,4,,1
```

demand.csv

```
o_zone_id,d_zone_id,volume
1,1,0
1,2,8000
2,1,0
2,2,0
```

settings.yml

```
assignment:
  number_of_iterations: 10
  route_output: 1
  simulation_output: 0
  number_of_cpu_processors: 4
  length_unit: mile
  speed_unit: mph
  UE_convergence_percentage: 0.001
  odme_activate: 0
```

4.4 Step 3: Visualize and Validate Network in NeXTA

Open NeXTA and navigate to "File" ¿ "Open Traffic Network Project" to select the node.csv file in your network data set. Check the network connectivity by calculating a simple path for an OD pair.

4.4.1 Example: Checking Connectivity

1. Select the node layer in the left-hand-side GIS panel and use the mouse to select node 1 and node 2. Alternatively, use the keyboard shortcut Control+F to search for these nodes.

2. Go to the path GIS layer, select node 1, right-click and choose "Check Connectivity from here".

3. Then select node 2, right-click and choose "Direction to here".

4. Alternatively, use the Control+F shortcut to specify the origin and destination for the path. Enter the ID of the origin node and the destination node separated by spaces, and click the "find" button to find the corresponding path.

4.5 Step 4: Run DTALite as a Windows Console Application

Place the executable of "DTALite.exe" with the input files in the same folder and click on it to perform traffic assignment and simulation.

For an OD demand of 7,000 on this network, the User Equilibrium method can be used to perform traffic assignment. As the path flow changes, the travel time on the two paths reaches the same equilibrium point, satisfying the requirement of User Equilibrium. The equilibrium solution is reached when the freeway flow is 5400 and the arterial flow is 1600 (7000-5400), leading to the same travel time of 30 minutes on both routes.

Parameters	Value
Freeway flow travel time (min): Freeway	20
Freeway flow travel time (min): Arterial	30
Capacity (vehicles/hour): Freeway	4000
Capacity (vehicles/hour): Arterial	3000
Demand	7000
BPR alpha	0.15
BPR beta	4

Table 1: Parameters for the User Equilibrium Example

4.6 Parameters and Calculations

The travel time function is:

- **Freeway_TT** = FFTT[1 + 0.15(v/c)$\hat{4}$]
- **Arterial_TT** = FFTT[1 + 0.15((demand-v)/c)$\hat{4}$]

where:

- **TT** = link travel time
- **FFTT** = free-flow travel time of the link
- **v** = link flow
- **c** = link capacity

4.7 Step 5: Visualize Output Files in NeXTA

The user can check the output files in Excel/CSV format for various performance summaries.

4.7.1 Visualization Types

- **Volume/Capacity (V/C) Ratio Visualization**
- **Volume Visualization**
- **Speed Visualization**

To view these visualizations:

1. Open NeXTA and import the network.
2. Choose the time period set in demand_period.csv.
3. Click the corresponding button (Volume/Capacity, Volume, Speed) to see the assignment outcome.

This chapter introduces the basic concepts of User Equilibrium using a simple example, guiding you through downloading, preparing, visualizing, and running a traffic assignment simulation with DTALite.

5 Introduction to User Equilibrium using 01_two_corridor Example

5.1 Learning Outcomes

By the end of this chapter, students will be able to:

- Understand the basic concept of User Equilibrium in network modeling.
- Prepare and manipulate input files for traffic assignment using DTALite.
- Visualize and validate network connectivity using NeXTA.
- Perform traffic assignment and analyze the results.
- Apply basic Intelligent Transportation Systems (ITS) concepts in network modeling.

5.2 Overview

This chapter uses the 01_two_corridor as an example to introduce the basic content and usage of DTALite. This example employs a simple case with a single origin-to-destination pair and two paths: $p = 1$ for the primary path (link_type=Freeway), and $p = 2$ for the alternative path (link_type=Arterial). Each path consists of two links. Path 1 has a free-flow travel time of 20 minutes, while Path 2 has a free-flow travel time of 30 minutes.

