CAMBRIDGE SYSTEMATICS

Northern Virginia Transportation Model

TransAction Model Version 1.0

User Guide

prepared for

Northern Virginia Transportation Authority (NVTA)

prepared by

Cambridge Systematics, Inc.

In association with

Arizona State University

April 24, 2023

www.camsys.com

user guide

Northern Virginia Transportation Model

TransAction Model Version 1.0

prepared for

Northern Virginia Transportation Authority (NVTA)

prepared by

Cambridge Systematics, Inc.
3 Bethesda Metro Center, Suite 310
Bethesda, MD 20814

In association with

Arizona State University

date

APRIL 24, 2023

Table of Contents

List of Tables

List of Figures

1.0 Model Overview

This User Guide addresses the structure and usage of the Northern Virginia Transportation Model (TransAction Model) Version 1.0. The model is maintained by the Northern Virginia Transportation Authority (NVTA). The TransAction Model supports system planning analyses at a regional level in Northern Virginia. To support the NVTA TransAction Update, the consulting team developed a modeling strategy that would balance tradeoffs between functionality and efficiency, ensure consistency with the TransAction performance measures and consider the ability to build in-house modeling capabilities to improve upon the existing model system.

The modeling system framework is shown in Figure 2.1.2, while Figure 2.1.2 shows the TransAction Model structure and its major components. Some of the key enhancements in this modeling strategy include:

- Integration of the COG/TPB model with a DTALite tool, which is an open-source, queue-based mesoscopic simulation package that provides a simpler, user-friendly, and more economical solution to conducting mesoscopic modeling at a large regional scale.
- New capability to model emerging travel behavior of transportation network company (TNC) travel
- New capability to conduct scenario analysis of travel via connected and automated vehicles (CAVs)
- Updating representation of travel behavior (trip rates and mode choices) reflecting the latest Regional Travel Survey (RTS 2017/8)
- Better representation and simulation of traffic congestion through dynamic traffic assignment (DTA)
- A robust scenario management system with flexibility for users and customized features
- A Modeling Dashboard that facilitates comparisons between scenarios and allows model users to quickly visualize information, with a variety of portable summary reports with a wealth of information about each scenario.
- Enhanced postprocessing utilities that will empower users with analytical capabilities to gain insights from the model results, with specialized module for easy use, such as select link analysis.

The macroscopic model calibration and validation were conducted using the latest observed data to make the macroscopic model better replicate observed data for the base year and produce more reasonable results in the study area, i.e., the Northern Virginia region. The focus is on the model components that have been refined, especially trip productions, mode choice, and traffic assignments. The DTALite model calibration and validation leveraged the RITIS speed data to identify the locations and extents of congestion at a high level of spatial and temporal detail, with a focus on key corridors in Northern Virginia.

In addition to the travel demand model, the TransAction evaluation included some off-model tools and procedures, as well as qualitative evaluation for some of the performance measures. These include a GIS-based process to quantify the accessibility by bicycle (the number of jobs accessible by bike within 30 minutes) used in the calculation of the C1 and C2 measures.

Additional detail on the methodology used for calculation of the TransAction measures is documented in the *Task 1.5 Technical Memorandum: Performance Measures Methodology*.

Figure 2.1.1 TransAction Model Structure

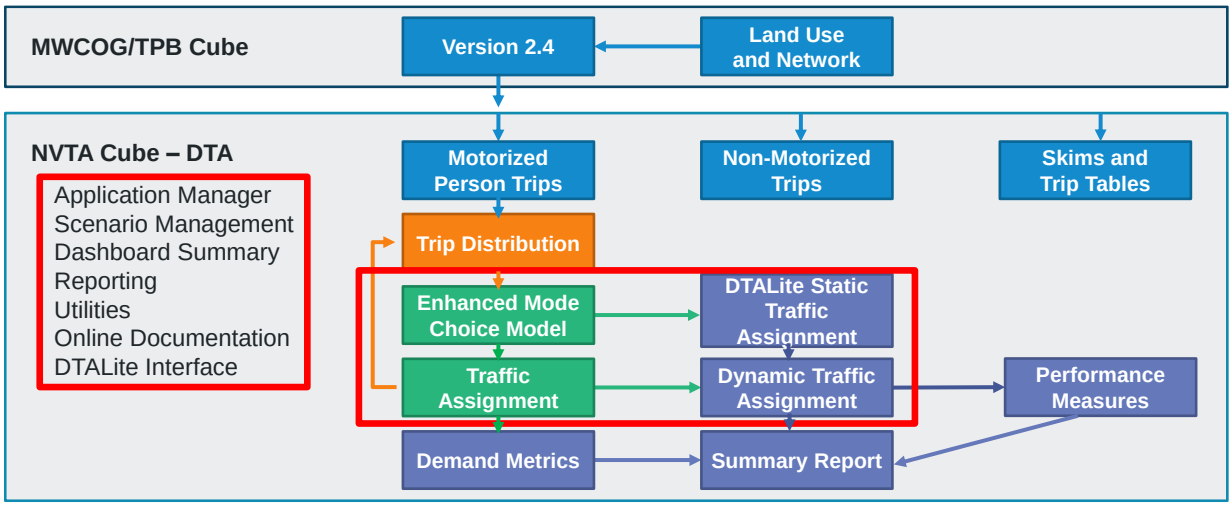
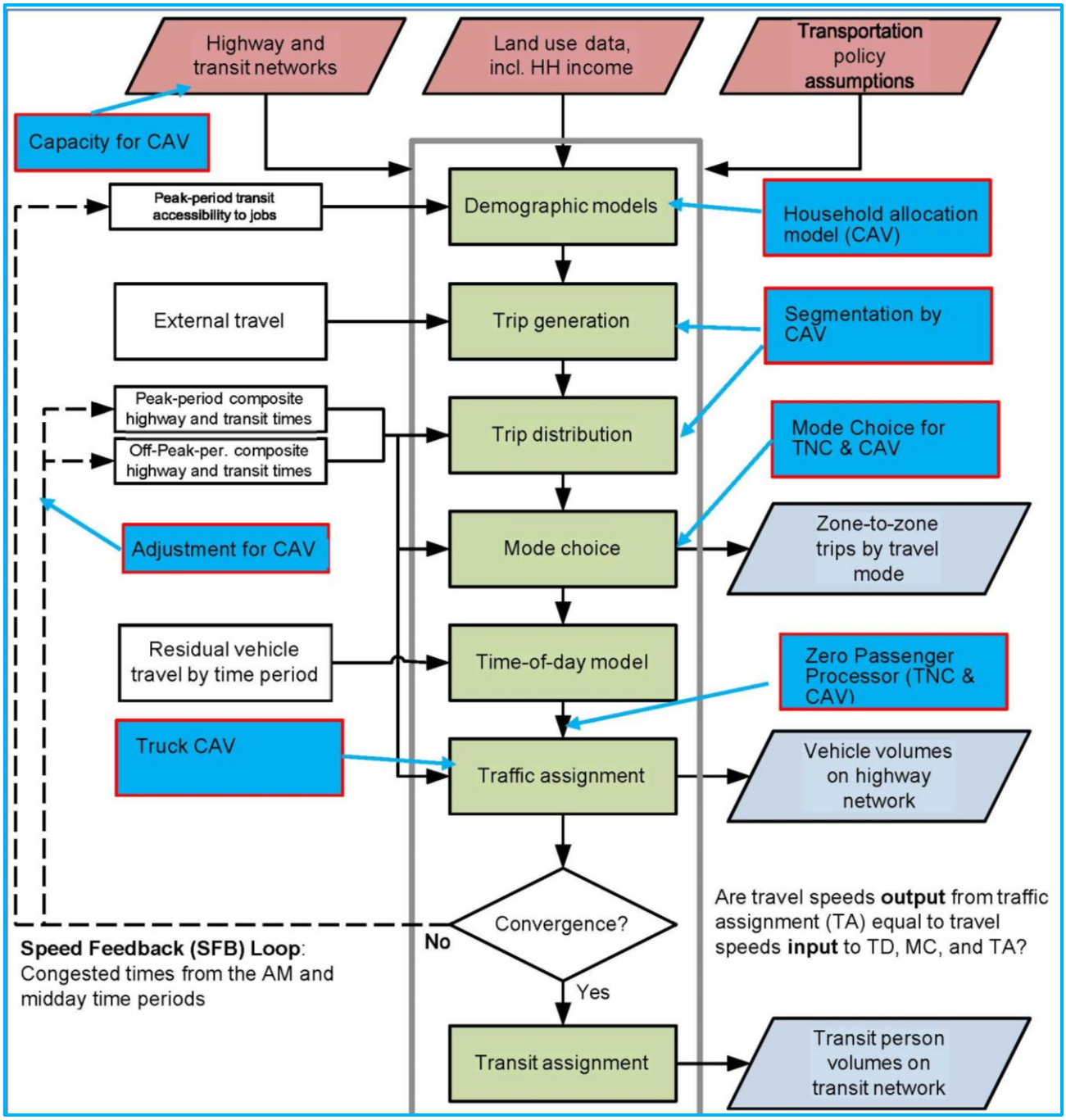


Figure 2.1.2 Travel Demand Forecasting Process



Source: Adapted from COG/TPB

2.0Hardware and Software Requirements and Installation

This section describes the hardware and software requirements and recommendations for running the TransAction Model and provides examples of the hardware used for model development and application.

2.1 HARDWARE

The TransAction Model has the hardware specifications that are needed for efficient execution, including the following:

- **Processor/Central processing unit (CPU):** Intel or AMD with 64-bit architecture. The faster the chip speed, the shorter the model run time. A faster chip/CPU is always preferred. A chip speed of around 3 GHz is recommended. An Intel CPU (2.1-2.5 GHz) was used for model runs.

- **Installed memory (RAM):** The upper limit for a 64-bit system depends on the edition of Windows system, varying from 128 GB on Windows 10 Home to 2 TB on Windows 10 Pro, Enterprise, and Education. 8 GB or more of RAM is recommended for running the TransAction Model. The workstations that we used for model runs have 64-384 GB RAM.

- **Storage space:** A hard drive with 2 TB or more space is recommended for the TransAction Model. The hard drive space needed depends on the complexity of a model scenario. Each model scenario/year typically takes more than 50 GB space but could take more than 1 TB with temporary files for a very complex scenario. The model was run on workstations with 2 TB or more free space.

- **Number of cores:** The TransAction Model uses Cube Cluster to run modules in parallel, the maximum threads is 16. It is recommended to run the model on a computer with 16 or more cores. The model can also be run on a computer with 2 or 4 cores. The computers we used for model runs had 24-40 cores.

2.2 SOFTWARE

The following define the software requirements for the TransAction Model:

- **Operating system**: Microsoft Windows (64-bit version), e.g., Windows 10, Windows Server 2008, or Windows Server 2012.

- **Citilabs Cube software**: The TransAction Model is implemented with Bentley Systems Cube software. The Cube software must be installed in order to run the model, for more information on Cube software, please visit https://www.bentley.com/software/cube/. The model has been tested under Cube version 6.4.5. The Cube modeling suite includes:

  - The system interface: Cube Base. Cube Base is the scenario management, model development, and data editing user interface.
  - Demand modeling: Cube Voyager is a modeling system with its own control and scripting language for transportation applications.
  - Cube Base and Cube Voyager are required for running the TransAction Model.

- **Process distributing**: Cube Voyager helps reduce model run-times by allocating calculation processes over multiple processors and machines.

- **DTALite:** The latest software release can be downloaded at the DTALite Github website. The basic control and interfaces of NeXTA are described in this user's guide document. The software package includes two software applications: NEXTA as GUI and data hub, DTALite as DTA simulation engine, which are discussed in detail in Chapter 8.

- **Python**: Version 3.8 or higher is needed to run scripts that will be discussed in Chapter 8. A user needs to install the following libraries: pandas, numpy, csv, time, shutil, sys, and os, under the dos command line, e.g., pip install pandas. If a user runs into some model run errors, he or she might want to consider installing "anaconda," which is an open-source package for managing python environments. A user can download the windows installer from here: https://www.anaconda.com/products/distribution. A user can run a Python script from either "Anaconda Prompt" or "Anaconda Powershell Prompt."

2.3 INSTALLATION PROCESS

This section describes the steps to install the model. The model user needs to

- Ensure that the required software is present on the system

- Choose a location on the computer to unzip the model files (provided as a zip file). The intermediate and final model results will be generated within the model folder, so select a location that has sufficient storage as recommended earlier.

Do not install the model to a directory that includes an "@" symbol. This symbol is interpreted by Cube to represent variables and will prevent Cube from following the file path. Note that folders under the "Users" directory, such as My Documents, include an "@" symbol in the file path. Also, do not install the model to a directory that begins with a lowercase "t". Cube scripts interpret "/t" as a tab character.

The catalog file is the NVTA.cat file within the NVTA Model folder. Once the catalog is open in Cube, the user will see the application groups. Accept all updates to the model directory.

2.4 FOLDER STRUCTURE

The unzip model folder will show the following folders:

- **Scenarios**: This is the scenario directory. The Scenario directory will contain the scenario folders. For example, for scenario Base2017, the user should see Base2017 folder. Each scenario folder (in this case Base2017) should have an "Inputs" folder for all the inputs of the scenario and "Outputs" folder. All the model outputs are written to the "Outputs" folder.

- **Scripts:** Cube application manager groups (*.APP) and script files (*.S)

    - The Cube script files are organized into subdirectories according to the application group.

    - The Cube script file names are set by the application group names and the program type. For example, a Matrix program in the Main Screen application can have a file name: HKHWY00A.S

- **Software:** It contains a list of external software as executable files that are used to generate summaries within the model.

- **Support:** It contains files and scripts that are applicable for all scenarios.

3.0 Scenario Manager

The Scenario Manager is where the full model is typically run. The base catalog file of the TransAction Model contains two scenarios in the base directory. These scenarios are shown in Figure 3.1.1.

3.1 MODEL SCENARIOS

Figure 3.1.1 TransAction Model Scenarios

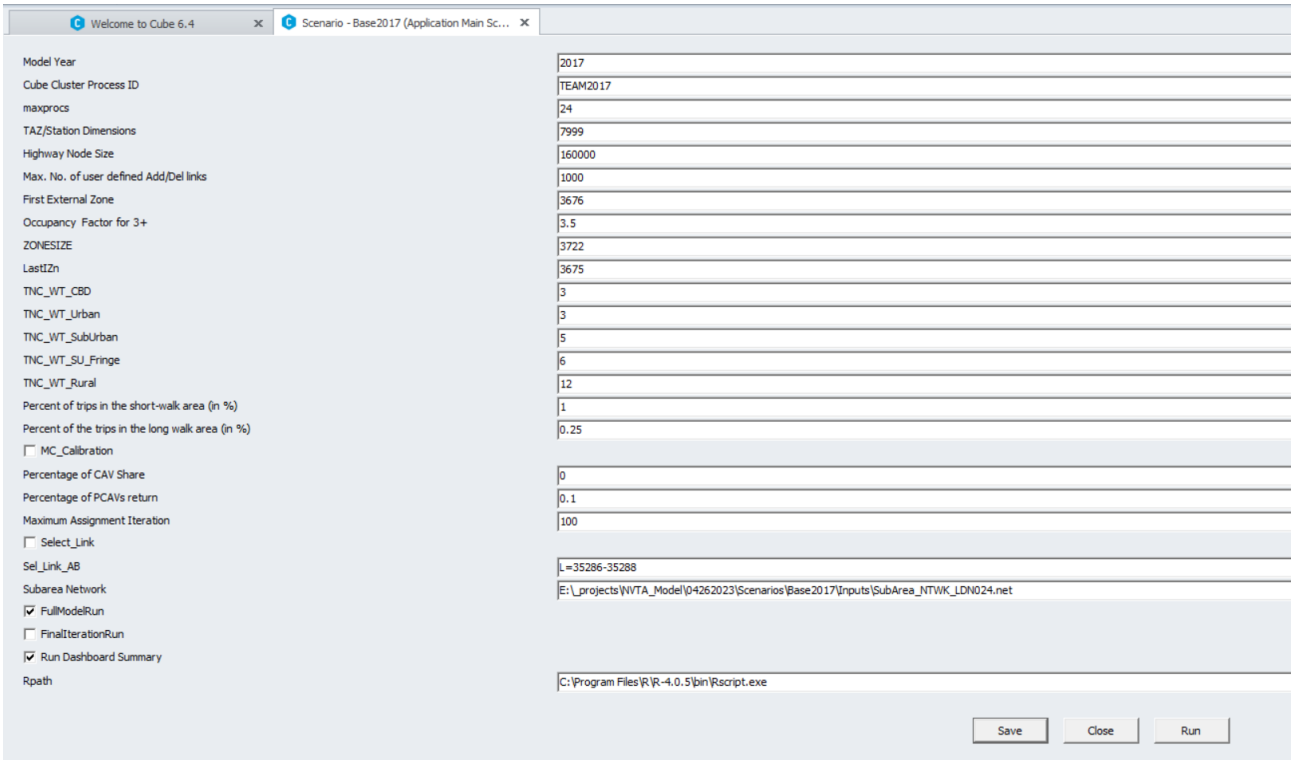
Within the Scenario Manager, there are "Parent", "Children" and "Siblings" directories. For the TransAction Model structure shown above, the Parent directory serves as the base for all the scenarios, with Base2017 and Base2045 scenarios relating as Children to the Base, and relating as Siblings to each other.

When creating new scenarios, either Children or Siblings can be added to the scenario manager. A Child will be created as a subfolder to the Parent that it is created from, whereas a Sibling will be created in a folder at the same level as the selected scenario. During the initial set-up, both Children and Siblings will be populated with the same data set as a Parent.

The TransAction Model is set up in such a way that the Base scenario is only a placeholder for all scenarios (in this case Base2017 and Base2045). The Base scenario is not designed to be the actual scenario to be run.

Figure 3.1.2 provides a screenshot of the opened Base2017 scenario within the Scenario Manager.

Figure 3.1.2 Scenario Manager – Initial Screen



The Scenario Manager consists of the Model Operation, Scenario Year, TNC wait times, and model run controls. These inputs are used in each of the submodel components. Each input on the Scenario Manager initial screen is described in the subsections below.

Catalog Key: Model Year – Scenario year

This integer edit box control is used to enter the scenario model year to be run in the model. The values here need to be consistent with a year of scenario to be run.

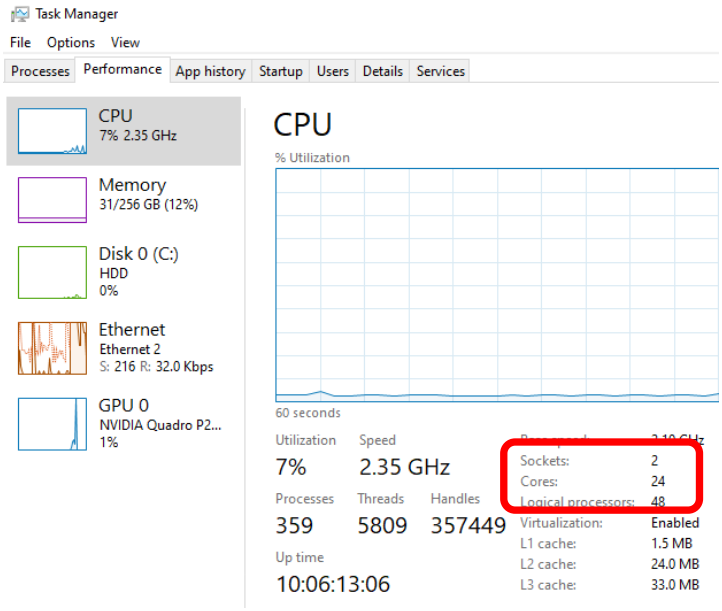
Catalog Key: Cube Cluster Process ID

This edit box control is used to enter the process id for the Cube clusters in the model. Any string as a combination of characters and numbers can be used here. This value does not require any changes during the normal model operation.

Catalog Key: maxprocs – Maximum number of threads for cluster

This edit box control allows the user to set an integer value for the number of processors that are used in running most of the MATRIX and HIGHWAY programs within the TransAction Model. The more processors used, the faster the model runs. If your server has 24 cores, set this to 20. Use fewer cores than the maximum available on your system. You can check the total cores available on your system in task manager as shown in the Figure 3.1.3.

Figure 3.1.3 System cores



Catalog Key: TAZ/Station Dimensions

This edit box specifies the dimensions of TAZ/Stations. This value is used to ensure that the matrix files and zone counts remain consistent. This value does not require any changes during the normal model operation.

Catalog Key: Highway Node Size

This edit box specifies the first highway node number in the highway network. This value does not require any changes during the normal model operation.

Catalog Key: Max No. of user defined add/del links

This value is used to ensure that the matrix files and zone counts remain consistent. This value does not require any changes during the normal model operation.

Catalog Key: First External Zone

This edit box specifies the first external zone number. This value is used to ensure that the matrix files and zone counts remain consistent. This value does not require any changes during the normal model operation.

Catalog Key: Occupancy Factor for 3+

This edit box specifies the occupancy factor for 3+ vehicle trips. The default value is set to 3.5. This value does not require any changes during the normal model operation.

Catalog Key: ZONESIZE

This edit box control allows the user to set the total number of TAZs. This value is used to ensure that the matrix files and zone counts remain consistent. This value does not require any changes during the normal model operation.

Catalog Key: LastIZn: Last internal zone number

This edit box control allows the user to set the last internal zone number in the model. This value is used to ensure that the matrix files and zone counts remain consistent. This value does not require any changes during the normal model operation.

Catalog Key: TNC_WT_CBD: TNC wait time in CBD

This edit control box allows the user to specify the wait time of TNC mode in a CBD area. This value is used in the mode choice model. This value does not require any changes during the normal model operation. The default value is set to 3 minutes.

Catalog Key: TNC_WT_Urban: TNC wait time in urban areas

This edit control box allows the user to specify the wait time of TNC mode in urban areas. This value is used in the mode choice model. This value does not require any changes during the normal model operation. The default value is set to 3 minutes.

Catalog Key: TNC_WT_SubUrban: TNC wait time in sub-urban areas

This edit control box allows the user to specify the wait time of TNC mode in sub-urban areas. This value is used in the mode choice model. This value does not require any changes during the normal model operation. The default value is set to 4 minutes.

Catalog Key: TNC_WT_SU_Fringe: TNC wait time in sub-urban fringe areas

This edit control box allows the user to specify the wait time of TNC mode in sub-urban fringe areas. This value is used in the mode choice model. This value does not require any changes during the normal model operation. The default value is set to 6 minutes.

Catalog Key: TNC_WT_Rural: TNC wait time in rural areas

This edit control box allows the user to specify the wait time of TNC mode in rural areas. This value is used in the mode choice model. This value does not require any changes during the normal model operation. The default value is set to 12 minutes.

Catalog Key: Percent of trips in the short-walk areas (in %)

This edit control box specifies the default value for percent of trips in the short-walk areas. This key is utilized in the mode choice model. The default value is 1. This value does not require any changes during the normal model operation.

Catalog Key: Percent of trips in the long walk areas (in %)

This edit box specifies the default value for percent of trips in the long walk areas. This value is utilized in the mode choice model. The default value is 0.25. This value does not require any changes during the normal model operation.

