FY 2014 Final Report

COG Contract 12-006:

Assistance with Development and Application of the National Capital Region Transportation Planning Board Travel Demand Model

Submitted to:

National Capital Region Transportation Planning Board
Metropolitan Washington Council of Governments

Submitted by:

AECOM

August 18, 2014

This page is intentionally blank

Table of Contents

List of Figures

List of Tables

1 Introduction

The National Capital Region Transportation Planning Board (NCRTPB or simply TPB) is the federally designated Metropolitan Planning Organization (MPO) for the Washington, D.C. metropolitan area and is also one of several policy boards that operate at the Metropolitan Washington Council of Governments (MWCOG or simply COG). The TPB is staffed by COG's Department of Transportation Planning (DTP). Since 2005 (FY 2006), COG/TPB staff has maintained a consultant-assisted project to assist with the development and application of the COG/TPB travel demand model. This project has frequently included a review and scan of modeling used by other MPOs across the U.S. Two of the main objectives of the project are to ensure that the TPB's modeling methods are in line with those of other MPOs and to provide guidance and advice in the area of travel demand modeling. The project contract was designed to operate on a fiscal-year basis and to be renewable for up to two additional fiscal years. This arrangement would allow the selected consultant to hold the contract for up to three years in total, at which time re-bidding of the contract would occur. The contract was re-bid at the end of FY 2011 and AECOM was chosen by a selection committee to be the consultant for FY 2012 (July 1, 2011 to June 30, 2012). About a year later, the contract was renewed for a second year (FY 2013). Finally, at the end of FY 2013, the contract was renewed for its third and final year (FY 2014). In FY 2014, four task orders were developed collaboratively between COG/TPB staff and AECOM:

- Task Order 10: Attend relevant meetings, such as the Travel Forecasting Subcommittee, and respond to ad hoc requests (this is a continuation of Task Order #7 from FY 2013 and Task Order #1 from FY 2012);
- Task Order 11: Cube-based Transit Walk Sheds: Integrate a GIS-based method of calculating transit walk sheds into the standard modeling process that does not require the user to own and operate ArcGIS software.
- Task Order 12: Traffic Assignment Improvements: Develop software tools and procedures that improve the current process used to model both high-occupancy vehicle (HOV) lanes and high-occupancy/toll (HOT) lanes ("express toll" lanes).
- Task Order 13: Mode Choice and Transit Modeling: Complete the conversion of MWCOG TRNBUILD networks to Public Transport (PT) format and migrate the MWCOG mode choice models from AEMS to the ModeChoice software.

This report documents the work done by AECOM to fulfill these task orders. Chapters 2 through 7 address Task Orders 10 through 13. Chapter 8 presents a summary of consultant recommendations. Many of the longer modeling scripts are included in the Appendix. **The procedures developed in Task Orders 11-13 are currently being tested by MWCOG staff**. After this review is complete, one or more of these procedures may be incorporated into MWCOG's production-use travel demand model.

This page is intentionally blank.

2 Meetings and Technical Assistance (Task Order 10)

AECOM participated in each of the six Travel Forecasting Subcommittee meetings of FY 2014 and prepared presentations for the July 19, 2013, January 24, 2014, March 21, 2014, and May 23, 2014 committee meetings. AECOM also met several times with the MWCOG staff to discuss work tasks, findings and recommendations.

This page is intentionally blank.

3 Cube-Based Transit Walk Sheds (Task Order 11)

The objective of Task Order 11 was to replace the ArcGIS-based procedures used by MWCOG to calculate transit walk sheds and zonal percent-walk-to-transit files with a Cube-based procedure that can be integrated into the standard modeling process as described in the Task Order 11 Memorandum from COG [1] and a subsequent proposed scope by AECOM[2]. This chapter documents this newly developed percent walk to transit ("PWT") process.

3.1 Background

The COG/TPB PWT process is currently ArcGIS-based and has suffered from stability problems in addition to software maintenance issues especially due to staff turnover. The VBA source in the COG/TPB PWT process is also currently inaccessible due to a lost password. In addition to improving reliability and overcoming reliance on ArcGIS software, the need to integrate the PWT process in the mainstream TPB modeling process was greatly felt.

The ideal PWT process would rely entirely on Cube Voyager scripts. While Cube Base (especially version 6) has made strides in extending GIS functionalities to users through its integration with ArcGIS, the GIS routines are not documented and are not available for use in Cube Voyager scripts. For example, Figure 3-1 shows the Cube's GIS Geoprocessing tool interface, which is accessed via a graphical user interface (GUI). Nonetheless, we wish to acknowledge the help provided by Citilabs' user support group for the work related to preparation of inputs as documented in this memo.

A self-contained custom program using square grids to approximate circular buffers was also considered as an alternative approach to PWT process. However COG/TPB staff did not prefer this approach to the one based on ArcPy [3] – a software package that provides access to ArcGIS's functions in a Python programming environment and which is available with Cube and ArcGIS installations at no additional cost.

Cube installation packages come in two versions – one, 'with ArcGIS' and two, 'without ArcGIS'. The 'with ArcGIS' version comes with an ArcGIS Engine Runtime ("ArcGIS runtime") and is expected to be used by users without an installation of ArcGIS Desktop ("full ArcGIS"). Note that each ArcGIS runtime has a version number that corresponds to its counterpart in the full ArcGIS package and both the ArcGIS runtime and full ArcGIS install Python as part of their installation. The 'without ArcGIS' version does not include the ArcGIS runtime and therefore Cube would rely on the user's full ArcGIS installation for its GIS functions.

---

[1] Memorandum dated July 22nd, 2013 from Mark S. Moran (COG/TPB) to David Roden (AECOM) titled "COG Contract 12-006, Year 3 (FY 2014): Proposal for a task order to develop a new way to calculate zonal percent-walk-to-transit values"

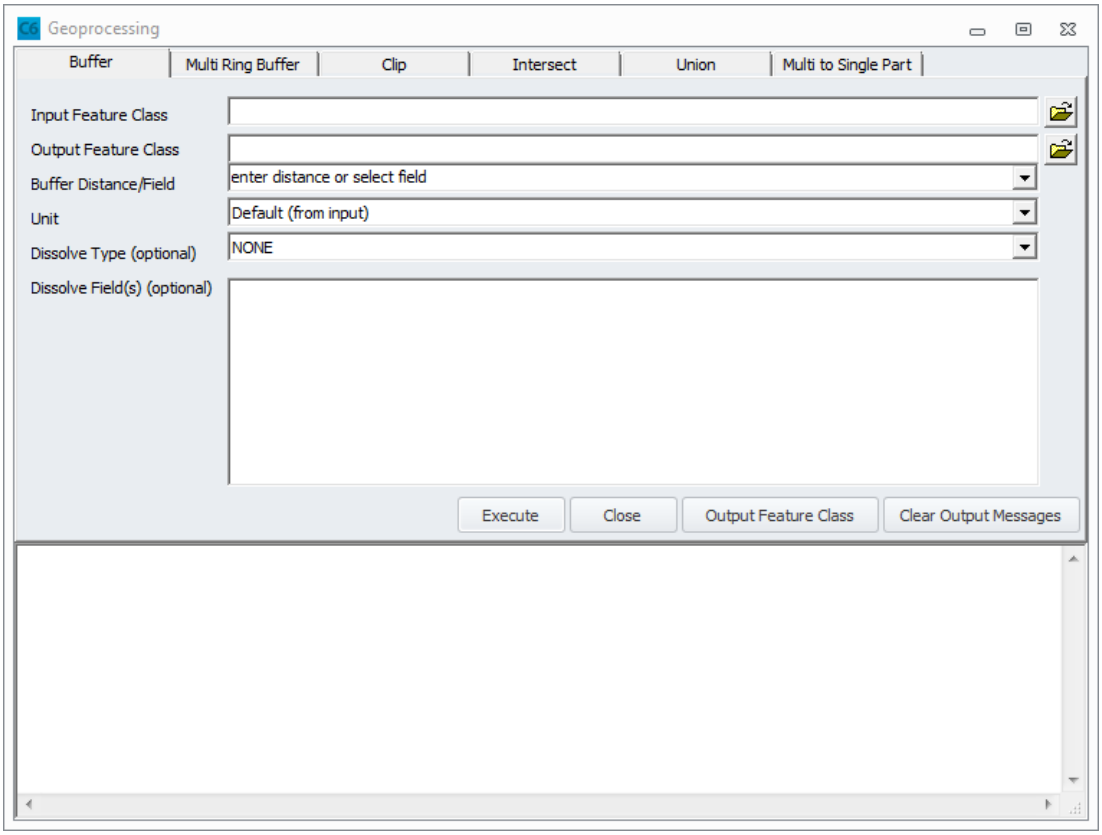
[2] Memorandum dated August 2nd, 2013 from David Roden (AECOM) to Mark S. Moran (COG/TPB) titled "Task Order 11 of COG Contract 12-006 – Cube-based Transit Walk Sheds"

[3] A term coined by ESRI by combining 'ArcGIS' and 'Python'. More information about ArcPy is available on ArcGIS online Help: http://resources.arcgis.com/en/help/main/10.1/index.html#//000v000000v7000000

It may be noted that ArcGIS has recently made major enhancements to its software from Version 9.x to Version 10.x. Additionally, several bugs in ArcGIS 10.0 were addressed in ArcGIS 10.1, so the later version is more reliable and stable.

The current TPB model version 2.3.52 has been designed with Cube 6.0.2, which comes with an ArcGIS runtime Version 10.0. This ArcGIS runtime corresponds to the ArcGIS Desktop 10.0. However, the latest Cube release – Cube 6.1.0 SP1 comes with an ArcGIS runtime Version 10.1, which corresponds to ArcGIS Desktop 10.1.

Figure 3-1: GIS Geoprocessing Tool Interface in Cube 6



3.2 Proposed ArcPy-Cube Based PWT Process

ArcPy was chosen as the basis for development because it provides convenient and powerful access to the GIS functionalities in a (Python) programming environment that is transparent and relatively easy to modify. However, this approach has its own challenges in maintaining backwards compatibility.

In order to simplify the implementation and boost its reliability, it was decided to choose the latest release of Cube – Cube 6.1.0 SP1 (with ArcGIS runtime 10.1), as the basis for the proposed PWT process using ArcPy and Cube. Nonetheless, users with Cube 6.0.2 and full ArcGIS 10.1 can also seamlessly use the new PWT process. However, users with Cube 6.0.2 and ArcGIS runtime 10.0 will not be able to use the new PWT process due to an issue in ArcPy within ArcGIS runtime 10.0. Refer to the conclusion section for more information.

The proposed PWT process utilizes both an ArcPy-based Python script and a Cube/Voyager script. The Voyager script prepares the inputs to the PWT GIS process directly from the TPB TRNBUILD/PT line files and the ArcPy Python script processes them to create intermediate walkshed buffers and the final output file (AreaWalk.txt). Alternatively, one may edit the ArcPy script to output PercentWalk.txt. The Version 2.3.52 Travel Model currently uses AreaWalk.txt as the input file.

Figure 3-2 presents an overview of the proposed PWT process and how it integrates (as an optional step) to the mainstream TPB model execution processes. Figure 3-3 provides an overview of the functions including in geo-processing steps performed within the Python script ('MWCOG_ArcPy_Walkshed_Process.py') which is at the heart of this proposed PWT process. The associated setup file, a modified complete TPB model system called 'Ver2.3.52_ModwithWalkshedProcess' has the following primary changes:

In the root folder:

- ArcPy_Walkshed_Process.bat - New Win. Batch File
- run_ModelSteps_Ver2.3.52_2010_Final.bat - Calls ArcPy batch file
  … (13 model steps files in total)
- run_ModelSteps_Ver2.3.52_2040_Final.bat - Calls ArcPy batch file

In the 'Scripts' sub-folder:

- MWCOG_ArcPy_Walkshed_Process.py - New Python Script
- MWCOG_Prepare_Inputs_to_Walkshed_Process_TRNBUILD.s - New Cube Script
- MWCOG_Prepare_Inputs_to_Walkshed_Process_PT.s - New Cube Script
- MWCOG_ArcPy_Walkshed_Process_TEMPLATE.mxd - New ArcGIS/Cube MXD
- Maryland1900Ft_ShapefileProjection_TEMPLATE.prj - New Text File

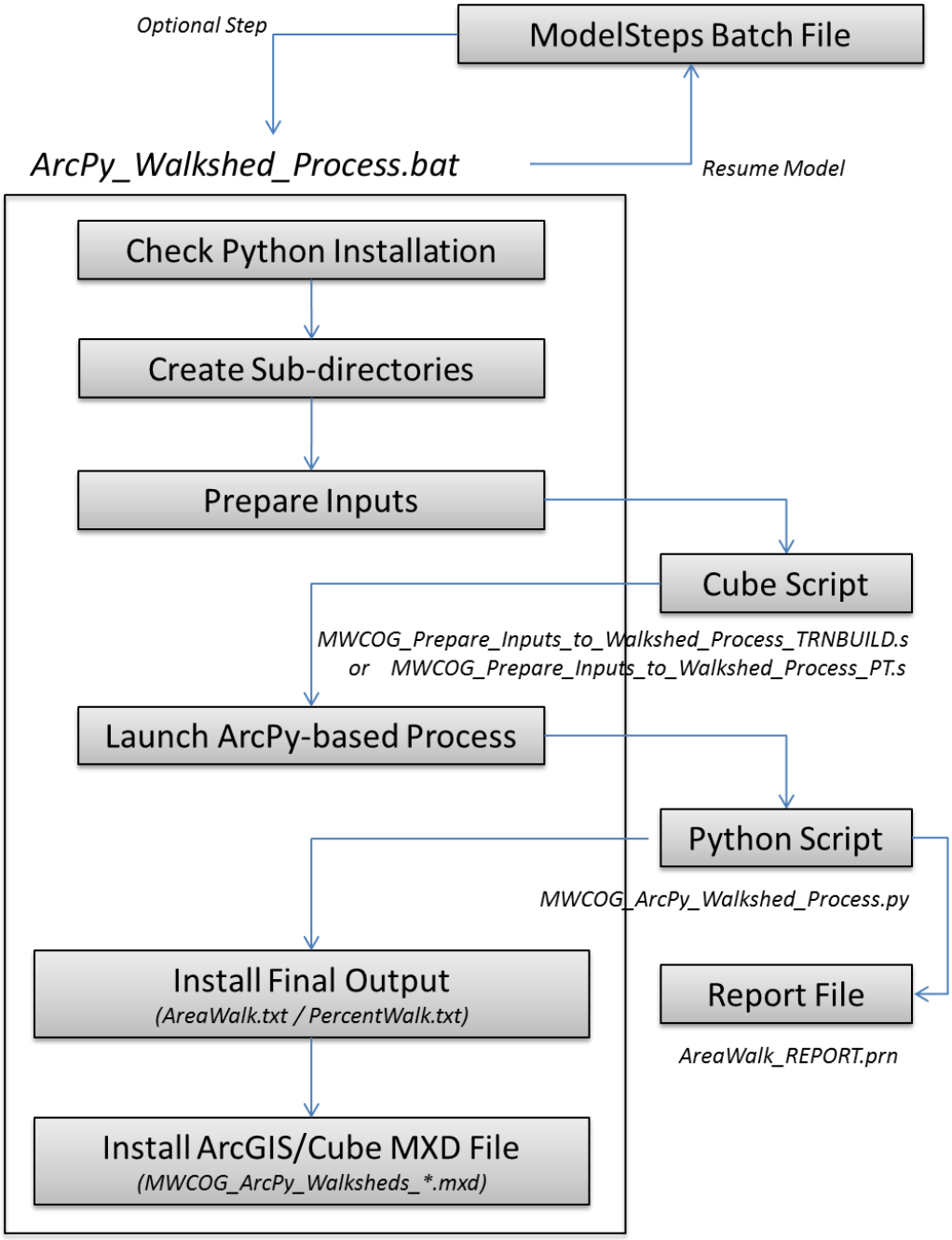
In the 'TPBTAZ3722_TPBMod' sub-folder:

- TAZLand3722.shp/.dbf/.shx/.prj - TAZ sorted shapefile
- TPBTAZ3722_TPBMod.shp/.dbf/.shx/.prj - TAZ sorted shapefile

The input TAZ polygons were manually sorted by TAZ in order to have a set of output files that were sorted by TAZ. However, during execution of the geo-processing steps, the input order of records is modified such that the input sort order is not maintained. Consequently, the final output text files (such as AreaWalk.txt and AreaWalk.csv) were not sorted by TAZ. This was also the case with the previous process. In order to make it easier to compare model input files between two modeling scenarios, based on discussions with COG/TPB staff, an additional step was added to physically sort the final shapefile (shapefile referred by 'temp_TAZ' was sorted by TAZ into a shapefile referred by 'sort_TAZ')

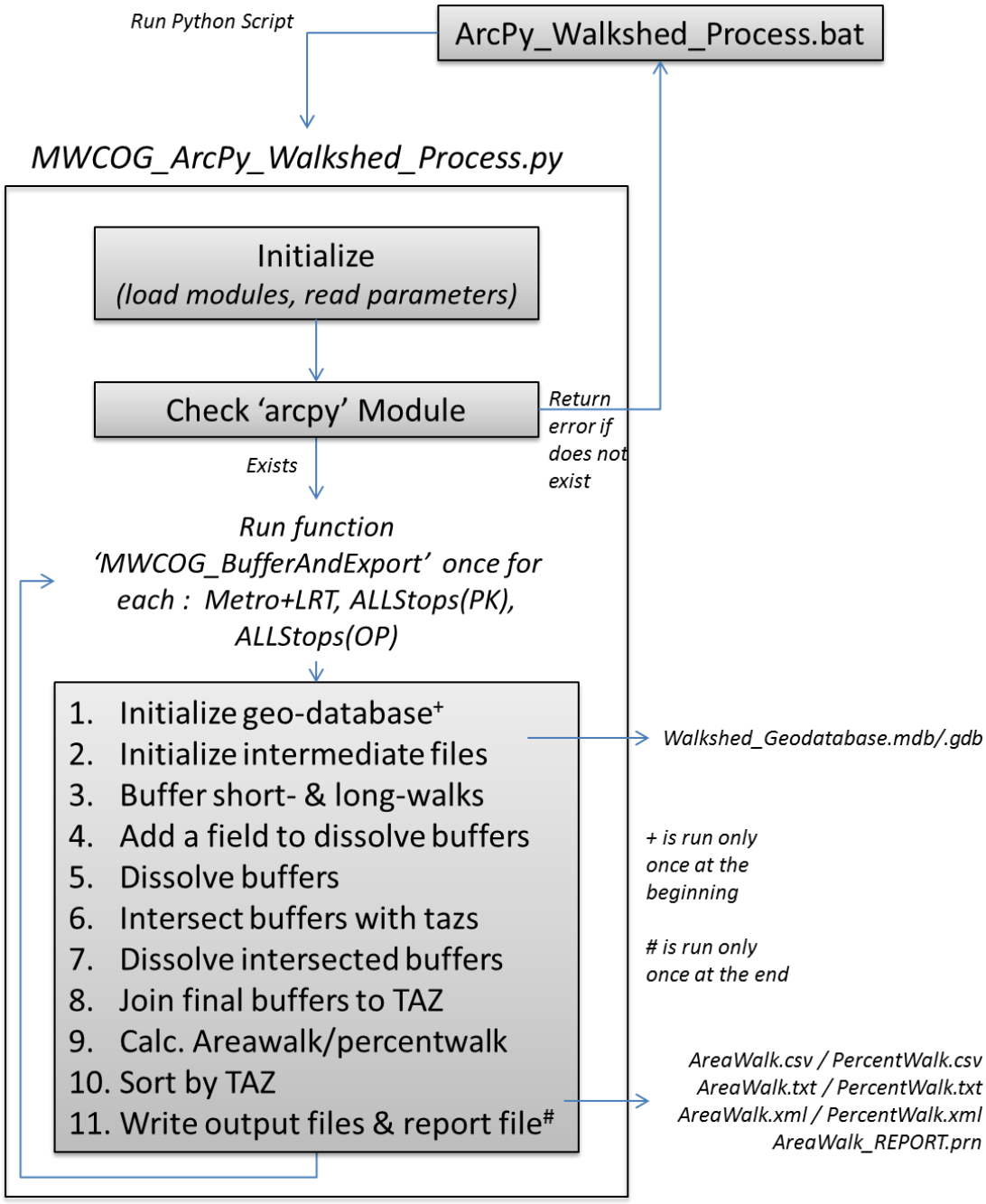
using the ArcPy's 'Sort_Management' command. This change makes the output files to be TAZ sorted and has negligible impact on the processing time.[4]

Figure 3-2: Overview of the proposed ArcPy-Cube based PWT Process



[4] Upon further inspection, COG/TPB staff found that output file was not completely sorted by TAZ. Staff added an additional "Sort_management" step in the ArcPy script, which resulted in the entire output file being sorted correctly.

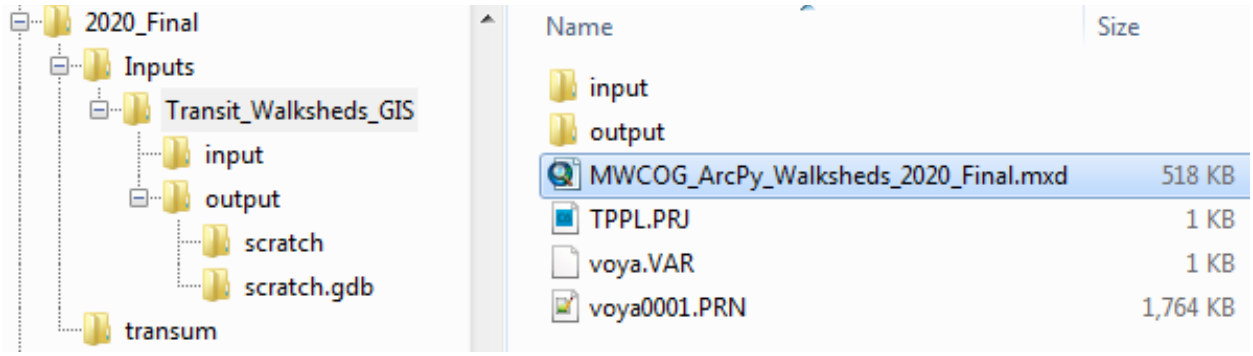
Figure 3-3: Flowchart of the Python script showing geo-processing steps



The proposed PWT process execution is optional and when executed creates intermediate folders within the scenario 'inputs' folder as shown in Figure 3-4 below. The top-level intermediate folder is called 'Transit_Walksheds_GIS' wherein the ArcGIS/Cube MXD file is placed for visualizing the walksheds buffers. Note that further subfolders 'input' and 'output' are also created, which are not to be confused with the scenario 'Inputs' folder.