5.3 Real-world Scenario: Desert City

Consider a city called 'Desert City', which is divided into different districts. There is a traffic flow demand between two districts: 'Downtown' and 'Marketplace'. These districts are connected by an arterial road and a freeway. The arterial road passes through a district called 'Suburbs', and the freeway passes through a district called 'Steel Plant'. Note that there is no production or attraction in the 'Suburbs' and 'Steel Plant' districts; the links merely pass through them in this scenario.

5.4 Step-by-Step Guide

5.4.1 Step 1: Download and Locate the Project Folder

First, download and unzip the released packages from GitHub. Locate the project folder of "01_two_corridor". Typically, copy DTALite.exe and NeXTA.exe into the same folder for easy access. The list of data files is as follows:

5.4.2 Step 2: Prepare Input Files

Users can prepare their own input files according to the data structure described in Section 3.1, which mainly consists of network files, demand files, and configuration files. For example, the core input files for 01_two_corridor are as follows:

node.csv

```
node_id,x_coord,y_coord,zone_id
1,0,-0.1,1
2,40.3,0.1,2
3,19.8,14.8,0
4,19.7,-9.7,0
```

link.csv

```
from_node_id,to_node_id,length,lanes,free_speed,capacity,link_type,VDF_alpha,VDF_beta,VDF_f
1,4,15,1,60,3000,2,0.15,4,,1
4,2,15,1,60,3000,2,0.15,4,,1
1,3,10,1,60,4000,1,0.15,4,,1
3,2,10,1,60,4000,1,0.15,4,,1
```

demand.csv

```
o_zone_id,d_zone_id,volume
1,1,0
1,2,8000
2,1,0
2,2,0
```

settings.yml

```
assignment:
  number_of_iterations: 10
  route_output: 1
  simulation_output: 0
  number_of_cpu_processors: 4
  length_unit: mile
  speed_unit: mph
  UE_convergence_percentage: 0.001
  odme_activate: 0
```

5.4.3 Step 3: Visualize and Validate Network in NeXTA

Open NeXTA and navigate to "File" ¿ "Open Traffic Network Project" to select the node.csv file in your network data set. Check the network connectivity by calculating a simple path for an OD pair.

Example: Checking Connectivity

1. Select the node layer in the left-hand-side GIS panel and use the mouse to select node 1 and node 2. Alternatively, use the keyboard shortcut Control+F to search for these nodes.

2. Go to the path GIS layer, select node 1, right-click and choose "Check Connectivity from here".

3. Then select node 2, right-click and choose "Direction to here".

4. Alternatively, use the Control+F shortcut to specify the origin and destination for the path. Enter the ID of the origin node and the destination node separated by spaces, and click the "find" button to find the corresponding path.

5.4.4 Step 4: Run DTALite as a Windows Console Application

Place the executable of "DTALite.exe" with the input files in the same folder and click on it to perform traffic assignment and simulation.

For an OD demand of 7,000 on this network, the User Equilibrium method can be used to perform traffic assignment. As the path flow changes, the travel time on the two paths reaches the same equilibrium point, satisfying the requirement of User Equilibrium. The equilibrium solution is reached when the freeway flow is 5400 and the arterial flow is 1600 (7000-5400), leading to the same travel time of 30 minutes on both routes.

Parameters	**Value**
Freeway flow travel time (min): Freeway	20
Freeway flow travel time (min): Arterial	30
Capacity (vehicles/hour): Freeway	4000
Capacity (vehicles/hour): Arterial	3000
Demand	7000
BPR alpha	0.15
BPR beta	4

Table 2: Parameters for the User Equilibrium Example

Parameters and Calculations The travel time function is:

- **Freeway_TT** = FFTT[1 + 0.15(v/c)$\hat{4}$]
- **Arterial_TT** = FFTT[1 + 0.15((demand-v)/c)$\hat{4}$]

where:

- **TT** = link travel time
- **FFTT** = free-flow travel time of the link
- **v** = link flow
- **c** = link capacity

5.4.5 Step 5: Visualize Output Files in NeXTA

The user can check the output files in Excel/CSV format for various performance summaries.