Catalog Key: MC_Calibration: mode choice calibration

This check box controls automatic calibration in mode choice for new target mode shares. The box should always be kept unchecked for model runs. This value does not require any changes during the normal model operation.

Catalog Key: Percentage of CAV Share

This edit box control allows the user to set the CAV market penetration rate. The values can range from 0 to 1.

Catalog Key: Select Link

This check box allows the user to run the select link analysis for the links mentioned in Sel_Link_AB catalog key.

Catalog Key: Sel_Link_AB

This character edit box control allows users to input the node numbers that define the links that are to have a select link analysis performed. The syntax must follow that as defined in the Cube documentation, for example L=665410-666990 or N=107,200-208.

Catalog Key: Subarea Network

This character edit box has the path the subarea network. The model user can edit the path to point to a specific location.

Catalog Key: FullModelRun

The edit check box allows the use run full model run. The check box should be checked if the user wants to run the model for all five iterations.

Catalog Key: FinalIterationRun

The check box, if selected, allows the use to run final iteration run. The model folder should contain outputs from previous iterations to run the model successfully.

Catalog Key: Run Dashboard Summary

The edit check box, if selected, creates the summary of the model run.

Catalog Key: Rpath

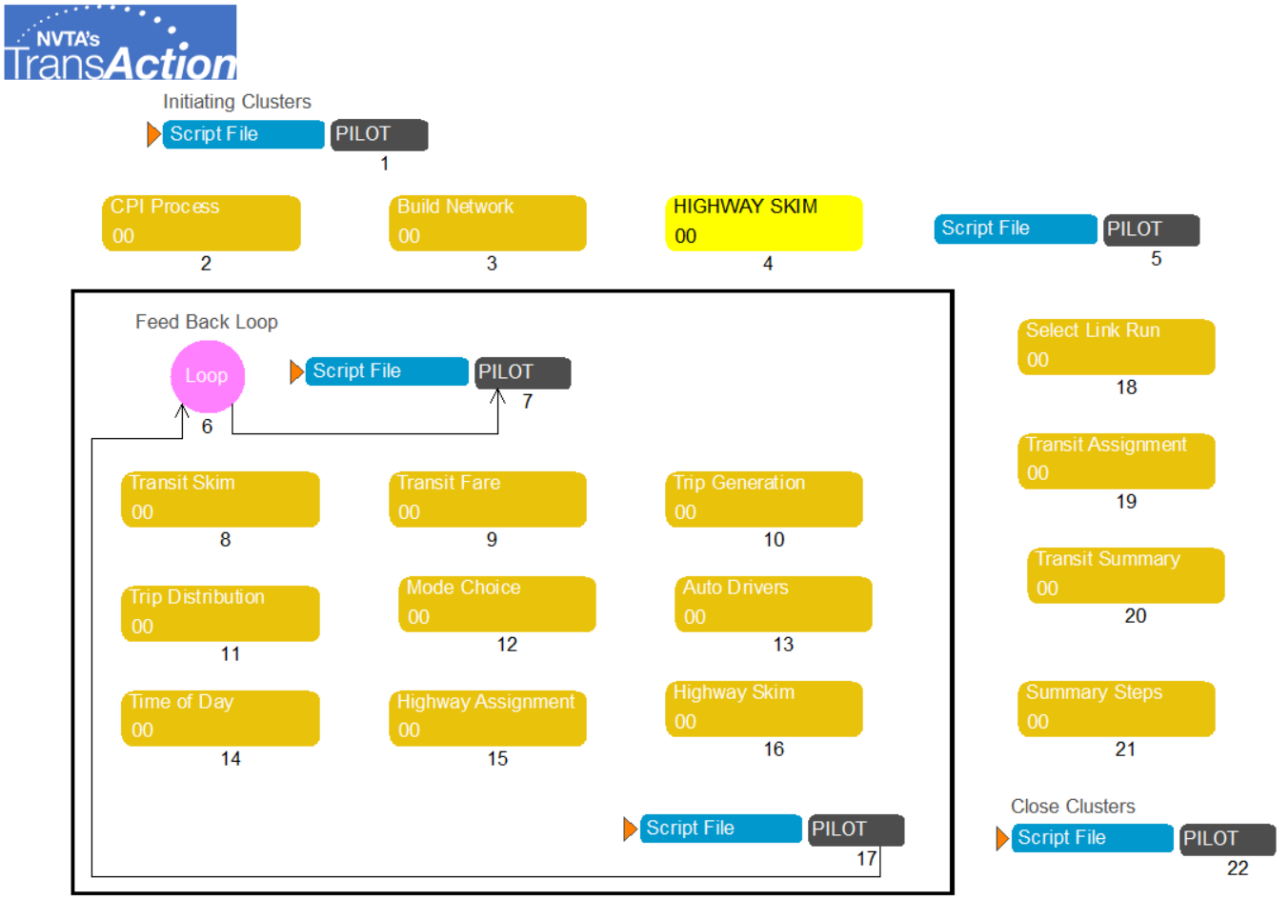
This character edit specifies the path to RScript.exe on the user's local system.

4.0 Application Manager

4.1 Main Screen

The Application Manager provides a graphic layout of the operation of the model (see Figure 4.1.1). The model is run for five iterations: pp, i1, i2, i3, and i4. The following sections describe each of the modeling steps and explains the model operation process.

Figure 4.1.1 Main Screen



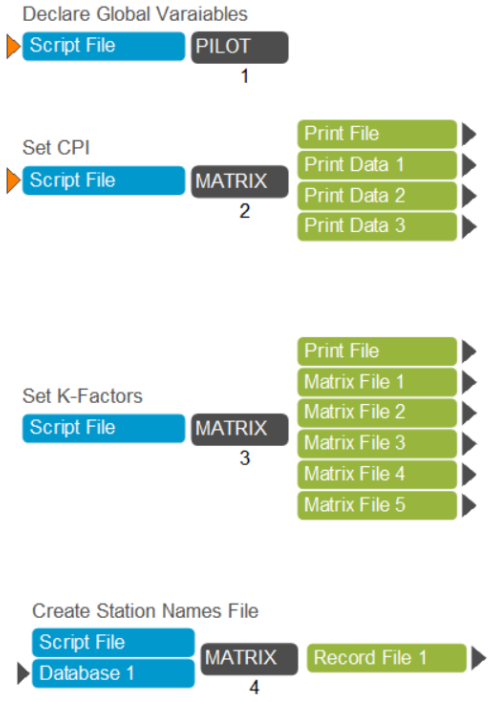
The initial Pilot program in the Application Manager opens the Cube clusters based on the value in the **maxprocs** catalog key.

4.2 Set CPI

The set CPI application group creates highway and transit deflators based on the model year, generates friction factors matrices for each purpose (home-based work, home based shopping,

home based other, non-home-based work, non-home based other) and creates a station file with ID and its corresponding names.

Figure 4.2.1 Set CPI

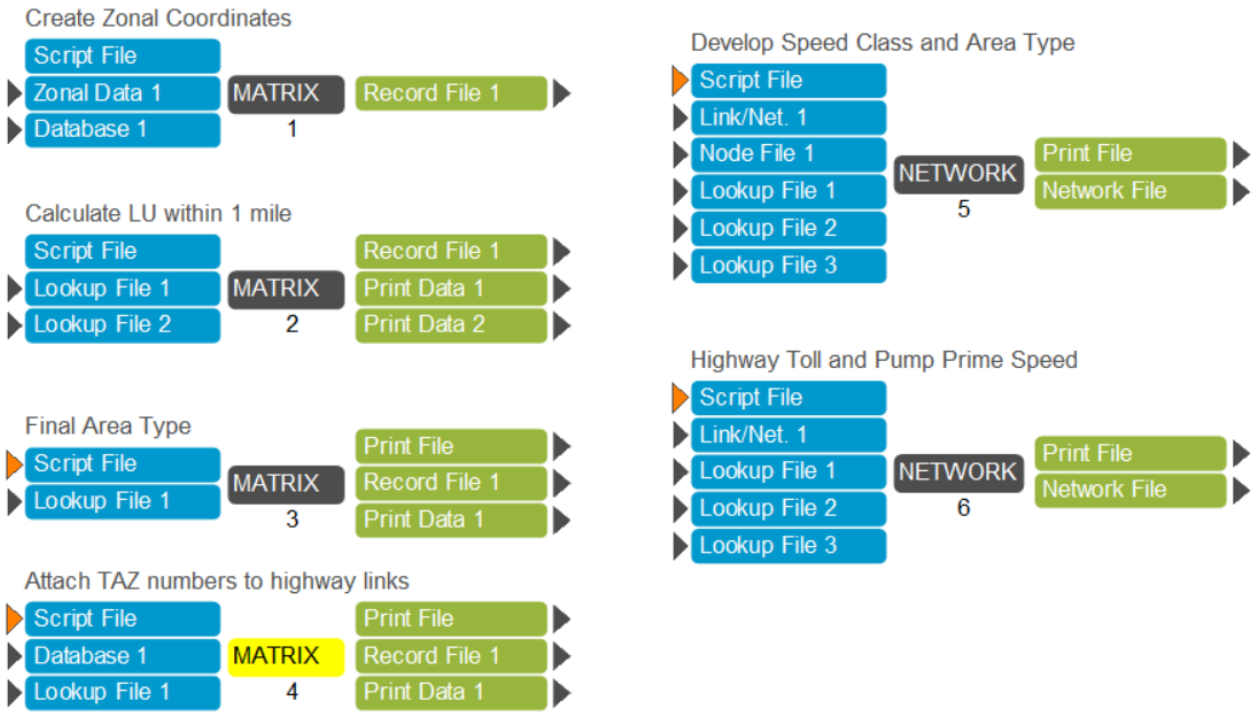


The pilot step declares global variables that are used in this application group. The matrix routine Step 2 creates the highway and transit deflation factors based on the model year. Step 3 creates K-factors, friction factors, and time penalties for trip purposes – home based work, home based shopping, home based other, non-home based work, and non-home based other. Step 4 creates the station names file from the stations input file.

4.3 BUILD NETWORK

The build network application group builds the highway network with the necessary attributes. (see Figure 4.3.1).

Figure 4.3.1 Build Network

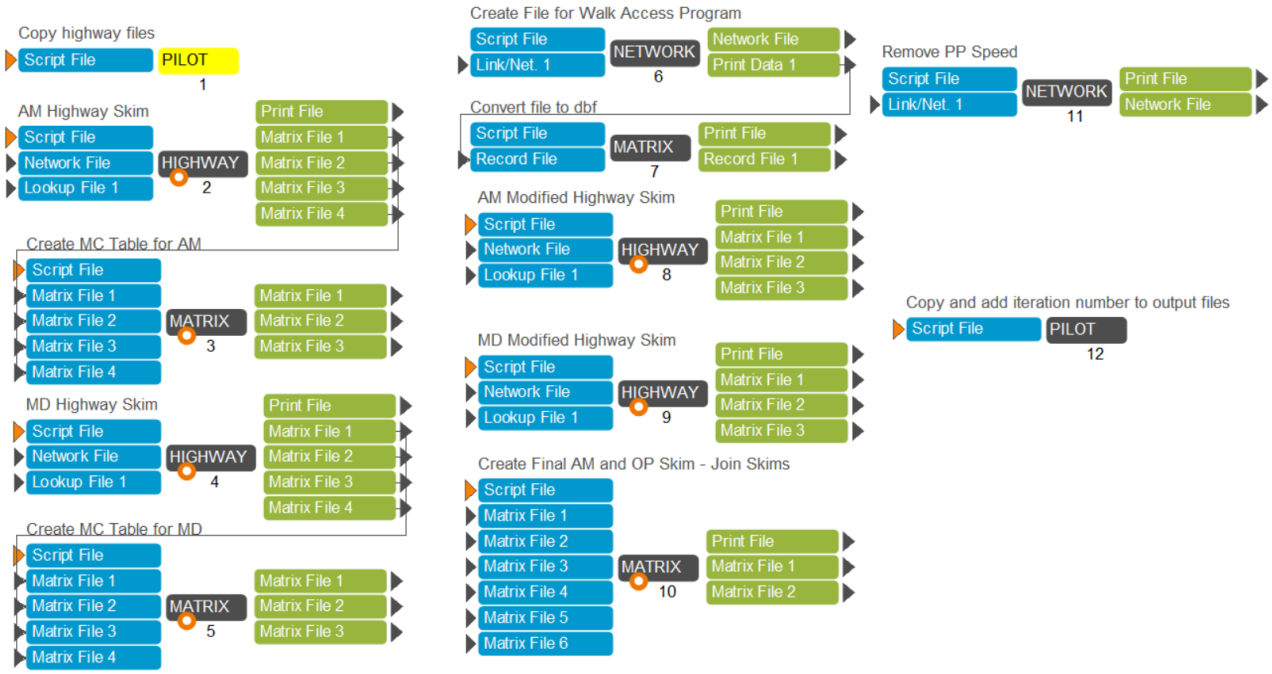


The matrix program from 1 to 4 creates TAZ file with X,Y coordinates, land use file with necessary attributes, and crosswalk between TAZ and highway link ids. The network programs 5 and 6 prepare the highway network with additional attributes such as area type, tolls, etc.

4.4 HIGHWAY SKIMS

In the highway skims application, the model creates skims for am and off-peak time period. In Figure 4.4.1, the highway Step 2 creates the highway skim files for am time period for SOV (single occupancy vehicle), HOV2 (high occupancy vehicle with two participants), HOV3+ (high occupancy vehicle with 3+ participants), and trucks. Matrix Step 3 further adjusts the AM time period skims (time, distance and toll values) for mode choice model. The highway Step 4 creates the highway skim files for MD time period for SOV (single occupancy vehicle), HOV2 (high occupancy vehicle with two participants), HOV3+ (high occupancy vehicle with 3+ participants) and trucks. Step 5 further adjusts the MD time period skims (time, distance and toll values) for mode choice model. Step 6 extracts A node , B node, distance, functional class type of the link and TAZ information of walk access links from the highway network. Some attributes of the network are further modified based on the link ids. Steps 8 and 9 reskim the modified network from Step 7 to create the skim files. Step 10 combines the skim cores from previous steps into two skim files for AM and MD time periods. Step 11 removes pump prime speeds on the network. The pilot program creates a copy of the output files from highway skims application manager with the iteration number ('pp' in this case) as suffix. The iteration numbers help in keeping track of files generated in each specific iteration.

Figure 4.4.1 Highway skims

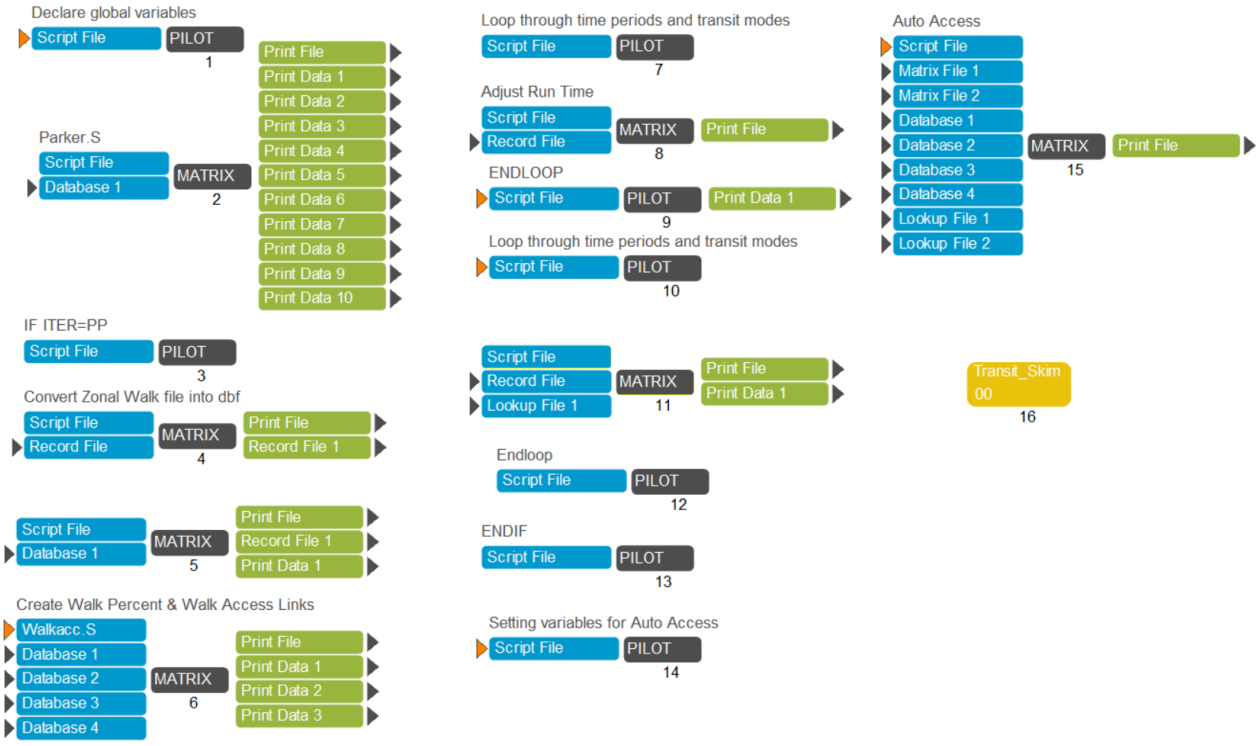


4.5 TRANSIT SKIM

The outer transit skim application manager group generates transit skims for commuter rail, Metrorail only, bus only, and combination of bus-Metrorail for each iteration of the feedback loop. This outer group generates the necessary access and egress link files that are used in the inner transit skim application. Figure 4.5.1 and Figure 4.5.2 shows the schematic layout of the outer and inner transit skims application manager, respectively.

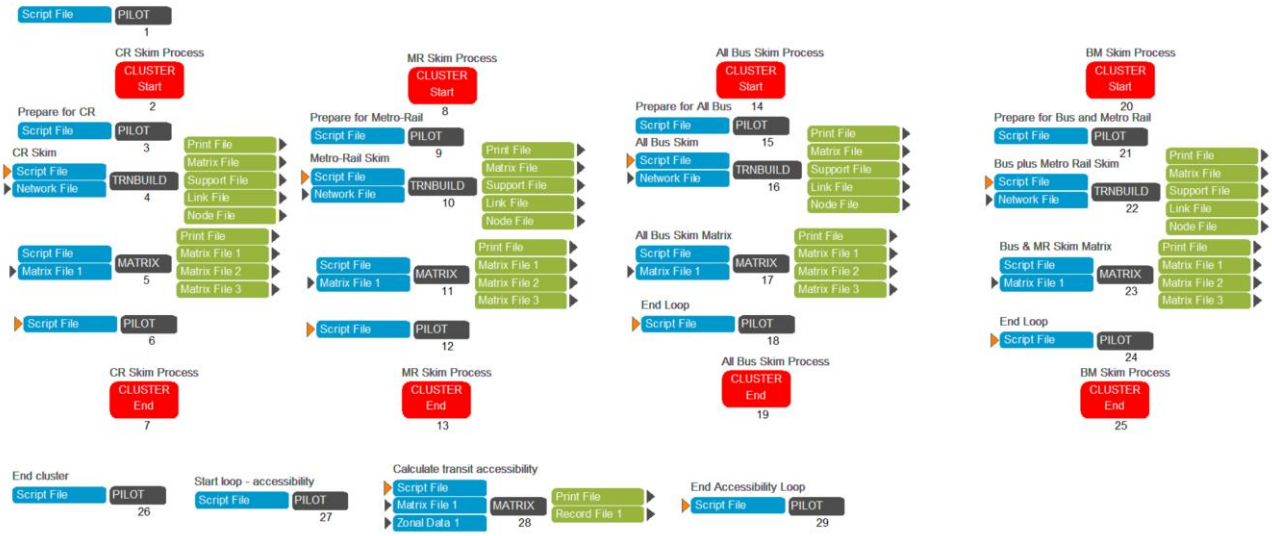
In Figure 4.5.1, the matrix program Step 2 uses the station file to generates AM and off-peak time period PNR to station transfer links for Metrorail, commuter rail, bus, light rail and BRT. The programs from Step 3 to Step 13 are executed only for the initial ("pump prime") iteration run. They create walk access transit links. Steps 14 and 15 create transit access links for all transit modes.

Figure 4.5.1 Transit Skims



The inner transit skim application program uses multistep parallel processing to generate the transit skims in parallel for each transit mode as shown in Figure 4.5.2. Step 1 to Step 25 generate transit skims for two time period (peak and off-peak) by sub-modes: commuter rail, Metrorail only, bus only, and combination of bus-Metrorail. These transit skims are further segmented by three access modes: walk, park and ride (PNR) and kiss and ride (KNR). Step 26 to Step 29 generate transit accessibility summary for total jobs accessible from a given zone within 35 min, 40 min, 45 min and 50 min. These summaries are used in the vehicle ownership model.