After execution, the final output file ('AreaWalk.txt' or 'PercentWalk.txt') is copied to the main scenario 'Inputs' folder to replace any existing copies.

Figure 3-4: ArcPy-Cube based PWT Folder Structure



3.3 Methodology

This section presents the contents of the batch files and scripts with an explanation of key aspects. The ArcPy batch file ("ArcPy_Walkshed_Process.bat) is the system manager which first checks for pre-requisites (Python/ArcPy), prepares inputs and calls the ArcPy processes and finally copies the output and MXD. This batch file has several execution checks included, so if any of the steps fail to execute successfully, the process stops and issues a message.

3.3.1 Batch file

The proposed PWT process is an optional step in the TPB model execution. Since its inputs do not change during speed-feedback iterations, it only needs to be run/called once during the model execution as part of the initial steps. To disable this proposed PWT process, the users can simply comment out the call to this process as shown in Figure 3-5.

Figure 3-5: Integration into Model Steps Batch File

```
< Example from run_ModelSteps_Ver2.3.52_2010_Final.bat >

rem ====== Pump Prime Iteration ========================================
set  iter =pp
set _prev_=pp
set  relGap =0.01

call ArcPy_Walkshed_Process.bat %1
call Set_CPI.bat                %1
```

Figure 9-1 in the Appendix presents the contents of the 'ArcPy_Walkshed_Process.bat' batch file, which searches for Python in the most common locations, i.e., the C, D, and E drives. Note in the unusual and deprecated circumstance where a user has both a full ArcGIS and an ArcGIS runtime installation, the full ArcGIS is preferred. If the user's installation of Python (which is installed as part of ArcGIS runtime or full ArcGIS) is not part of C, D, or E, then that path can be added to this batch file.

Note that this batch file creates intermediate directories and deletes any existing intermediate files in them (this could happen when the process is re-run).

3.3.2 Inputs

The three input point-Shapefiles (All Stops for Peak Period, All Stops for Off-Peak Period and Metro+LRT Stops for All-Day) are prepared using the MODE[1-10]AM.TB and MODE[1-10]OP.TB files in the scenario 'Inputs' folder. The All-Day Metro and LRT stops are prepared using only the Mode 3 and Mode 5, AM and OP line files. In the batch file, the presence of PT line files (assuming a naming convention of MODE[1-10]AM.Lin and MODE[1-10]OP.Lin) is detected and given preference, if both TRNBUILD and PT line files exist. The batch file then chooses the appropriate Cube script to process these line files. Figure 9-2 in the Appendix presents the input preparation process for TRNBUILD line files only. The Cube script to process the PT line files is included in the setup.

Figure 9-2 makes use of the integration of Cube and ArcGIS to export point-shapefiles using the 'FORMAT=SHP' feature in Cube which unfortunately does not write out the projection information of the shapefile which is required for reliable GIS operations. The projection used for TPB model inputs is NAD 1983, Maryland State Plane, National Geodetic Survey code 1900, in units of feet.[5] A template projection file is used to include projection information for each of the input shapefiles as shown in Figure 3-6.

Figure 3-6: Maryland 1900 Feet StatePlane/ Shapefile Projection Information

```
< Contents of Maryland1900Ft ShapefileProjection TEMPLATE.prj >

PROJCS["NAD_1983_StatePlane_Maryland_FIPS_1900_Feet",GEOGCS["GCS_North_American_1983",DATUM["D_No
rth_American_1983",SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["De
gree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],PARAMETER["False_Easting",131233
3.333333333],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-
77.0],PARAMETER["Standard_Parallel_1",38.3],PARAMETER["Standard_Parallel_2",39.45],PARAMETER["Lat
itude_Of_Origin",37.66666666666666],UNIT["Foot_US",0.3048006096012192]]
```

3.3.3 ArcPy, Python based Processing

After the input point-shapefiles are produced, short (1/2 mile) and long (one mile) buffers are created for each point and processed with the input TAZ boundary shapefile[6] to compute the PWT values. All these processes are carried out using ArcPy functions from a Python script as shown in Figure 9-3. The buffers and other intermediate files are stored in a geodatabase for encapsulation and expediting the processing speed.

This Python script is designed to provide flexibility to the user to:

- Specify the short and long walk distances (default to 2640 feet and 5280 feet respectively).
- Specify the type of output (defaults to 'AreaWalk.txt', the other option is 'PercentWalk.txt')
    - AreaWalk.txt

[5] When Esri implemented the NGS codes, they were part of a proposed Federal Information Processing Standard (FIPS). For that reason, Esri identifies the NGS zones as FIPS zones. Although that proposed standard was withdrawn, Esri decided to maintain the FIPS name in its projection files for continuity. Source: "ArcGIS Help 10.1 - State Plane Coordinate System," July 31, 2013.

[6] Currently, the input TAZ shapefile has had the water area for each TAZ subtracted out, before the processing begins.

This file includes the area (in squared miles) of each type of walk distance within a TAZ.

- PercentWalk.txt
  This file includes the percentage of walk-to-transit of each type of walk distance in a TAZ.

- Specify the type of geodatabase: personal- versus file- geodatabase (default is personal geodatabase)
- Specify the TAZ-area field optionally (defaults to 'TAZ_Area', used if provided, otherwise computed).

The Python script is primarily written around a custom function ('MWCOG_BufferAndExport') and is called three times, once with each input shapefile. This function performs all the buffering and processing operations for each of the two walk-distance types in sequence. Note that ArcPy is not a thread-safe library, i.e., multiple instances of ArcPy (especially w.r.t. buffering) can't be reliably executed in parallel using multiple cores, as simultaneous instances tend to freeze the process or make it longer than the normal amount of processing time to complete. Considerable testing was done in the initial stages to perform buffering operations in parallel – however only the sequential operations were found to be reliable.

The current complete execution times on a computer with 2.7 GHz CPU and more than 4 GB of physical RAM are approximately 30-35 minutes for 2010 and approximately 40-45 minutes for future years such as 2040. These runtimes can be considered reasonable given their integration with the TPB modeling processes. During execution, the peak RAM used by the process was found to be approximately up to 900 MB, hence users should ensure that a minimum of 1GB of RAM is available to avoid any memory related complications. Additionally, if the user intends to launch multiple runs, **it is recommended to start the runs approximately one hour apart to allow sufficient time for the corresponding PWT processes to finish** and thereby avoid any complications associated with running overlapping PWT processes.

Figure 9-3 in the Appendix presents the contents of the ArcPy-based Python script and Figure 9-4 shows a sample of the typical screen output generated as part of the proposed PWT process execution.

3.3.4 Outputs

After a successful execution of the PWT process, the 'ouput' subfolder contains the files 'AreaWalk.csv' and 'AreaWalk.txt' (or, optionally, 'PercentWalk.csv' and 'PercentWalk.txt'). The 'AreaWalk.csv' file is a comma-delimited (CSV) file as directly exported from the final intermediate shapefile stored in the geodatabase. The 'AreaWalk.txt' file is a fixed-column-format version of this file without the 'X_COORD' and 'Y_COORD' fields, similar in structure to the AreaWalk.txt file that is expected as an input to the Version 2.3.52 Travel Model. The structures of these files are shown Figure 3-7 in Figure 3-8 and for 'AreaWalk.csv' and 'AreaWalk.txt', respectively. In addition to these files, a report file 'AreaWalk_REPORT.prn' is also written out in the 'output' subfolder, which contains the ArcPy execution screen output and a summary report. The summary report provides statistics and other information such as:

- The total area, and percentage of land area, within the long-walk transit walkshed for the peak period
- The total area, and percentage of land area, within the long-walk transit walkshed for the off-peak period
- A list of TAZs with zero walk area
- A list of all cases where the peak period walk area is less than the off-peak walk area, which could indicate a network coding error, since one would generally expect more transit service in the peak period than the off-peak period.

Figure 3-7: Structure of Output File: Areawalk.txt

TAZID	TAZ_AREA	MTLRTSHR	MTLRTLNG	ALLPKSHR	ALLPKLNG	ALLOPSHR	ALLOPLNG
74	0.2004	0.1611	0.2004	0.2004	0.2004	0.2004	0.2004
75	0.2581	0.0000	0.2008	0.2581	0.2581	0.2581	0.2581
76	0.1900	0.0000	0.0000	0.1900	0.1900	0.1900	0.1900
77	0.0914	0.0000	0.0109	0.0914	0.0914	0.0914	0.0914
78	0.0862	0.0000	0.0000	0.0862	0.0862	0.0862	0.0862
1	0.0424	0.0228	0.0424	0.0424	0.0424	0.0424	0.0424
2	0.1576	0.0000	0.1576	0.1576	0.1576	0.1576	0.1576
3	0.2022	0.0393	0.2022	0.1173	0.2022	0.0866	0.2022
4	0.1596	0.1144	0.1596	0.1596	0.1596	0.1397	0.1596
5	0.0319	0.0044	0.0319	0.0312	0.0319	0.0312	0.0319
6	0.0412	0.0043	0.0412	0.0371	0.0412	0.0369	0.0412
7	0.0708	0.0000	0.0708	0.0708	0.0708	0.0708	0.0708
8	0.0708	0.0530	0.0708	0.0708	0.0708	0.0708	0.0708

Figure 3-8: Structure of Output File: AreaWalk.csv

XCoord	YCoord	TAZ	TAZ_AREA	MTLRTSHR	MTLRTLNG		ALLPKSHR		ALLPKLNG	
	ALLOPSHR		ALLOPLNG							
1340551.878	479699.1855	902	42407225.76	0	0.1176	1.2017	1.5211	1.2017	1.5211	
1317237.895	478948.0289	943	5058470.488	0	0	0.1814	0.1814	0.1814	0.1814	
1317199.459	476932.3885	942	4422449.642	0	0	0.1586	0.1586	0.1586	0.1586	
1319406.825	477215.1768	944	10555784.55	0	0.0107	0.3786	0.3786	0.3786	0.3786	
1321471.993	477058.5197	975	12258329.01	0	0.2012	0.4397	0.4397	0.4397	0.4397	
1323778.218	477110.5839	976	4919180.965	0	0.1765	0.1765	0.1765	0.1765	0.1765	
1323903.837	475414.4119	973	6664729.176	0.0699	0.2391	0.2391	0.2391	0.2391	0.2391	
1334080.647	475678.9387	995	7491842.816	0.1391	0.2687	0.2687	0.2687	0.2687	0.2687	
1332754.619	474818.5554	983	6011435.262	0.0639	0.2156	0.2156	0.2156	0.2156	0.2156	
1327738.632	468407.2309	961	6726101.977	0	0.0966	0.2413	0.2413	0.2413	0.2413	
1322599.252	467540.6576	967	3516414.603	0.0734	0.1261	0.1261	0.1261	0.1261	0.1261	
1320348.819	466029.2313	950	2901385.344	0	0.1041	0.1041	0.1041	0.1041	0.1041	
1322614.257	465824.8056	954	1452031.033	0	0.0521	0.0521	0.0521	0.0521	0.0521	
1327422.825	464257.0283	959	10618701.29	0	0	0.3809	0.3809	0.3809	0.3809	
1324560.126	461899.7234	957	5346147.916	0	0	0.1918	0.1918	0.1918	0.1918	
1330254.655	465673.1624	960	9232038.91	0	0	0.3312	0.3312	0.3312	0.3312	

Additionally, the template MXD file is copied to the 'Transit_Walksheds_GIS' folder with a suffix of the scenario name such as MWCOG_ArcPy_Walksheds_2010_Final.mxd for scenario "2010_Final". This MXD file can be opened in ArcGIS 10.1 or directly in Cube 6.1.0 SP1 as shown in Figure 3-9 and Figure 3-10, respectively.

Figure 3-9: Output 2010 Walkshed Buffers in ArcGIS

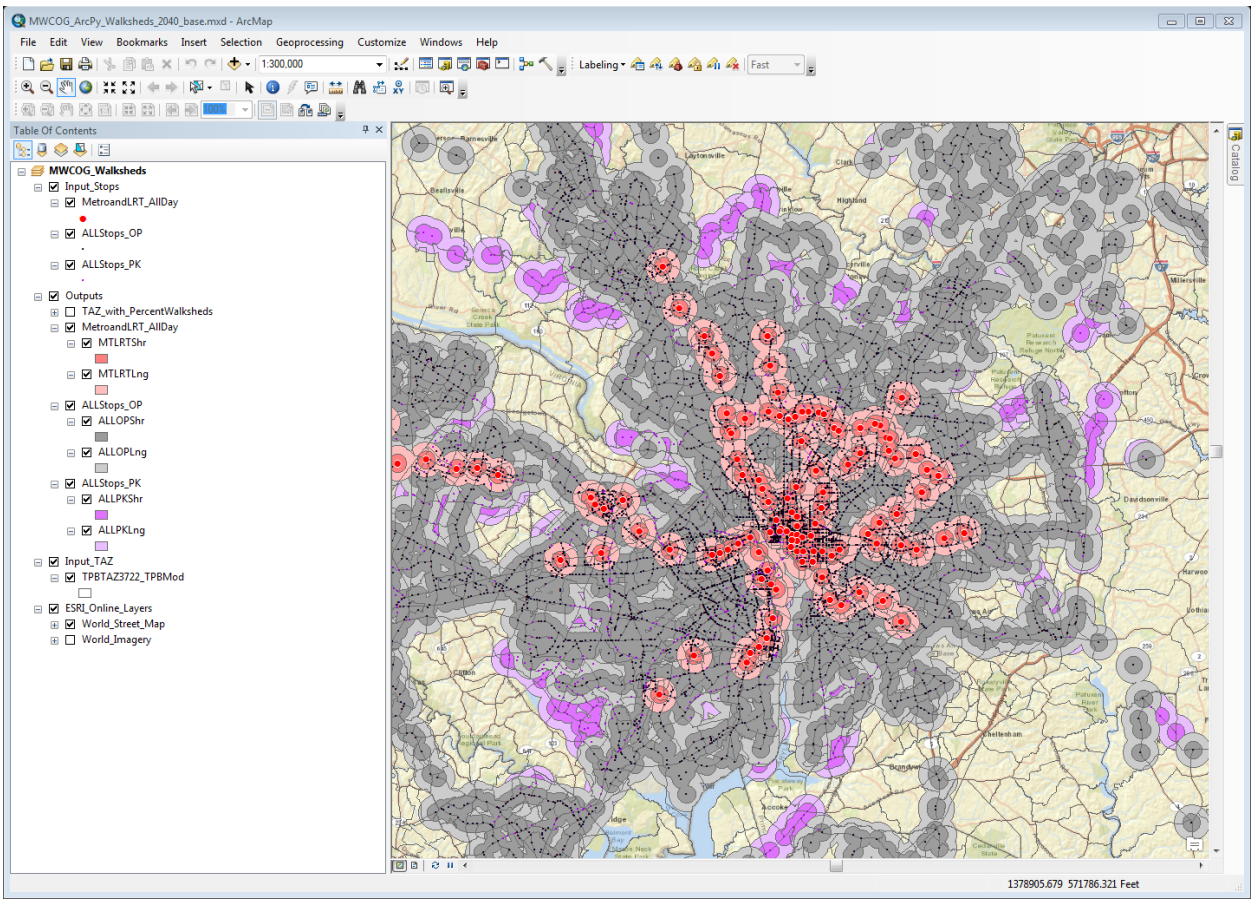
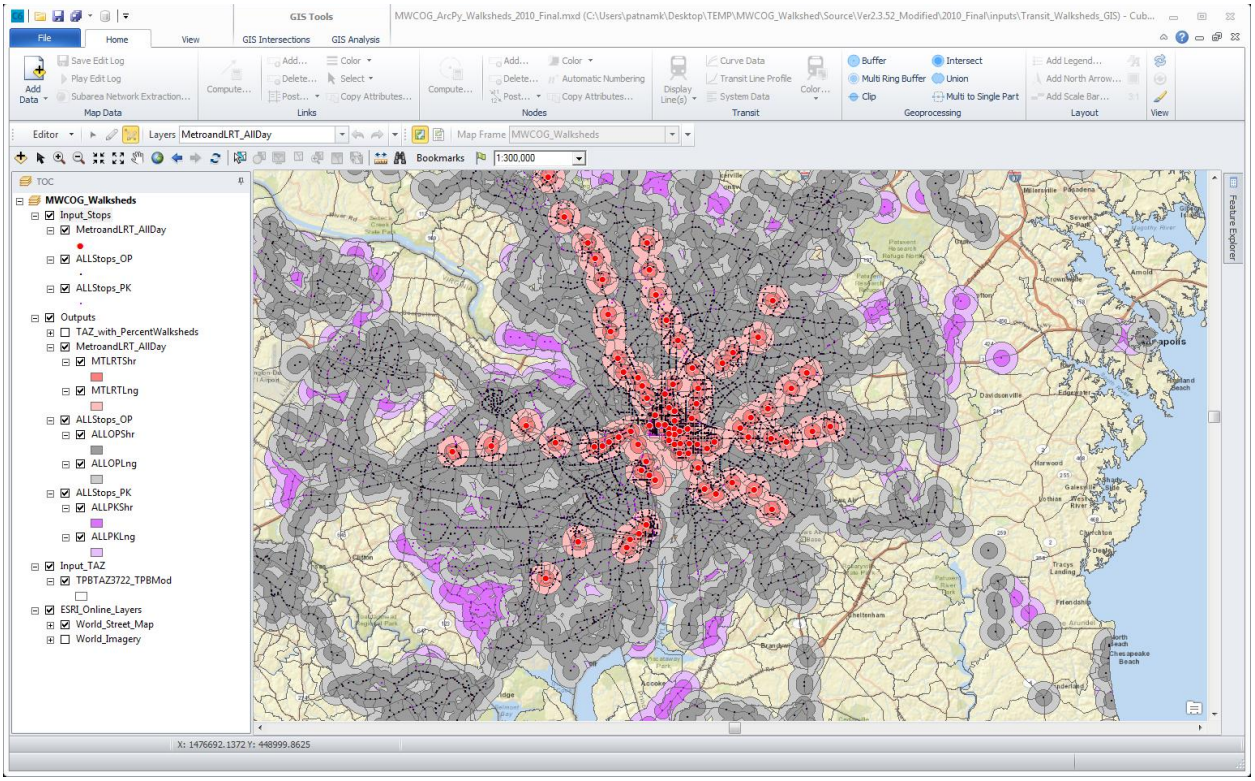


Figure 3-10: Output 2020 Walkshed Buffers in Cube



3.4 Conclusion and Future Improvements

The proposed PWT process using ArcPy and Cube was successfully integrated with the TPB modeling process while removing the dependency on a full implementation of ArcGIS, but still maintaining compatibility with it. To improve the reliability of the PWT process, the latest release from Cube – Cube 6.1.0 SP1 was chosen as the desired platform. Cube 6.0.2 could not be chosen because the ArcGIS runtime that is included with it (ArcGIS runtime version 10.0) has an issue[7] in its ArcPy installation which prevents a function ('Join') from executing. Nonetheless, it is anticipated that future releases of Cube and ArcGIS will be compatible with this proposed PWT process. It is worth noting that users with an existing ArcGIS runtime (likely 9.3.1 or 10.0) who want to upgrade to ArcGIS runtime 10.1 should perform a "clean" install by removing the previous ArcGIS runtime prior to installing ArcGIS runtime 10.1.

The results of the proposed PWT process with a TAZ polygon containing total TAZ (TPBTAZ3722_TPBMod.shp/.dbf/.shx/.prj) area were found to be consistent with the current COG 'AreaWalk.txt' file [8] with the exception of certain TAZs with water-bodies. However, after using a TAZ

[7] This issue causes ArcPy to require an elevated ('ArcInfo') type of license to execute the 'Join_Management' ArcPy function in ArcGIS 10.0 or its equivalent runtime.

[8] AreaWalk.txt dated 9/10/2013 9:34 AM size 280 KB as packaged with '2010_Final' scenario of the TPB model

polygon that included only the TAZ land area (TAZLand3722.shp/.dbf/.shx/.prj), the results were found to be practically identical.

As part of internal testing, COG investigated[9] and found the primary source of the difference for the TAZs where the proposed PWT process's results were not exactly identical to those from COG's current process. This source relates to the way Cube handles stop nodes in TRNBUILD line files, wherein, the native behavior of TRNBUILD treats the first transit node in a transit route and the last node in a transit route as a stop node, even if the node is coded as a "through" or "non-stop" node (i.e., preceded by a minus sign). COG also found that this behavior was limited to TRNBUILD and did not extend to PT. Since the proposed PWT process automates the preparation of input shapefiles from TRNBUILD/PT line files, the first and last transit nodes of all line files are treated (by TRNBUILD) as stop nodes whereas as the inputs for COG's process are manually prepared wherein the stop-nodes are appropriately handled, thereby causing the results to differ slightly. This issue is expected to go away when the transit modeling is converted from TRNBUILD to PT and all the transit line files are coded in PT. Furthermore, even for TRNBUILD-format line files, cases where the first and last node are both coded as non-stop nodes are generally not correct, so the coding, in these cases, can be corrected. It is anticipated that minor revisions may be necessary to the Cube Voyager script (MWCOG_Prepare_Inputs_to_Walkshed_Process_PT.s) that prepares the input shapefiles to accommodate any changes associated with full conversion to PT files.

Preparation of other GIS-based transit network inputs to the TPB model, such as PEF (pedestrian environment factor), could be implemented using ArcPy as well. Further investigation into improving the ArcPy processing times may also be worth considering.

[9] Memorandum *Comments on your Cube/ArcPy-based transit walkshed process and its associated memo* dated March 25, 2013 from Mark S. Moran and Dzung Ngo, COG/TPB to David Roden and Krishna Patnam, AECOM, May 15, 2014

4 HOV Modeling (Task Order 12)

This chapter documents the work performed for Task Order 12 related to HOV choice modeling. These procedures have been integrated into the COG/TPB model version 2.3.52 setup for application testing.