Visualization Types

- **Volume/Capacity (V/C) Ratio Visualization**
- **Volume Visualization**
- **Speed Visualization**

To view these visualizations:

1. Open NeXTA and import the network.
2. Choose the time period set in demand_period.csv.
3. Click the corresponding button (Volume/Capacity, Volume, Speed) to see the assignment outcome.

5.5 Appendix A: From Mathematical Modeling to Network-Based Assignment and Simulation

5.5.1 Link Volume-Delay Function in Static Traffic Assignment

There are a number of key components for the static traffic assignment procedure:

- Input trip table describes the flow per hour from each origin zone to each destination zone.
- A traffic network consisting of nodes, links, and link volume-delay functions.
- Volume-delay function such as BPR (Bureau‘‘latex of Public Roads) relationship that shows increased link travel time as an increase in link volume:

$$\mathrm{TT} = \mathrm{FFTT}\left[1 + 0.15\left(\frac{v}{c}\right)^4\right]$$

where:

- **TT** = link travel time
- **FFTT** = free-flow travel time of the link
- **v** = link flow
- **c** = link capacity

Remark: The link travel time function typically depends only on its own flow, while ignoring link volume on opposing or conflicting directions. The link capacity might not be a strict upper limit on flow, e.g., specified by the Highway Capacity Manual.

As one of the simplest cases of behavior, the User Equilibrium (UE) Principle assumes users are "greedy" and familiar with the system. Equilibrium requires iteration to reach the following two principles:

- **Principle A**: No individual trip maker can reduce their path costs by switching routes.
- **Principle B**: All used routes between an O-D pair have equal and minimum costs, while all unused routes have greater or equal costs (compared to the used path costs).

Wardrop (1952) proposed the user equilibrium and system optimal principles of route choice behavior in his seminal paper, and Beckman et al. (1956) formulated the static user equilibrium traffic assignment problem as an equivalent convex mathematical programming problem. Since their influential contributions, the development of static network assignment formulations, algorithms, and applications have made remarkable progress. The books by Sheffi (1985) and Patriksson (1994) provide the most comprehensive coverage on the static traffic assignment problem and its variants.

5.5.2 General Mathematical Descriptions of Traffic Assignment

Traffic assignment loads an origin-destination (OD) trip matrix onto links of a traffic network, while satisfying a certain route choice behavioral model, e.g., deterministic user equilibrium. Traffic assignment is used to predict/estimate how trip-makers may shift to other routes or departure times in response to various strategies such as road pricing, incidents, road capacity improvement, and traffic signal re-timing.

For example, tolling typically leads to traffic diversion on alternative routes and/or other transportation modes, and many traffic congestion mitigation strategies should be developed to improve the capacity to which the traffic may be diverted, for example, signal optimization, traveler information provision, and transit operation.

The common time periods include morning peak, afternoon peak, and off-peak, and we can use the time of day factor to calculate the trip in the peak hour (e.g., morning peak may be 11

By using a simplified static traffic assignment formulation, the following mathematical description adopts the related sections in the paper titled "Equivalent Gap Function-Based Reformulation and Solution Algorithm for the Dynamic User Equilibrium Problem" by Lu, Mahmassani, and Zhou (2009). One can consider the extended DTA formulation by adding a time index dimension.