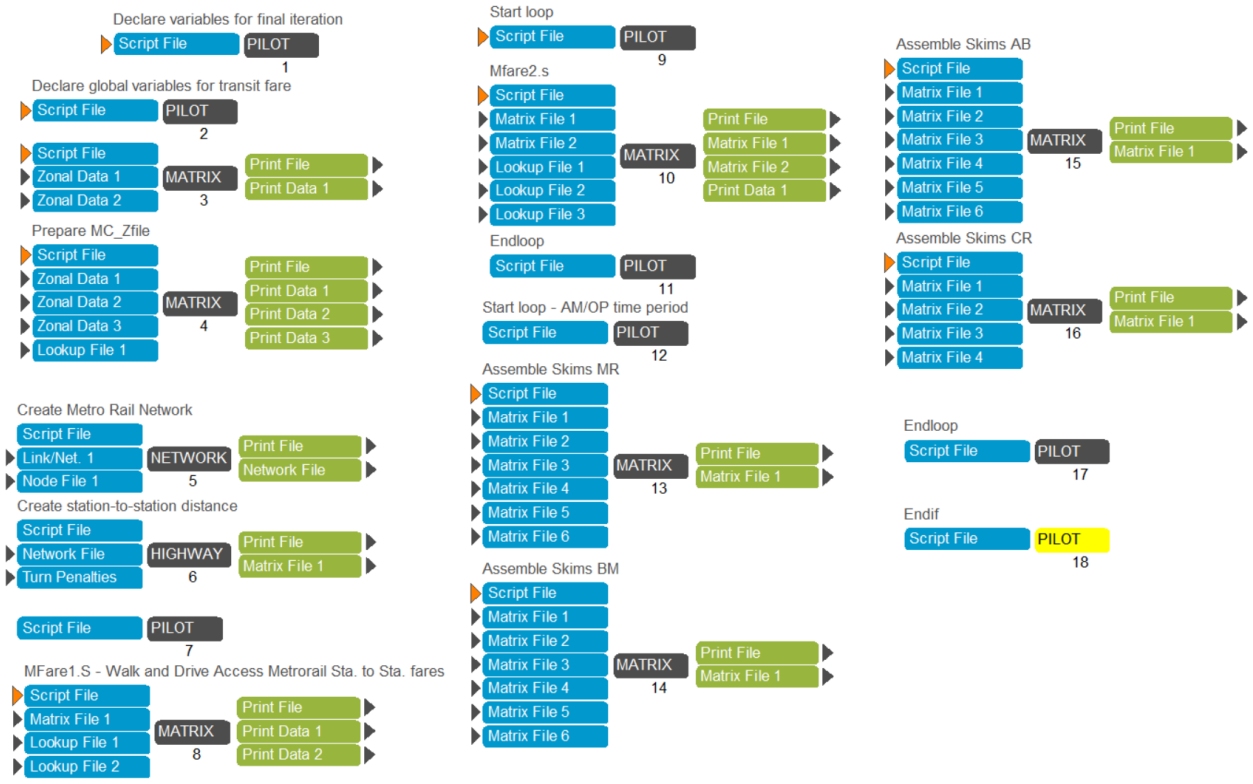
Figure 4.5.2 Transit Skims



4.6 TRANSIT FARE

Application Manager group generates transit fares for each iteration of the feedback loop. (see Figure 4.6.1)

Figure 4.6.1 Transit Fare

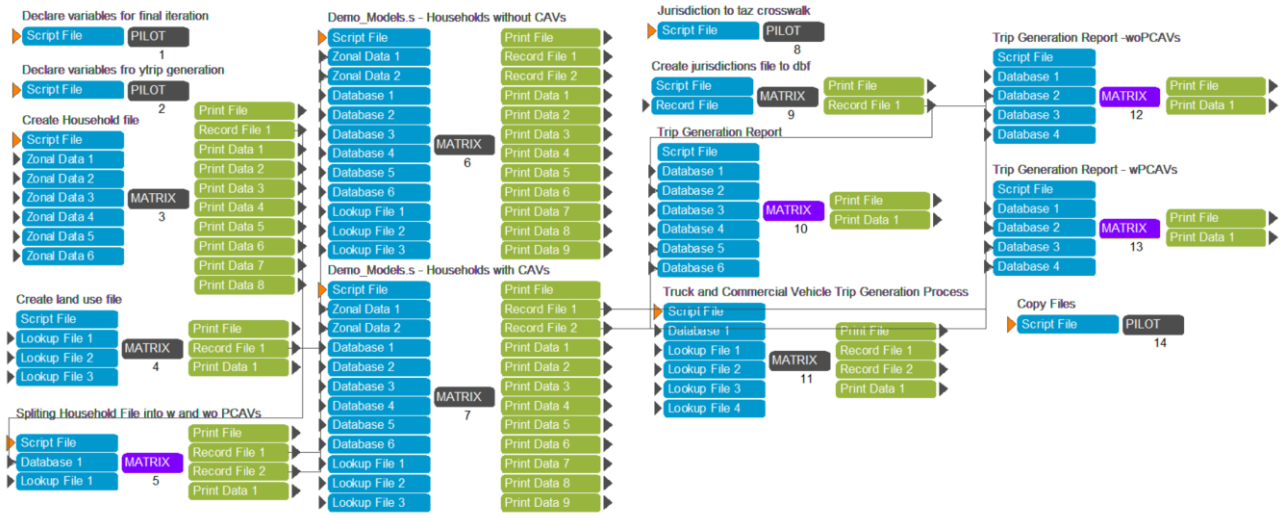


Step 2 declares global variables that are used in the subsequent programs. Step 3 merges the Metrorail walk percent information into the zonal equivalency file. The resulting file is used to calculate transit fares for each zone and zonal parking costs used in the mode choice model in Step 4. In Steps 5 and 6, the script develops Metrorail station-to-station distance skims. Step 8 calculates Metrorail station to station fares for AM and off-peak periods. Step 10 calculates the total transit fare between TAZs for AM and off-peak periods. Steps 13 to 16 consolidate the binary fare files into a fare file for each transit sub-mode which is further used in the mode choice model.

4.7 TRIP GENERATION

The trip generation application manager estimates productions and attractions for all zones. Figure 4.7.1 shows the schematic layout of the process.

Figure 4.7.1 Trip Generation



In matrix Step 3, the program uses zone data, area types, transit accessibility files, and other control totals to generate total number of households segmented by household income, household size, and vehicle availability in the household. Matrix Step 4 furthers generates a land use file containing information such as total population, total employment by various industry types, employment density, area type, etc. Matrix Step 5 further segments the households with two categories – Households without CAVs and Households with CAVs. It uses a lookup file which contains the percentage of households owning CAVs for each income group based on the CAVs market penetration rate. Matrix Steps 6 and 7 generate the zonal productions and attractions for households with CAVs and without CAVs, respectively. Please note that the productions and attractions for external trips are only created for households without CAVs. Pilot program 8 and matrix Step 9 generates a crosswalk of jurisdiction and zones. Matrix Step 10 generates the summary of productions and attractions for all household within the region. Matrix Steps 12 and 13 produce the trip generation summary for households with and without CAVs, respectively. The trip generation summary includes land activity by jurisdiction, area type, motorized/non-motorized/home-based trip production/attractions by purpose/area type/jurisdictions. Step 11 uses the zonal land use file, area type file, truck and commercial trip model coefficients to generate medium truck, heavy truck, and commercial vehicle zonal trips.

4.8 TRIP DISTRIBUTION

Trip distribution application manager has two components: trip distribution for external and trip distribution for internal trips. The "trip distribution external" application manager distributes external trip ends and produces external trip tables by purpose. The application manger "trip distribution internal" calculates the internal trip ends, distributes them and finally combines with external trip tables for households with and without CAVs. Figure 4.8.1 shows the schematic layout of trip distribution application manager.

Figure 4.8.1 Trip Distribution

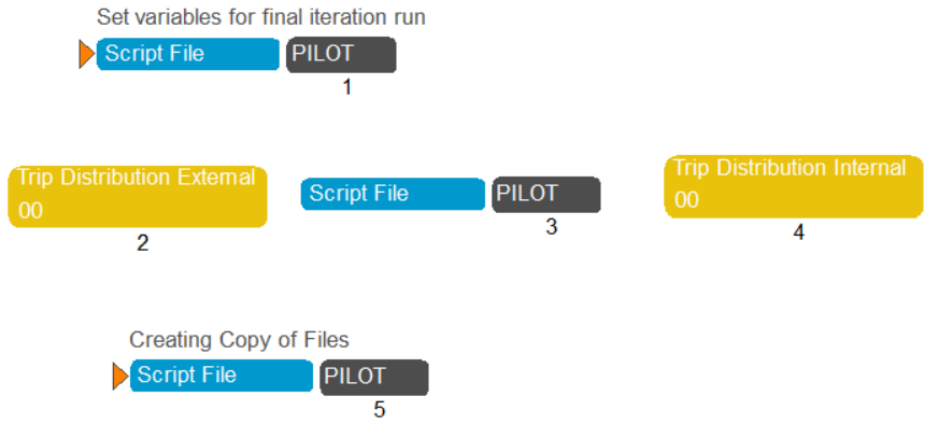
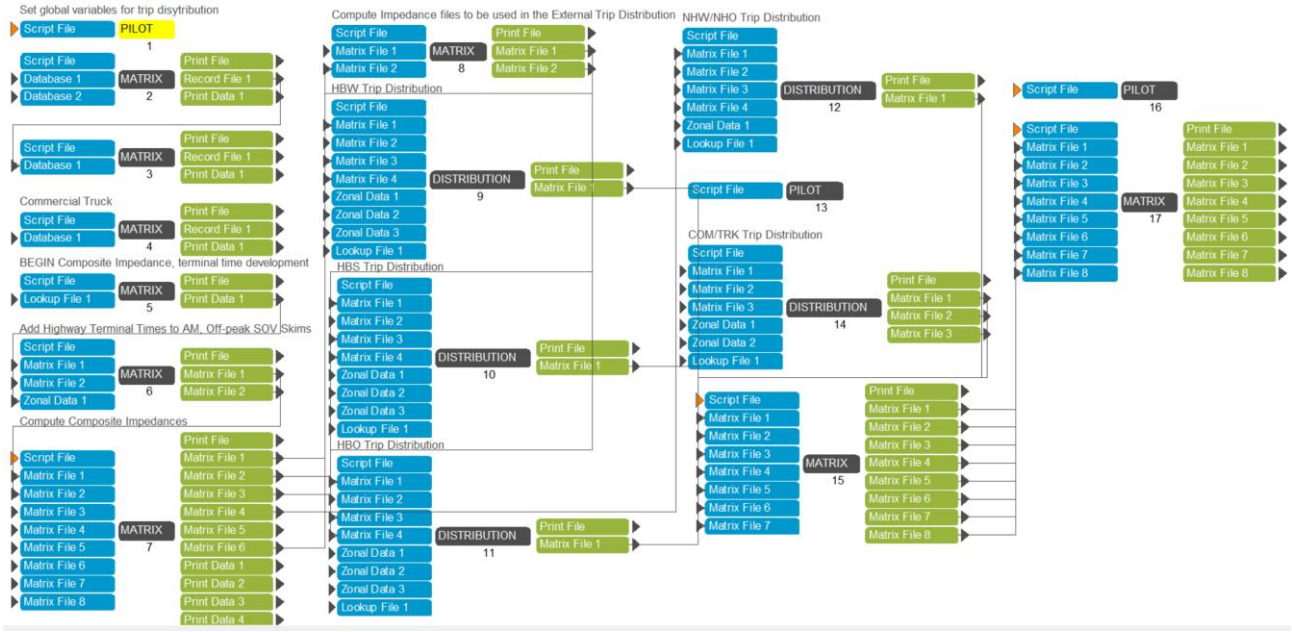


Figure 4.8.2 shows the schematic layout of the external trip distribution application manager. The application manager distributes external auto person trips by purpose. Steps 2 and 3 uses zonal production and attraction from trip generation to prepare trip ends that are suitable for applying trip distribution for external productions and attractions only. Step 4 produces external truck trip ends for commercial, medium and heavy trucks. Steps 5 to 8 generate zonal impedances that are used in both the distribution of external and internal trips. The AM peak travel impedances are used for distributing home based work trips, and midday impedances are used for all remaining purposes.

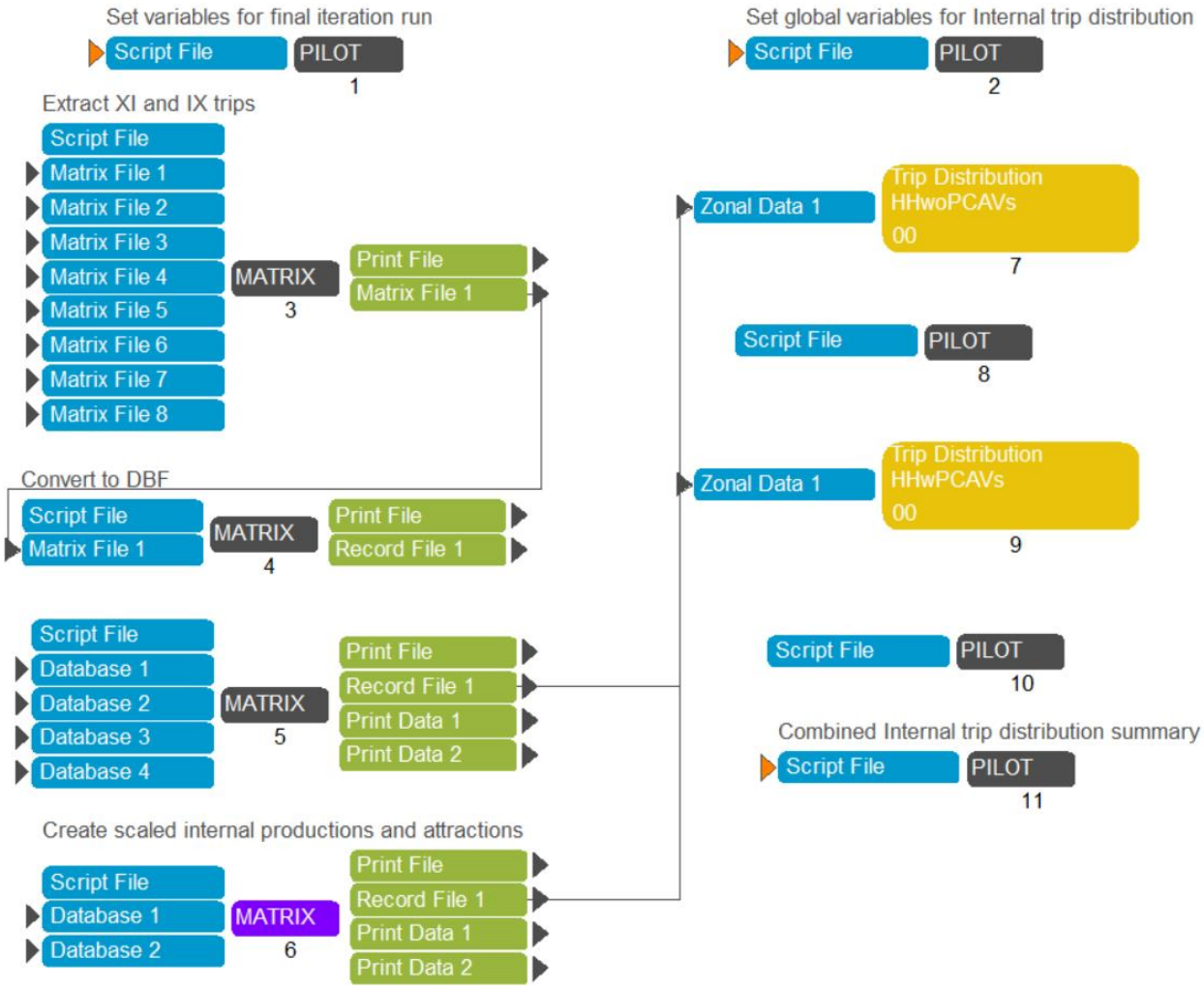
Figure 4.8.2 Trip Distribution External



In Figure 4.8.3, the trip distribution internal application manager distributes internal motorized person trips by purpose. Steps 3 to 6 read the external trip tables created in the external trip distribution application manager and summarize the trip end from those trip tables. The program scripts also read the internal trip-ends from the trip generation process and subtract internal

external trips from the total computed trip productions by purpose to calculate the final internal trip productions. These final productions are further used to compute the internal trip attraction scaling factor.

Figure 4.8.3 Trip Distribution Internal



In Figure 4.8.4, Steps 1 to 5 run final trip productions and attractions through distribution by home based work (HBW), home based shopping (HBS), home-based other (HBO), non-home based work/other (NHW/NHO), and Commercial/medium/heavy trucks. Steps 6 computes the total person trips by combining internal and external trips. Steps 7 to 12 summarize the trip tables generated in Step 6 for households who do not own PCAVs. Figure 4.8.5 shows the schematic layout of internal trip distribution of households with PCAVs. This application group distributes trips made by households who own PCAVs and summarizes the trip ends at the end.

Figure 4.8.4 Trip Distribution Internal: Household without PCAVs

s

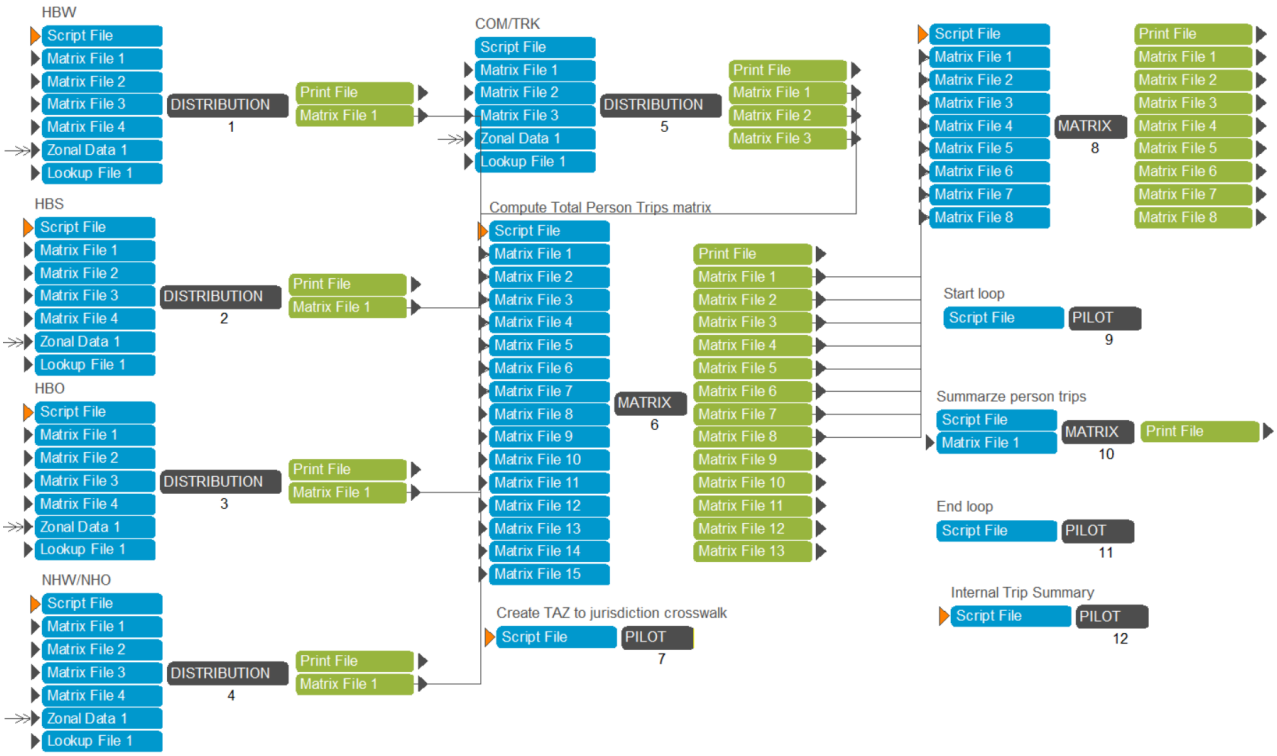
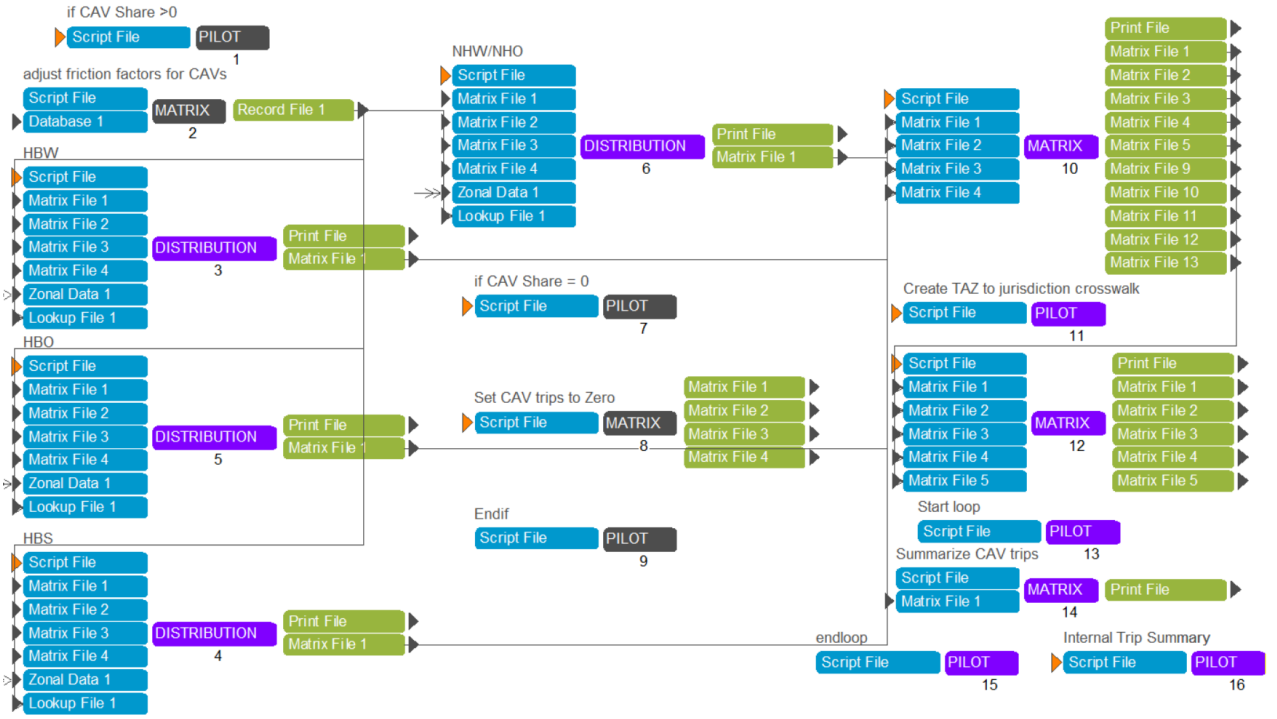


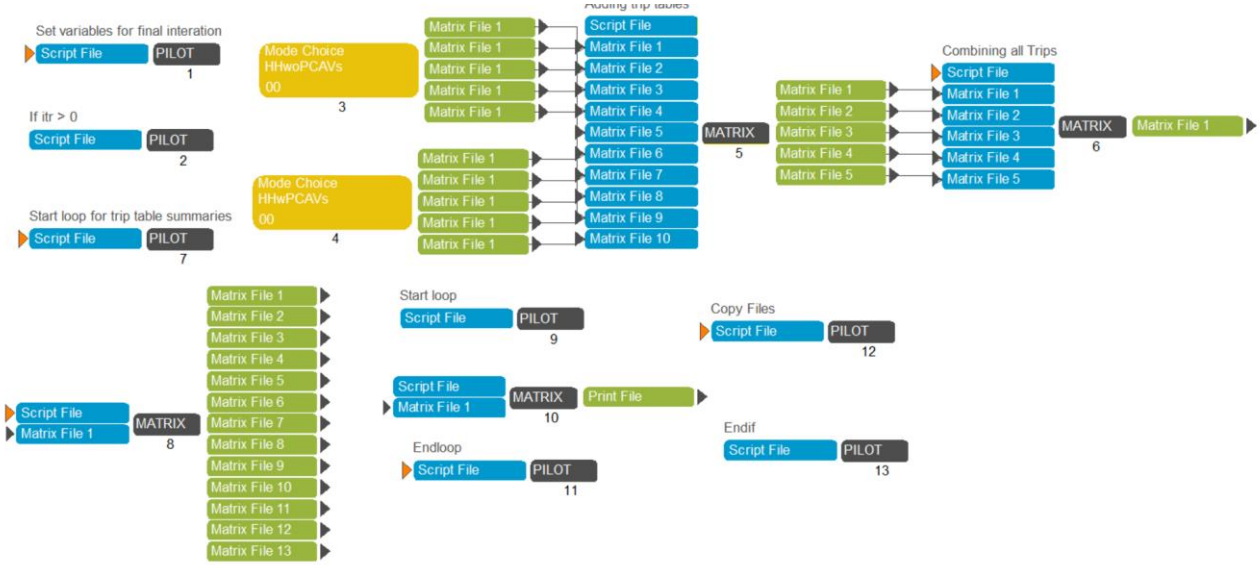
Figure 4.8.5 Trip Distribution Internal: Household with PCAVs



4.9 MODE CHOICE

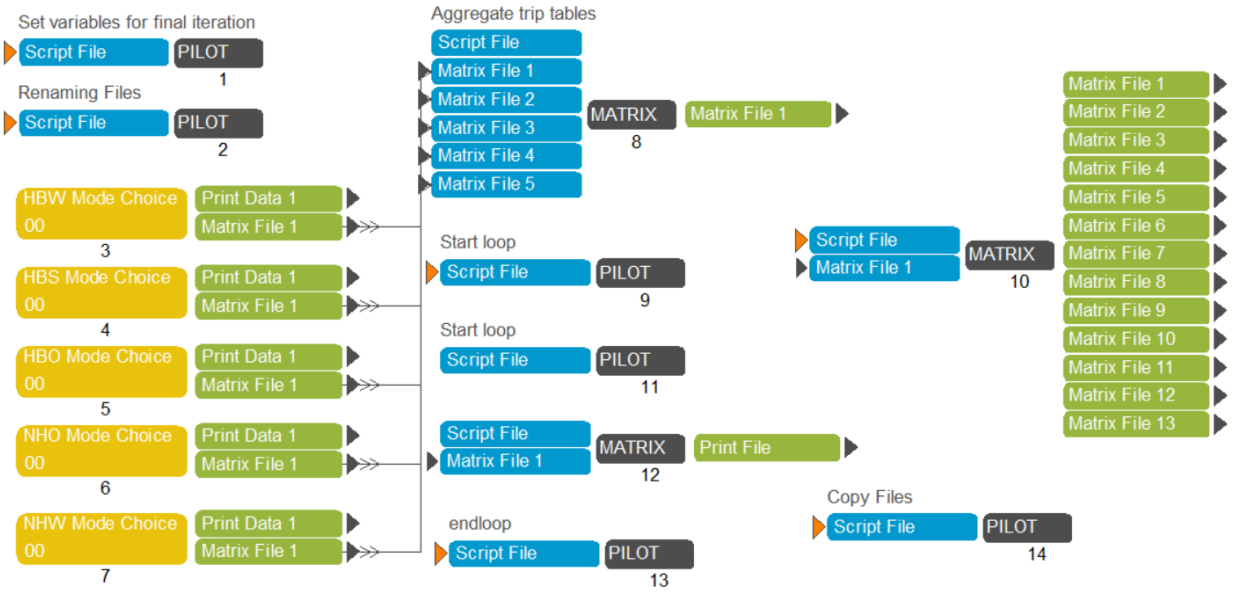
The mode choice Application Manager includes households with CAVs and households without CAVs as shown in Figure 4.9.1. For each type of households, the mode choice is carried out for each trip purpose (HBW, HBS, HBO, NHW and NHO) in different application manager. Figure 4.9.2 shows the steps involved in each mode choice component for HBW trip purpose. Similar steps are followed for other trip purposes in their respective application managers.