4.1 Background

The COG/TPB Version 2.3.52 Travel Model includes a series of procedures for modeling high-occupancy vehicle (HOV) lanes and high-occupancy/toll (HOT) lanes. HOT lanes are also referred to as "express toll lanes" or "express lanes," such as the I-495 Express Lanes. In the Washington, D.C. area, HOT lanes exist only in Virginia and the occupancy requirement for using the lanes for free is HOV3+. Both HOV lanes and HOT lanes are considered managed-lane facilities. However, HOT lanes are a special case, since they involve both HOV traffic and single-occupant vehicle (SOV) traffic on the same facility. The traffic assignment process used in the Version 2.3.52 Travel Model is a multi-class assignment, which means that there are separate markets or user classes (trip tables) that are loaded during each assignment. Currently six user classes are differentiated:

1. Single-occupant vehicle (SOV)
2. High-occupant vehicle with two persons (HOV2)
3. High-occupant vehicle with three or more persons (HOV3+)
4. Medium and heavy trucks
5. Commercial vehicles
6. Airport passengers traveling to/from the three commercial airports in the region

In theory, one would simply use these six user classes in a multi-class traffic assignment. However, work done in 2008 with an earlier version of the travel model (Ver. 2.2) showed that traffic on the Beltway HOV lanes was overestimated and traffic on the I-395/Shirley Highway HOV lanes was underestimated during the peak-period traffic assignments.[10] To correct this issue, COG/TPB staff divided the peak-period highway assignment (both AM and PM) into two parts: HOV3+ traffic and non-HOV3+ traffic (i.e., SOV, HOV2, trucks, and airport passengers). This HOV assignment technique is known as the "**two-step assignment**," since it split the AM and PM peak period assignment into two steps.[11]

In addition to the two-step assignment, a second special modeling procedure was developed to better model travel on HOT-lane facilities. This HOT-lane modeling procedure is called the "**HOV3+ skim replacement**" (HSR) procedure, or the "multi-run" procedure, since it involves using multiple model runs. When toll setting is required for HOT lanes being modeled, there are four model runs:

1. Toll Base

[10] Jinchul Park to Files, "Two Step Traffic Assignment for HOT Lane Modeling in 2008 CLRP," Memorandum, (December 2, 2008).

[11] Ronald Milone et al., *User's Guide for the MWCOG/NCRTPB Travel Forecasting Model, Version 2.3, Build 52*, Draft Report (Washington, D.C.: Metropolitan Washington Council of Governments, National Capital Region Transportation Planning Board, September 18, 2013), 180–182, http://www.mwcog.org/transportation/activities/models/documentation.asp.

2. Toll Pump Prime
3. Toll Setting
4. Final

Details about these four steps can be found in a memo dated April 26, 2010.[12]

In many cases, tolls would has already been set by another modeler. For example, the standard transmittal of the COG/TPB travel model includes a set of toll values for the HOT lanes that were estimated by COG/TPB staff. In these cases, the HOT-lane tolls are taken as given at the start of a series of model runs. Even so, using the current COG/TPB process, multiple runs are still required for each scenario, but the number of runs is only two, not four:

1. Base
2. Final

The "**base**" run is designed to generate the HOV travel times used in the mode choice model. Since HOT lanes exist only in the Virginia portion of the region and since the occupancy requirement for using the lanes for free is HOV3+, the use of the term "HOV" in this section implicitly refers to HOV3+. According to HOV policy in Virginia, HOVs should have priority when using HOV facilities and the introduction of HOT lane facilities should not significantly deteriorate travel times of HOVs. To reflect this policy in the travel model, COG/TPB staff developed a process where, in the "base" run, HOV travel times are calculated by designating all HOT-lane facilities as HOV-lane facilities.[13] This captures the travel time for unimpeded flow of HOV traffic on HOT lanes, consistent with VDOT's stated policy. In the "**final**" run, HOT-lane facilities are designated as HOT-lane facilities, but all HOV skims used for the "final" run are actually taken from the "base" run (hence "HOV3+ skim replacement"). Skims for all other modes are taken from the "final" run. In this methodology, the sole purpose of conducting the "base" run is for measuring the travel times of HOV traffic on HOT-lane facilities.[14]

The "HOV3+ skim replacement" technique helps keep the HOT-lane facilities attractive in the mode choice model and the "two-step assignment" technique helps dis-incentivize non-HOV-link paths, thereby together acting to improve the assignment of HOV trips onto HOT-lanes and HOV-facilities. One of the main drawbacks to using the "HOV3+ skim replacement" technique to model HOT-lane facilities is that it requires multiple model runs per modeling scenario. So, in a case where tolls are already set, modeling each scenario (e.g., 2020) requires two model runs, and, in a case where toll setting needs to be accomplished, the four model runs can take over a week of time. Thus, one of the primary motivations behind Task Order 12 (and its predecessor, Task Order 8) was to develop a new HOT-lane modeling process that would not require as much run time, but would still respect Virginia Department of Transportation (VDOT) policies about HOV travel. It was also felt that there may be a

[12] Jinchul Park to Files, "HOT Lane Modeling Process of MWCOG/TPB (Draft)," Memorandum, (April 26, 2010).
[13] Ibid.
[14] Milone et al., *User's Guide for the MWCOG/NCRTPB Travel Forecasting Model, Version 2.3, Build 52*, 19.

way to reduce model run times for HOV-lane modeling, since the "two-step" assignment requires longer model run times than not using the "two-step" assignment.

Task Order 8 of the FY 2013 final report[15] was successful in developing and testing procedures to improve HOT-lane modeling. Similar to its two subtasks, the Task Order 8 work was composed of two primary components: (1) an HOV choice model and (2) a revised traffic assignment procedures to perform integrated, but optional, toll-setting. The HOV choice model was successful in improving the HOV3+ assignment on the I-395/I-95 corridor in a multi-class traffic assignment by adjusting SOV/HOV trips based on HOV travel time savings for work trips. This HOV choice model consisted of a secondary HOV-choice model, made immediately after and outside of the main mode choice model, and was limited in its calibration due to a lack of HOV traffic counts on facilities other than the I-95/I-395 corridor. This HOV choice model was originally envisioned as an initial proof of concept toward the replacement of the "two-step assignment" with multi-class assignments and to support the development of HOT lane modeling. The plan was to ultimately integrate the model into a restructured and re-calibrated mode choice model in the future. Nonetheless, in the interest of clarity during the transition from the 'AEMS' to 'ModeChoice' software, as part of Task Order 13, the existing structure of the MWCOG mode choice model was retained for now. In other words, re-estimation of the mode choice model, although desirable, was not performed as part of the software transition. Retaining this "interim" HOV choice model is anticipated to help obtain better HOV volume assignments to HOT lanes.

Task Order 12 was scoped to incorporate the recommendations and procedures from Task Order 8 into the COG/TPB Version 2.3.52 Travel Model and refine inputs to the extent data is available, and document the process. The primary objective was to enhance the capability of a future version of the MWCOG model, rather than to replicate the results of the current MWCOG Version 2.3.52 Travel Model.

This chapter documents the enhanced procedures, originally developed as part of Task Order 8, which have been incorporated into the COG/TPB Version 2.3.52 Travel Model. The HOV choice model is presented first followed by HOT lane modeling procedures in Chapter 5.

4.2 HOV Choice

Unlike the previous effort in Task Order 8, where only the work trips were modified after supplying the mode choice model with SOV skims in place of HOV skims, Task Order 12 expanded model calibration to include all trip purposes without changing any inputs to the mode choice models. This meant that the HOV choice model developed earlier was re-designed and re-calibrated to choose between SOV, HOV2 and HOV3+ modes based on additional traffic counts. As this Task Order was primarily geared towards implementation of the Task Order 8 procedures, some of the relevant portions of Task Order 8 are repeated here to provide context.

[15] FY 2013 Draft Report, Assistance with Development and Application of the National Capital Region Transportation Planning Board Travel Demand Model, submitted to MWCOG by AECOM with Stump Hausman, July 1st, 2013

As mentioned earlier, COG/TPB model version 2.3 assigns HOT-lane traffic using a multi-run traffic assignment. The main goal of this technique is to ensure that travel speeds for the HOV 3+ traffic on the HOT lanes are not degraded by the other traffic using the HOT lanes. The key to properly modeling HOV facilities is to model travelers who choose to form a carpool or use transit in order to take advantage of the travel time or cost savings offered by the HOV lane differently from travelers who carpool for reasons other than time savings (such as a family traveling together). What this means from a behavioral modeling perspective is that an HOV lane should be considered differently in a mode choice model than it will be in an assignment model. In the assignment model, any vehicle that qualifies under the occupancy restriction can choose to use the HOV lane if it saves the travelers travel time or cost. The mode choice model, however, needs to limit the HOV choice to those trips that have an HOV alternative and model general auto occupancy as a household travel decision. A mode choice model calibrated with indistinguishable modes is not likely to perform effectively as a predictive tool, especially given HOT lane incentives.

It makes sense that auto occupancy will vary by trip purpose and time of day. Unfortunately, it is hard to find observed data that shows these variations, since traffic counts typically cannot indicate trip purpose. Consequently, modeled data are used to highlight a few points. Table 4-1 shows daily mode shares for single- and high-occupancy vehicles (SOV and HOV) in the Washington D.C. region.[16] This table shows how, in the model, and, presumably in real life, auto occupancy varies by trip purpose. However, despite the variation in mode share, very few of these HOV trips have different travel characteristics from the corresponding SOV trips. The vast majority of HBO and NHB trips are joint household trips that are formed without regard to the relative performance of competing modes. This includes parents taking their kids to school, family members going out to eat or shop, and business associates traveling to a meeting or to lunch.

Table 4-1: Daily Mode Shares for SOV and HOV Trips

Mode	HBW	HBO	HBS	NHO	NHW	Total
SOV	67.4%	38.1%	43.0%	42.9%	65.2%	47.7%
HOV	11.7%	58.8%	56.3%	56.1%	29.0%	46.4%

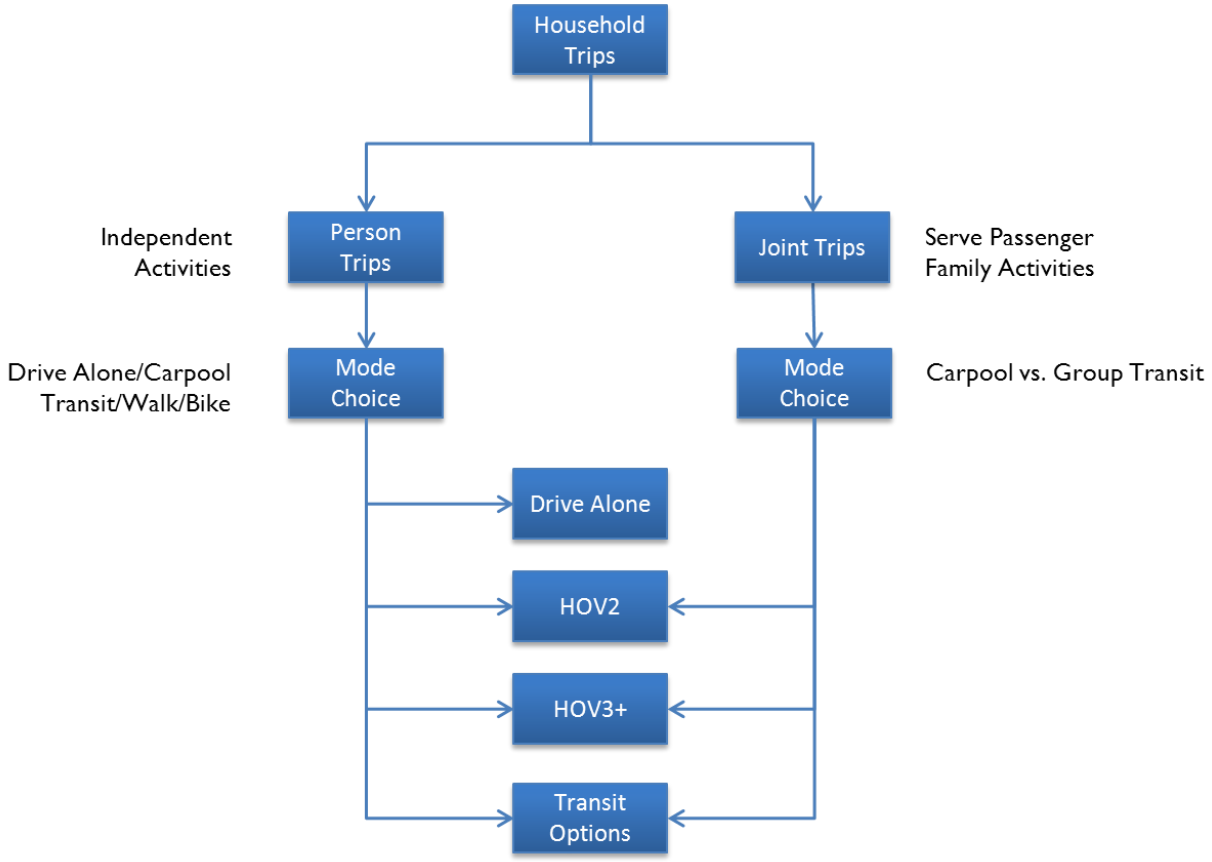
Source: MWCOG Model for 2040 Base Conditions

4.3 HOV Choice Models

The general structure of a comprehensive HOV choice model is shown in Figure 4-1 In this case, the household trips are assigned to independent person trips or joint household trips based on the trip purpose and joint travel rates recorded in the household survey. The joint trip probability model should be estimated using travel markets that do not provide a significant travel time or cost advantage for HOV trips. In other words, the objective is to estimate group travel based on household and general trip characteristics, rather than mode-specific information.

[16] The estimated data is from a model run which represents year-2040 conditions.

Figure 4-1: HOV Choice Structure: Comprehensive HOV Choice Model

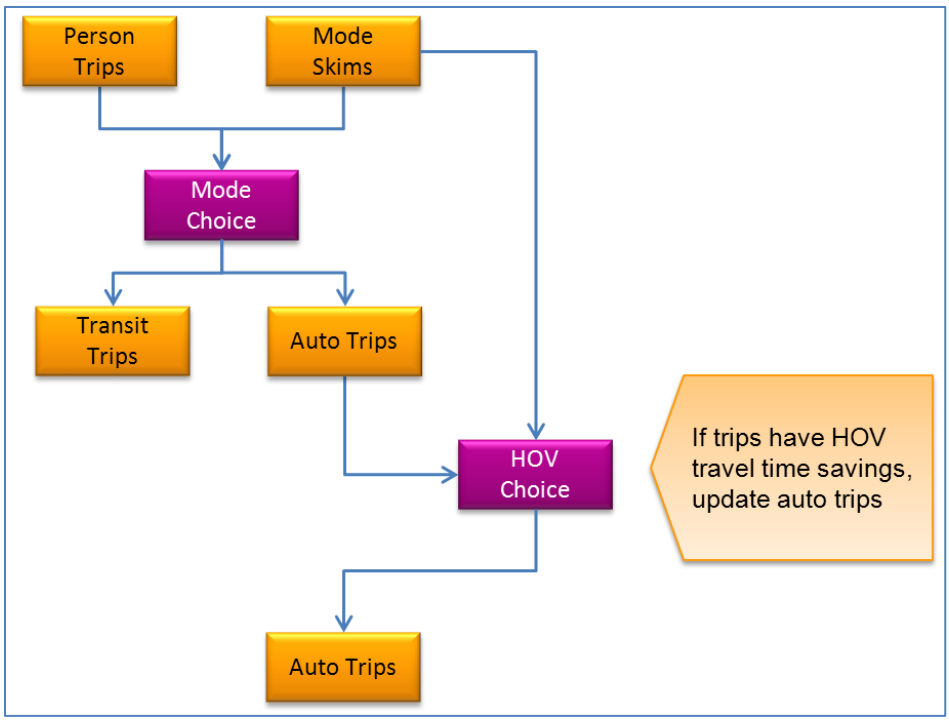


For joint or group trips, the mode choice model can be relatively simple. It does not include drive-alone options, so the basic choice is between a standard carpool mode and traveling as a group on transit. In other words, the HOV travel time and cost are compared to the transit travel time and group fares. Since the market for family travel by transit is quite limited (e.g., zero-car households and activity locations where parking is extremely limited or expensive, such as sports events, etc.), the vast majority of joint household trips will be made by automobile.

The independent person trip mode choice model includes the full array of travel modes, but will generally limit the HOV2 and HOV3+ options to trip interchanges with a travel time or cost advantage. (This is not a hard and fast rule, but it does simplify the model calibration to some extent).

Given the resources of this task order, it was not possible to estimate the comprehensive HOV choice model shown in Figure 4-1. Consequently, a simple HOV choice model was developed and tested, as shown in Figure 4-2.

Figure 4-2: HOV Choice Structure: Simplified HOV Choice Model



Source: Roden, D. B. (2013, March 22). *HOV/HOT Lane Modeling and Public Transport Research*. Presented at the March 22, 2013 meeting of the Travel Forecasting Subcommittee, held at the Metropolitan Washington Council of Governments, Washington, D.C.

This simplified HOV choice model was re-calibrated using the following procedure:

1. Use the HOV skims in the current mode choice models for HOV modes
2. Assign all trips to the network using appropriate HOV restrictions.
3. Compare the major highways HOV volumes to observed counts (they should be low).
4. Compare the AM peak period LOV skims to the HOV skims in the major corridors and identify zone-to-zone interchanges that have an HOV travel time advantage.
5. Estimate a generic binary choice model that adjusts the splits between SOV, HOV2 and HOV3+ trips based on travel time difference. Calibrate the model to match major highway volumes to observed counts.

Since the mode choice and the HOV choice models are applied to daily trips, the final trips for each mode are divided into four time periods in a separate procedure to be used in traffic assignment.

4.4 HOV Assignment Models

The output of the HOV choice and auto occupancy models are SOV, HOV2 and HOV3+ trips that can be aggregated by time of day for traffic assignment. Given the improved method of estimating HOV demand, it is likely the HOV volumes assigned to the HOV lanes during a single multi-class assignment will match the observed counts reasonably well. This should eliminate the need for a two-step assignment process and improve the overall consistency of the traffic assignment.

The probability of success could be further enhanced by modeling HOV links using perceived travel times rather than actual travel times in the generalized cost function. This perceived travel time is likely to reflect the reliability of the travel time on the HOV lanes as opposed to the general purpose lanes. The day-to-day impacts of non-recurring congestion in the general purpose lanes are frequently considered by travelers as they select roadways and schedule trips. If the lane is a managed HOT lane, a reliable speed should be maintained 95 percent of the time.

The use of perceived travel time and user-based tolls in a multi-class assignment imply that each vehicle class has a different generalized cost function for each link in the network. There are also link-use restrictions that determine the types of links the model can use to construct the path. These include HOV2, HOV3+, bus-only, and truck restrictions. They are also likely to include toll and no-toll restrictions if a toll choice model is included. This perceived travel time is implemented using vehicle-class specific valuations of time as discussed in the HOT-lanes modeling chapter.

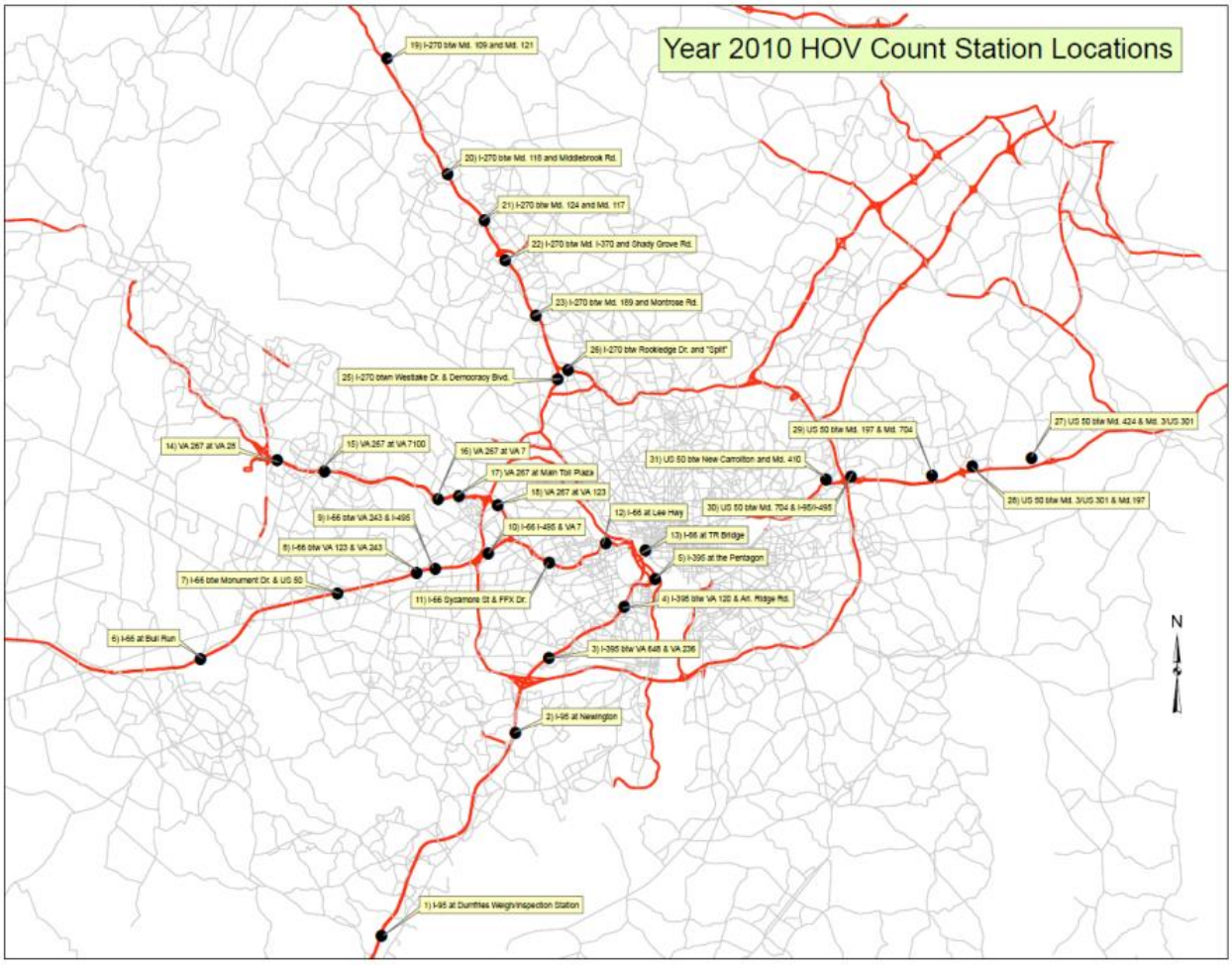
4.5 HOV Model Calibration

The calibration effort for the simplified HOV choice model involved the five steps outlined above. The HOV demand resulting from steps 1 and 2 is considered the background HOV demand. These trips chose HOV modes naturally and not due to travel time savings associated with that mode. Under step 2, the highway assignment was performed using a single multi-class assignment in which HOV3+ trips are assigned along with the rest of the modes.

The resulting volumes in the general purpose (GP) and HOV lanes were compared with the available count data[17]. Daily HOV counts were available for the 16 locations along I-95/I-395, I-66, VA-267, and I-270 shown on Figure 4-3. Table 4-2 describes the location and HOV type (occupancy requirement of 2 or 3 persons) of each count station. HOV counts during the AM and PM peak periods were available at all locations. A comparison of the observed and model-estimated volumes for AM and PM peak periods is shown in Table 4-3 and Table 4-4, respectively.