Consider a network $G = (N, A)$, where N is a finite set of nodes and A is a finite set of directed links (i, j), $i \in N$ and $j \in N$. Associated with each link (i, j)

is the link travel time $s_{ij}(t)$ required to traverse link (i, j) when departing at time interval $t \in S$ from node i. For simplicity and without loss of generality, $s_{ij}(t)$ is regarded as link travel time, though it can be generalized to include travel time, out-of-pocket cost, and other travel impedances that may incur when traversing link (i, j) at time t. Travel time and cost are used interchangeably in this paper. Other important notation and variables are summarized below:

- O: subset of origin nodes; $O \subseteq N$
- D: subset of destination nodes; $D \subseteq N$
- T: set of departure time intervals
- o: subscript for an origin node, $o \in O$
- d: subscript for a destination node, $d \in D$
- $P(o, d)$: set of all feasible paths for a given triplet (o, d)
- p: subscript for a path $p \in P(o, d)$
- q_{od}: number of trips departing from node o to node d
- r_{odp}: number of trips departing from o to d and assigned to path $p \in P(o, d)$
- r: path flow vector, $r = \{r_{odp}, \forall o \in O, d \in D, p \in P(o, d)\}$
- $c_{odp}(r)$: path travel cost (or time) for the travelers departing from o to d and assigned to path $p \in P(o, d)$; $c_{odp}(r) = \sum_{(i,j)\in p} s_{ij}$, and is a function of the path flow vector r
- $c(r)$: vector of path travel costs; $c(r) = \{c_{odp}(r), \forall o \in O, d \in D, p \in P(o, d)\}$

The OD demand pattern for the entire planning horizon (i.e., $q_{od}, \forall o, d$) is assumed to be known a priori. The key behavioral assumption for the path choice decision is as follows: in a disutility-minimization framework, each trip-maker is rational and chooses a path that minimizes the travel cost. Specifically, for each trip-maker in (o, d), a path $p^* \in P(o, d)$ will be selected if and only if $c_{odp}(r) = \min_{p\in P(o,d)} c_{odp}(r)$.

Given the assumptions above, the problem is to solve the UE traffic assignment problem, with a given OD demand, to obtain a path flow pattern satisfying the UE conditions. Specifically, the goal is to determine a UE path flow vector (routing policies) over a vehicular network for each OD pair and each departure time interval (i.e., $r^* \equiv \{r^*_{odp}, \forall o, d, p \in P(o, d)\}$).

By the above UE definition, all trips in a network are equilibrated in terms of actual experienced path costs, so it is necessary to determine the experienced path costs $c(r)$ for a given path flow vector r. To this end, a simulation-based dynamic traffic (or network loading) model is used to obtain the experienced

path cost vector. It should be noted that the algorithm is independent of the specific dynamic traffic model selected; any (macroscopic, microscopic, or mesoscopic) dynamic traffic model capable of capturing complex traffic flow dynamics, in particular the effect of physical queuing, as well as preventing violations of the first-in-first-out property, can be embedded into the proposed solution algorithm.

With the introduction of the gap function $\text{Gap}(r, \pi)$, the proposed nonlinear minimization problem (NMP) is presented as the following:

$$\min_{r,\pi} \sum_{o \in O} \sum_{d \in D} \sum_{\tau \in T} \sum_{p \in P(o,d)} r_{odp}[c_{odp}(r) - \pi_{od}]$$

subject to:

$$\sum_{p \in P(o,d)} r_{odp} = q_{od}, \forall o, d$$

$$c_{odp}(r) - \pi_{od} \geq 0, \forall o, d, p \in P(o, d)$$

$$r_{odp} \geq 0, \forall o, d, p \in P(o, d)$$

In the above NMP reformulation, both π and r are independent decision variables and hence the gap function is a function of both r and π (i.e., $\text{Gap}(r, \pi)$), where π and r are connected with each other through inequality constraint (3). $\text{Gap}(r, \pi)$ provides a measure of the violation of the UE conditions in terms of the difference between the total actual experienced path travel cost and the total shortest path cost evaluated at any given path flow pattern $r \in \Omega$. The difference vanishes when the path flow vector r^* satisfies the UE conditions. Thus, solving the UE problem can be viewed as a process of finding the path flow vector $r^* \in \Omega$ and π^* such that $\text{Gap}(r^*, \pi^*) = 0$.