Figure 4.9.1 Mode Choice



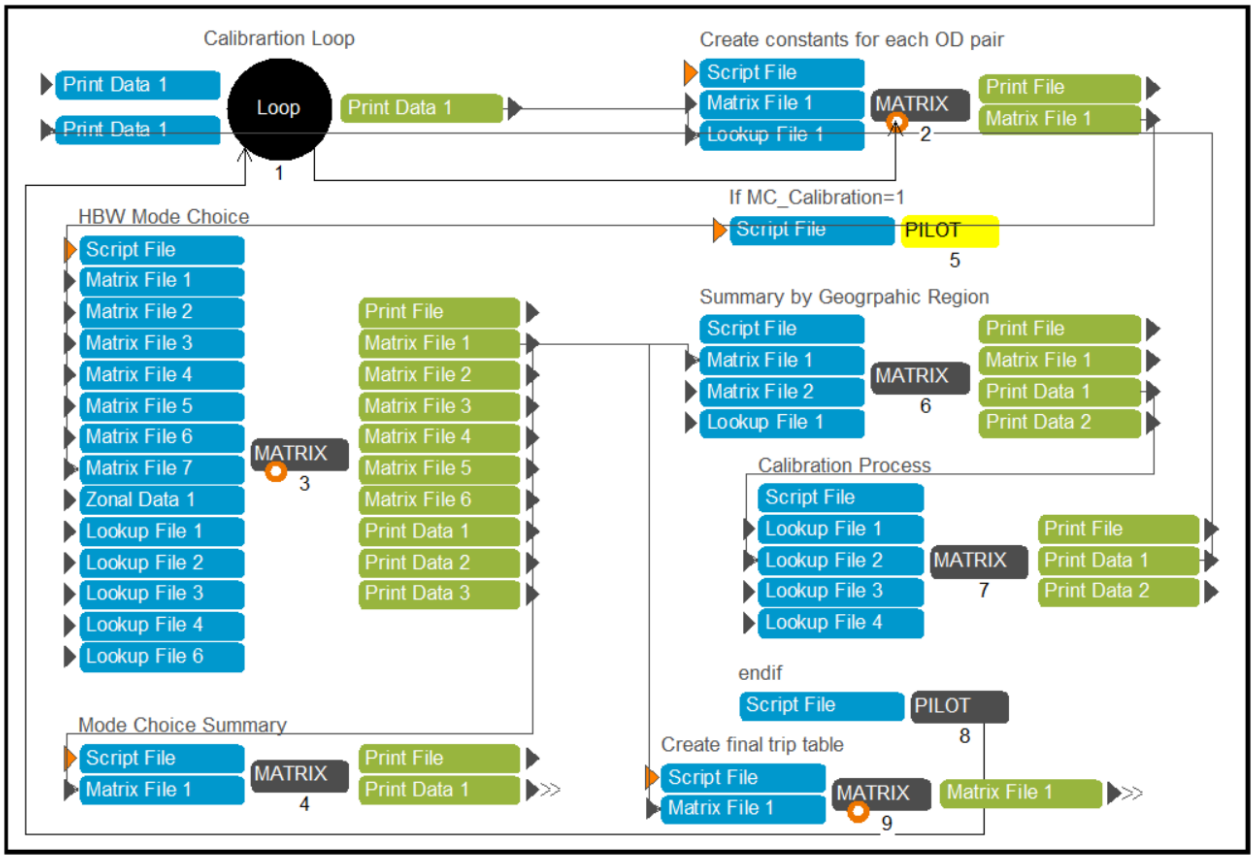
In Figure 4.9.1, matrix Step 5 aggregates trip tables by trip mode for each trip purpose (HBW, HBS, HBO, NHW, NHO) for households with and without PCAVs into trip tables by trip mode for each trip purpose. Step 6 further aggregates the trip tables for each trip purpose into single trip table by mode. Step 7 to Step 13 summarize the trips by purpose in each jurisdiction.

Figure 4.9.2 Mode Choice - HHwoPCAVs



In Figure 4.9.2, the application manager for HBW Mode Choice, HBS Mode Choice, HBO Mode Choice, NHO Mode Choice and NHW Mode Choice creates the final person trip tables. Step 8 aggregates the trip tables for each purpose to a single trip table. Steps 9 to Step 13 further summarize the trips for each jurisdiction by trip purposes.

Figure 4.9.3 Mode Choice - HBW

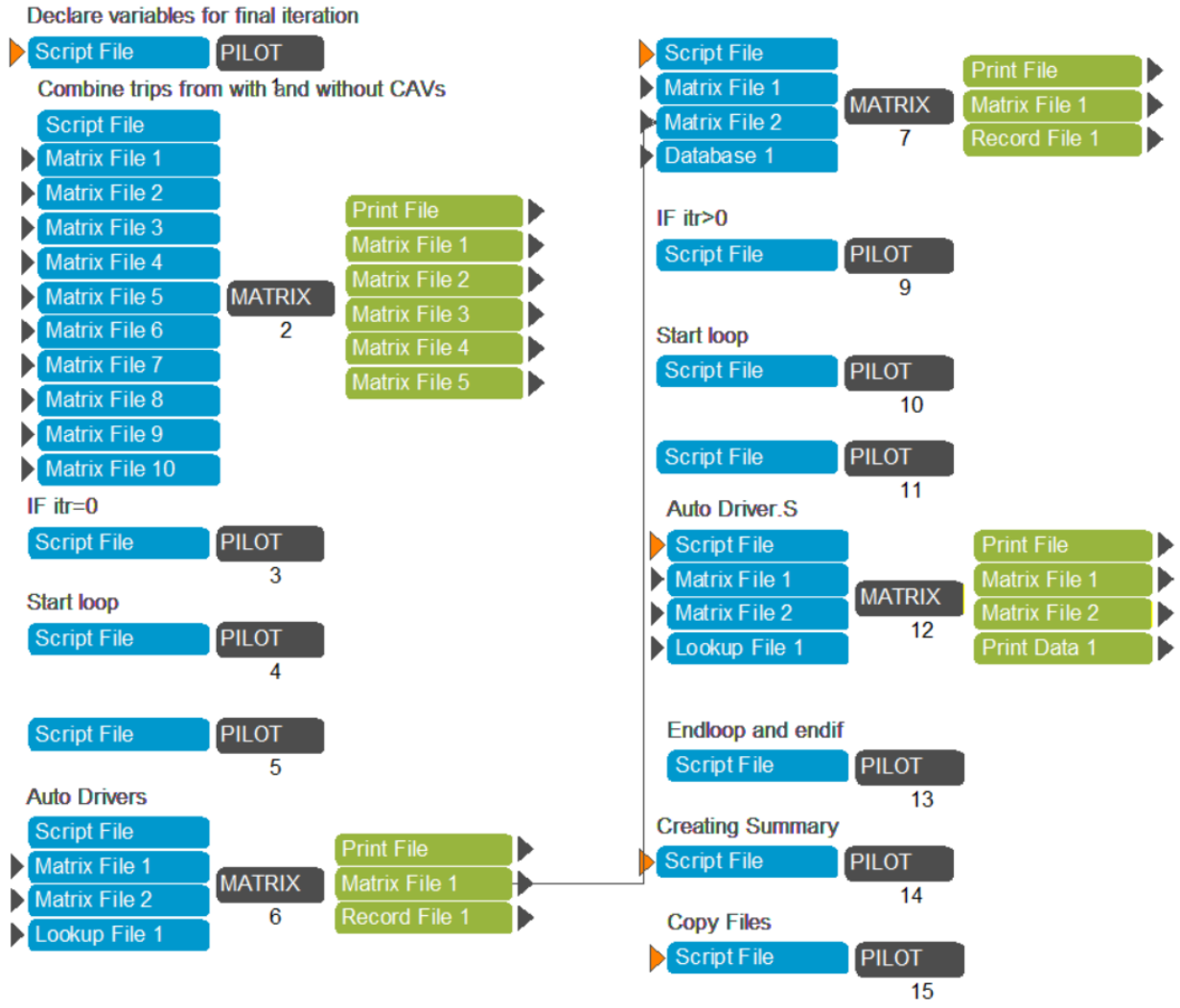


In Figure 4.9.3, loop 1 shows calibration loop which has been set to run just for one iteration as the trip mode choice is already calibrated to match the target shares. Step 2 creates a matrix file that shows the HBW constant for each origin destination pair. Step 3 uses skims, HBW trips segmented by income group, HBW constants, model coefficients, mode choice parameters and area type definitions to apply mode choice model to the input trips to create trip tables by trip modes. Step 4 creates a regional total trip summary. Steps 5 to Step 8 are only executed if the catalog key MC_Calibration is checked. Steps 5 to Step 8 summarize the trips by 21 jurisdictions and trip model and readjust the HBW constants by comparing the target and current mode shares. Step 9 creates the final trip tables which aggregates the commuter rail trips of KNR and PNR trip mode. The same above steps are followed for other trip purposes.

4.10 AUTO DRIVERS

The auto driver application Manager converts trip tables from mode choice to vehicle trips (see Figure 4.10.1).

Figure 4.10.1 Auto Drivers

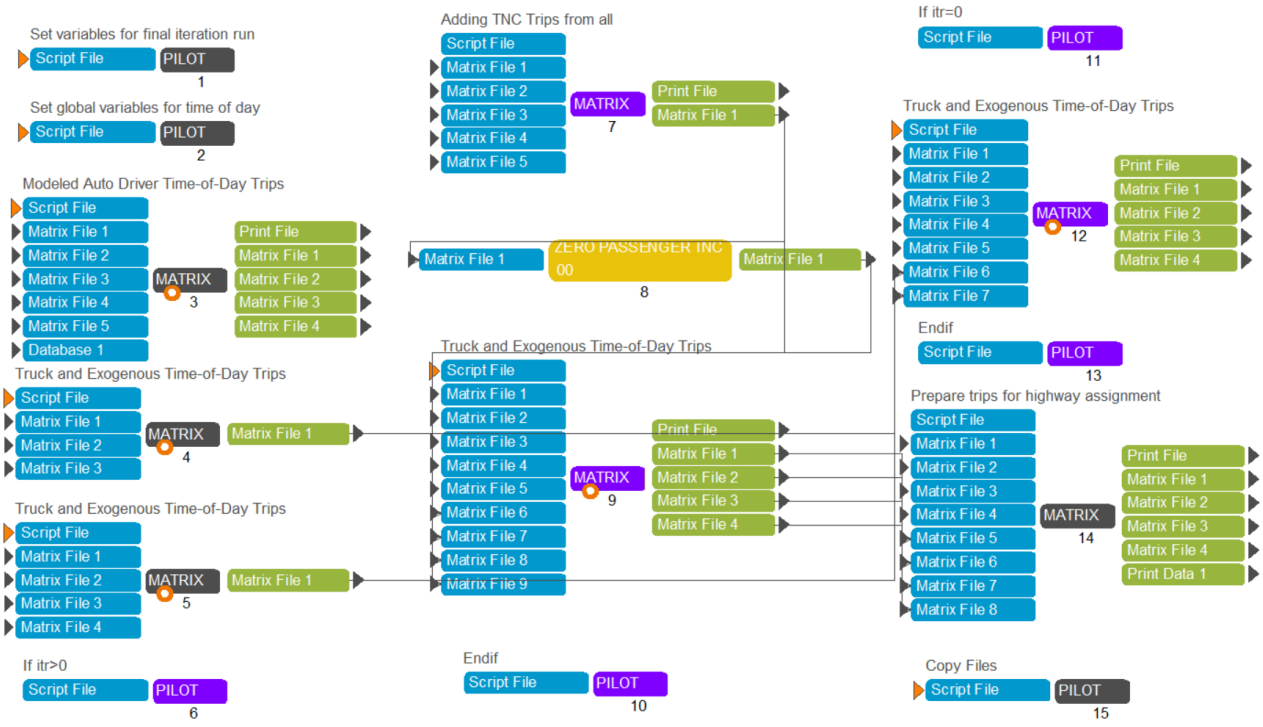


Step 1 declares global variables. Step 2 aggregates the mode choice outputs of households with/without CAVs to person trips for each trip purpose. Steps 3 to Step 7 are executed only for iteration 'pp' (pump prime). It generates the vehicle trips for each trip purpose. Steps 8 to Step 11 are executed for iterations 1 to 4. They generate the vehicle trips for each trip purpose.

4.11 TIME OF DAY

The time-of-day application manager converts the daily vehicles trips to time-of-day vehicles for the four modeled time periods (see Figure 4.11.1).

Figure 4.11.1 Time of Day

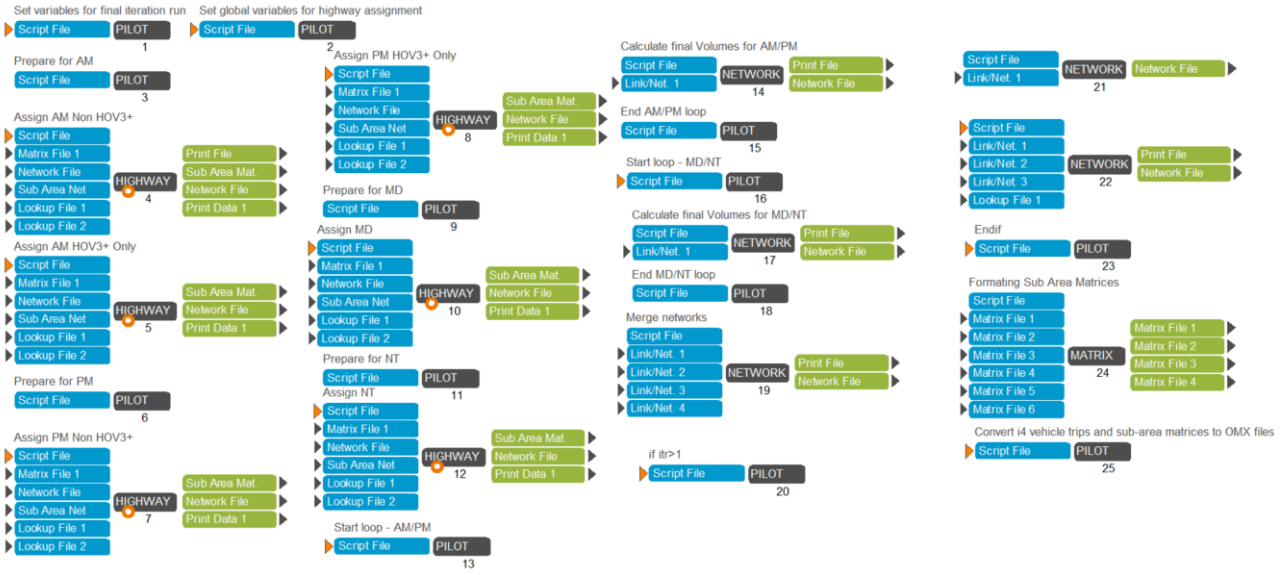


Step 1 declares global variables for final iteration. Step 2 declares global variables related to time-of-day application manager. Step 3 converts the vehicle trips to time-of-day trips. Step 4 and 5 converts the truck trips to time-of-day trips. Step 6 to Step 10 are executed for iterations one to four. Steps 7 and 8 create zero passenger TNC trips. Step 9 converts miscellaneous trips like external, visitor trips, airport trips, truck trips and zero passenger TNC trips to time of day miscellaneous trips. Steps 11 to 13 are executed for pump prime iteration. It again creates the miscellaneous time of day vehicle trips for the pump prime iteration. Step 14 merges the vehicle trips and miscellaneous vehicles trips into single vehicle trip files for each modeled time period.

4.12 HIGHWAY ASSIGNMENT

The highway assignment application manager assigns the time-of-day trips to the network. The assignment procedure is split into two parts: non HOV3+ (i.e., SOV, HOV2, trucks and airport passengers) and HOV 3+ (see Figure 4.12.1).

Figure 4.12.1 Highway Assignment

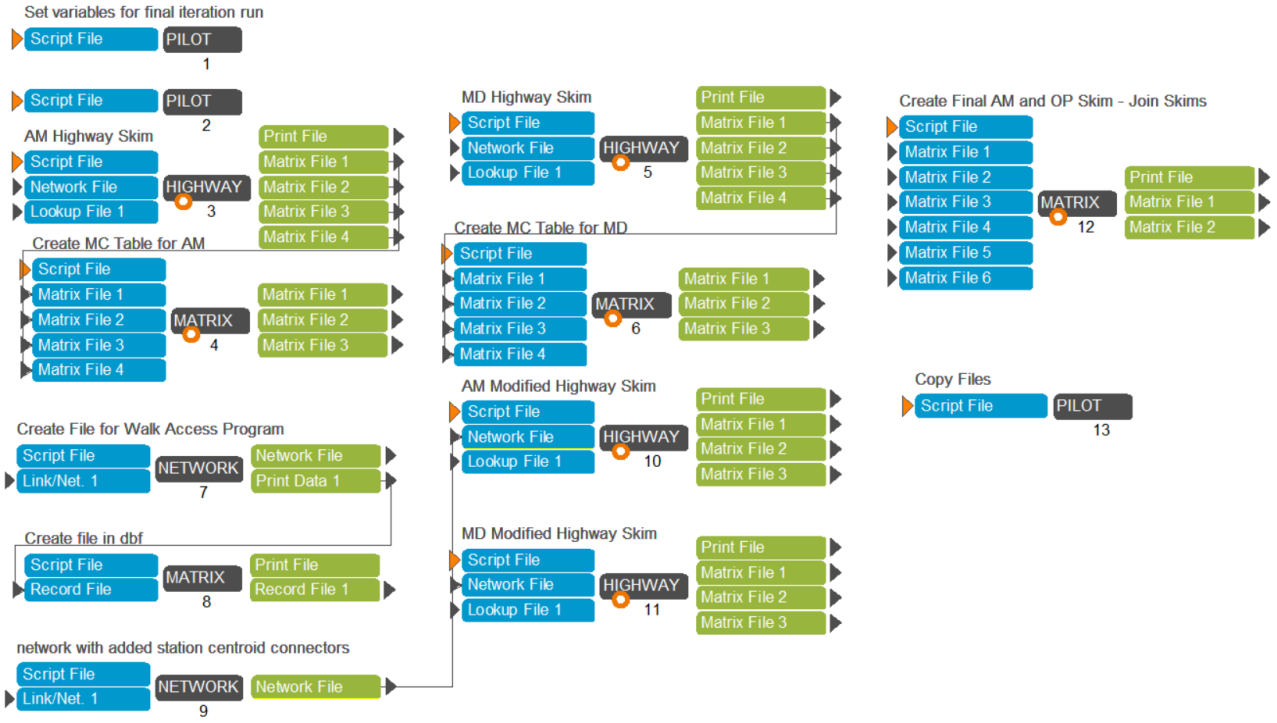


Steps 4 and 5 perform the highway assignment for AM time period. Steps 7 and 8 perform the highway assignment for PM time period. Steps 9 and 10 perform the highway assignment for MD time period. Steps 11 and 12 perform the highway assignment for NT time period. The final volumes on loaded network are added onto to the temporary network in Step 13 to Step 18. In the Step 19, all volumes for each time period are further added to a single loaded network. The attributes of the loaded network include volumes by each time period and vehicle class. Steps 20 to 23 average the volumes on the links for iteration greater than one. Step 24 aggregates the sub-area matrices from highway assignment for each time period.

4.13 HIGHWAY SKIM

The highway skim application manager is exactly same as the application manger 4.4 except that the current application manager is run for iterations within the feedback loop (see Figure 4.13.1).