[17] Memorandum "Transmittal of 2010 HOV count data to AECOM for Task Order 12 (Traffic Assignment) of COG Contract 12-006" dated October 11 2013 from Mark Moran and Meseret Seifu (COG/TPB) to David Roden (AECOM)

Figure 4-3: Traffic Count Locations



Source: Traffic counts compiled by MWCOG staff

Table 4-2: Traffic Count Stations

Location	Facility	Count Station	HOV Type
1	I-95/I-395	I-95 at Dumfries	3
2	I-95/I-395	I-95 north of Newington	3
3	I-95/I-395	I-395 between VA 648 (Edsall Rd) & VA 236 (Duke St)	3
4	I-95/I-395	I-395 between VA 120 (S. Glebe Rd) & Arlington Ridge Rd	3
5	I-95/I-395	I-395 at the Pentagon	3
6	I-66	I-66 between VA 234 Business and Bull Run	2
7	I-66	I-66 between Monument Drive and U.S. 50	2
8	I-66	I-66 between VA 123 and VA 243 (Nutley Street)	2
9	I-66	I-66 between VA 243 (Nutley Street) and I-495	2
10	VA-267	VA267 at VA7100 (Fairfax County Pkwy)	2
11	VA-267	VA267 between Trap Road and VA7 (Leesburg Pike)	2
12	I-270	I-270 between I-370 and Shady Grove Road	2
13	I-270	I-270 between Md. 189 (Falls Rd) and Montrose Road	2
14	I-270	I-270 between Montrose Road and the "Split" (max load point)	2
15	I-270	I-270 Spur between the "Split" and Democracy Blvd	2
16	I-270	I-270 at Rockledge Dr.	2

Table 4-3: 2010 Background GP and HOV Assignments along the Regional Major Corridors (AM Peak)

Loc	HOV Type	GP_OBS	GP_EST	EST/OBS	HOV_OBS	HOV_EST	EST/OBS	OBS	EST	EST/OBS
1	3	10,692	14,792	138%	3,107	1,299	42%	13,799	16,091	117%
2	3	16,523	19,905	116%	8,248	2,974	36%	24,771	22,069	89%
3	3	15,138	15,498	102%	7,228	3,008	42%	22,366	18,506	83%
4	3	21,911	16,435	75%	8,496	3,709	44%	30,407	20,144	66%
5	3	17,957	7,254	40%	8,784	1,644	19%	26,741	8,898	33%
6	2	10,467	15,381	147%	4,938	4,660	94%	15,405	20,041	130%
7	2	11,321	13,872	123%	3,271	4,182	128%	14,592	18,054	124%
8	2	11,388	14,556	128%	3,841	4,390	114%	15,229	18,946	124%
9	2	14,436	11,691	81%	3,821	4,390	115%	18,257	16,081	88%
10	2	11,560	14,482	125%	3,458	3,201	93%	15,018	17,683	118%
11	2	14,054	17,229	123%	4,540	5,082	112%	18,594	22,311	120%
12	2	13,107	20,852	159%	3,442	4,197	122%	16,549	25,049	151%
13	2	22,324	23,092	103%	3,470	4,451	128%	25,794	27,543	107%
14	2	26,987	24,349	90%	4,260	5,195	122%	31,247	29,544	95%
15	2	12,511	13,910	111%	1,837	2,325	127%	14,348	16,235	113%
16	2	14,476	10,455	72%	2,423	2,782	115%	16,899	13,237	78%
All		**244,852**	**252,944**	**103%**	**75,164**	**57,490**	**76%**	**320,016**	**310,434**	**97%**

Loc – Count Location, GP – General Purpose Lane, OBS – Observed, EST - Estimated

Sources for observed counts:

1) Zilliacus, C. P., & Reschovsky, C. (2011). 2010 Performance of High-Occupancy Vehicle Facilities on Freeways in the Washington Region. Washington, D.C.: National Capital Region Transportation Planning Board.
2) Moran, M. S., & Seifu, M. (2013, October 11). Transmittal of 2010 HOV count data to AECOM for Task Order 12 (Traffic Assignment) of COG Contract 12-006. Memorandum.

Notes: Observed counts are single-day counts, but do not include the entire day. The COG/TPB counts focus only on the **peak periods in the peak direction**, e.g., I-395 northbound in the AM and I-395 southbound in the PM.

HOV Type = HOV occupancy requirement for the link (e.g., 2 => 2+ persons).

Counts were taken on both the HOV lanes and the general purpose (GP) lanes. Although the GP counts will be mainly SOV, they will also include some HOV, since not all vehicles carrying multiple occupants use the HOV lanes. Similarly, although the HOV counts will include mainly HOV, they may also include LOV traffic, such as hybrid cars and HOV violators.

Table 4-4: 2010 Background GP and HOV Assignments along the Regional Major Corridors (PM Peak)

Loc	HOV Type	GP_OBS	GP_EST	EST/OBS	HOV_OBS	HOV_EST	EST/OBS	OBS	EST	EST/OBS
1	3	14,490	20,789	143%	3,527	2,344	66%	18,017	23,133	128%
2	3	15,034	25,852	172%	13,613	5,436	40%	28,647	31,288	109%
3	3	21,790	22,249	102%	12,207	4,836	40%	33,997	27,085	80%
4	3	31,927	23,302	72%	12,403	6,214	50%	44,330	29,516	67%
5	3	21,033	14,210	68%	10,056	3,008	30%	31,089	17,218	55%
6	2	17,023	20,371	120%	5,155	6,597	128%	22,178	26,968	122%
7	2	12,554	19,561	156%	4,957	5,824	117%	17,511	25,385	145%
8	2	16,626	20,516	123%	4,403	6,563	149%	21,029	27,079	129%
9	2	16,894	17,870	106%	4,346	6,563	151%	21,240	24,433	115%
10	2	18,010	19,240	107%	3,725	5,508	148%	21,735	24,748	114%
11	2	20,805	21,879	105%	5,019	7,500	149%	25,824	29,379	114%
12	2	31,215	29,913	96%	4,341	6,246	144%	35,556	36,159	102%
13	2	29,164	33,068	113%	5,001	6,540	131%	34,165	39,608	116%
14	2	31,139	31,590	101%	8,280	7,994	97%	39,419	39,584	100%
15	2	16,384	15,616	95%	4,069	4,453	109%	20,453	20,069	98%
16	2	14,755	15,974	108%	4,211	3,309	79%	18,966	19,283	102%
All		**328,843**	**352,001**	**107%**	**105,313**	**88,937**	**84%**	**434,156**	**440,937**	**102%**

Sources for observed counts:

1) Zilliacus, C. P., & Reschovsky, C. (2011). 2010 Performance of High-Occupancy Vehicle Facilities on Freeways in the Washington Region. Washington, D.C.: National Capital Region Transportation Planning Board.
2) Moran, M. S., & Seifu, M. (2013, October 11). Transmittal of 2010 HOV count data to AECOM for Task Order 12 (Traffic Assignment) of COG Contract 12-006. Memorandum.

The comparison indicates that the AM and PM period GP and HOV model volumes are comparable to the observed counts at the regional level. The performance by individual station varies quite a bit. Averaged across all sixteen stations, the GP model volumes are higher and the HOV model volumes are lower than the counts, though, again, there is wide variation at the station level. This suggests that the GP volumes at the daily level should be decreased to match the observed counts. Table 4-3 and Table 4-4 also show that HOV3+ model volumes are significantly lower than counts (EST/OBS is less than 100% for count locations 1-5), while model volumes at most of HOV2 locations are higher than counts (EST/OBS is mostly greater than 100% for count locations 6-16). This observation suggests that trips should be shifted from HOV2 to HOV3+ to make the model volumes more consistent with the counts.

The comparison also suggests that the estimated HOV volumes in the AM peak period are considerably lower than the counts (see, for example the estimated-to-observed ratio of 76% in the bottom of Table 4-3). One of the reasons for the discrepancy between AM peak and daily model volumes could be that the model does not account for the closure of HOV lanes for a few hours during the day.

In the original HOV choice model,[18] an incremental logit model was developed to increase the peak HOV3+ volumes on major highways. In expanding this concept to all trip purposes and HOV2+ facilities, the project team decided to replace the incremental model with a standard logit model. The equation of the previous model did not completely obey the structure of an incremental logit model, while the new model includes the well-known equation which is based on discrete choice model rules. The incremental logit model is used when the choice probabilities of the alternatives exist and one or more independent variables change; while, in the HOV choice model travel times come from the original model unchanged. In fact, HOV facilities save users' travel time in comparison to general purpose lanes, but these travel times belong to two competitive modes. If for any reason the travel time of any mode changes after the primary mode choice model, the incremental logit model would be the best way to recalculate the new mode shares. Since the travel times do not change, a standard logit model was developed to estimate the peak HOV3+ volumes on major highways. The shift from HOV2 to HOV3+ is based on network travel time benefits for HOV3+ from the HOV lanes. The model was calibrated with the objective to match the AM and PM peak HOV model volumes to the counts. The probability of choosing one alternative from two alternatives according to a binary logit model is equal to:

$$Pr_i = \frac{e^{u_i}}{e^{u_i} + e^{u_j}}$$

Where:

Pr_i is the probability of choosing alternative "*i*" ; and

u_i is the utility of alternative "*i*".

This equation can be written in another form[19] as follows:

$$Pr_a = \frac{e^{u_i}}{e^{u_i} + e^{u_j}} = \frac{e^{u_i}/e^{u_j}}{e^{u_i}/e^{u_j} + e^{u_j}/e^{u_j}} = \frac{e^{\Delta u_{ij}}}{1 + e^{\Delta u_{ij}}}$$

Where:

Δu_{ij} is the difference between utilities of alternative "*i*" and alternative "*j*".

[18] FY 2013 Draft Report, Assistance with Development and Application of the National Capital Region Transportation Planning Board Travel Demand Model, submitted to MWCOG by AECOM with Stump Hausman, July 1st, 2013.

[19] Ben-Akiva, M., & Lerman, S. (1985). Discrete Choice Analysis: Theory and Application to Travel Demand (1st ed.). The MIT Press.

The same concept was used in the HOV choice model to calculate revised HOV3+ demand as follows:

$$R\_HOV3_{ij} = K_{ij} * HOV2_{ij} * \frac{e^{\lambda \Delta T_{ij}}}{1 + e^{\lambda \Delta T_{ij}}}$$

$$K_{ij} = \frac{HOV3_{ij}}{\beta * HOV2_{ij}}$$

Where:

R_HOV3_{ij} is the revised HOV3+ trips between *i* and *j* zones;

$HOV2_{ij}$ is the original HOV2 trips between *i* and *j* zones;

$HOV3_{ij}$ is the original HOV3+ trips between *i* and *j* zones;

ΔT_{ij} is the travel time saving by using the HOV3+ lane network; and

$\lambda \; and \; \beta$ are the model parameters.

Another difference between the previous version and the current version of the HOV choice model is the trip purposes affected by the model. The previous version of the HOV choice model was applied to only HBW trips, while the new model is applied to all purposes. Nonetheless, HOV trips in the off-peak time periods generally do not have access to HOV lanes (since they are not in operation), so they must use general purpose lanes. Since most of the peak-period trips are HBW trips, applying the HOV choice model to all purposes does not affect the results significantly, but this decision makes the HOV choice model consistent with the general mode choice model.

The suggested logit model has two λ parameters and one β parameter. The first λ is applied to interchanges with more than half-a-minute travel time benefit. The other λ is applied to interchanges with less travel time benefit. The model was designed with two λ parameters in order to provide flexibility to vary the effect on interchanges based on travel time criteria. If, however, it is desired to simplify the model, it can be changed to use only one λ parameter. β is the other model parameter that affects the portion of HOV2 trips influenced by this model. A lower β will shift more trips from HOV2 to HOV3+.

The additional HOV3+ demand is deducted from HOV2 demand sets. The model was calibrated through multiple iterations that seek to match the HOV volumes at count locations. The best model was found by setting λ_1, λ_2 , and β to 5, 5, and 0.05, respectively. Table 4-5 and Table 4-6 show the calibrated model volumes at count locations for AM and PM peak periods, respectively.

Table 4-5: 2010 Calibrated Model GP and HOV Volumes along the Regional Major Corridors (AM Peak)

Loc	HOV Type	GP_OBS	GP_EST	EST/OBS	HOV_OBS	HOV_EST	EST/OBS	OBS	EST	EST/OBS
1	3	10,692	13,320	125%	3,107	1,628	52%	13,799	14,948	108%
2	3	16,523	17,895	108%	8,248	4,071	49%	24,771	21,966	89%
3	3	15,138	15,314	101%	7,228	4,019	56%	22,366	19,333	86%
4	3	21,911	16,099	73%	8,496	5,559	65%	30,407	21,658	71%
5	3	17,957	7,440	41%	8,784	2,242	26%	26,741	9,682	36%
6	2	10,467	15,455	148%	4,938	4,308	87%	15,405	19,762	128%
7	2	11,321	13,794	122%	3,271	3,726	114%	14,592	17,520	120%
8	2	11,388	13,831	121%	3,841	4,056	106%	15,229	17,887	117%
9	2	14,436	11,467	79%	3,821	4,056	106%	18,257	15,523	85%
10	2	11,560	13,577	117%	3,458	2,903	84%	15,018	16,480	110%
11	2	14,054	16,783	119%	4,540	4,339	96%	18,594	21,122	114%
12	2	13,107	20,617	157%	3,442	4,024	117%	16,549	24,640	149%
13	2	22,324	22,743	102%	3,470	4,199	121%	25,794	26,942	104%
14	2	26,987	24,190	90%	4,260	4,893	115%	31,247	29,083	93%
15	2	12,511	13,857	111%	1,837	2,221	121%	14,348	16,078	112%
16	2	14,476	10,333	71%	2,423	2,559	106%	16,899	12,892	76%
All		**244,852**	**246,715**	**101%**	**75,164**	**58,801**	**78%**	**320,016**	**305,516**	**95%**

OBS – Observed, EST – Estimated, GP – General Purpose Lane

Table 4-6: 2010 Calibrated Model GP and HOV Volumes along the Regional Major Corridors (PM Peak)

Loc	HOV Type	GP_OBS	GP_EST	EST/OBS	HOV_OBS	HOV_EST	EST/OBS	OBS	EST	EST/OBS
1	3	14,490	17,195	119%	3,527	2,956	84%	18,017	20,151	112%
2	3	15,034	23,234	155%	13,613	7,542	55%	28,647	30,776	107%
3	3	21,790	20,168	93%	12,207	6,086	50%	33,997	26,254	77%
4	3	31,927	21,967	69%	12,403	8,548	69%	44,330	30,516	69%
5	3	21,033	13,019	62%	10,056	3,671	37%	31,089	16,690	54%
6	2	17,023	20,464	120%	5,155	6,395	124%	22,178	26,859	121%
7	2	12,554	18,802	150%	4,957	5,703	115%	17,511	24,506	140%
8	2	16,626	18,659	112%	4,403	6,286	143%	21,029	24,945	119%
9	2	16,894	16,537	98%	4,346	6,286	145%	21,240	22,823	107%
10	2	18,010	18,184	101%	3,725	5,139	138%	21,735	23,323	107%
11	2	20,805	21,067	101%	5,019	6,973	139%	25,824	28,040	109%
12	2	31,215	26,916	86%	4,341	6,121	141%	35,556	33,037	93%
13	2	29,164	29,711	102%	5,001	6,341	127%	34,165	36,053	106%
14	2	31,139	29,705	95%	8,280	7,725	93%	39,419	37,430	95%
15	2	16,384	15,115	92%	4,069	4,618	114%	20,453	19,734	96%
16	2	14,755	14,590	99%	4,211	2,836	67%	18,966	17,425	92%
All		**328,843**	**325,334**	**99%**	**105,313**	**93,226**	**89%**	**434,156**	**418,562**	**96%**

OBS – Observed, EST – Estimated, GP – General Purpose Lane

Note that the main goal of this HOV choice model is to ensure that travel speeds for the HOV 3+ traffic on the HOT lanes are not degraded by other traffic using the HOT lanes. This is currently done in the COG model using a combination of “multi-run” and “two-step” assignments. Since the HOT-lane modeling procedure, as discussed in the next chapter, consolidates the “multi-run” and “two-step” assignments, this HOV choice model becomes a key modeling step to help compensate for the removal of the “multi-run” and “two-steps” by matching or improving the currently assigned HOV volumes on HOV facilities. Therefore it is desirable and sufficient to have the new HOV choice model produce HOV volumes on HOV facilities that are approximately the same as or better than the background COG model. It may also be noted that this analysis is based on observed data (counts) for the year 2010, which does not have any HOT-lane facilities in operation and therefore is not modeled with a “multi-run” in the COG model. Hence, this HOV choice model could only be calibrated to compensate for the removal of the “two-step” assignment. Looking at the summaries presented in Table 4-3, Table 4-4, Table 4-5, and Table 4-6, the HOV choice model results in peak HOV volumes closer to the count. The

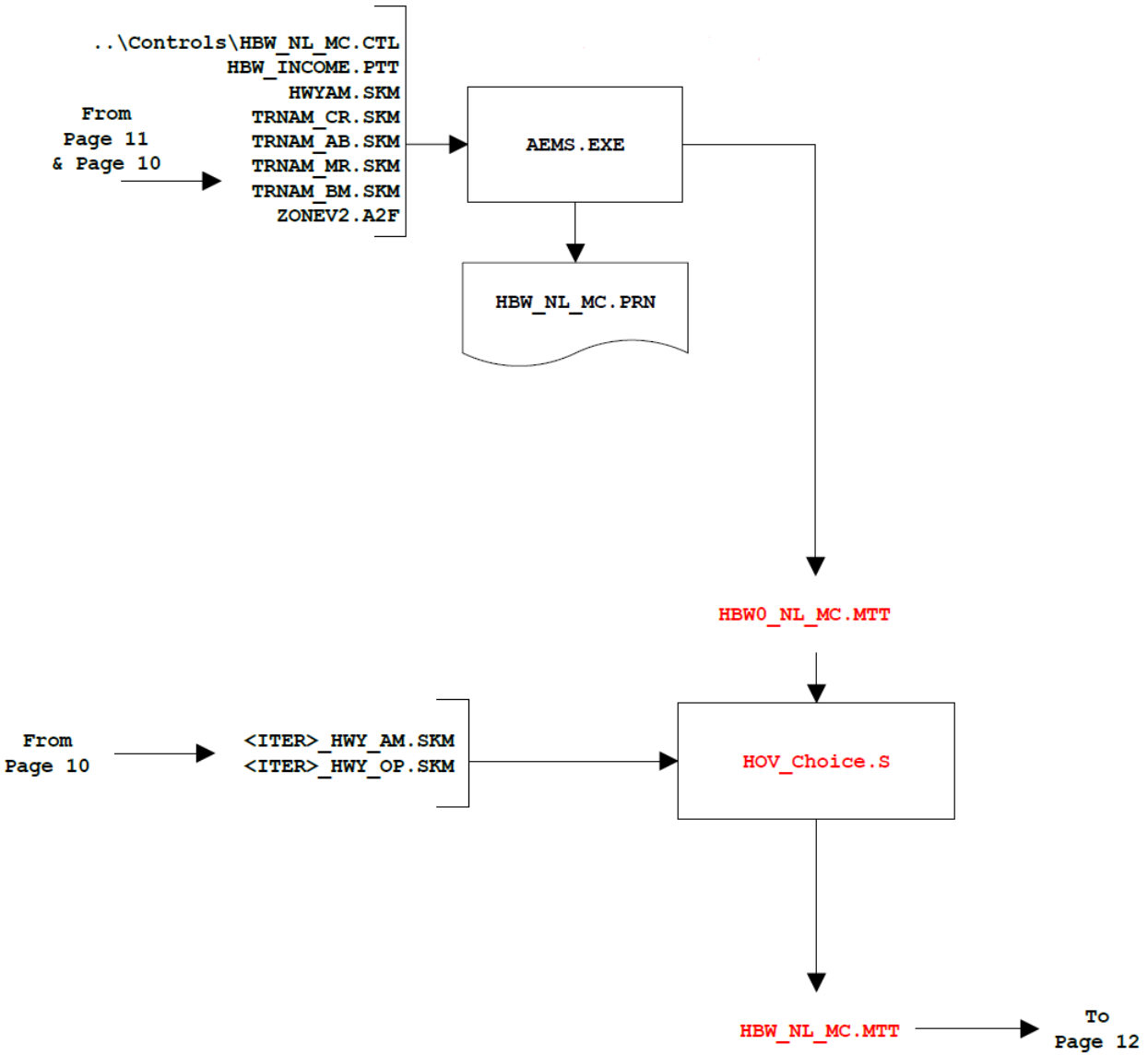
HOV choice model improved PM peak volumes more than AM peak volumes. Not only are the HOV volumes on the facilities closer to counts, but also the volumes on general purpose lanes have been improved by 2 percent and 7 percent in AM and PM peak periods, respectively. Table 4-5 and Table 4-6 indicate that the total volume on HOV facilities is lower than the count. However, it is higher than in the original model (before applying the new HOV choice model) and is closer to the counts. This suggests that a general lack of attraction to an HOV corridor, and not just the HOV mode share, is a reason for low HOV volumes.

In summary, the HOV choice model can be calibrated to achieve desired HOV volumes on the HOV facilities. However, a careful review of the HOV count data must be conducted before applying the HOV choice model.

For application purposes, the HOV choice model and the multi-class assignment procedures were integrated into the current mode choice and assignment procedures. Modified model procedure flow charts are shown in Figure 4-4, which shows the HOV choice process that generates the revised HOV and SOV shares for work trips as an example. Note that the Task Order 12 work was done independently of the Task Order 13 work hence this procedure uses AEMS.exe and not ModeChoice.exe for the general mode choice software.

Figure 4-5 and Figure 4-6 present the changes to the model “wrapper” batch file and the contents of the HOV choice model script, respectively.

Figure 4-4: Proposed HOV Choice Model



Source: Process flowchart: Application of the TPB Version 2.3 Travel Model (Build 48) – Page A-17

Figure 4-5: Changes to the "modelsteps" batch files for HOV choice

```
set _hwy_HOV_lambda1_=5.0
set _hwy_HOV_lambda2_=5.0
set _hwy_HOV_Beta_=0.05
```

Figure 4-6: Contents of HOV choice script (HOV_Choice.s) applied for all trip-purposes

```
;travel time saving thresholds to shift demand among sov, hov2 and hov3
low_ben_sov_hov2    =  0.05 ; lower limit to travel time saving for sov to shift to hov2
sig_ben_sov_hov2    =  1.0  ; limit for significant travel time saving (different lambda) for sov
to shift to hov2
low_ben_sov_hov3    =  0.05 ; lower limit to travel time saving for sov to shift to hov3
```

```
sig_ben_sov_hov3   =  1.0  ; limit for significant travel time saving (different lambda) for sov
to shift to hov3
low_ben_hov2_hov3  =  0.05 ; lower limit to travel time saving for hov2 to shift to hov3
sig ben hov2 hov3  =  0.5  ; limit for significant travel time saving (different lambda) for hov2
to shift to hov3

RUN PGM = MATRIX

    PARAMETERS ZONES=3722

    MATI[1] = %_prev_%_am_sov_MC.skm       ; %_iter_%_am_hov3_hovrestrict.skm
    MATI[2] = %_prev_%_am_hov2_MC.skm      ; %_iter_%_am_hov3_hovrestrict.skm
    MATI[3] = %_prev_%_am_hov3_MC.skm      ; %_iter_%_am_hov3_hovallow.skm

    MATI[4] = %_iter_%_%_purpi_%_NL_MC.MTT

    FILLMW MW[20] = MI.4.4(11)             ; carry forward transit trips

    MATO = %_purpo_%_NL_MC.MTT, DEC=3*D, MO=17,18,19,20-30, NAME = SOV, HOV2, HOV3, WK_CR,
WK_BUS, WK_BUS_MR, WK_MR, PNR_KNR_CR, PNR_BUS, KNR_BUS, PNR_BUS_MR, KNR_BUS_MR, PNR_MR, KNR_MR

    JLOOP

        ; working matrices for sov, hov2 and hov3
        MW[1]  = MI.4.1                    ; background SOV
        MW[2]  = MI.4.2                    ; background HOV2
        MW[3]  = MI.4.3                    ; background HOV3

        MW[4]  = MW[1] + MW[2] + MW[3]     ; background SOV  + HOV2 + HOV3
        MW[5]  = MW[1] + MW[2]             ; background SOV  + HOV2
        MW[6]  = MW[1] + MW[3]             ; background SOV  + HOV3
        MW[7]  = MW[2] + MW[3]             ; background HOV2 + HOV3

        MW[8]  = (MI.1.1 - MI.2.1)         ; HOV2 travel time benefit over SOV       (+ve implies
savings)
        MW[9]  = (MI.1.1 - MI.3.1)         ; HOV3+ travel time benefit over SOV      (+ve implies
savings)
        MW[10] = (MI.2.1 - MI.3.1)         ; HOV3+ travel time benefit over HOV2     (+ve implies
savings)

        ; initialize final matrices
        MW[11] = MW[1]
        MW[12] = MW[2]
        MW[13] = MW[3]

        ;shifting HOV2 to HOV3
        IF ((%_hwy_HOV_lambda1_% = 0) && (%_hwy_HOV_lambda2_% = 0))
            MW[16] = 0
        ELSEIF ((MW[7]>0) && (MW[2]>0) && (MW[10]>@sig_ben_hov2_hov3@))
            MW[13] = min(MW[7], MW[2] * (MW[3] / (% hwy HOV Beta % * MW[2])) *
exp(%_hwy_HOV_lambda1_% * MW[10])/(1+ exp(%_hwy_HOV_lambda1_% * MW[10])))
            MW[16] = MW[13] - MW[3]
        ELSEIF ((MW[7]>0) && (MW[2]>0) && (MW[10]<@sig_ben_hov2_hov3@) && (MW[10] >
@low_ben_hov2_hov3@))
            MW[13] = min(MW[7], MW[2] * (MW[3] / (%_hwy_HOV_Beta_% * MW[2])) *
exp(% hwy HOV lambda2 % * MW[10])/(1+ exp(% hwy HOV lambda2 % * MW[10])))
            MW[16] = MW[13] - MW[3]
        ENDIF

        MW[17] =  MW[1]
        MW[18] =  MW[2] - MW[16]
        MW[19] =  MW[3] + MW[16]

    ENDJLOOP

ENDRUN
```

5 HOT Lane and Toll Modeling (Task Order 12)

As mentioned earlier in the HOV choice chapter, this Task Order 12 was primarily geared towards implementation of the Task Order 8 procedures, hence some of the relevant portions of Task Order 8 are repeated here to provide context.

5.1 Background

The goal of this effort is to integrate the HOT lane modeling procedure developed as part of Task Order 8 into the COG/TPB model version 2.3.52. The settings are configured to perform the previously most-successful alternative (A-1) while retaining all other "knobs" including the option to fully turn-off toll-setting. When toll-setting is not performed, the input tolls are utilized.[20] However, no efforts are made to replicate the results from the COG/TPB model version 2.3.52. The previous effort for HOT lane modeling was most successful in implementing toll-setting when starting from well "seasoned" seed tolls. The seed tolls were prepared using the input tolls in COG/TPB model version 2.3.52. During the integration process, further opportunities were explored to achieve run time savings within traffic assignment steps. The toll groups are also consolidated.

Task 8.1 (HOT-Lane Modeling) identified the sensitivity of the highway traffic assignments to the newly introduced value-of-time (VOT) distributions in toll-choice modeling. The VOT distributions used in task 8.1 were derived from the mean wage in the Washington metropolitan region from the Bureau of Labor Statistics (BLS); however, the VOT distribution was not varied for each of the six vehicle assignment classes. Ideally, VOT distributions should also consider income, time-of-day, purpose and occupancy for use in toll choice modeling. However since income stratification information is not available at the traffic assignment stage, in the interest of time and budget, it is desirable to focus this task on only developing value of time distributions that can be directly applied in the highway assignment step. In other words, VOT distributions will be prepared by the four modeled time-periods (AM, MD, PM and NT) for each of the six vehicle assignment classes. These VOT distributions will need two data points for each time-period and vehicle-class combination: (1) mean wage rate and (2) median wage rate. Lognormal[21] distributions are often used[22] to represent value-of-time distributions. Using the mean and median wage rate values, one can generate a series of associated lognormal curves. This data would need to correspond to the base-year dollars of the model (2007 $).

[20] The input tolls can be found in the toll parameter file (toll_esc.dbf).

[21] A log-normal (or lognormal) distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed. http://en.wikipedia.org/wiki/Log-normal_distribution

[22] Section 5.2.3 and Appendix A.2.4, NCHRP 722 Assessing Highway Tolling and Pricing Options and Impacts, Volume 2: Travel Demand Forecasting Tools, published by the Transportation Research Board (TRB). 2012. http://onlinepubs.trb.org/onlinepubs/nchrp/nchrp_rpt_722v2.pdf

&

Peter Samuel, "Average Value of Time Saved Irrelevant to Toll Express Lanes - Randall Pozdena ECONorthwest," Toll Road News, August 30, 2013, http://tollroadsnews.com/news/average-value-of-time-saved-irrelevant-to-toll-express-lanes---randall-pozdena-econorthwest.

Similarly, task 8.3 (speed validation) suggested improvements to the volume-delay functions based on observed data especially for the freeways. However, since INRIX data primarily consists of only speed information, additional data compilation/collection is required to pair the INRIX speed to traffic volumes. In this regard, it will be important to gather freeway counts that match INRIX speeds to fit speed-flow curves to these data for deriving freeway volume-delay function (VDF). This task was skipped because of the difficulty in obtaining volume data corresponding to the speed data from INRIX.

The remaining sections of the chapter discuss the concept of modeling managed lanes and provide an overview of the HOT lanes procedures followed by details of the modifications.

5.2 HOT Lanes

HOT lanes combine HOV choice with toll lanes whose toll values are actively managed. HOV travelers can use the toll lanes for free or at a reduced price, but other travelers must pay the toll. In Virginia, three or more occupants are needed to qualify for HOV (i.e., HOV3+). If the lanes are actively managed, the toll is dynamically set to ensure that the speed in the toll lanes is maintained at a reasonably high level, typically 50 mph or better. As demand increases, the toll is increased to discourage toll-paying vehicles from using the facility. The toll rate assigned to a given vehicle is based on the time of day and location where the vehicle enters the facility. This means that different vehicles can have different toll rates while traveling on the same link at the same time.

Replicating this level of complexity is well beyond the capabilities of traditional travel demand forecasting models. The best approximation is to estimate the average toll for a given link over the assignment period. For the MWCOG model, assigning trips in four time periods implies that four toll rates will need to be estimated for each link to maintain the desired level of service.

5.3 Dynamically Managed Lanes

Implementing a toll choice model or a multi-class traffic assignment using fixed tolls is relatively straightforward[23]. The toll becomes a component of the generalized cost function using one or more values of time for different vehicle classes. The more difficult challenge, however, is identifying a toll that maintains a given performance level within a dynamically managed lane. For a HOT lane facility, the managed lanes include both tolled and non-tolled traffic (i.e., HOV vehicles). The system manager sets the toll rate for the tolled vehicles based on the speed of the combined traffic volume. Since reducing the amount of tolled traffic often results in an increase in the amount of HOV traffic (due to

[23] Though, according to Boyce (2010), "Consistent forecasts of multiple-class link flows are needed for the evaluation of proposed facilities for which vehicle classes are treated differently, such as toll roads, bridges, HOV lanes and HOT lanes... However, class-specific link flows ... are not uniquely determined by the standard user-equilibrium traffic assignment model deployed in currently available software systems" (p. 1)

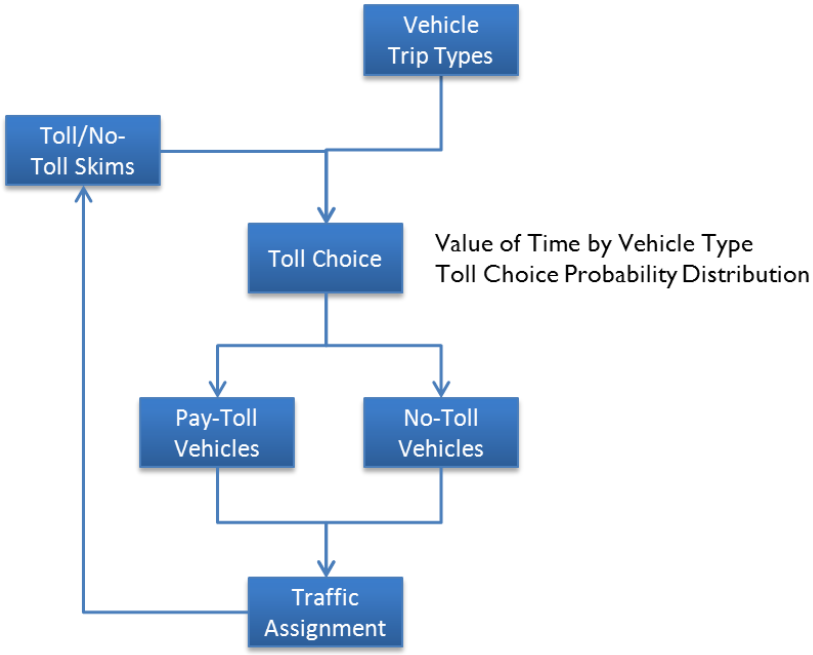
Boyce, David E., Yu (Marco) Nie, Hillel Bar-Gera, Yang Liu, and Yucong Hu. *Field Test of a Method for Finding Consistent Route Flows and Multiple-Class Link Flows in Road Traffic Assignments*. Travel Model Improvement Program (TMIP). Washington, D.C.: U.S. Department of Transportation, Federal Highway Administration, March 8, 2010. http://www.transportation.northwestern.edu/docs/research/Boyce_FieldTestConsistentRouteFlows.pdf.

increased congestion in the general purpose facilities and lanes), setting the toll rate is generally an iterative process of converging to an optimal solution.

Currently MWCOG identifies the toll rate for managed lanes through an off-line process[24] that iterates through a series of traffic assignments with toll adjustments during each iteration. The toll is incrementally adjusted until the volume-to-capacity ratio on each managed link or group of links is between 0.95 and 1.01 (i.e., the minimum speed threshold). The resulting toll rate is then inserted as a fixed toll on each link by time period for use in a full application of the model.

This task implemented one of the several options researched for integrating a similar toll setting process with HOV and toll choice models within the overall traffic assignment step in the standard modeling process – similar to a method described in NCHRP 364 Toll and Demand Estimation[25]. The overall design of the traffic assignment process is outlined in Figure 5-1. The primary challenge is to develop a procedure that identifies a reasonable toll rate in a computationally efficient way.

Figure 5-1: Overall Traffic Assignment Process



The assignment process outlined in Figure 5-1 differs from the current MWCOG model in two primary ways. The first difference is that the toll choice model in the proposed process uses value-of-time distributions instead of a single value of time to split the vehicle trip tables into toll and no-toll classes for input to the multi-class assignment. The second difference is the iterative process needed to identify

[24] Jinchul Park to Files, "HOT Lane Modeling Process of MWCOG/TPB (Draft)," Memorandum, (April 26, 2010).
[25] Described as method numbered 3 – "modeled within the trip assignment component of a model" on page 16 of NCHRP 364 Estimating Toll Road Demand and Revenue, A Synthesis of Highway Practice published by the Transportation Research Board (TRB). 2006. www.onlinepubs.trb.org/onlinepubs/nchrp/nchrp_syn_364.pdf

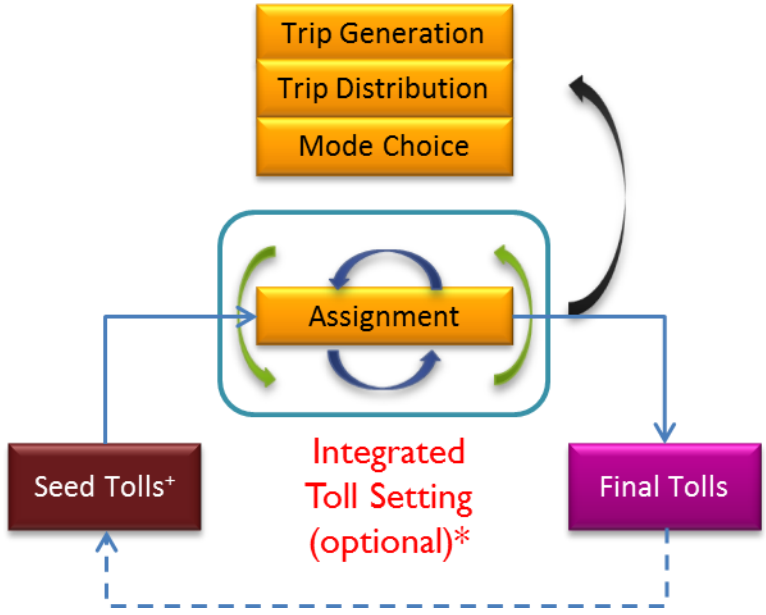
a toll rate that achieves the objective function (e.g., maximize revenue or maximize throughput given a minimum performance standard).

5.4 Overview of Integrated Toll Setting in Traffic Assignment

One of the key changes to the highway assignment process involved adapting and integrating MWCOG's off-line toll setting procedure into the standard assignment procedure. A toll file and a value-of-time distribution file for each time period and vehicle class were added to the inputs to facilitate the integrated toll-choice modeling and highway assignment. Additional parameters were introduced in the model-steps batch file and assignment scripts to allow the user the ability to control and customize toll-setting behavior, including the option to turn it off. These changes are discussed in relevant sections of this chapter.

Figure 5-2 presents an overview of the highway assignment process. Speed-feedback iterations are shown with a black arrow, all-or-nothing convergence assignments with blue arrows and the newly integrated toll-setting process with green arrows.

Figure 5-2: Modified Highway Assignment Process



+ Fixed tolls or outputs from the toll setting process of the previous speed-feedback iteration

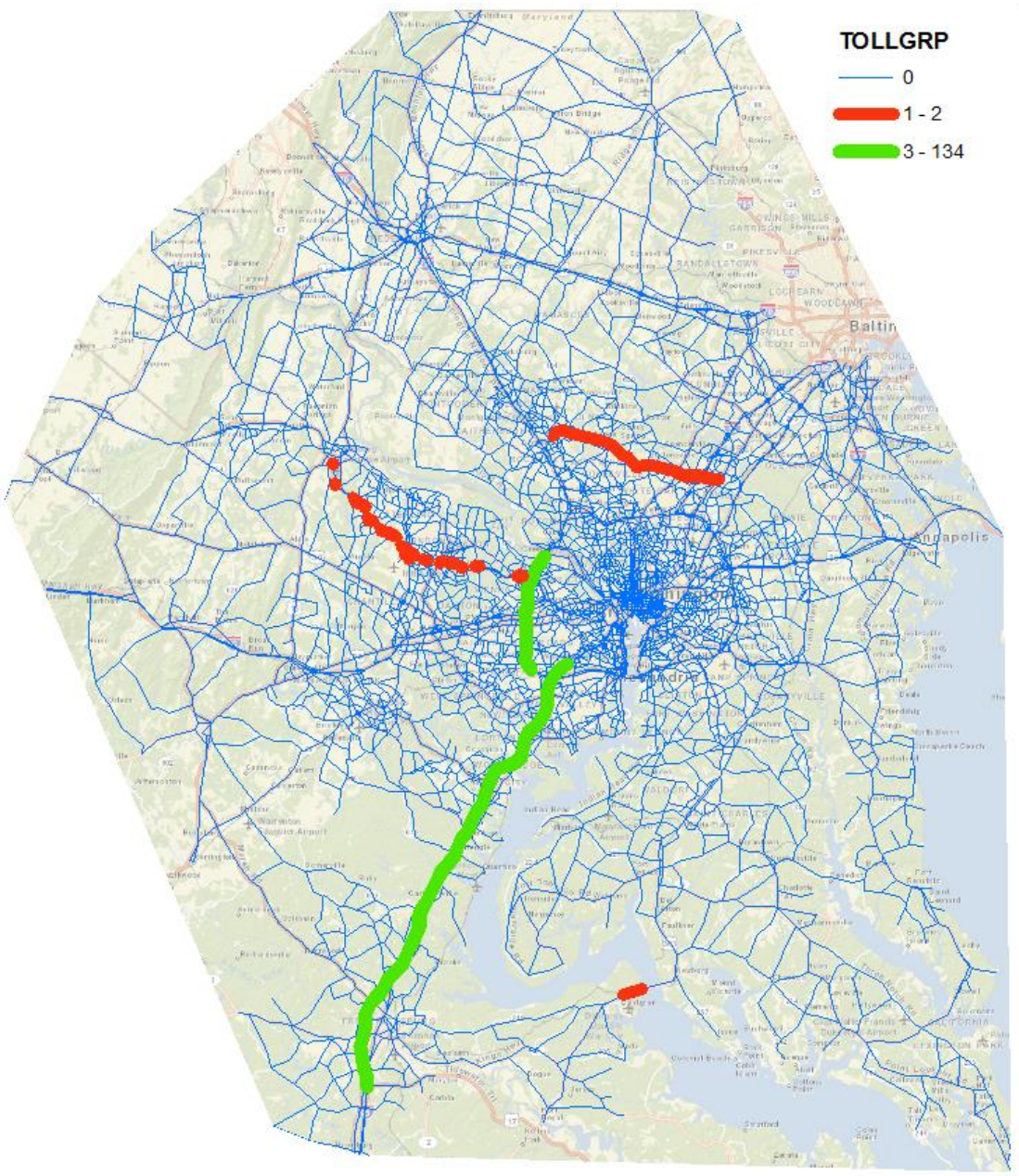
* Two levels of toll setting convergence criteria and search methods

The integrated toll-setting procedure was designed to adjust only the non-fixed tolls in the network; however the toll-choice model considered all tolls in the generalized cost function to fully account for all costs associated with paths. Currently there are 134 toll groups in the model that include all toll links in the network. The toll groups numbered 1 and 2 use static tolls, whereas toll groups numbered 3 and above use dynamic tolls (i.e., HOT lanes). Toll group 1 was designated by MWCOG staff for toll roads

that existed at the time of calibration to 2007 conditions. Toll group 2 was reserved for the Intercounty Connector (ICC).[26]

Figure 5-3 depicts these toll groups in the MWCOG region. Although the toll groups were consolidated from 134 to 91 as described later, the incorporated modeling procedures are compatible with both toll groups, provided all input files are consistent in toll-group coding.

Figure 5-3: Existing Toll Groups in 2020



[26] Milone et al., *User's Guide for the MWCOG/NCRTPB Travel Forecasting Model, Version 2.3, Build 52*, 91.

5.5 Traffic Assignment Process Modifications

Similar to the previous effort, all of the HOT lane assignment procedures were primarily tested using year-2020 data, but on COG/TPB model version 2.3.52 with Round 8.2 Cooperative Land-Use Forecasts instead of the COG/TPB model version 2.3.48 with Round 8.1 Cooperative Land-Use Forecasts. The year 2020 was chosen due to the availability of coded HOT lanes on the Beltway and I-95 in Virginia to enable testing variably priced facilities.

The current MWCOG model utilizes Windows batch files to manage the execution of Cube Voyager-based scripts and other custom programs for a typical model application. All of these batch files are generic except two, a "wrapper" batch file and a "model-steps" batch file, that are year or scenario specific. As part of this research effort, the overall structure of the model execution was retained while streamlining three areas: the wait-process between sub-steps, Cube Cluster management, and the overall highway assignment process. All of these changes were implemented by modifying the model-steps batch file, relevant Voyager scripts, and the associated "helper" batch files where applicable. Details of changes to the batch files and the Voyager scripts are presented later in this chapter.

The following is a summary of the miscellaneous changes made as part of the highway assignment model development effort in order to streamline the model system management and execution:

1. Currently, the Windows "ping" command is used to pause processing for a 10-second period to prevent possible conflicts of certain processes running in parallel. We have changed the "ping" command to the "choice" command, as shown in Figure 5-4, to increase reliability and reduce clutter in the log and batch files. Using the ping command to pause processing is somewhat obscure, whereas using the choice command is more straightforward. For example, in the choice command, one can specify the number of seconds to wait (in this example, "/T:10" causes the choice command to wait 10 seconds), whereas, the ping command depends on network status and hardware performance, and it requires an estimate of number of retries, which generates multiple output lines in the log file.

Figure 5-4: Wait-process changed from PING to CHOICE

```
BEFORE:
@ping -n 11 127.0.0.1

AFTER:
CHOICE /C C /M "Performing a brief wait  ... ; press C to resume immediately" /D:C /T:10
```

2. Created a new directory 'Cygwin' under the 'Software' directory and moved Cygwin binaries ('head.exe', 'tail.exe') and 'TIMETHIS.exe' into this folder. Additionally, four Cygwin[27] DLLs, which are not included with the COG/TPB model version 2.3.52, but are required to execute Cygwin binaries, were identified and placed in this folder (Figure 5-5).