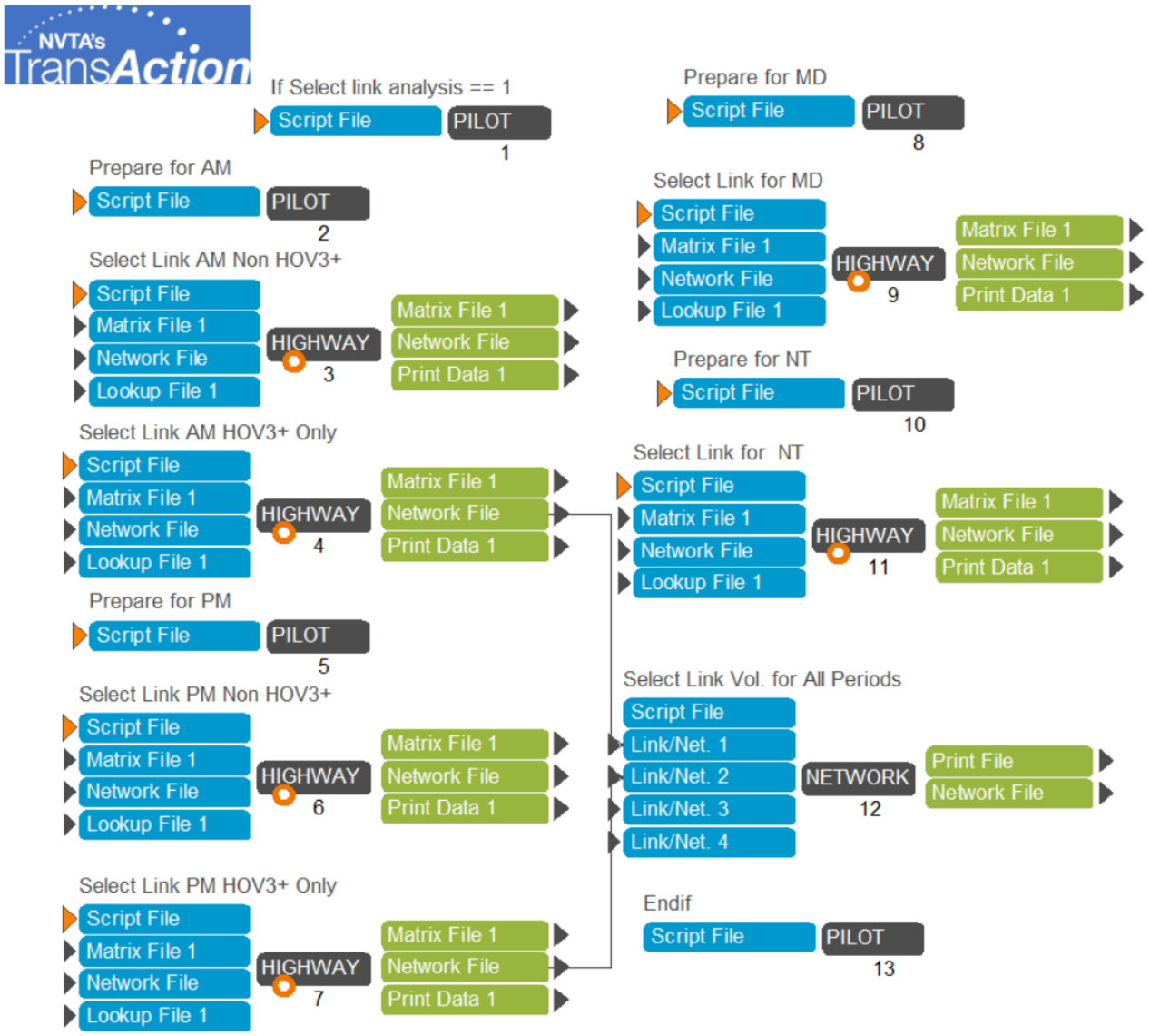
Figure 4.13.1 Highway Skim



4.14 SELECT LINK RUN

The select link application manager runs the select analysis on the selected link defined by the user in the catalog keys. The select link analysis is run only if the user has checked the select link analysis checkbox. It is also performed at the end of the iteration 4. Figure 4.14.1 shows the procedure of select link analysis.

Figure 4.14.1 Select Link Analysis

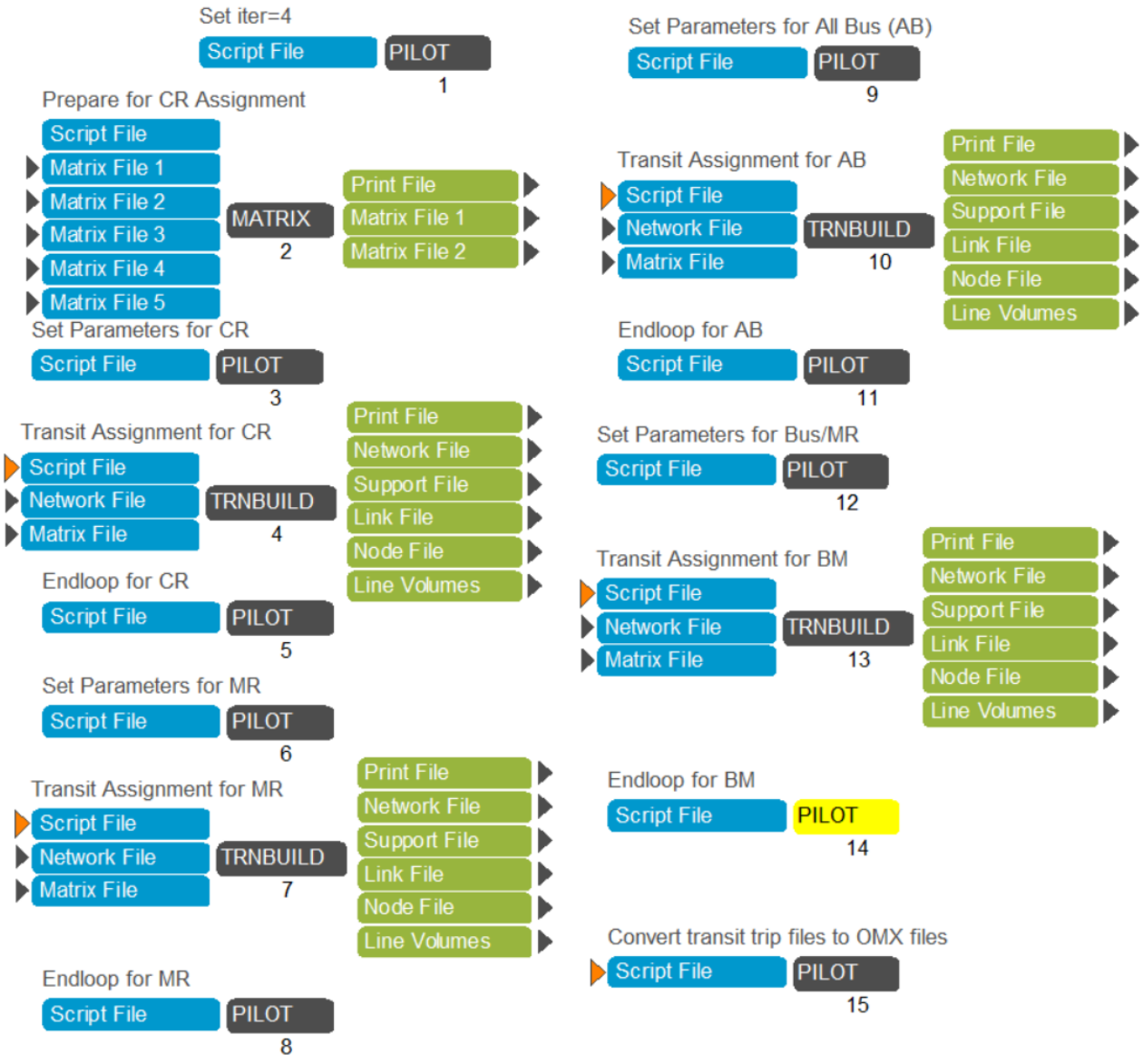


Steps 2 to 4, Steps 5 to 7, Steps 8 to 9 and Steps 10 to 11 conduct the select link analysis by running the highway assignment for AM, PM, MD and NT time periods. Step 12 aggregates the volumes on the select links to single network file.

4.15 TRANSIT ASSIGNMENT

The transit assignment application manager performs the transit assignment procedure on transit trips. It is conducted only for the final iteration. Figure 4.15.1 shows the flow chart of transit assignment procedure.

Figure 4.15.1 Transit Assignment

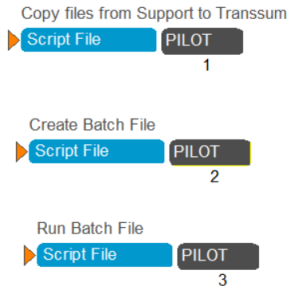


Step 2 prepares the transit trip tables for each transit mode. Pilot 3 defines the input for transit assignment for commuter rail. Transit build program of Step 4 conducts the transit assignment for commuter rail. Pilot 5 ends the loop for commuter rail transit assignment. Similarly, Steps 6 to 8, Steps 9 to 11, and Steps 12 to 14 conduct transit assignment for Metrorail, all bus, and bus-Metrorail, respectively.

4.16 TRANSIT SUMMARY

The transit summary application manager (see Figure 4.16.1) creates the batch files to summarize the boardings at each station and also generate transit assignment volume DBF files into two summary volume files: PK_VOL.dbf representing peak transit boardings and volumes and OP_VOL.dbf representing off-peak transit boardings and volumes.

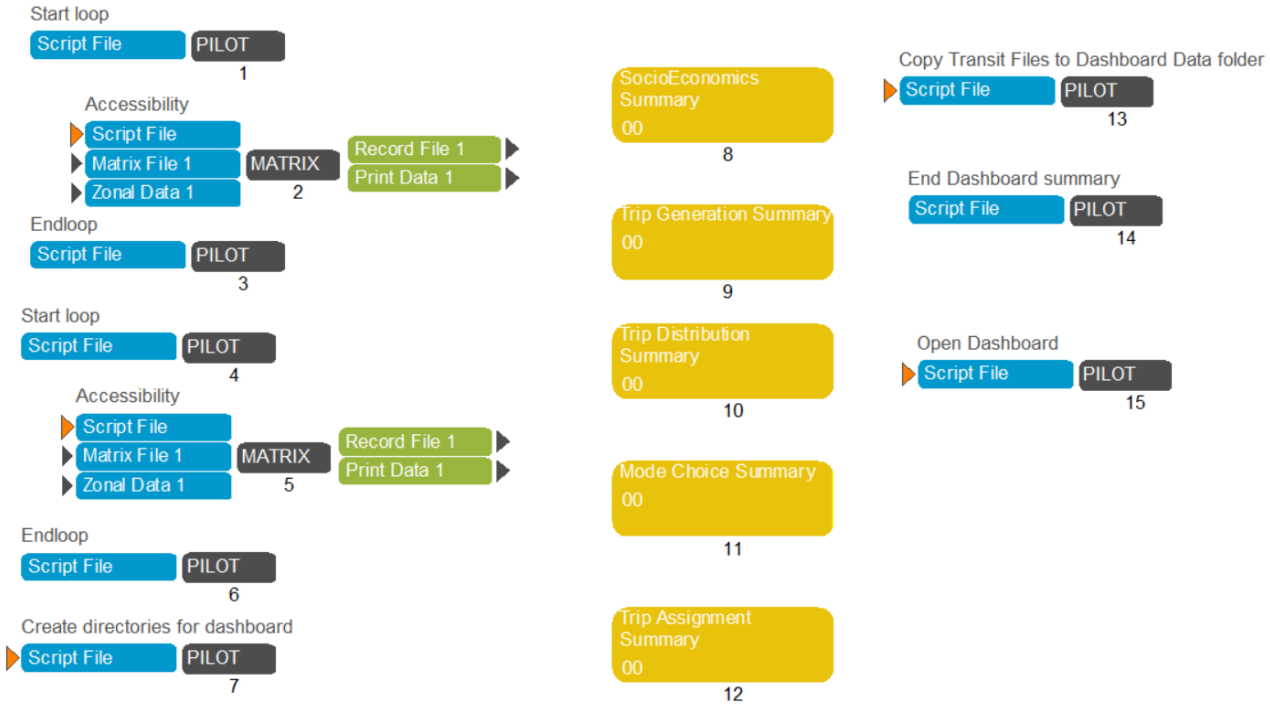
Figure 4.16.1 Transit Summary



4.17 SUMMARY STEPS

The summary steps application manager (see Figure 4.17.1) is run only if the user selects the RunDashboard check box in the catalog keys. This application summarizes various components of the model that are crucial to populate the fields of the RShiny dashboard. Steps 1 to 7 calculate accessibility for different travel time thresholds. Socioeconomic summary application manager summarizes the population, employments for the entire model domain, NVTA region, and other sub regions. Trip generation summarizes motorized/non-motorized trips by households with and without PCAVs. Trip distribution summary application manager summarizes the trips between the sub-regions. Mode choice summary application manager summarizes the trips by different modes and purposes for the entire model domain, NVTA region, and other sub-regions. The trip assignment summary application group summarizes the vehicle miles travelled and vehicle hours travelled in the entire region.

Figure 4.17.1 Summary Steps



5.0 Running the Macroscopic Model

The Cube model can be run in two different ways as discussed below. After the Cube model is run, the output matrix and network files are processed to use as inputs to the DTALite model. The detailed discussion on how to run the DTALite model can be found in Chapter 8.

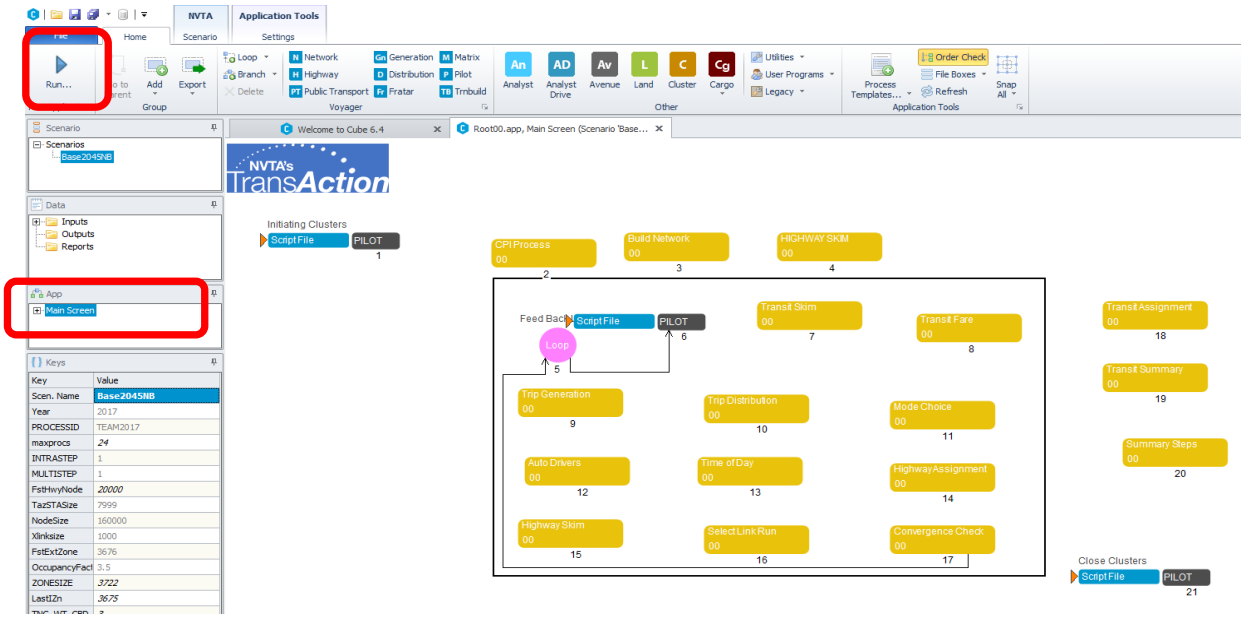
Method 1:

- Make sure all the files are present in the input folder of the scenario.
- Double click on the scenario on Cube catalog. A window will appear with scenario specific settings.
- Change the catalog keys as needed.
- Click on save and run. This will start running the scenario from the beginning.

Method 2:

- Click on main-screen application group shown in figure. This will open up the UI of the model as shown in Figure 4.17.1.
- Click the "Run" button at the top left of the Cube window as shown in Figure 4.17.1.

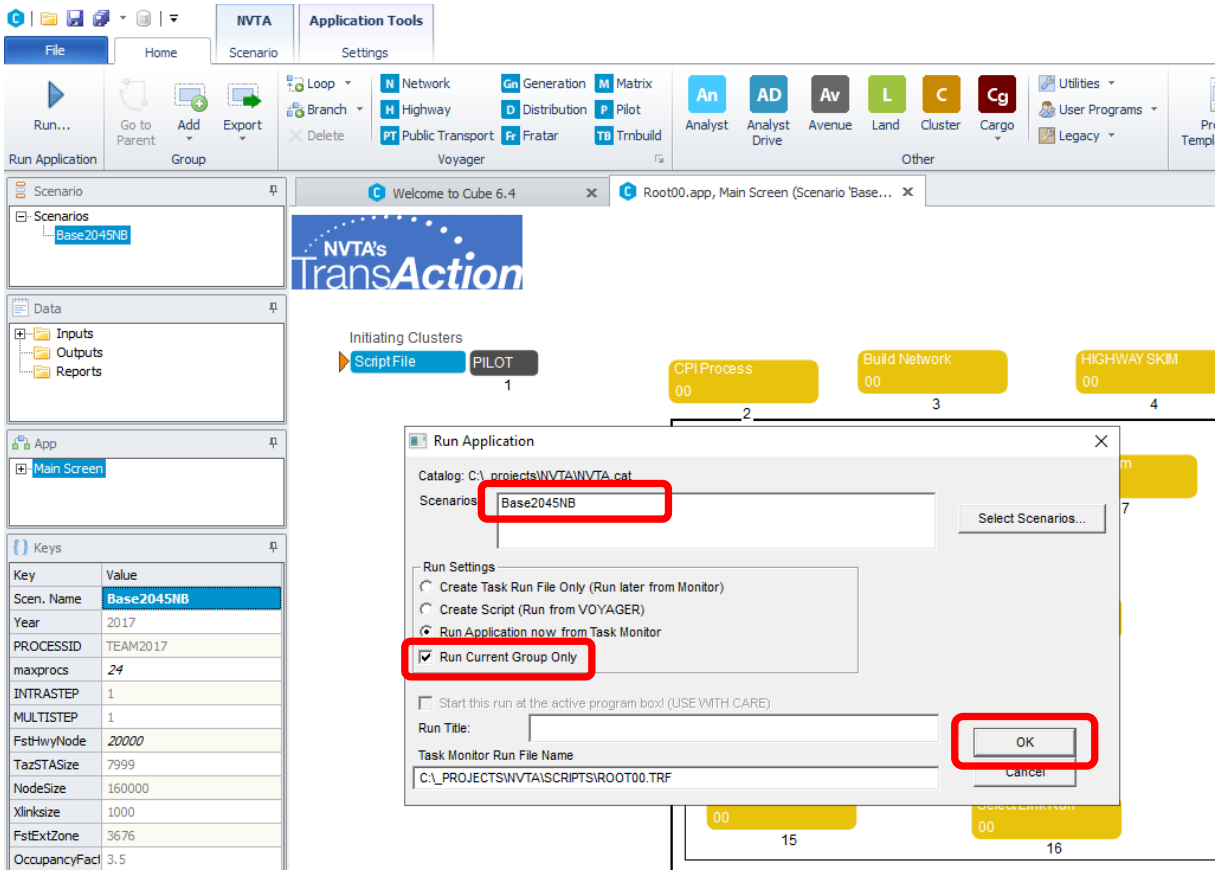
Figure 4.17.1 Cube application group



- Run Application window shall pop up

- Make sure to check "Run Current Group Only" and "Run Application now from Task Monitor" are selected.
- Make sure there is a scenario selected. "Scenarios" allows you to select other scenarios if required.
- Click on "OK" to start the run.
- For any further messages or pop up, click "OK" (see Figure 4.17.2).

Figure 4.17.2 Run Application



6.0Model Dashboard

This chapter describes the model dashboard that was developed to visualize some of major model results, as related to the NVTA region, the model domain, and individual jurisdictions in the NVTA region.

6.1 DASHBOARD

TransAction Model Dashboard displays summary statistics and performance measures of trip-making, including:

- Socioeconomics
- Trip Generation
- Trip Distribution
- Mode Choice
- Traffic Assignment
- Transit Ridership
- Congestion Level

The summaries can be shown for the model domain, the NVTA region, and jurisdictions in the NVTA region. The data used for tables and graphics in the Dashboard can be downloaded. See Figures 6.1.1 through 6.1.5 for example illustrations of the Dashboard.