[27] Cygwin is a collection of tools that allow one to execute Linux/UNIX commands in Microsoft Windows. As an alternative to this step, one can also install Cygwin and update the Windows PATH environment variable so that it finds the four DLL files in the Cygwin installation folder.

Figure 5-5: Cygwin DLLs added to Software\Cygwin Folder

3. Some programs, when they finish running, produce a "return code" that indicates whether the program completed successfully or not. In Windows batch files, the return code is called ERRORLEVEL. After a command is run in a Windows batch file, one may check the return code by using the IF ERRORLEVEL command directly after the program whose return code is being checked. In the original helper batch files, the IF ERRORLEVEL command was placed after the Cluster.exe command, which means it would have checked the return code of Cluster.exe. We have moved the IF ERRORLEVEL command so that it comes immediately after the command that runs the Voyager script (see Figure 5-10). In this way, the IF ERRORLEVEL checks the return code of the Cube Voyage script, not the Cube Cluster command that is used to close the Cube Cluster node. The previous highway assignment process was setup to detect an ERRORLEVEL code of 1 and the previous highway assignment script was actually returning an ERRORLEVEL code of 1 (a warning), but due to the misplacement of the detection it was not being trapped. The ERROLEVEL code check was changed to detect 2 (an error) after this change to prevent runs from halting.

4. The use of Cube Cluster was modified in two ways: (1) the cluster ID setup was streamlined to improve its operational efficiency including naming convention changes to make the implementation more user-friendly – users are also provided the option to change the name of the cluster ID (previously hard-coded to 'AMsubnode' and 'MDsubnode') in the wrapper batch file and (2) the Cube Cluster windows were changed to launch in minimized mode using the key "starthide" option, instead of the "start" option. This change minimizes screen clutter during runtime. This change and the changed mentioned above corrected the originally misplaced checks to detect failure of sub-steps in the model while supporting the option to perform simultaneous model runs within the same root folder by providing separate and unique nodes for each scenario run. These changes can also be noticed in Figure 5-10.

5. The assignment steps were modified to move all the functions and link work-variables (LW) to the ADJUST phase of the HIGHWAY program to ensure that all link work-variables are synchronized across the various Cube Cluster subnodes between equilibrium iterations.

6. One of our analysis checks found that some links with zero lanes were resulting in non-zero assigned volumes. Such links were very limited and largely located in Washington D.C. While the existence of links with zero lanes appears to be a network coding issue, the assignment

procedures should have ideally not assigned any volume to such links. This may be attributed to the way volume-delay functions are implemented using lookup tables. If a lookup table returns zero as the travel-time ratio for erroneous inputs instead of large values, the path builder may actually see it as a favorable path option. In order to prevent such assignments, a new pathgroup 32 (highest allowed by Cube) was used to trap and exclude zero-lane links from assignments.

5.5.1 Procedural Changes

The biggest noticeable change is the consolidation of the "base" and "final" folders corresponding to the "multi-run" setup into a single folder, suffixed with "EC", standing for existing plus committed. One option was to simply re-use either the "base" or the "final" folder names, but doing so could potentially cause confusion amongst users or make them look for the other. Hence a new name was chosen to label the folders to prevent any source of confusion. Figure 5-6 shows an example for the year 2020. With this change, the batch file performing the HOV3+ skim substitution - "HSR_Highway_Skims.bat" was no longer required and therefore removed from the model setup. The calls to this batch file from the model-steps batch files were also removed. Another corresponding change was the removal of the environment variable "_HOV3PATH_" as it was no longer necessary, given the consolidation of the "base" and the "final" folders.

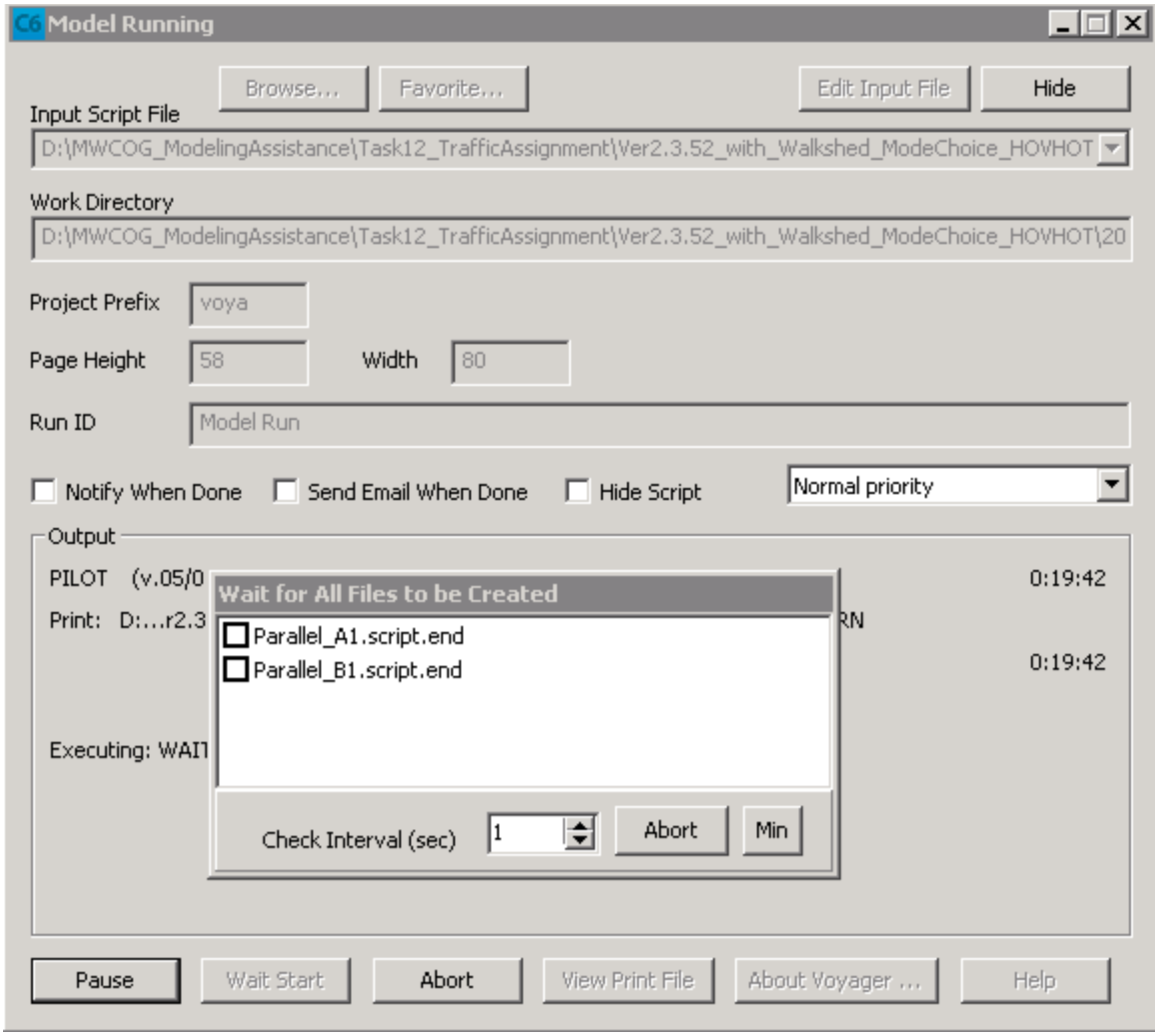
Figure 5-6: Consolidation of "base" and "Final" folders

As part of changes to the Cube Cluster management, the sub-nodes were re-organized to streamline the cluster management and also make it more user-friendly. Currently, the use of 'AMsubnode' and 'MDsubnode' names as MDP processIDs can cause confusion for the user since these names manage four time-periods, namely, AM, PM, MD and NT amongst themselves. To eliminate the source of this confusion, in the wrapper batch file, the 'AMsubnode' and 'MDsubnode' environment variables were renamed to '**DP_A_ProcessList**' and '**DP_B_ProcessList**' environment variables. In addition, two corresponding new environment variables were added – '**DP_A_ProcessID**' (="Parallel_A") and '**DP_B_ProcessID**' (= "Parallel_B") to use as Cube Cluster IDs. These cluster IDs (for e.g.: "Parallel_A") can be changed by the user, if so desired, so long as they are not identical and do not include spaces, without impacting the Cube Cluster operations. The term DP (distributed processing), ProcessID and ProcessList are directly related to their functions and descriptions in the Cube Voyager documentation. Hence, the user will find it easier to understand their implementation in the updated COG/TPB model version 2.3.52.

Another change in the Cube Cluster setup was the way cluster nodes were being used. Currently the main Cube Voyager script controlling the MDP/IDP execution is also given a task along with its worker nodes. This can cause some confusion in the way Cube Cluster is set up in the COG/TPB model version 2.3.52. For example, on an 8-core machine, the 'AMsubnode' is typically set to "1-4" whereas the

"MDsubnode" is typically set to "2-4" instead of "1-4". In this updated COG/TPB model version 2.3.52, the "DP_A_ProcessList" and "DP_B_ProcessList" are both set identically to "1-4" (or "1-5", "1-6" etc., depending on the number of cores/nodes the user intends to use), so that the main Cube Voyager script simply issues parallel processing tasks to its worker nodes and waits for them to finish. This setup is easier to maintain and was found to marginally improve CPU utilization during testing. The resulting typical Cube Voyager window with such a setup is shown in Figure 5-7.

Figure 5-7: Modified Cube Cluster Setup - Main Script Waits for Worker Nodes



The 'subnode' environment variable was retained as is because it was not used in the highway assignment process. Additionally, several parameters controlling the configuration of toll setting in the highway assignment model were introduced as environment variables in the model-steps batch file. These changes are shown in Figure 5-8 and Figure 5-9. Figure 5-10 shows the changes to the Cube Cluster setup.

Figure 5-8: Changes to the "wrapper" batch files

```
:: Removed :: set AMsubnode=1-4
:: Removed :: set MDsubnode=2-4

set DP A ProcessID=Parallel A
set DP_A_ProcessList=1-4

set DP B ProcessID=Parallel B
set DP_B_ProcessList=1-4
```

Figure 5-9: Changes to the "modelsteps" batch files for HOT lane modeling

```
set _hwy_HOT_perform_toll_setting_=0
set _hwy_HOT_lower_relgap_for_toll_setting_=0
set _hwy_HOT_use_last_tolls_=0
set  hwy HOT use vot distrib in toll setting = 1
```

The four parameters listed in Figure 5-9 are described later in this section.

Figure 5-10: Contents of Batch file: Highway_Assignment_Parallel.bat

```
CD %1

REM  Highway Assignment

if exist voya*.*  del voya*.*
if exist % iter % Highway Assignment.rpt   del % iter % Highway Assignment.rpt

if exist %DP_A_ProcessID%*.script* del %DP_A_ProcessID%*.script*
if exist %DP B ProcessID%*.script* del %DP B ProcessID%*.script*

Cluster.exe %DP A ProcessID% %DP A ProcessList% starthide exit
Cluster.exe %DP_B_ProcessID% %DP_B_ProcessList% starthide exit

start /w Voyager.exe ..\scripts\Highway_Assignment_Parallel.s  /start -Pvoya -S..\%1
if errorlevel 2 goto error

copy Voya*.prn       % iter % Highway Assignment.rpt /y
goto end

:error
REM  Processing Error....
PAUSE

:end
Cluster.exe %DP_A_ProcessID% %DP_A_ProcessList% close exit
Cluster.exe %DP_B_ProcessID% %DP_B_ProcessList% close exit
CD..
```

The next major change is to the highway assignment script, as might be expected, due to the integration of the toll choice model and the toll setting procedures. The key change to the script was “off-shoring” of portions of the code to external files in order to make it easier to maintain, debug and follow the top-level logic of the assignment script. This enables the functionality to be logically grouped, to be selectively re-used, and to have a non-redundant consistent code across all time-periods, and thereby help in reducing errors and help in maintaining “clean” code. The final assignment script was thus

broken down into one main script and eight sub-scripts stored in external files as shown in the following list:

- Highway_Assignment_Parallel.bat
    - Highway_Assignment_Parallel.s
        - Highway_Assignment_Parallel_part1_Main.s
        - Highway_Assignment_Parallel_part2_Initialize.s
        - Highway_Assignment_Parallel_part3_SummarizeTolls.s
        - Highway_Assignment_Parallel_part4_ApplyVOTSplitTollNonToll.s
        - Highway_Assignment_Parallel_part5_BuildPaths.s
        - Highway_Assignment_Parallel_part6_CalculateRestrainedFinalVolSpdVC.s
        - Highway_Assignment_Parallel_part7_SummarizeAndAdjustTolls.s
        - Highway_Assignment_Parallel_part8_TollEvalTerminationCheck.s

These nine scripts are presented in the Appendix in Figure 9-5, Figure 9-6, Figure 9-7, Figure 9-8 ,Figure 9-9, Figure 9-10, Figure 9-11, Figure 9-12, and Figure 9-13. The main script (Highway_Assignment_Parallel.s) makes multiple calls to the "part" code blocks using Cube Voyager's "READ FILE" commands. When execution begins, Cube Voyager uses these calls to pull together various "part" code blocks to create a larger script(s) in preparation for execution.

Although having nine scripts for highway assignment may seem more complex than having just one, the total number of lines of code has been reduced even after inclusion of toll-setting capability. Prior to dividing the highway assignment into code blocks, there were 2,006 lines of code in the single Highway_Assignment_Parallel.s script. After the division into nine files, including the shortened Highway_Assignment_Parallel.s script, the total number of lines of code was reduced to 1,834. Without such a division of scripts, following the approach of duplicating code for each of the four time-periods, it is estimated that a single highway assignment script would have contained approximately 6,500 lines of code (298 + 4 x 1,536) – something that would be very difficult to maintain. The number of lines of code in each of the scripts is shown in Table 5-1.

Table 5-1: Highway Assignment Scripts: Number of Lines

Highway Assignment Code Block	Number of Lines
Highway_Assignment_Parallel.s (Ver. 2.3.52, before integrated toll-setting)	**2,006**
Highway_Assignment_Parallel.s	298
Highway_Assignment_Parallel_part1_Main.s	369
Highway_Assignment_Parallel_part2_Initialize.s	48
Highway_Assignment_Parallel_part3_SummarizeTolls.s	133
Highway_Assignment_Parallel_part4_ApplyVOTSplitTollNonToll.s	354
Highway_Assignment_Parallel_part5_BuildPaths.s	284
Highway_Assignment_Parallel_part6_CalculateRestrainedFinalVolSpdVC.s	87

Highway_Assignment_Parallel_part7_SummarizeAndAdjustTolls.s	200
Highway_Assignment_Parallel_part8_TollEvalTerminationCheck.s	61
Subtotal of Parts 1 to 8	1,536
Total (after integrated toll-setting)	**1,834**

The final integrated assignment procedure starts with populating the network with the seed tolls, the latest tolls from the last toll-setting iteration or the latest tolls from the last speed feedback iteration, depending on the settings in the model-steps batch file. Seed tolls are stored in a series of text files in the "inputs" folder with names beginning with "SEED_TOLLS". The next step writes out the recently encoded tolls for testing purposes. The toll/no-toll skims are prepared next and the trips are split into toll and no-toll trip tables using a toll-choice distribution curve. This split is performed only once per toll-setting iteration so as not to affect the highway assignment convergence. Next, two single multi-class assignments are performed using these trip tables. Following the traffic assignment, the restrained speeds are computed and adjusted on the network. After this stage, the loaded network is read to adjust the tolls for each toll group and the adjusted tolls are evaluated against the old tolls and convergence criteria to determine if further iterations are required.

The traffic assignment type remains unchanged as Bi-conjugate Frank-Wolfe and uses the progressively stringent convergence (relative-gap) criteria as coded in the COG/TPB model version 2.3.52 ranging from a threshold relative-gap/maximum-iterations of 0.01/1000 for the pump-prime iteration to 0.0001/1000 for the final 'i4' iteration.

The following is a summary of the controls/variables that the user can specify. Each of the five variables can take on a value of either "0" (off/disabled) or "1" (on/enabled).

1. DO_TOLL_SETTING = '%_hwy_HOT_perform_toll_setting_%'

   This parameter is the master parameter controlling the toll-setting process and is specified in the model-steps batch file. It is referenced in three scripts: Highway_Assignment_Parallel.s, Highway_Assignment_Parallel_part1_Main.s, and Highway_Assignment_Parallel_part4_ApplyVOTSplitTollNonToll.s. It allows the user to turn on or entirely off the toll-setting procedure. The parameter can be set to a value of "0" (toll setting off) or "1" (toll setting on). When turned off, the input seed toll files are completely skipped and the highway assignments are carried on using the input toll parameter file (toll_esc.dbf) as coded in the zonehwy.net file. However, "check" toll files are created for inspection or potential use as seed toll files.

2. LOWER_RELGAP_FOR_TOLL_SETTING = '%_hwy_HOT_lower_relgap_for_toll_setting_%'

   This parameter is specified in the model-steps batch file. When enabled (set to '1'), it allows the user to optionally lower the threshold for assignment convergence during toll-setting iterations. When enabled, the toll-setting is performed with a lowered relative gap threshold (currently hard coded in the Highway_Assignment_Parallel.s script under the variable 'low_rel_gap' to

0.1), after which one final assignment with the standard convergence threshold is performed using the final tolls. This functionality can be disabled by setting this parameter to '0'.

3. USE_LAST_TOLLS = '%_hwy_HOT_use_last_tolls_%'

   This parameter is specified in the model-steps batch file and, when enabled (set to '1'), it allows for using "evolving" tolls by using the latest tolls from the last speed-feedback iteration as the starting tolls in the current speed-feedback iteration. This functionality can be disabled by setting this parameter to '0'.

4. APPLY_VOT_TO_SPLIT_VTT = '%_hwy_use_vot_distrib_in_toll_setting_%'

   VTT stands for vehicle trip table. This parameter is also specified in the model-steps batch file and allows the user to optionally turn off (when set to '0') the process of splitting trips into toll and no-toll trip tables. When turned off, all trips have access to all facilities including toll facilities. In other words, the highway OD trip table is not bifurcated into toll-users and non-toll users. Note that in such a bifurcation, the non-toll users do not have access to the tolled-facilities whereas the toll-users have access to all facilities. When this bifurcation (toll-choice) is not made, all users have access to all facilities including toll facilities. This parameter should always remain on (set to '1') in normal toll-setting applications and should only be turned off (set to '0') when one wants to see the impact of using the continuous value of time distributions.

5. ALLOW_EXCLUSIVE_TOLL_REDUCTION_LOOPS = 0

   This parameter is specified only in the main highway assignment script. It is recommended that this parameter be set to zero for normal applications. Under normal application (set to zero), the toll adjustment functions adjust the tolls both up and down based on the relationship between the current V/C ratios and the threshold V/C ratios. However, when none of the toll groups increase their toll and either all or some of them only lower their toll, the toll-search is considered done when this parameter is set to zero. When it is set to '1', the toll-search will always continue even if the tolls are only decreased. Therefore this key has a major impact on processing times and should only be considered for enhanced toll-search criteria. Note that the criteria for maintaining speeds on HOT lanes assures only a minimum speed corresponding to an upper threshold V/C ratio. Activating this option will cause the algorithm to try to maximize the number of vehicles that will use the toll facility during a given time period. If/when enabled, this key would make the toll-search process look for a solution in which all toll-groups fall within the threshold V/C ratio range of 0.95 and 1.01 instead of a solution where none of the toll groups exceed the maximum threshold V/C ratio of 1.01. Note that the former solution may not even exist or its search could easily exhaust the maximum number of toll-setting iterations (currently set to 99), and expend a great deal of processing time meanwhile.

Following are some configuration of these parameters for different types of runs:

To make a run with toll setting disabled (note if the first parameter is zero, values for other parameters do not matter):

```
set _hwy_HOT_perform_toll_setting_=0
set _hwy_HOT_lower_relgap_for_toll_setting_=0
set _hwy_HOT_use_last_tolls_=0
set _hwy_HOT_use_vot_distrib_in_toll_setting_=0
```

To make a run with toll setting enabled, with lowered relative gap and using last tolls (this configuration was found to take the least processing time in the test performed by COG/TPB staff[28]):

```
set _hwy_HOT_perform_toll_setting_=1
set _hwy_HOT_lower_relgap_for_toll_setting_=1
set _hwy_HOT_use_last_tolls_=1
set _hwy_HOT_use_vot_distrib_in_toll_setting_=1
```

To make a run with toll setting enabled, with full relative gap:

```
set _hwy_HOT_perform_toll_setting_=1
set _hwy_HOT_lower_relgap_for_toll_setting_=0
set _hwy_HOT_use_last_tolls_=0
set _hwy_HOT_use_vot_distrib_in_toll_setting_=1
```

Note that the integrated toll-setting, when enabled, is currently performed in all of the speed-feedback iterations as opposed to COG's current offline process where toll-setting is performed on the results of the final (i4) speed-feedback iteration. However, it may be noted that the new design allows the flexibility to selectively perform toll-setting for different speed feedbacks iterations.

To disable toll-setting for pp, i1, i2, i3 and enable it only for i4 (other parameters can potentially also be configured differently between speed-feedback iterations):

```
set _hwy_HOT_perform_toll_setting_=0
set _hwy_HOT_lower_relgap_for_toll_setting_=1
set _hwy_HOT_use_last_tolls_=1
set _hwy_HOT_use_vot_distrib_in_toll_setting_=1
.
.
.

set _iter_=pp
set _prev_=pp
set _relGap_=0.01
.
.
.
set _iter_=i3
set _prev_=i2
set _relGap_=0.001

.
.
.

set _hwy_HOT_perform_toll_setting_=1
```

[28] Mark Moran (COG/TPB) to David Roden and Krishna Patnam (AECOM), *Comments on your draft report chapter for Task Order 12 (Traffic assignment improvements related to modeling HOV facilities and HOT-lane facilities) and associated modeling files*, June 19, 2014

```
set _iter_=i4
set _prev_=i3
set _relGap_=0.0001
```

Unlike the previous implementation in Task Order 8, the Cube Cluster setup to use both MDP and IDP was retained, as it is done in the COG/TPB model version 2.3.52. In Task Order 8, the MDP setup was removed and only IDP was used since most applications are done on a single physical machine, albeit with multiple cores. However, since it is known that beyond a threshold point further parallelization has diminishing returns (i.e., the benefits of parallelization do not scale infinitely), it is better to run two MDP sessions with half the number of IDP cores each (for example, four) than to run a single IDP session with twice the number of cores (for example, eight). This is so because, through initial testing with the COG/TPB model version 2.3.52 on an eight-core machine (quad-core processor with hyper-threading enabled), it was found that the AM and PM assignments can be completed somewhat quicker when run in parallel using MDP and with 4 IDP nodes each, than when run in sequence with 8 IDP nodes each time.

The overview of the highway assignment procedures can be seen from a series of flowcharts as shown in Figure 5-11. Figure 5-12, Figure 5-13 and Figure 5-14. These flowcharts successively “drill-down” into the toll-setting integrated highway assignment process. Figure 5-11 shows the flow of execution starting from the model-steps batch file including Cube Cluster setup, Figure 5-12 shows the contents of the top-level highway assignment script, and its sub-parts, including toll-setting are shown in Figure 5-13 and Figure 5-14.

Figure 5-11: Overall Highway Assignment Batch Process

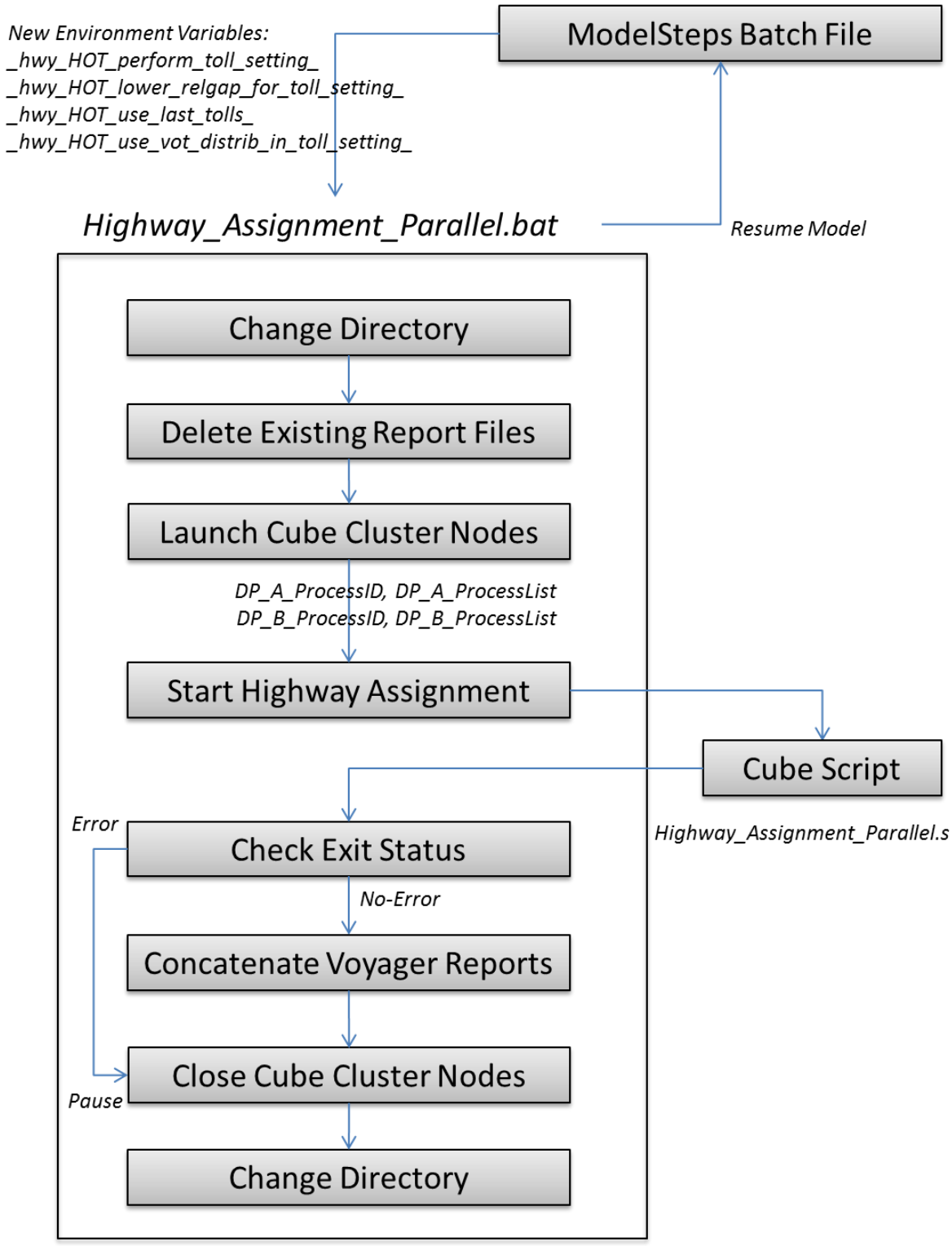


Figure 5-12: Highway Assignment Parallel Processing

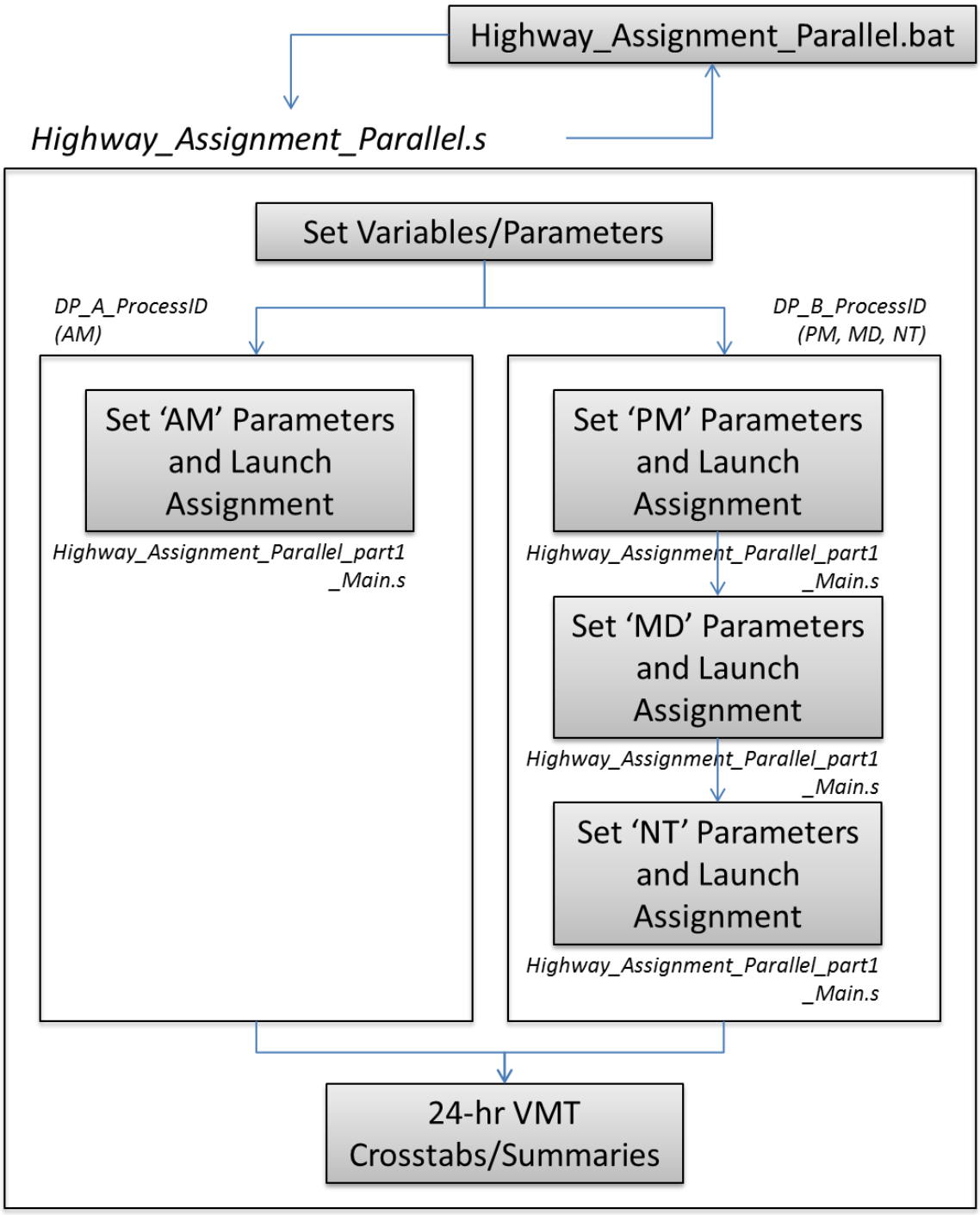


Figure 5-13: Toll Processing Option Controls

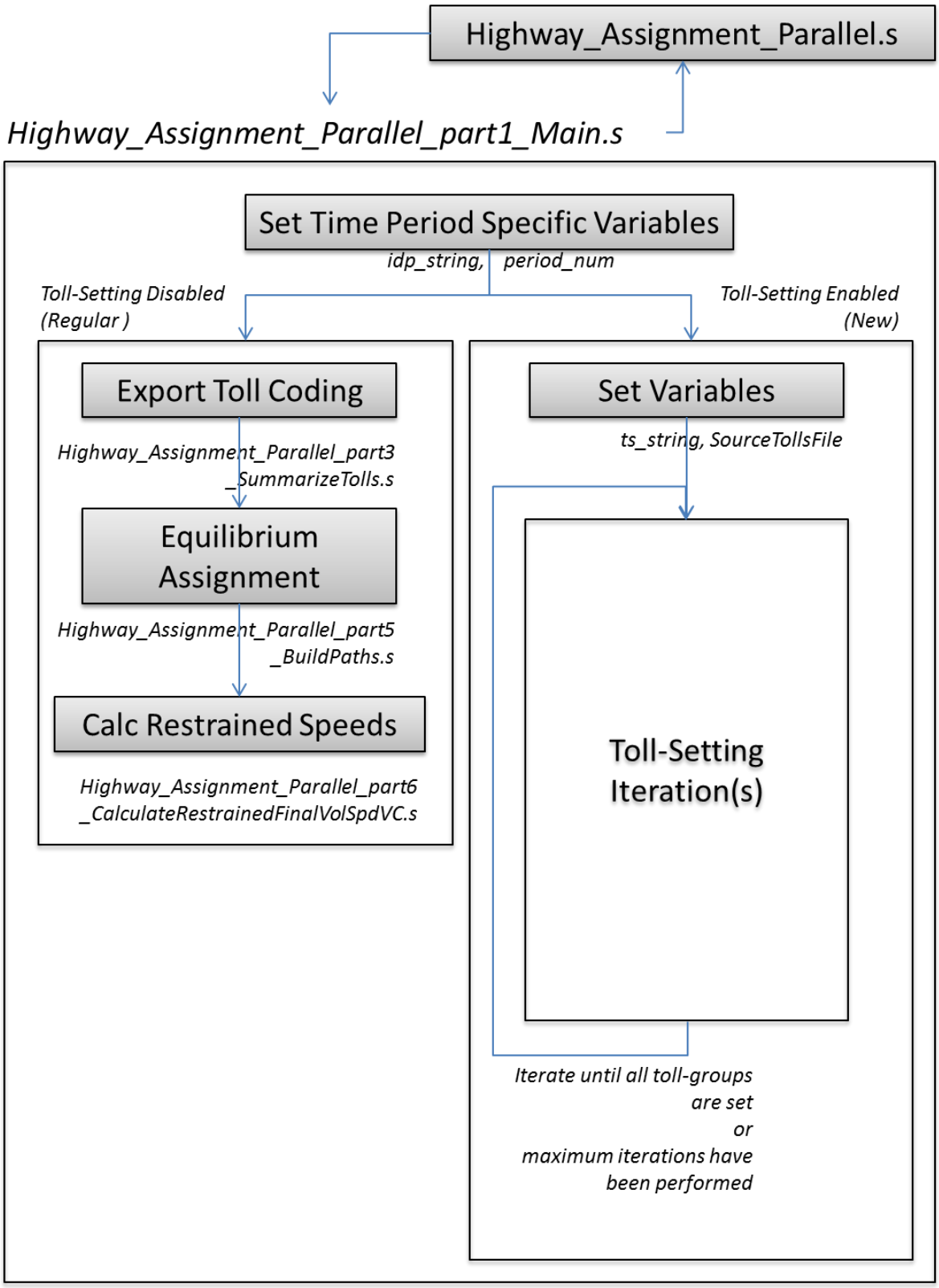
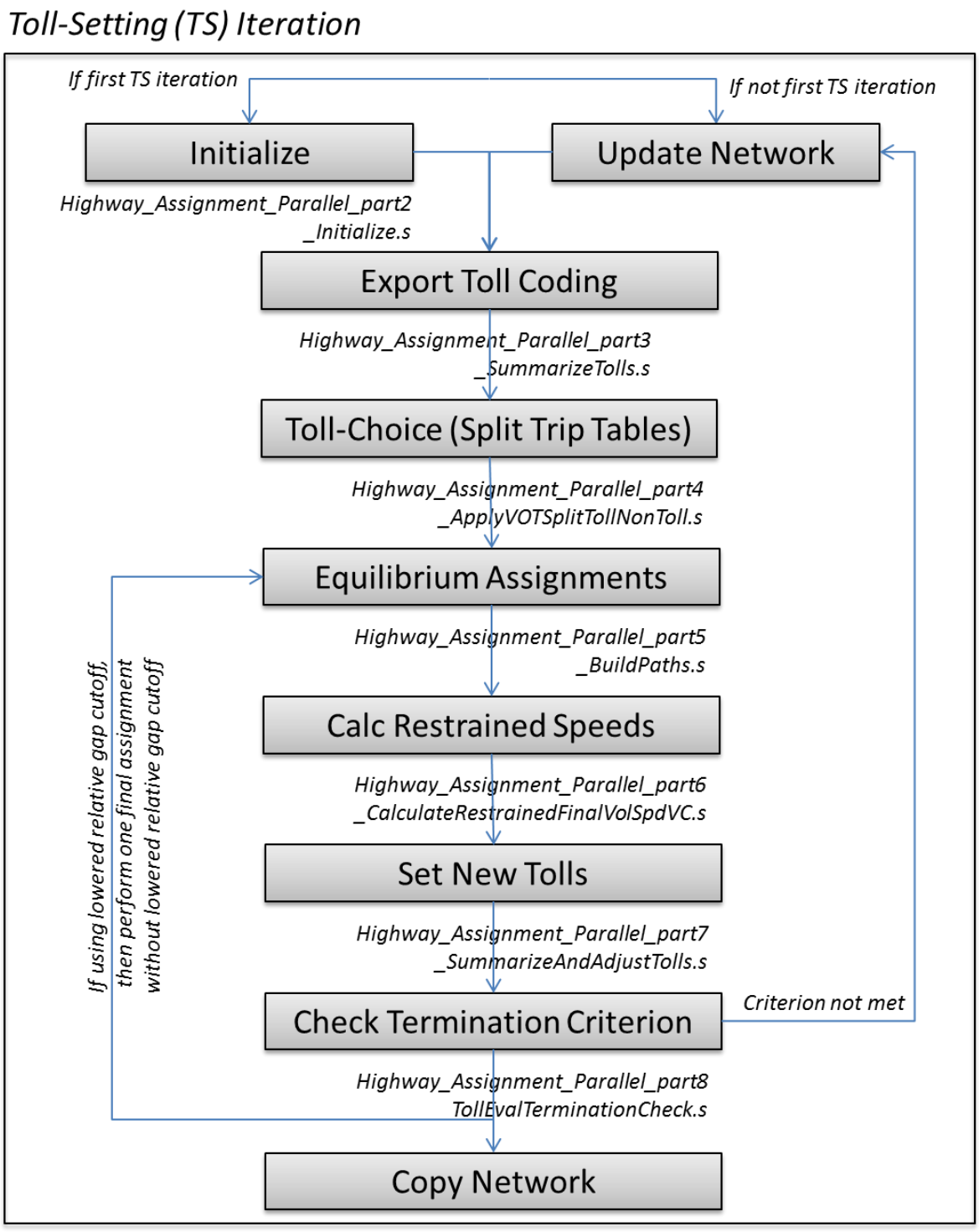


Figure 5-14: Toll-Setting Iteration Process



5.5.2 Toll Rate File

The existing MWCOG model uses a single input "Toll_Esc.dbf" file[29] to store the tolls for both the fixed- and variable-toll facilities. This file includes toll rates specified in cents/mile and factors for adjusting these toll rates for each of the four time periods (AM, MD, PM and NT) by toll group. These toll rates are then posted on individual links, based on its coded toll group, during the highway network preparation step, as a toll cost per link in cents. While this file structure provides convenient access to the tolls across all four time periods, it is not convenient for parallel processing of time periods (e.g., if a file contains four sets of toll values, updating two or more sets simultaneously could cause sharing violations in the operating system or other complications). Hence a new toll-rate file, based on the structure previously in use in COG's original offline toll setting procedure, was used to store and update toll-rates during integrated toll setting. Only when toll-setting is enabled, during the highway assignment process, are the tolls from this file used to encode the "TOLL" and "TOLL_VP" fields on the links and replacing their previous values. The structure of this file is shown in Table 5-2. This file stores tolls as toll rates in cents/mile for only variably prices facilities (toll group ≥ 3), and has place holders for two sets of tolls, old and new, to aid in evaluating toll change. The last field in this file contains a flag that indicates satisfying the toll-termination criteria (satisfied when set to '1' and un-satisfied when set to '0'). Note that in this file, the 5th column, NEW_TOLL, contains the updated tolls.

Table 5-2: Toll Rate File Structure

TOLL_GROUP	NUM_LINKS	AVG_FTYPE	OLD_TOLL	NEW_TOLL	AVG_SPD	AVG_VC	VMT	DIFF_VC	FLAG
3	0	0	0	0	0	0	0	-1.01	1
4	0	0	0	0	0	0	0	-1.01	1
5	0	0	0	0	0	0	0	-1.01	1
6	1	1	224.5356	224.5356	31.7646	1.002	1922.35	-0.008	1
7	1	1	20	20	45.9461	0.9179	1496.83	-0.0921	1
8	3	1	224.5356	224.5356	31.7646	1.002	4950.05	-0.008	1
9	1	1	20	20	59.7217	0.7114	5732.11	-0.2986	1
10	1	1	20	20	62.7909	0.501	3364.11	-0.509	1
11	8	1	20	20	53.9948	0.8442	24049.84	-0.1658	1

The following is a description of various toll files that are input or output during the toll-setting assignment process. They all have the identical structure described earlier.

<TOD> (Time of day) = AM/PM/MD/NT
<itr> (Speed feedback iteration) = pp/i1/i2/i3/i4
<tsitr> (Toll-setting iteration) = 1/2/3/4/5… etc.

1. 'Inputs\SEED_TOLLS_<TOD>.TXT'

   For example: 'inputs\SEED_TOLLS_AM.TXT'

[29] See page 2-3 of *User's Guide for the TPB Travel Forecasting Model, Version 2.3, Build 38, on the 3,722-Zone Area System*. Final Report. Washington, D.C.: National Capital Region Transportation Planning Board, January 20, 2012.

These four files correspond to the four time-of-day periods and are included in the 'inputs' directory for every scenario and are the first set of tolls that are evaluated. The values used for the seed tolls are critical to the computational efficiency of a model run, due to their impact on the number of toll setting iterations required to reach convergence.

2. '<scenario>\LATEST_TOLLS_<TOD>.TXT'

   For example: '2020_EC \LATEST_TOLLS_AM.TXT'

   Similar to the four seed toll files, these files are created by the assignment script at the end of every toll setting iteration for potential use in the next toll iteration or speed feedback iteration depending on the user-configuration.

3. '<scenario>\<itr>_NEW_<tsiter>_ TOLLS _<TOD>.TXT'

   For example: '2020_EC\i4_NEW_24_TOLLS_AM.TXT'

   These are intermediate toll files that are newly determined and have not been evaluated yet.

4. '<scenario>\<itr>_CHECK_<tsiter>_TOLLS_<TOD>.TXT'

   For example: '2020_EC\i4_CHECK_12_TOLLS_AM.TXT'

   These are also intermediate toll-rate files created for the purpose of checking what tolls have been coded on the network. These "check" toll-rate files from a current toll setting iteration should contain identical toll rates (in the 5th column of the toll file) to the "new" toll files from a previous toll setting iteration.

It is worth noting that any of these toll-rate files can also double up as a seed toll-rate file when named appropriately and placed in the inputs directory. This design consideration extends some convenience to the user in terms of creating seed toll-rate files.

It may also be noted that the "Toll_Esc.dbf" stores toll-rates in current-year dollars or inflated dollars. By contrast, all tolls (rates) in the new toll-rate files are saved in deflated dollars to avoid the process of deflating them when reading from these files and inflating them while writing to them. This simplification helps avoid errors when updating assignment scripts and keeps the generalized cost computation consistent with the rest of the model. Another advantage of using deflated dollars is that these seed/intermediate/latest toll-rate files become more portable, i.e., they can potentially be carried across years/scenarios, if so desired, without needing to factor them first.

The tolls as coded in the COG/TPB model version 2.3.52 were used to create seed toll-rate files for the consolidated toll groups in such a manner as to effectively result in the same toll on their constituent links. The toll-group consolidation is discussed in the next section. These seed toll-rate files are all in deflated dollars and are presented in the Appendix in Figure 9-14 for AM, in Figure 9-15 for PM and in Figure 9-16 for off-peak (MD and NT). The AM seed toll-rates range from 18.0574 cents/mile (default/minimum toll of 20 cents/mile in current year dollars) to over $4.78/mile (corresponding to toll

group 76) whereas the PM seed tolls range from 18.0574 cents/mile (default/minimum of 20 cents/mile in current year dollars) to over $3.8/mile (corresponding to toll group 82). The MD and NT seed tolls are set to a flat value of 13.543 cents/mile (corresponding to 15 cents/mile in current year dollars). In both the peak period seed tolls, only 15 toll groups have tolls higher than the default/minimum toll rate of 20 cents/mile. Also, some toll groups appear bundled together sharing common toll-rates, for example, toll groups 15, 50 and 73.

5.5.3 Toll Choice and Value of Time Distributions

There are a number of different ways tolls can be modeled within a regional modeling process. Pricing considerations are sometimes implemented through a toll choice model or as a sub-mode within a nested logit mode choice model. In this case, the choice of paying a toll or not paying a toll is based on the attributes of the traveler, the trip purpose and the generalized cost of the two alternatives. Such models can capture traveler behavior effectively, but they are difficult to estimate and calibrate because observed toll choice data at the household level is not generally available.

The overall assumption of a toll choice model, however, is that the decision or choice is made for the whole trip. The model compares a minimum impedance path that includes toll options to a minimum impedance path that does not. The probability of choosing one path over the other is determined by the toll choice model. As with the HOV choice, it is best to limit the estimation and application of a toll choice model to trips that have at least two viable alternatives.