Figure 6.1.1 TransAction Model Dashboard: an Example of Socioeconomics

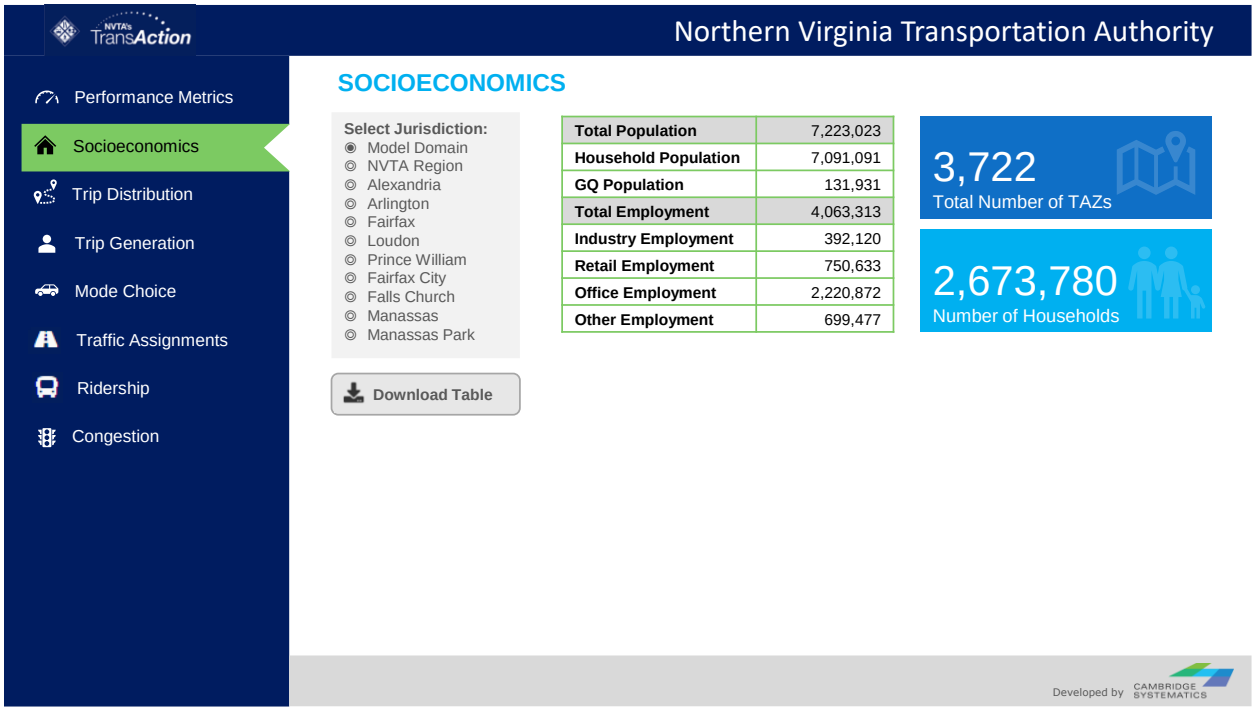


Figure 6.1.2 TransAction Model Dashboard: an Example of Trip Generation

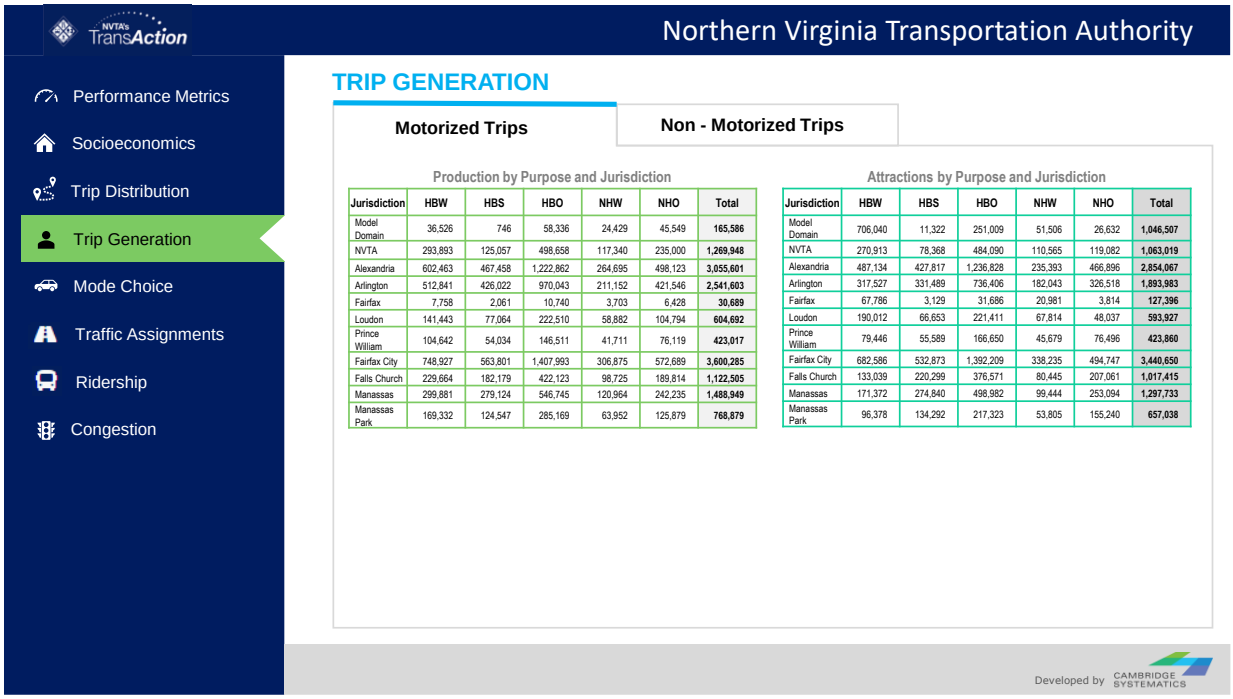


Figure 6.1.3 TransAction Model Dashboard: an Example of Trip Distribution

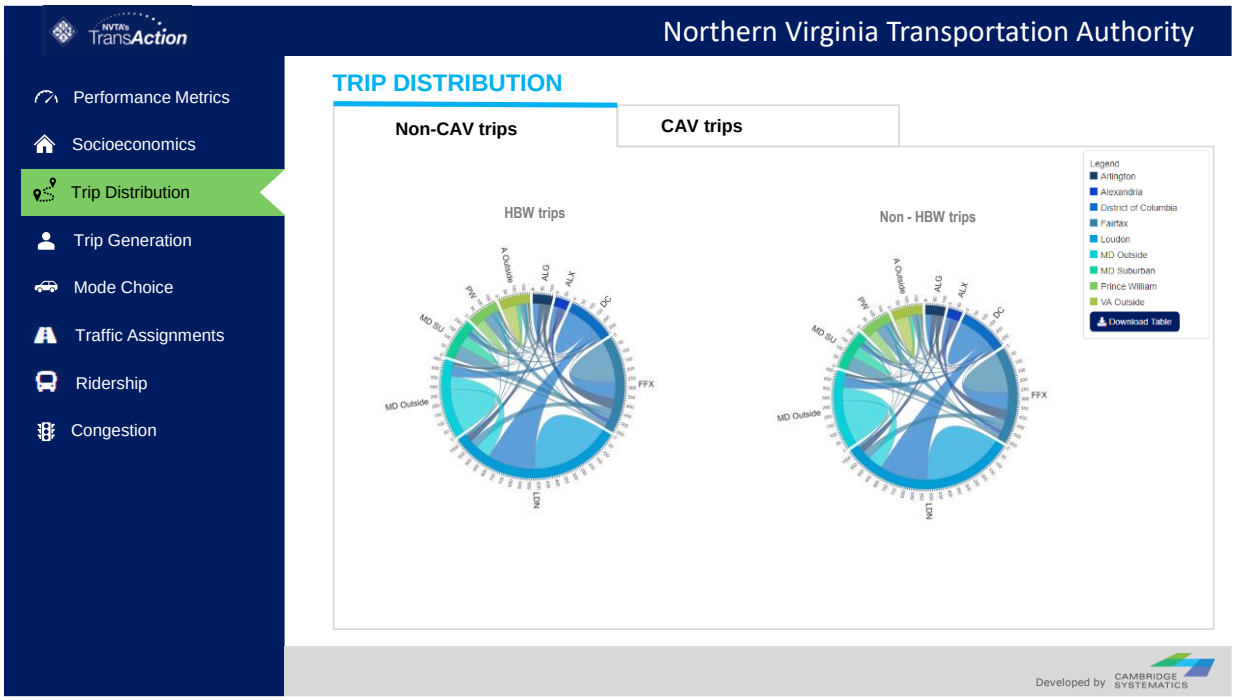


Figure 6.1.4 TransAction Model Dashboard: an Example of Mode Choice

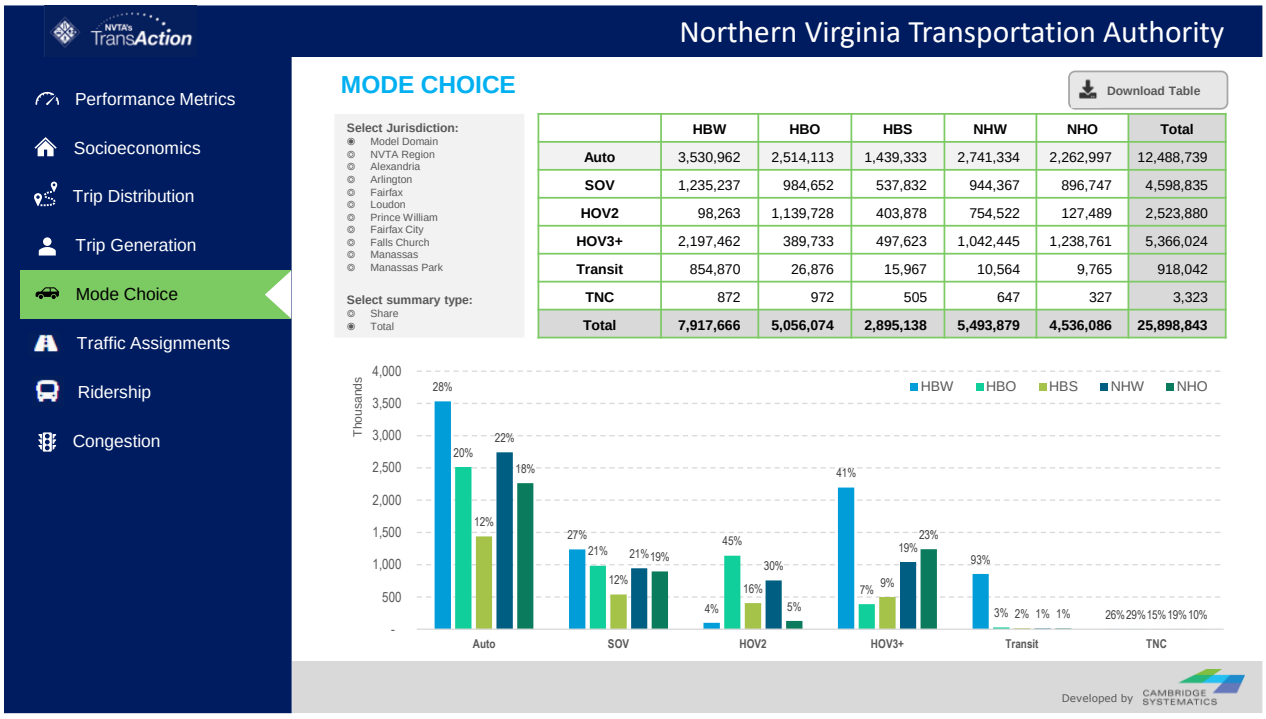


Figure 6.1.5 TransAction Model Dashboard: an Example of Traffic Assignments

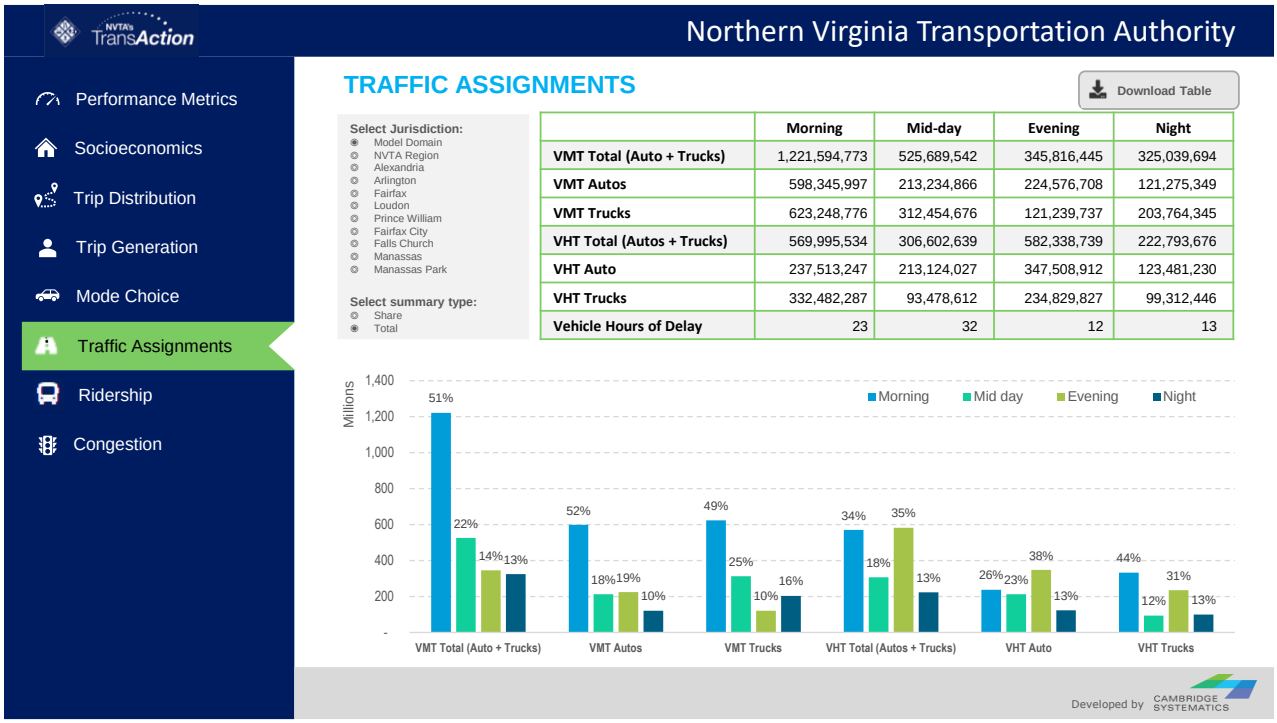
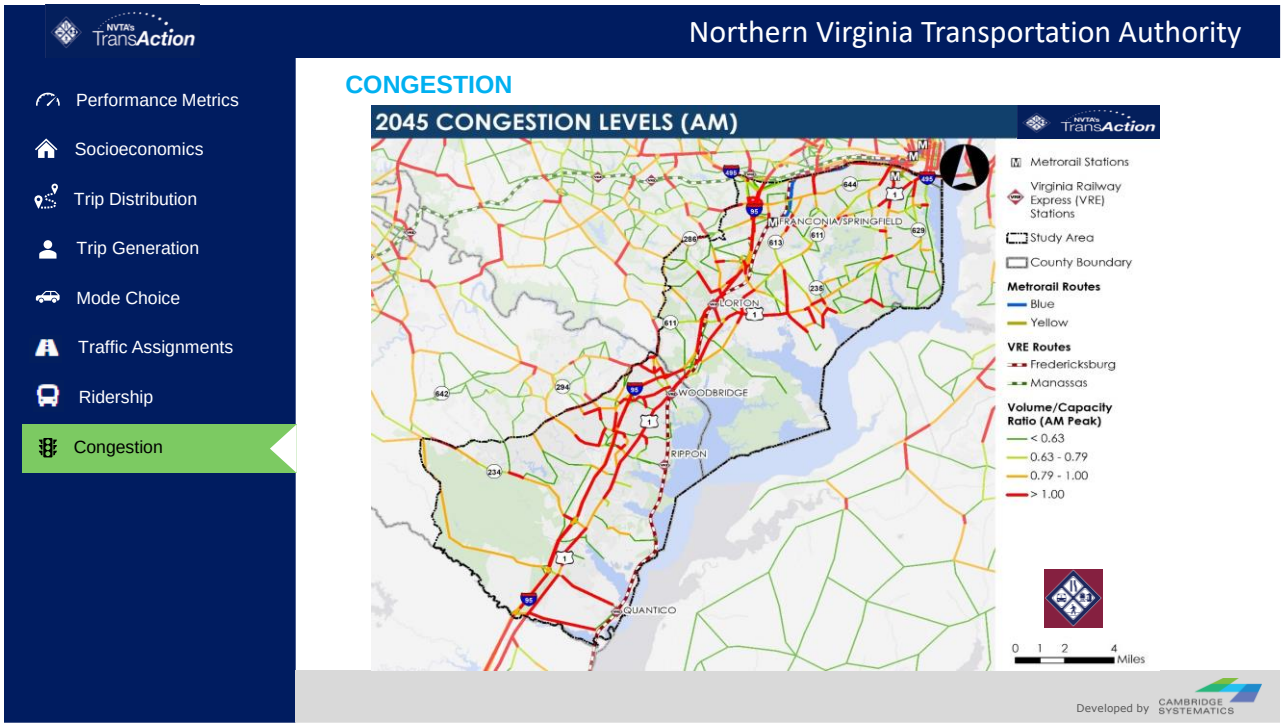


Figure 6.1.6 TransAction Model Dashboard: an Example of Congestion Level



6.2 REPORT AND RESULT SUMMARIES

In addition to the Model Dashboard, the TransAction Model generates detailed summaries and reports as part of a model run. For each model, the scenario "Outputs" folder has three sub-folders:

- Dashboard Data, which stores the data to be used for creating a dashboard for a scenario.
- Summary, which includes detailed tables and summaries of results from individual model components.
- Transum, which has detailed summaries for transit assignments.

7.0 Model Network

This chapter summarizes the processes of coding projects in the model networks (Cube and DTALite) in support of the TransAction Update and Six Year Program Update. Sample coding processes for different types of projects are outlined, but each project will need to be handled uniquely based on the project description. The network coding of projects starts with in Cube, which are then converted to the DTALite network.

7.1 CUBE MODEL NETWORK

Network coding in Cube can typically be completed either through Cube's graphic user interface, or by altering the input files directly. The following section highlights how the most common types of projects can be represented in the Cube network files.

Highway Network Coding

The highway network in Cube is represented by two files: link.dbf and node.dbf. The highway network node file, node.dbf (Table 7.1), contains the XY coordinates for both TAZ centroids and highway nodes. The link.dbf file contains the attributes of individual highway segments (links), which are shown in Table 7.2. A row record in this DBF file is uniquely defined by the A- Node/B-Node pair, with basic characteristics of individual highway segments including distance, the number of directional lanes by time-of-day period (??LANE), directional user limits (??LIMIT), and facility type (FTYPE). Full definitions of each field and attribute can be found in the MWCOG/TPB model documentation.

Table 7.1 Highway node file description (node.dbf)

Variable Name	Description
N	TAZ or Highway Node Number
X	X - Coordinates (NAD83-based in whole feet)
Y	Y- Coordinates (NAD83-based in whole feet)

Source: COG/TPB. 2020.

Different types of highway projects require different types of changes to the link and node files, including adding new links and/or nodes or changing attributes. Some of the more common types of projects and the required coding changes include:

- Roadway Widenings: Change the number of lanes (??LANE) by time period.
- HOV/HOT Conversion: Change the user limit code, as outlined in Table 7.3.
- New/Extended Roadways: Add rows to the link file and populate all attributes. Nodes may also need to be added, according to the numbers in Table 7.4.

- Interchanges: Based on any design graphics, new links and nodes are added to represent the interchange geometry. Facility Types are changed on links as appropriate.

Table 7.2 Base highway link file description (link.dbf)

Variable Name	Description
A	A-Node
B	B-Node
DISTANCE	Link distance (in whole miles w/explicit decimal)
JUR	Jurisdiction Code (0-23) *0/DC, 1/MTG, 2/PG, 3/ALR/, 4/ALX,5/ FFX, 6/LDN, 7/ PW, 8/(unused), 9/ FRD, 10/HOW, 11/AA, 12/CHS, 13/(unused), 14/CAR, 15/CAL, 16/STM, 17/ KG, 18/FBG, 19/STF, 20/SPTS, 21/FAU, 22/CLK, 23/JEF*
SCREEN	Screenline Code
FTYPE	Link Facility Type Code (0-6)
	*0/centroids, 1/Freeways, 2/Major Art., 3/Minor Art, 4/ Collector, 5/*Expressway, 6/ *Ramp*
TOLL	Toll Value in current year dollars
TOLLGRP	Toll Group Code
AMLANE	AM Peak No. of Lanes
AMLIMIT	AM Peak Limit Code (0-9)
PMLANE	PM Peak No. of Lanes
PMLIMIT	PM Peak Limit Code (0-9)
OPLANE	Off-Peak No. of Lanes
OPLIMIT	Off-Peak Limit Code (0-9)
EDGEID	Geometry network link identifier
LINKID	Logical network link identifier

Variable Name	Description
NETYEAR	Planning year of network link
SHAPE_LENGT	Geometry length of network link (in feet)
PROJECTID	Project identifier

Source:COG/TPB. 2020. Highway and Transit Networks used in the Air Quality Conformity Analysis of the 2020 Amendment to Visualize 2045 and the FY 2021-2024 TIP (Ver. 2.3.78 Travel Model).

Table 7.3 Limit codes

Limit Code	Vehicles Allowed
0	All Vehicles
2	HOV 2+ Occ. Vehicles
3	HOV 3+ Occ. Vehicles
4	All Vehicles, other than trucks
5	Airport Passenger Auto Driver Trips
9	Transit Only

Source:COG/TPB. 2020.

Table 7.4 Allocated highway node ranges by jurisdiction

Jurisdiction	Beginning Node	Ending Node	Allocated Nodes
Arlington Co., Va.	30000	31999	2000
City of Alexandria, Va.	32000	33999	2000
Fairfax Co. Va.	34000	37999	4000
Loudoun Co., Va.	38000	39999	2000

Prince William Co., Va.	40000	41999	2000
Reserved Nodes	90000	90999	1000

Source:COG/TPB. 2020.

Transit Network Coding

Transit coding in Cube can be significantly more complex than highway coding and requires changes to a wider range of input files as shown in Table 7.5 depending on the transit sub-mode being coded. This memo does not provide a full instruction manual on transit coding in Cube but highlights important coding changes that will be made for common transit project types.

Table 7.5 Transit network input files

Filename	Description	Type
Station.dbf	Station file: Metrorail, Comm. Rail, LRT stations/PNR lots & bus PNR lots	DBF
AreaWalk.txt	Used to calculate zonal percent walk to transit values	Text
met_node.tb	Metrorail stations	Text
com_node.tb	Commuter rail stations	Text
lrt_node.tb	LRT stations/stops	Text
new_node.tb	BRT/streetcar stations/stops	Text
met_pnrn.tb	Metrorail PNR lots	Text
com_pnrn.tb	Commuter rail PNR lots	Text
bus_pnrn.tb	Bus PNR lots	Text
lrt_pnrn.tb	LRT PNR lots	Text
new_pnrn.tb	BRT/streetcar PNR lots	Text
met_link.tb	Metrorail links	Text
com_link.tb	Commuter rail links	Text

Filename	Description	Type
lrt_link.tb	LRT links	Text
new_link.tb	BRT/streetcar links	Text
com_bus.tb	Transfer link (walk) between commuter rail station and bus & LRT stop	Text
lrt_bus.tb	Transfer link (walk) between LRT station and bus stop	Text
new_bus.tb	Transfer link (walk) between BRT/streetcar stop and bus stop	Text
MODE1AM…,MODE10AM.tb	AM transit line files	Text
MODE1OP…,MODE10OP.tb	Off-peak transit line files	Text
com_bus.tb	Transfer link (walk) between commuter rail station and bus & LRT stop	Text
lrt_bus.tb	Transfer link (walk) between LRT station and bus stop	Text

Source:COG/TPB. 2020.

Transit service changes (changes to frequency, service span, or routing within existing transit right-of-way) are implemented in the transit line files (e.g., MODE??_AM). These files are divided by mode, operator, and service type as defined in Table 7.6 and include information about service levels and routing by time of day. Routing for bus services (modes 1,2, and 6-9) are defined by highway nodes from the node.dbf file, while other modes require more input files and coding. New or increased bus service can be implemented in the Cube model by only editing the transit line files.

Table 7.6 Transit mode codes

Mode Code	Mode Description
1	Local bus: Metrobus (also includes DC Circulator bus)
2	Express bus: Metrobus
3	Metrorail

Mode Code	Mode Description
4	Commuter rail
5	Light rail
6	Local bus: Other primary service (inner jurisdictions)
7	Express bus: Other primary service (inner jurisdictions)
8	Local bus: Other secondary service (outer jurisdictions)
9	Express bus: Other secondary service (outer jurisdictions)
10	BRT/streetcar

Source:COG/TPB. 2020.

New fixed guideway projects such as Metrorail, commuter rail, light rail, bus-rapid transit (BRT), and streetcar, require additional coding to implement including changes to the transit line files, addition of transit-only links to the network (???_link.tb), and additions/changes to the transit station file (station.dbf). Table 7.7 lists the variables included in the station file, which must be included for all fixed-guideway transit stations, and any park-and-ride stations (regardless of transit mode).

Table 7.7 Variables in Transit Station file (Station.dbf)

Name	Type	Field Description
SEQNO	N	Sequence Number
MM	C	Mode Code (M=Metrorail, C=Commuter rail, B=Bus, L=Light rail, N= BRT/streetcar)
NCT	N	Access distance code (1, 2, 3, 0, 9, 8)
STAPARK	C	Does the station have a park-and-ride lot? (Y=yes; blank=no)
STAUSE	C	Is the station in use for the given year? (Y=yes; blank=no)
SNAME	C	Station Name/PNR lot name
STAC	N	Station centroid number (5001-7999), also known as a park-and-ride (PNR) lot centroid or a dummy PNR centroid"

Name	Type	Field Description
STAZ	N	For the purposes of path building, the TAZ (1-3722) that represents the location of the station PNR lot. Usually, the closest TAZ to the PNR lot.
STAT	N	Station Node (8000-8999, 9000-9999, 10000-10999)
STAP	N	Station park-and-ride (PNR) node number (11000-13999)
STAN1	N	Station bus node #1 (used to generate a station-to-bus-node connector)
STAN2	N	Station bus node #2 (used to generate a station-to-bus-node connector)
STAN3	N	Station bus node #3 (used to generate a station-to-bus-node connector)
STAN4	N	Station bus node #4 (used to generate a station-to-bus-node connector)
STAPCAP	N	Parking capacity (number of spaces at the PNR lot)
STAX	N	X coordinate of station/PNR lot (MD State Plane, NAD83, feet)
STAY	N	Y coordinate of station/PNR lot (MD State Plane, NAD83, feet)
STAPKCOST	N	Peak period parking cost (daily cost, cents)
STAOPCOST	N	Off-peak parking cost (hourly cost, cents)
STAPKSHAD	N	Peak-period shadow price **(currently not used)**
STAOPSHAD	N	Off-peak-period shadow price **(currently not used)**
FIRSTYR	N	Year of Station/PNR lot Opening (unused by scripts, but used as metadata)
STA_CEND	N	Project ID (Metadata)
	C	Scenario name, or left blank (Metadata)
	C	Comments, if any, regarding the file, since file cannot accept comment lines preceding the data lines

Source:COG/TPB. 2020.

Coding transit stations can be complex and requires the addition of multiple nodes (within specific numerical ranges as shown in Table 7.8), and access links to ensure connections between bus stops, rail stations, and park-and-ride facilities all function appropriately. Stations with park-and-ride facilities require additional detail including any pricing and capacity assumptions.

Table 7.8 Station centroid and station node range by mode

Mode	Mode Code	Station Centroid Range	Station Node Range
Metrorail (Mode 3)	M	5000-5999	8000-8999
Commuter rail (Mode 4)	C	6000-6999	9000-9999
Light rail transit (Mode 5)	L	7000-7999	10000-10499
Bus rapid transit/streetcar (Mode 10)	N	Not used	10500-10999
Bus (Modes 1, 2, 6-9)	B	Not used	Not used

Source:COG/TPB. 2020.

Transit priority projects can be coded in two ways, depending on the level of prioritization. In the case of fully dedicated transit lanes, new transit only links (and associated nodes) can be added to the highway network. In the case of other prioritization treatments (e.g., queue jumps or signal priority), the route run times which are hardcoded in the transit line files can be updated to represent the expected travel time savings.

Coding Other Project Types

The Cube model network does not include a pedestrian or bicycle network, and those projects will not be included in the model network. Bicycle network improvements will be considered off-model as part of the Accessibility performance measures (see the Performance Measures Methodology Memo for more details). Signal improvements at intersections are also not able to be represented in the Cube networks but can be addressed in the DTALite Network. ITS projects are also best handled in the DTALite model, although it is possible that the impacts of some large-scale ITS projects could be represented in the Cube model through changes to the link-level attributes discussed in the section on highway project coding. Each project will be considered individually to identify the best way to represent it in the modeling tools.

7.2 DTALITE NETWORK

Highway Network Coding

The DTALite network uses General Modeling Network Specification (GMNS), in which all GMNS data files are in CSV format. A generic network used for GMNS includes a set of three layers: node (Table 7.9), link (Table 7.10) and movement (Table 7.11). Roadway network data defines the basic node-link structure, along with attributes for each link and node. A link is defined using upstream node and downstream node IDs, with essential attributes such as length, free-flow speed, lanes, and capacity, typically required for traffic assignment. Additionally, nodes are related to movements, which contain the individual's movement from node to node.

Table 7.9 Node Data Structure (Node.csv)

Field Name	Description	Sample Value
name	Optional for visualization only	Main street @ Highland Dr.
node_id	Node identification number	1001
ctrl_type	Intersection control type	5
node_type	Optional text label for visualization and identifies of node	1
x_coord	Longitude or horizontal coordinate in any arbitrary geographic coordinate system.	100
y_coord	Latitude or vertical coordinate horizontal coordinate in any arbitrary geographic coordinate system	200
geometry	Text string used to describe node location https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry	

Source: ASU, 2021, User Guide for NeXTA for GMNS.

Table 7.10 Link Data Structure (Link.csv)

Field Name	Description	Sample Value
Name	Optional for visualization purposes	Main Street
link_id	Link identification number of the road	101

Field Name	Description	Sample Value
from_node_id	Upstream node number of the link, must already defined in input_node.csv	2
to_node_id	Downstream node number of the link, must already defined in input_node.csv	3
link_type	Optional text label for visualization and data checking purposes	1
length	The length of the link (between end nodes), measured in units of miles.	1.0
lanes	The number of lanes on the link	2
free_speed	Free-flow speed on defined link. Suggested Unit: mph or kmph	20
capacity	The number of vehicles per hour per lane.	1500
geometry	Text string used to describe link shape and location (typically in WKT geographic coordinate system). The initial value can be empty, and NeXTA will generate the text string based on the coordinates of upstream and downstream nodes.	LINESTRING (30 10, 10 30, 40 40)

Source: ASU, 2021, User Guide for NeXTA for GMNS.

Table 7.11 Movement Data Structure (Movement.csv)

Field Name	Description	Sample Value
mvmt_id	Movement identification number	1
node_id	Node identification number	1001
name		Main Street

Field Name	Description	Sample Value
ib_link_id	upstream link identification number of the movement	100002 1
ib_lane	Lane number of inbound link	1
ob_link_id	Downstream link identification number of the movement	1 100002
ob_lane	Lane number of outbound link	1
type	Optional text label for visualization and identifies the direction of movement	U-Turn
penalty		50
capacity	Maximum service flow rate for each lane of the movement, in vehicles per hour.	1500
ctrl_type	Intersection control type	2

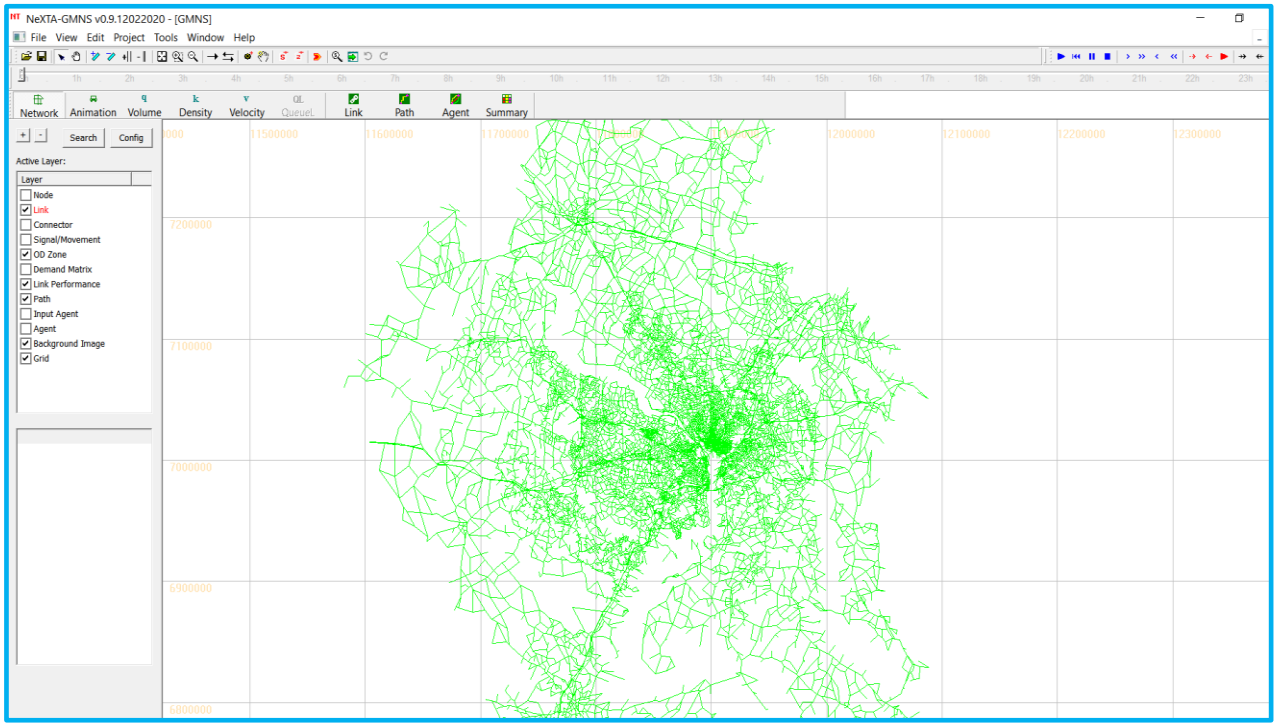
Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS.

The roadway network is visualized (see Figure 7.2.1) and validated in NeXTA (Network EXplorer for Traffic Analysis), which is a graphical user interface to facilitate preparation, post-processing and analysis of simulation-based dynamic traffic assignment datasets. NeXTA Version 3 uses DTALite as fast dynamic traffic assignment engine for transportation network analysis. NeXTA has been used as a multiresolution data hub in the FHWA analysis, modeling, and simulation (AMS) data hub concept of operations project. NEXTA is distributed as a free open-source software package for transportation analysis and simulation.

In general, for projects that are already coded in the CUBE network, the CUBE network files will be converted to the DTALite file format. The types of projects include:

- Roadway Widenings: the number of lanes by time period.
- HOV/HOT Conversion: "allowed use" in DTALite is in correspondence with the user limit code in CUBE.
- New/Extended Roadways: Add rows to the link file and populate all attributes. Nodes may also need to be added.
- Interchanges: Based on any design graphics, new links and nodes are added to represent the interchange geometry, with link attributes populated.

Figure 7.2.1 Network Visualization and Management in NeXTA



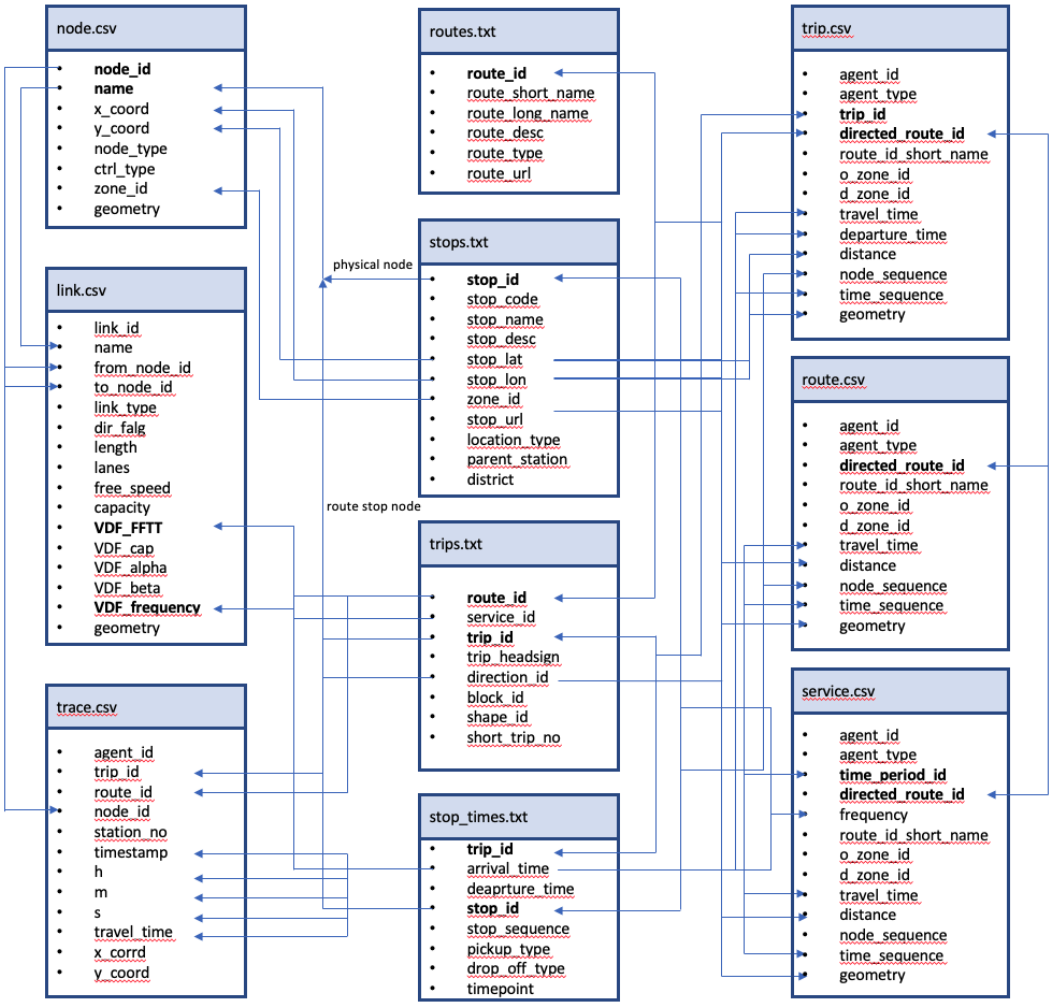
Transit Network Coding

The transit network was developed using the General Transit Feed Specification (GTFS) data from the transit service providers in the region. The GTFS defines a common format for public transportation schedules and associated geographic information, including the information of basic routes, trips, stops and stop times. The GTFS data files (e.g., trip.txt, route.txt, stop_times.txt. and stops.txt) were converted into the standard node and link network files using the GMNS, which follows the data structure and conversion flows outlined in Figure 7.2.2.

The node data table (see Table 7.12) includes eight columns in which each row represents an individual node, and the columns contain the attributes of each node. The field "node_id" divides into 2 parts containing physical stops and route stops. The name of the physical stop is the original stop_id in standard GTFS, and the name of the route stop combines 3 keys, route_id, direction_id and stop_id, in standard GTFS. The x_coord and the y_coord are the longitude and latitude. The purpose of node_type is to provide a reference distinguishing the physical stops and the route stops. Other attributes include the node control type (e.g., signal) and the zone_id in which the node is located.

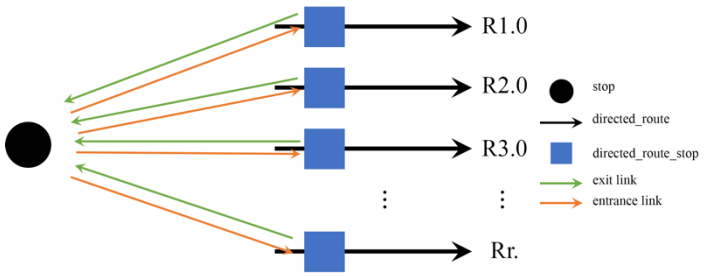
For each directed route stop node, there are service links to connect the corresponding transit stop (Figure 7.2.3). The entrance link is a boarding link, defined with the cost of the entrance including the passenger's waiting time. The exit link is a deboarding link whose cost is zero.

Figure 7.2.2 Transit Network Data Conversion from GTFS (in the Middle) to DTALite



Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS Model.

Figure 7.2.3 Relationship between physical stop and route stops



Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS Model.

Table 7.12. Transit Node Data Structure Example (Node.csv)

node_id	name	x_coord	y_coord	node_type	ctrl_type	zone_id	geometry
1	1	-112.1122625	33.47016013	stop		1	
2	2	-112.1109737	33.74600419	stop		2	
3	3	-112.1403316	33.8749822	stop		3	
4	4	-112.1510393	34.11327557	stop		4	
10001	R1.0.1	-112.1123625	33.47006013	directed_route_stop			
10002	R1.0.2	-112.1110737	33.74590419	directed_route_stop			
10003	R1.0.3	-112.1404316	33.8748822	directed_route_stop			
10004	R1.0.4	-112.1511393	34.11317557	directed_route_stop			
10005	R6.0.2	-112.1124625	33.46996013	directed_route_stop			
10006	R6.0.3	-112.1111737	33.74580419	directed_route_stop			

Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS Model.

Table 7.13. Transit Node Data Structure Example (Link.csv)

name	from_node_id	to_node_id	link_type	dir_flag	length	lanes
1->2	1	2	physical link	1	20	1
R1.0.1.entrance	1	10001	entrance	1	0	1
R1.0.1.exit	10001	1	exit	1	0	1
R1.0.1->R1.0.2	10001	10002	transportation	1	20	1

free_speed	capacity	VDF_FFTT1	VDF_cap1	VDF_alpha1	VDF_beta1	VDF_frequency1

name	from_node_id	to_node_id	link_type	dir_flag	length	lanes
60	49500	0.33	49500	0.15	4	1
60	49500	0.25	49500	0.15	4	6
60	49500	0	49500	0.15	4	6
60	49500	0.33	49500	0.15	4	6

Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS Model.

The link data table (see Table 7.13) includes fourteen columns in which each row represents an individual link, and the columns contain the attributes of that link. The name is divided into 4 parts containing the physical links, entrance links, exit links and transportation links. The physical links means the links of the physical network. The entrance link and exit link are used to connect with directed route stops and physical stops. The transportation links are the segment between two directed route stops. The VDF_FFTT (in minutes) is the free flow travel time of this link. The field “VDF_frequency” is the total number of trips along a certain route over this time period.

The service data table (see Table 7.14) includes thirteen columns in which each row represents an individual service. Each route over a time period provides a single service, and the columns contain the information of each route. The field "time_period_id" means which time period this service belongs to. The field "directed_route_id" is combined by two keys, the indexes of route_id and direction_id, in standard GTFS data. The table also includes the information of frequency of a certain route for each service. The field "node_sequence" represents the directed route stop sequence of each route. The corresponding time sequence and latitude and longitude for each directed route stop are filled in the fields "time_sequence" and "geometry", respectively. In addition, the link data table includes the information of the agent ID, agent type, the short name, total travel time and distance of each route, and the zone IDs original stops and destination stops of each route belong to.

Table 7.14. Transit Node Data Structure Example (Service.csv)

agent_id	agent_type	time_period_id	directed_route_id	frequency	route_id_short_name	o_zone_id
1	transit	0	R1.0	6	route 1	1
2	transit	0	R2.0	6	route 2	2
3	transit	1	R1.0	3	route 1	1
4	transit	1	R2.0	6	route 2	2

agent_id	agent_type	time_period_id	directed_route_id	frequency	route_id_short_name	o_zone_id
d_zone_id	travel_time	distance	node_sequence	time_sequence	geometry	
4	1	50	10001;10002;10003;10004			
3	0.35	10	10005;10006			
4	1	50	10001;10002;10003;10004			
3	0.35	10	10005;10006			

Source: ASU, 2021, User Guide for NeXTA for GMNS GTFS-GMNS Model.

8.0DTALite

8.1 MODEL FUNCTIONALITY

Model Architecture

The enhanced COG/TPB model will estimate travel demand by modes and by time periods, under a static user equilibrium assignment process, for the model area. These demands will be refined before use in the DTALite mesoscopic modeling which uses a dynamic traffic assignment process to assign traffic more accurately across the regional network. The process and architecture are displayed in Figure 8.1.1, with components listed below:

a. Network Data includes two essential files, node.csv and link.csv for the mesoscopic network representation.

b. OD Demand Meta Database includes the setting.csv as the configuration file that describes information such as agent type, demand period, and demand file list, which help users to represent the OD demand information for different user types at specific demand periods.

c. Traffic Assignment Module includes the key steps of the assignment, including the BPR Volume Delay Function, Shortest Path Tree Generation, and Flow Assignment, which generates the path flow and link flow according to the User Equilibrium principle.

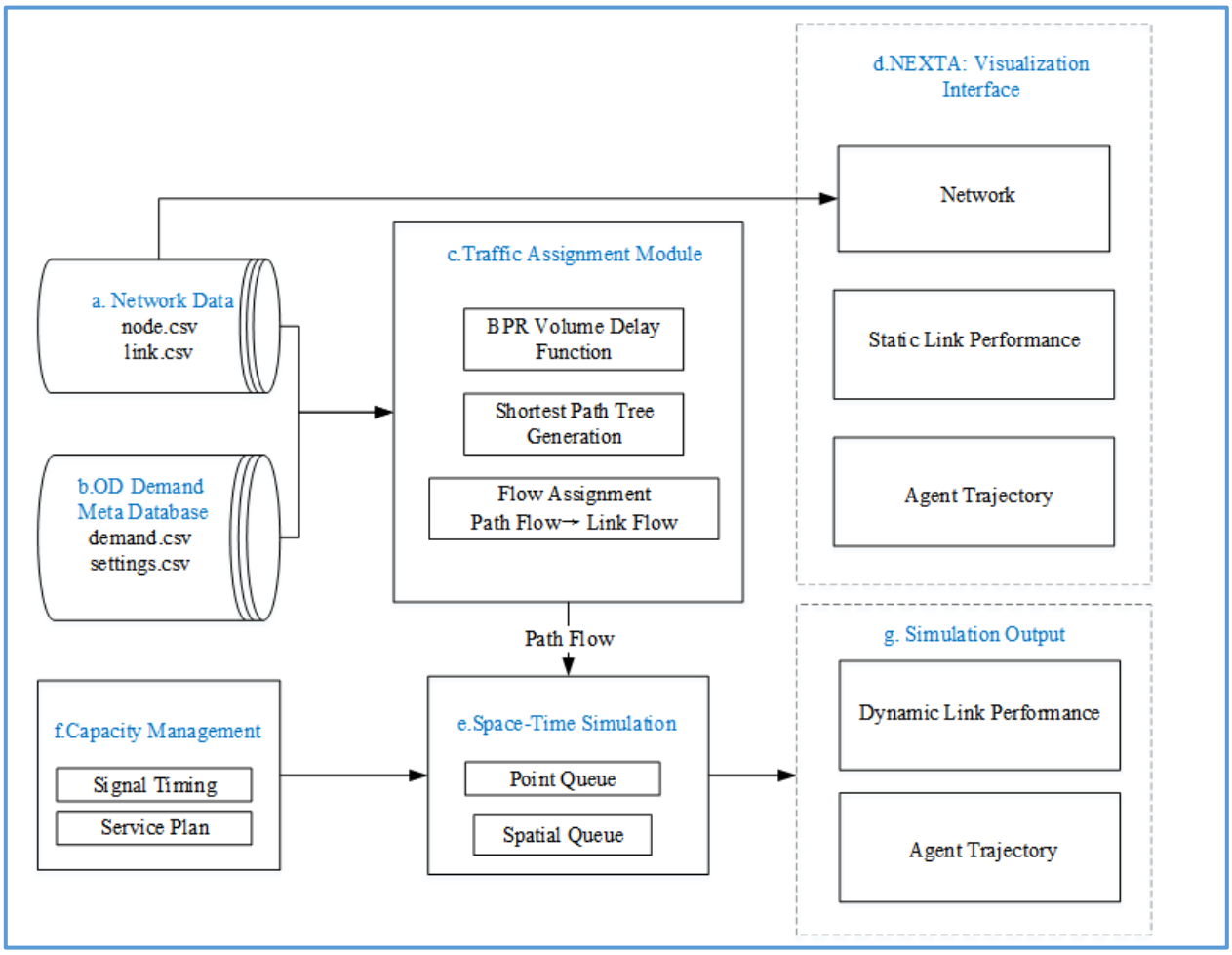
d. NEXTA: Visualization Interface Module is able to visualize the network and the output of traffic assignment, including Static Link Performance and Agent Trajectory.

e. Space-Time Simulation Module utilizes the path flow output of Traffic Assignment Module to perform Space-Time Simulation. The underlying traffic flow models in the Space-Time Simulation Module are Point Queue (PQ) and Spatial Queue (SQ). A simplified kinematic wave (KW) model can be also used in an advanced mode.

f. Capacity Management aims to manage the static and time-dependent link capacity input for Space-Time Simulation, such as signal timing plans and multi-modal service plans.

g. Simulation Output Module covers the output file of Space-Time Simulation Module, including Dynamic Link Performance and Agent Trajectory in terms of link_performance.csv and agent.csv, which can be visualized in NeXTA.

Figure 8.1.1 DTALite Model Architecture



Model Package and Data Flow

Figure 8.1.2 shows a data flow for performing dynamic traffic analysis in the bundle of DTALite and NeXTA, including the system importing, analysis and exporting. In particular, the solution to import GIS data from multiple planning packages is to develop an open data format that allows DTALite users to feasibly convert their own data in proprietary format to a unifying data structure and widely used longitude and latitude coordinate system (WGS 84 used by Google Maps, Google Earth and on-line Google Fusion Tables), akin to the Open Document format which creates a standardized file format that allows users to open, edit, and save Microsoft Word documents from other applications such as Google Docs.

The latest software release can be downloaded at the DTALite Github website. The basic control and interfaces of NeXTA are described in this user's guide document. A working sample project folder can be found here (Chicago_Sketch).

The software package includes two software applications: NEXTA as GUI and data hub; DTALite as DTA simulation engine. For a general use of NeXTA and DTALite, the user can use the following specific instructions:

Step 1: make sure that you have installed the Microsoft Visual C++ 2015 Redistributable Package (x86) for parallel computing in DTALite.

Step 2: make sure that you have installed Gnuplot Software for some visualization functions in NeXTA (http://www.gnuplot.info/)

Step 3: Download and unzip the NeXTA/DTALite software package.

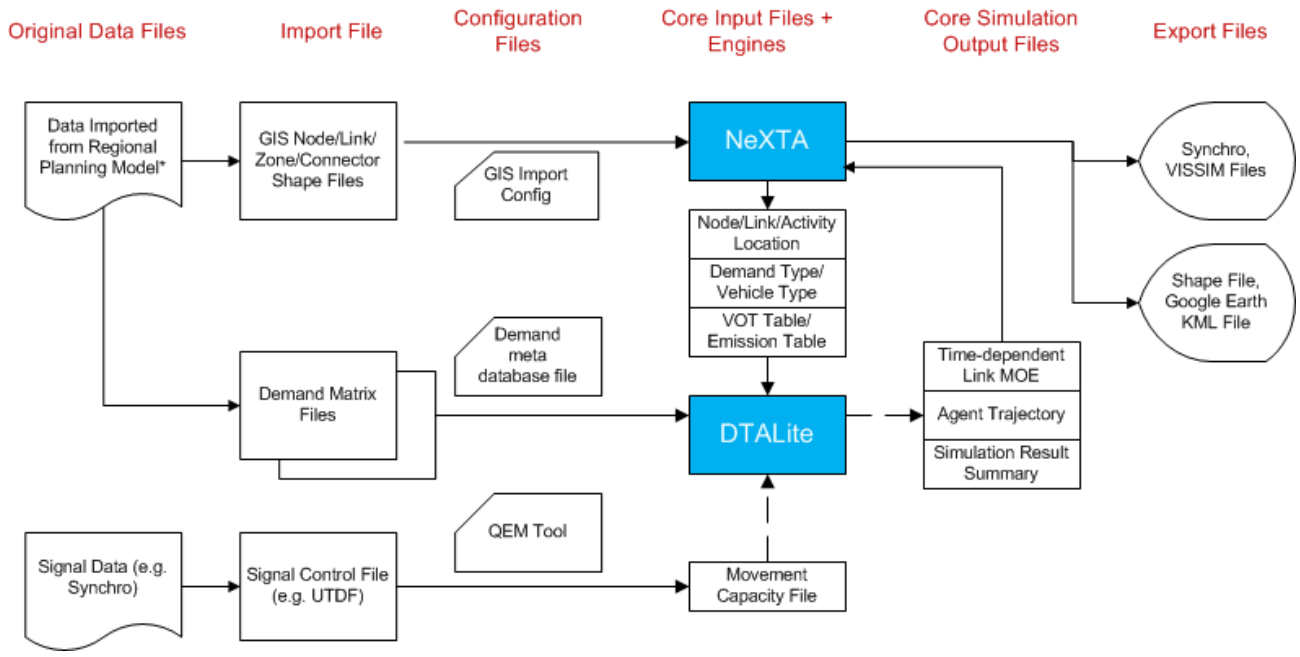
Step 4: Click "NeXTA"--"File"--"Open Traffic Network Project" to choose the tnp file in your network data set.

Step 5: Click "Project"--"Perform Traffic Assignment" to input your settings and run "DTALite" for dynamic traffic assignment and simulation.

For importing the GIS information of traffic networks from external files, such as GIS shape files, please download the NeXTA_GIS version at the DTALite Github website.

To run the DTALite for the TransAction Model, see Section 8.2 below for details.

Figure 8.1.2 Data Flow for Performing Dynamic Traffic Analysis



8.2 MODEL SCENARIO RUNNING

Running a scenario in DTALite involves several steps, including network and demand conversion, setting up the package settings, and post-processing the results. In this section, we will walk you through the steps required to run a scenario in DTALite.

Step 1: Network and Demand Conversion

Before running a scenario in DTALite, the first step is to convert the network and demand data into the required format. To convert the network data, run **cube2gmns.py**, which takes a Cube network shapefile as input and converts it into a GMNS standard network format. You will need to specify the input folder directory that contain the shapefile. This code will generate the required network files in the GMNS format.

In order to properly load a geographical dataset, it is imperative to ensure that the folder containing the shapefile also includes the accompanying **.prj**, **.dbf**, and **.shx** files. Neglecting to include any of these files may result in erroneous representations of the geographic data. Thus, it is essential to verify that all necessary files are present and accounted for prior to loading the dataset.

In order to utilize the Python code for network conversion, it is necessary to provide the shape file directory in the format depicted in Figure 8.2.1.

Figure 8.2.1 Network Conversion Python Code User Input

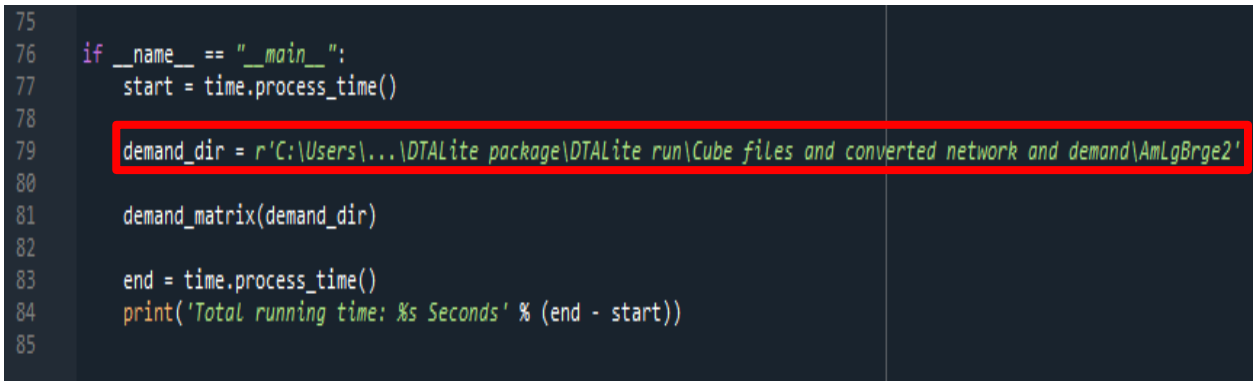
```
335
336 if __name__ == '__main__':
337     start_time = time.process_time()
338
339     shapfile_path = r'C:\Users\...\DTALite package\DTALite run\Cube files and converted network and demand\Network_AmLgBrge2'
340     output_folder = shapfile_path
341
342     network = _buildnet(shapfile_path)
343     _outputNode(network, output_folder)
344     _outputLink(network, output_folder)
345
346
347     end_time = time.process_time()
348     print('Total Running time: %s Seconds' % (end_time - start_time))
349
```

Next, you will need to convert the demand data from OMX format to O-D trip tables in CSV format using another Python package, **omx2csv.py**. This will generate the O-D demand tables needed for the DTALite simulation.

To make use of the Python code for demand conversion, it is imperative to furnish the demand file directory in the format illustrated in Figure 8.2.2.

Please ensure that the names of the demand files are consistent with their corresponding analysis period. They should be named in the format of "...AM...omx" (or MD, PM, NT), for example, "I4_AM_2045BD2.omx" for the AM period.

Figure 8.2.2 Demand Conversion Python Code User Input



```
75
76  if __name__ == "__main__":
77      start = time.process_time()
78
79      demand_dir = r'C:\Users\...\DTALite package\DTALite run\Cube files and converted network and demand\AmLgBrge2'
80
81      demand_matrix(demand_dir)
82
83      end = time.process_time()
84      print('Total running time: %s Seconds' % (end - start))
85
```

Step 2: Running DTALite Package

Once the network and demand data have been converted, you can set up the DTALite settings by creating a settings file. The settings file specifies various features such as agent type, facility type, periods start time and end time, and the type of dynamic traffic assignment (simulation or DTA).

To create the settings file, an existing setting.csv (a template file provided) can be modified. The template file contains default values for the dynamic traffic assignment settings, and they can be modified based on specific requirements. The important sections of the setting file can be modified as described in the following. The remaining parts are recommended to be kept as default values and need not be modified.

To run DTALite, the network and demand file, along with the DTALite.exe, need to be placed in a single folder. DTALite can be executed either by running the exe file directly or by using the assignment.py, which simplifies the creation of necessary files and the execution of multiple time period analysis. When the Python code is executed, it is important to ensure that separate setting files are created that are consistent with the analysis time periods, following the format "setting_am.csv". For instance, if the code is being run for the entire day, the following files should be created: "setting_am.csv", "setting_md.csv", "setting_pm.csv", and "setting_nt.csv."

A csv file called "whole_net_link_qvdf.csv" contains the calibrated parameter for the entire northern Virginia region is provided. To conduct an analysis in a specific region, the user must extract the values from "whole_net_link_qvdf.csv" and create a file named "link_qvdf.csv". This file must be placed in the same folder as DTALite.exe. However, by using "**assignment.py**," this process can be automated, as a function is defined in the python code to extract the required values and create the file.

To use the assignment python code, it is necessary to specify the analysis time period, modify the parent folder directory, and the folder name in the format illustrated in Figure 8.2.3.

Figure 8.2.3 Assignment Python Code User Input

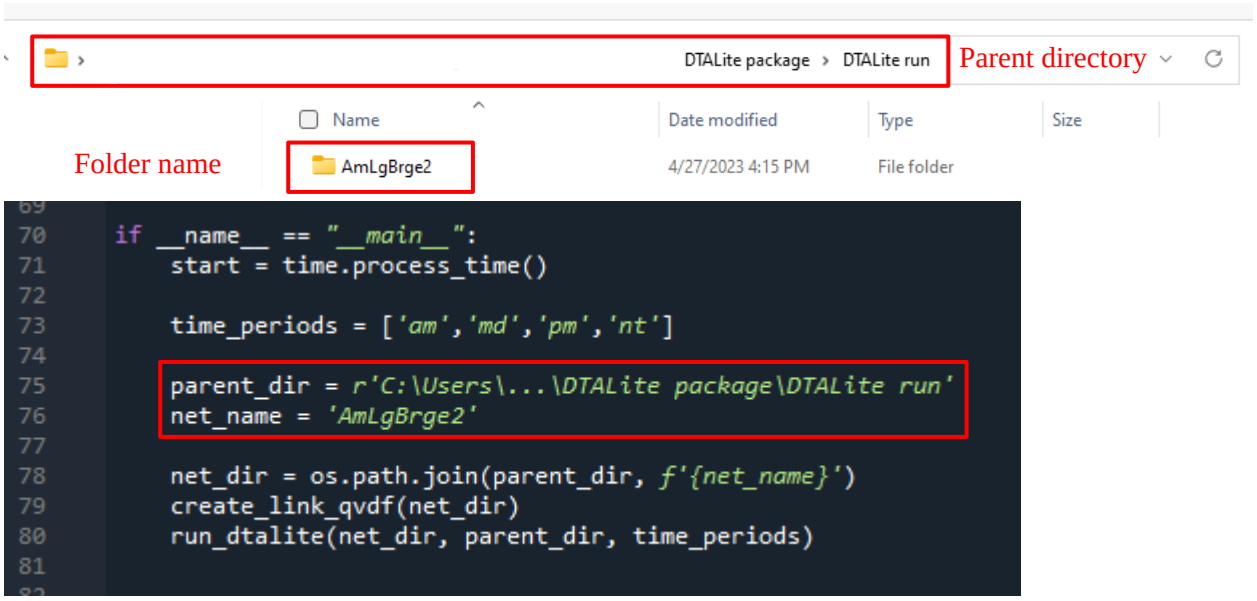


Figure 8.2.4 Setting File for DTALite

	A	B	C	D	E	F	G	H	I
1	[assignment]		assignment_mode	column_ge	column_updating_iterations	odme_iterations	simulation_iterations	sensitivity_analysis	number_
2	1		lue	50	0	-1	1	1	35
3									
4	[agent_type]	agent_type	name		vot	flow_type	pce	person_occupancy	
5		sov	DRIVE		24	0	1	1	
6		hov2	HOV2		40	0	1	2	
7	2	hov3	HOV3		60	0	1	3.5	
8		com	COMMERCIAL TRUCK		30	0	1	1	
9		trk	TRUCK		30	0	1	1	
10		apv	AIR		30	0	1	1.6	
11									
12	[link_type]	link_type	link_type_name		agent_type_blocklist	type_code	traffic_flow_code	vdf_type	
13		0	Centroids			c	0	qvdf	
14		100	Centroids			c	0	qvdf	
15		101	Freeways			f	0	qvdf	
16		102	Major Arterial			a	0	qvdf	
17	3	103	Minor Arterial			a	0	qvdf	
18		104	Collector			a	0	qvdf	
19		105	Expressway			f	0	qvdf	
20		106	Ramp			r	0	qvdf	
56									
57									
58	[demand_period]	demand_period_id	demand_period		time_period				
59		1	AM		0600_0900	3			
60	4								
61									
62									
63	[demand_file_list]	file_sequence_no	file_name		format_type	demand_period	agent_type		
64		1	sov_am.csv		column	AM	sov		
65		2	trk_am.csv		column	AM	trk		
66	5	3	hv2_am.csv		column	AM	hov2		
67		4	hv3_am.csv		column	AM	hov3		
68		5	apv_am.csv		column	AM	apv		
69		6	com_am.csv		column	AM	com		
70									
88									
89	[capacity_scenario]		from_node_id	to_node_id	time_window	time_interval	travel_time_delta	capacity	

Figure 8.2.4 depicts the sections required of setting file to modify in order to utilize the NVTA DTALite model. The following descriptions outline the required modifications for each section:

Section 1 (assignment setting): The column C, labeled as "assignment_mode," includes two modes: "lue" and "dta." The "lue" mode corresponds to the DTA simulation and analytical integrated engine, which is fast but does not generate vehicle trajectory. On the other hand, the "dta" mode is the simulation engine that generates vehicle trajectory. Other settings in this section can be left as default values, as shown in Figure 8.2.4.

Section 2 (agent type): The column B, labeled as "agent_type," specifies various agent types in the NVTA DTALite model, which can be set according to the values shown in Figure 8.2.4. The "vot" column (E) specifies the value of "vot" for each agent for different analysis periods (AM, MD, PM, and NT), which can be set based on the values in Figure 8.2.5. The rest of the settings in this section can be set as default, as shown in Figure 8.2.4.

Figure 8.2.5 Values of "vot" for Each Agent in Different Analysis Periods

	AM	Midday	PM	Night
SOV	24	20	20	20
HOV 2	40	15	30	15
HOV 3+	60	15	60	15
commerci	30	30	30	30
Truck	30	30	30	30
Airport pa	30	30	30	30

Section 3 (link type): This section includes link types as specified according to the various types in the NVTA DTALite model. The settings in this section can be left as default, as shown in Figure 8.2.4

Section 4 (demand period): In this section, the demand period (column C) and its corresponding time period (column E) must be specified based on the current analysis period.

Section 5 (demand file list): In this section, all the required demand files for the specified analysis period in Section 4 must be listed under "file_name" (column C), and the demand_period (column F) must be consistent with the analysis period. The rest of the settings can be set as the values in Figure 8.2.4.

Step 3: Post-Processing and Traffic State Statistics

After the simulation has completed, traffic state statistics can be generated using the provided Python codes. The first Python code will generate traffic delay, person delay, person mile, and average speed for jurisdiction level.

In order to utilize the Python code, **perf_based_stats.py**, for jurisdiction level performance statistics, the directory of the link_performance.csv files (DTALite model results) and the analysis periods, as well as the network files, including link.csv and node.csv, must be provided. All of these files should be placed in the same folder, and the directory of the folder containing all of these required files should be specified in the Python code, as illustrated in Figure 8.2.6. It is

important to note that the link performance files must be renamed in the following format: 'link_performance_{periods}.csv'. For instance, the link performance file for the AM analysis period must be renamed as link_performance_am.csv, which is also automated in the assignment python code.

Figure 8.2.6 Required Input for Jurisdiction-based Traffic Performance Statistics

```
147
148     if __name__ == "__main__":
149         start = time.process_time()
150
151
152         time_periods = ['am','md','pm','nt']
153
154
155         parent_dir = r'C:\Users\...\DTALite package\DTALite run'
156         sub_net = 'AmLgBrge2'
157
158         perf_based_stat(time_periods, parent_dir, sub_net)
159
160         end = time.process_time()
161         print('Total Running time: %s Seconds' % (end - start))
162
```

Figure 8.2.7 illustrates an example of traffic performance at the jurisdiction level.

Figure 8.2.7 Jurisdiction-based Traffic State Statistical Analysis

Jur_name	delay	person_delay	person_hour	person_mile	mile/hour	delay/hour	max_severe_congestion_duration	mean_severe_congestion_duration(normal avg)	mean_severe_congestion_duration(length avg)
District of Columbia	8.8	42892.5	93889.4	2189455.7	23.3	0.5	3	0.17	0.15
Montgomery	7.2	39442.2	137149.6	5191190.4	37.9	0.3	2.25	0.05	0.04
Prince George's	5.9	36453.9	146485.0	6119992.9	41.8	0.2	2.583	0.03	0.03
Arlington	2.4	13139.9	34926.0	1113646.4	31.9	0.4	3	0.12	0.11
Alexandria	1.6	9925.2	24940.3	736914.8	29.5	0.4	2.917	0.15	0.13
Fairfax	13.9	81224.5	226147.2	7641139.0	33.8	0.4	3	0.12	0.09
Fairfax City	0.5	1878.0	4313.9	101813.3	23.6	0.4	2.083	0.23	0.20
Falls Church	0.2	565.8	1614.6	41728.7	25.8	0.4	1.417	0.10	0.10
Loudoun	5.0	21613.9	67120.7	2250290.5	33.5	0.3	3	0.06	0.06
Prince William	5.9	39095.5	97003.6	3074653.5	31.7	0.4	3	0.09	0.07
Manassas	0.2	483.9	2609.2	92405.8	35.4	0.2	0.75	0.02	0.02
Manassas Park	0.0	122.8	216.6	3432.9	15.9	0.6	1.583	0.47	0.35
Frederick	2.6	14546.6	59462.1	2705346.1	45.5	0.2	2.25	0.04	0.02
Howard	3.7	25329.8	76919.6	3144167.6	40.9	0.3	3	0.12	0.08
Anne Arundel	4.7	32350.6	100304.1	4152038.7	41.4	0.3	2.25	0.09	0.07
Charles	1.9	9570.0	29870.7	1078799.7	36.1	0.3	2.583	0.06	0.04
Carroll	2.2	7541.1	28440.1	1146342.3	40.3	0.3	2.417	0.05	0.03
Calvert	0.5	2477.6	11218.4	476614.8	42.5	0.2	0.917	0.01	0.00
St. Mary's	1.0	4420.9	15763.9	600742.5	38.1	0.3	1.583	0.04	0.03
King George	0.3	1010.6	6979.6	314482.7	45.1	0.1	0	0.00	0.00
Fredericksburg	0.3	1505.5	6718.4	295417.3	44.0	0.2	1.083	0.08	0.04
Stafford	1.5	13266.6	39691.5	1569402.9	39.5	0.3	1.917	0.05	0.03
Spotsylvania	0.8	4460.8	19210.4	860696.9	44.8	0.2	2.417	0.02	0.02
Fauquier	1.4	5899.2	27615.2	1275264.7	46.2	0.2	1.417	0.04	0.03
Clarke	1.3	7537.7	14921.0	402181.8	27.0	0.5	2.083	0.23	0.24
Jefferson	0.6	3310.2	11925.3	492613.0	41.3	0.3	1.417	0.06	0.03
TOTAL_AM	74.4	420065.3	1285456.2	47070774.8	36.6	0.3	3	0.06	0.04

The second Python code, "**link_performance_comparison.py**," will generate link performance comparison, which involves comparing the performance statistics for built and non-built case scenarios. This code will generate comparison statistics for various performance measures, including delay, travel time, and vehicle miles traveled.

The python code requires the user to specify the necessary input parameters as illustrated in Figure 8.2.8. Firstly, the analysis time periods and their corresponding duration should be specified in the first box. Subsequently, the directory containing the built and non-built networks (or the network selected for comparison) must be specified in the second box.

Figure 8.2.8 Link Performance Comparison Python Code User Input

```
145
146  if __name__ == "__main__":
147      start = time.process_time()
148
149      period_length_dict = {'am':3,'md':6,'pm':4,'nt':11,'pm_r':4}
150
151      time_periods = ['am','md','pm','nt']
152
153      parent_dir = r'C:\Users\...\DTALite package\DTALite run'
154
155      bd_net = 'AmLgBrge2'
156      nb_net = 'TA2045NB'
157
158      getdiff(time_periods, period_length_dict, parent_dir, nb_net, bd_net)
159
```

Finally, the last Python code will be used to generate Speed Class Statistics, which will calculate traffic state and performance statistics for different speed categories (speed ranges) for built and non-built case scenarios.

The Python code for generating Speed Class Statistics, "**spd_class_statistics.py**," can be utilized by first specifying the analysis time periods and their corresponding durations in the first box, as shown in Figure 8.2.9. Next, the directory that contains the built and non-built networks (or the network selected for comparison) must be indicated in the second box as illustrated in Figure 8.2.9.

Figure 8.2.9 Speed Class Statistics Python Code User Input

```
213  if __name__ == "__main__":
214
215      period_length_dict = {'am':3,'md':6,'pm':4,'nt':11,'pm_r':4}
216
217      time_periods = ['am','md','pm','nt']
218
219      parent_dir = r'C:\Users\...\DTALite run'
220
221      bd_net = 'AmLgBrge2'
222      nb_net = 'TA2045NB'
223
224      getSpdstat(parent_dir, nb_net, bd_net, time_periods, period_length_dict)
```