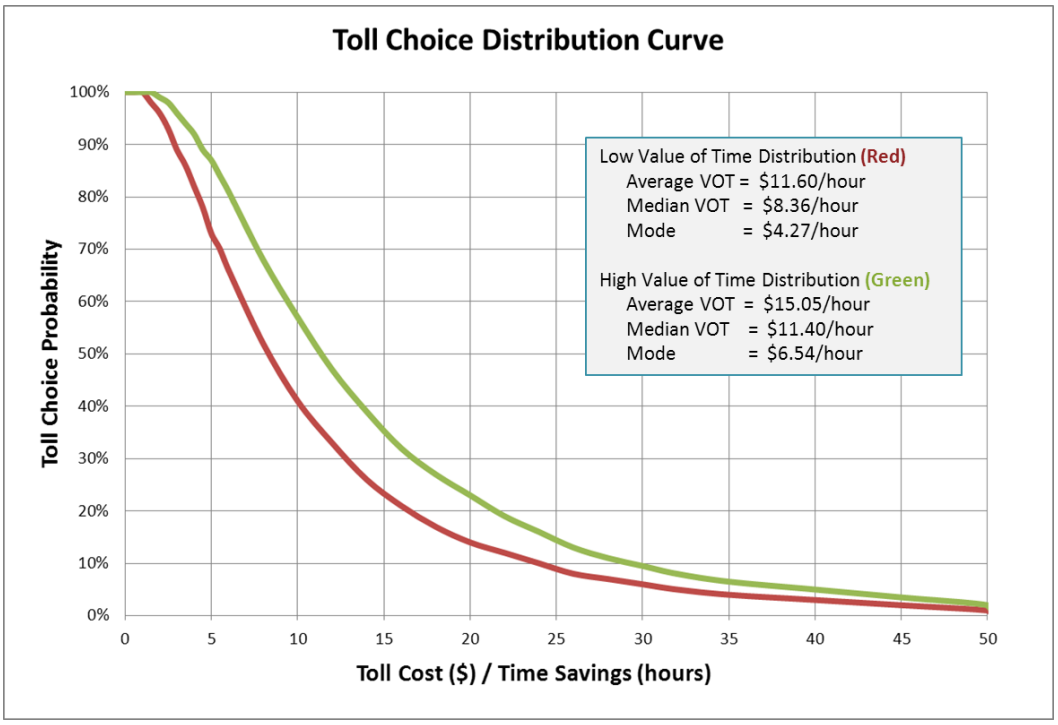
Tolls may also be modeled using a path builder or assignment model. In this case, the decision to pay a toll is based on the impact of the toll on the generalized cost function of the minimum impedance path. This becomes a simple trade-off between travel time and cost, based on the value of time. In other words, a single path is built that minimizes the weighted combination of travel time and cost for each trip interchange and all trips on that interchange use that path. In this approach, the toll is buried in the link impedance and is typically unaffected by the attributes of the traveler or the trip purpose.

In reality, different travelers have significantly different values of time for different types of trips. In addition, the value of time for a given trip is not a simple function of the traveler's income. A low income person may have a high value of time for their trip to work or their trip to pick-up their child at the day care center because there are significant costs or other impacts associated with being late. This means that the decision to pay a toll is not an all-or-nothing choice based on the minimum impedance between two points, but a choice probability based on a value-of-time distribution.

A reasonable compromise between the all-or-nothing nature of a path-based toll choice approach and the complexity of a toll nest within a mode choice model is to apply a probability function based on a value-of-time distribution to the toll/no-toll path options. The travel time savings provided by the toll-based path are used to determine the probability that trips on that interchange will choose the toll or no-toll option. The trips are split accordingly and assigned to the corresponding path. This makes the assignment approach sensitive to the input value-of-time distribution and distributes the trips between toll and no-toll options more realistically.

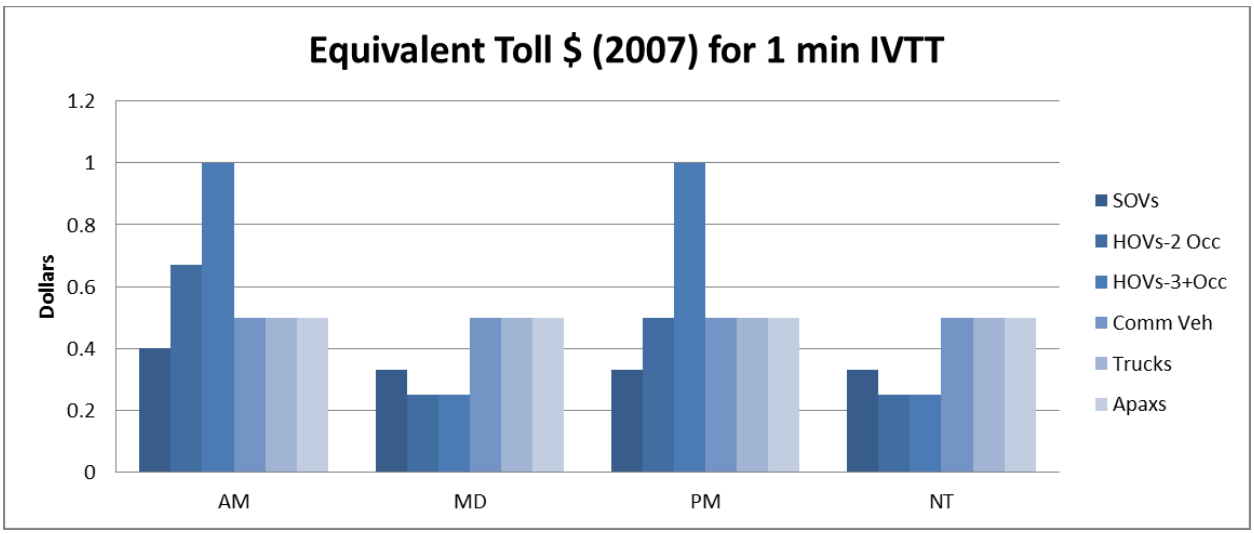
Figure 5-15 shows an example of a toll-choice probability function. This example shows a high and low value-of-time distribution. The low distribution (red) was derived from a small stated-preference survey about people's willingness to pay tolls, and the high distribution was estimated using a regional income distribution. In this example, if the toll option costs $10 but saves an hour of travel time, 40 percent of the travelers would pay the toll under the low value-of-time distribution and 57 percent of the travelers would pay the toll under the high value-of-time distribution. In other words, under the low value-of-time example, 40 percent of the trips on this interchange will be assigned to the toll path and 60 percent of the trips will be assigned to the no-toll path.

Figure 5-15: Toll Probability Example



Source: Tampa-Hillsborough Expressway Authority Bus Toll Lane Feasibility Study

Figure 5-16: Existing Equivalent Toll Minutes



Source: MWCOG. These value-of-time values are derived from information in the equivalent toll-minutes file (Toll_Minutes.TXT). For details, see page 103 of the September 18, 2013 user's guide for the Version 2.3 Travel Model.

The current MWCOG process uses equivalent toll minutes for converting toll costs to a generalized time for each of the six vehicle classes during path building as shown in Figure 5-16.

In managed lane applications, typically, the value-of-time distributions are assumed to be of lognormal nature, where the toll-revenue is closely related to the length of the "tail" of this distribution. Statistically, the mean and the median values of a distribution can be used to generate a representative normal or a lognormal curve. As part of the toll choice model, toll choice distribution curves based on the value of time distributions for each of the six vehicle classes and four time-of-day periods were generated using the data provided by COG/TPB staff[30] and shown in Table 5-3. These VOT distributions ("probability density functions" or PDFs) are then used to derive toll-choice probability curves by computing 1 minus the cumulative probability distribution function (CDF), and are used to separate the OD trips into two groups: toll trips and no-toll trips. Note that this split is performed only once per toll setting iteration in order to avoid affecting the highway assignment's convergence. Toll and no-toll skims are prepared by building paths with and without access to all of the tolled links to assist in this process.

The value-of-time distributions and toll-choice probability distributions for each of six vehicle classes can be seen in Figure 5-17 for AM, Figure 5-18 for MD, Figure 5-19 for PM and Figure 5-20 for NT. In each of these figures, there are two sets of curves: one set that starts from 0% on the left Y-axis and another set that starts from 100% on the left Y-axis. The set that starts from 0% is the value-of-time PDF corresponding to a given vehicle-class. These curves have the distinct tail as mentioned earlier. The other set that starts from 100% ("waterfall curves") are the toll-choice probability curves (1 – CDF)

[30] Memorandum on "Value of Time (VOT) by Time Period and Vehicle Classification" dated October 24, 2013 from Ron Milone (COG/TPB) to David Roden (AECOM).

indicating the likelihood (Y axis) of paying certain toll (X axis) in exchange for an hour of travel-time savings.

Table 5-3: Time Valuations by Time Period and Vehicle Class (Source: MWCOG)

	Line No.		
Assumption Set 1	1	**Hourly Median Wage Rate ('07$/Hr):**	$20.51
	2	**Hourly Mean Wage Rate ('07$/Hr):**	$26.11
	3	**Ratio of Mean/Median Wage Rate:**	1.27
		Median VOT ('07$/Hr) by Trip Purpose :	
	4	**Work-related (100% of the avg. wage rate):**	$20.51
	5	**Non-Work Related (50% of the avg. wage rate):**	$10.26

Assumption Set 2

	Cumulative VOT Factor						
		1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
6	Work	1.0000	2.0000	3.5000	1.5000	1.5000	1.0000
7	NonWork	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000

Assumption Set 3

	AM period	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
8	Work	0.5618	0.5618	0.5618	1.0000	1.0000	1.0000
9	Non-Work	0.4382	0.4382	0.4382	0.0000	0.0000	0.0000
	Midday Period	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
10	Work	0.2391	0.2391	0.2391	1.0000	1.0000	1.0000
11	Non-Work	0.7609	0.7609	0.7609	0.0000	0.0000	0.0000
	PM Period	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
12	Work	0.3918	0.3918	0.3918	1.0000	1.0000	1.0000
13	Non-Work	0.6082	0.6082	0.6082	0.0000	0.0000	0.0000
	Night/Early Period	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
14	Work	0.2681	0.2681	0.2681	1.0000	1.0000	1.0000
15	Non-Work	0.7319	0.7319	0.7319	0.0000	0.0000	0.0000

Table 5-4: Time Valuations by Time Period and Vehicle Class (contd.) (Source: MWCOG)

Result 1

	Median VOT ($/Hour) by Time period and Vehicle Type	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
16	AM	16.02	27.54	44.82	30.77	30.77	20.51
17	Midday	12.71	17.61	24.97	30.77	30.77	20.51
18	PM	14.27	22.31	34.36	30.77	30.77	20.51
19	Night/Early	13.00	18.50	26.75	30.77	30.77	20.51

Result 2

	Mean VOT ($/Hour) by Time period and Vehicle Type	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
20	AM	20.39	35.06	57.06	39.17	39.17	26.11
21	Midday	16.18	22.42	31.78	39.17	39.17	26.11
22	PM	18.17	28.40	43.74	39.17	39.17	26.11
23	Night/Early	16.56	23.56	34.06	39.17	39.17	26.11

Minutes per dollar Implied by Result 1

Result 3

	Median VOT (Min/Dollar) by Time period and Vehicle Type	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
24	AM	4	2	1	2	2	3
25	Midday	5	3	2	2	2	3
26	PM	4	3	2	2	2	3
27	Night/Early	5	3	2	2	2	3

Existing Version 2.3 Minutes per dollar

Median VOT (Min/Dollar) by Time period and Vehicle Type	1-Occ. Auto Dr.	2-Occ. Auto Dr.	3+Occ. Auto Dr.	Comm. Veh	Med/Hvy Truck	Airport Auto Dr.
AM	3	2	1	2	2	2
Midday	3	4	4	2	2	2
PM	3	2	1	2	2	2
Night/Early	3	4	4	2	2	2

Figure 5-17: Value of Time Distributions – AM Peak Period

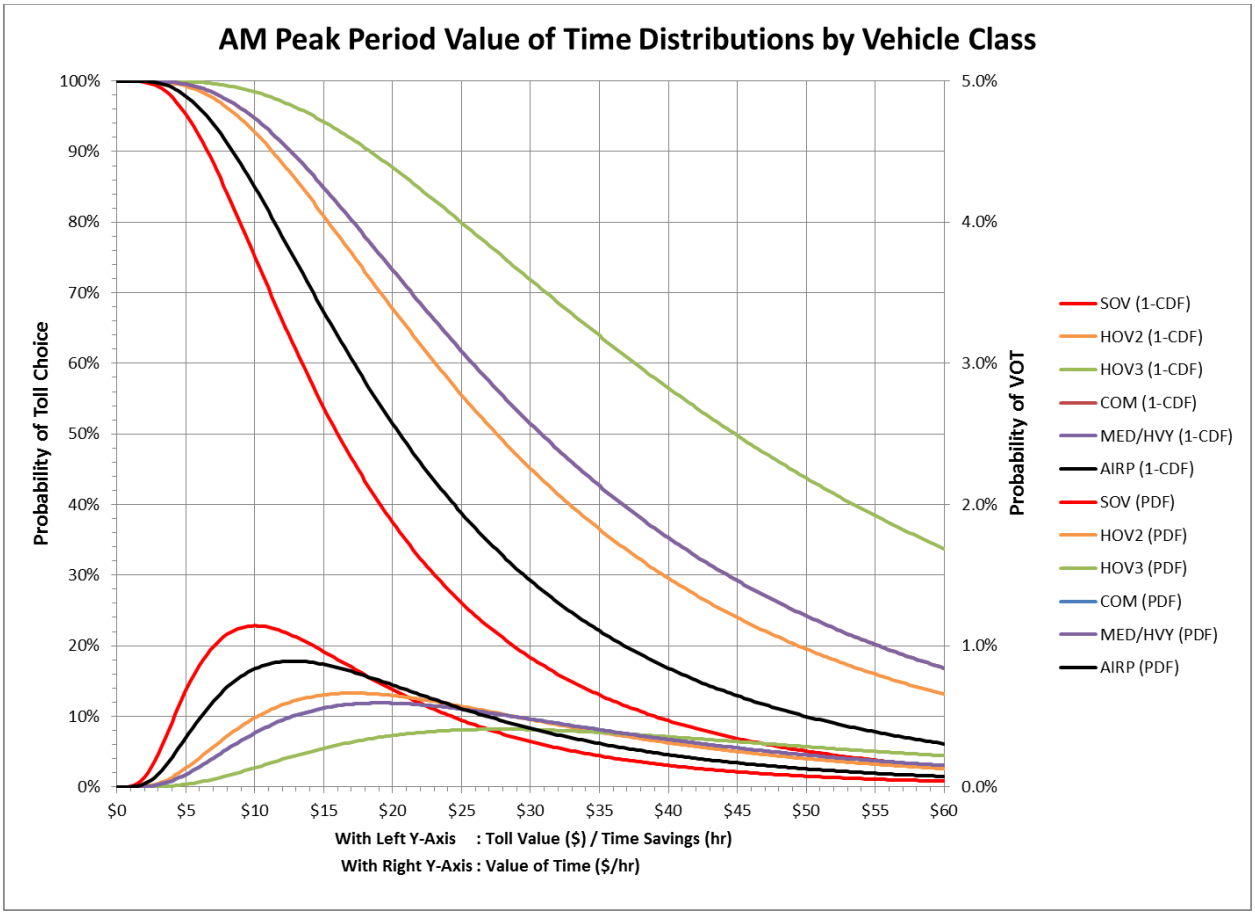


Figure 5-18: Value of Time Distributions - Mid-Day

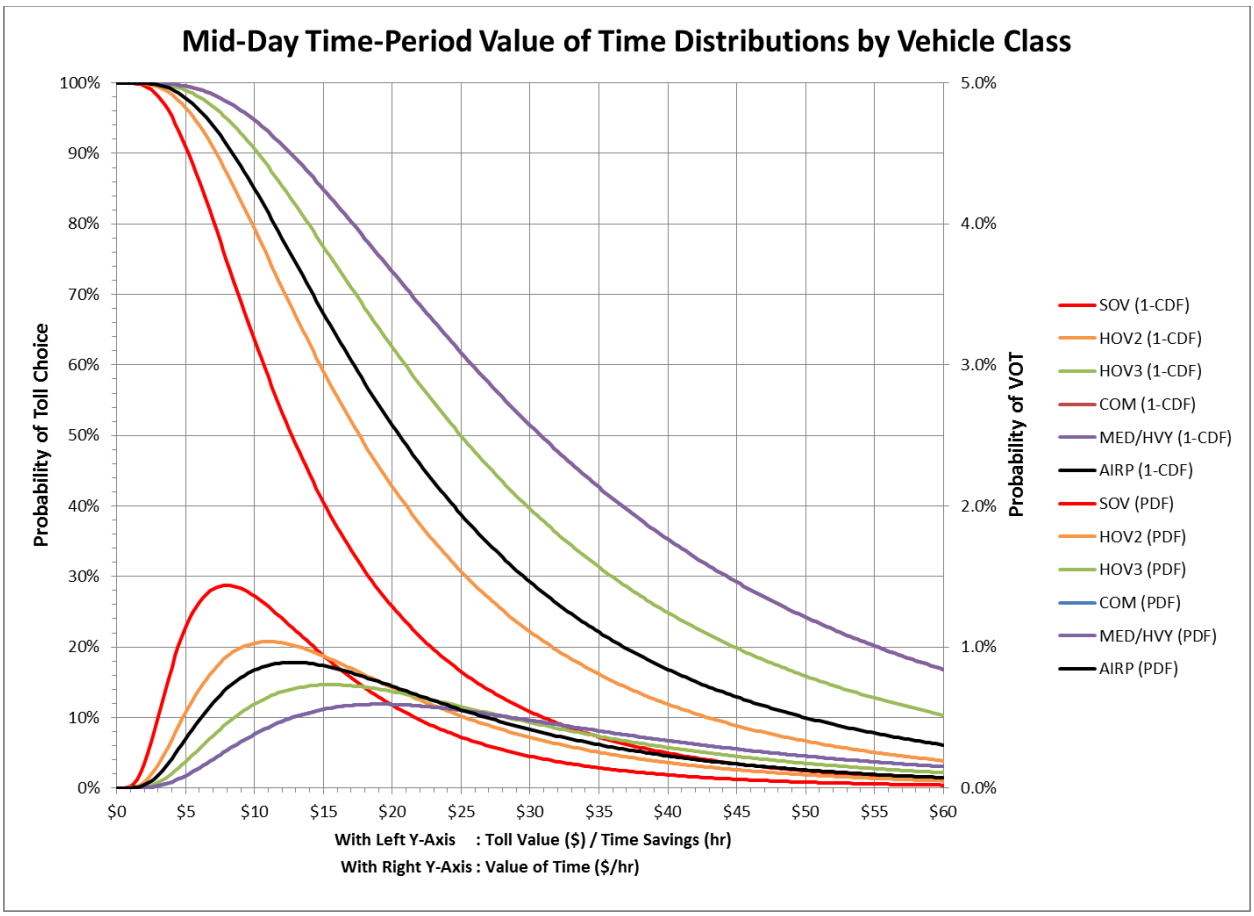


Figure 5-19: Value of Time Distributions - PM Peak Period

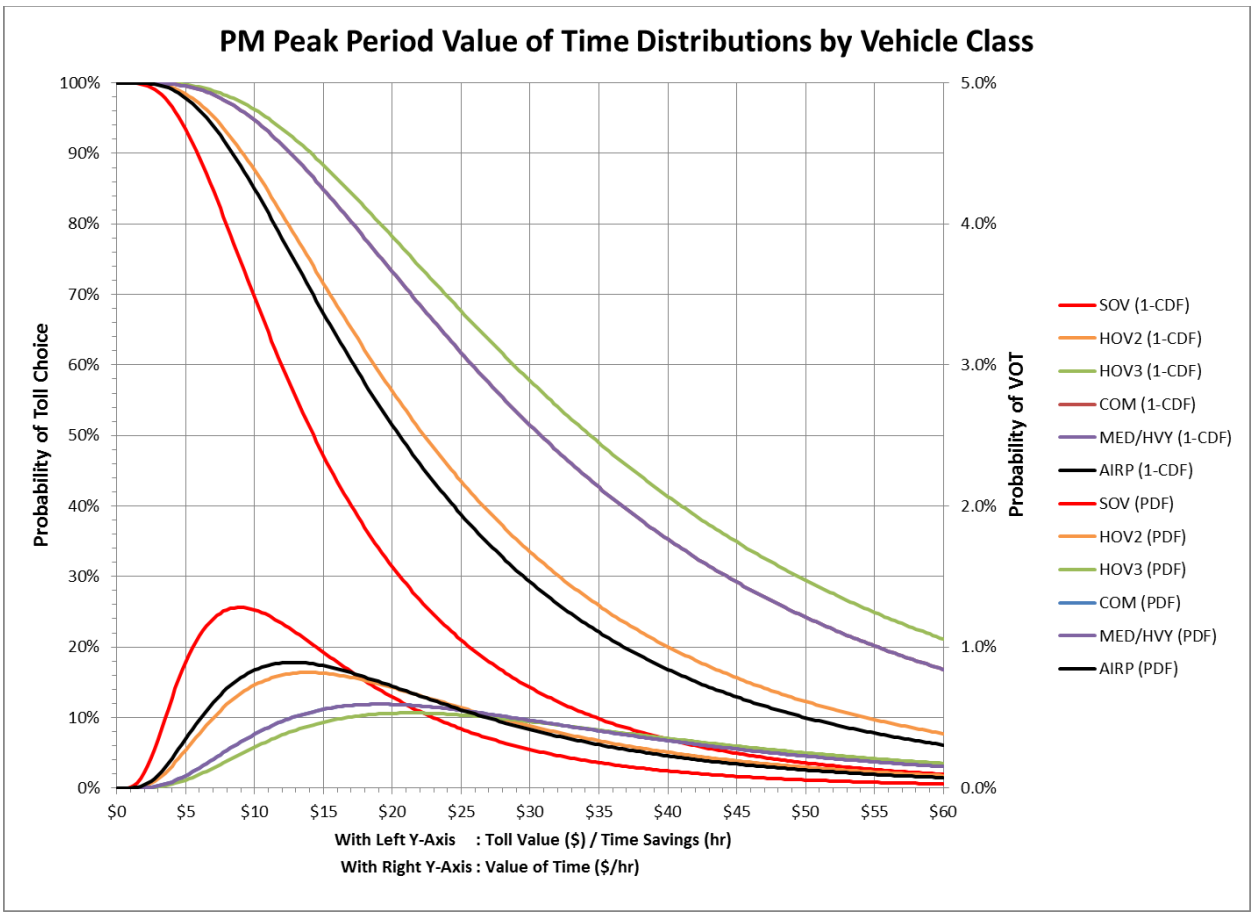
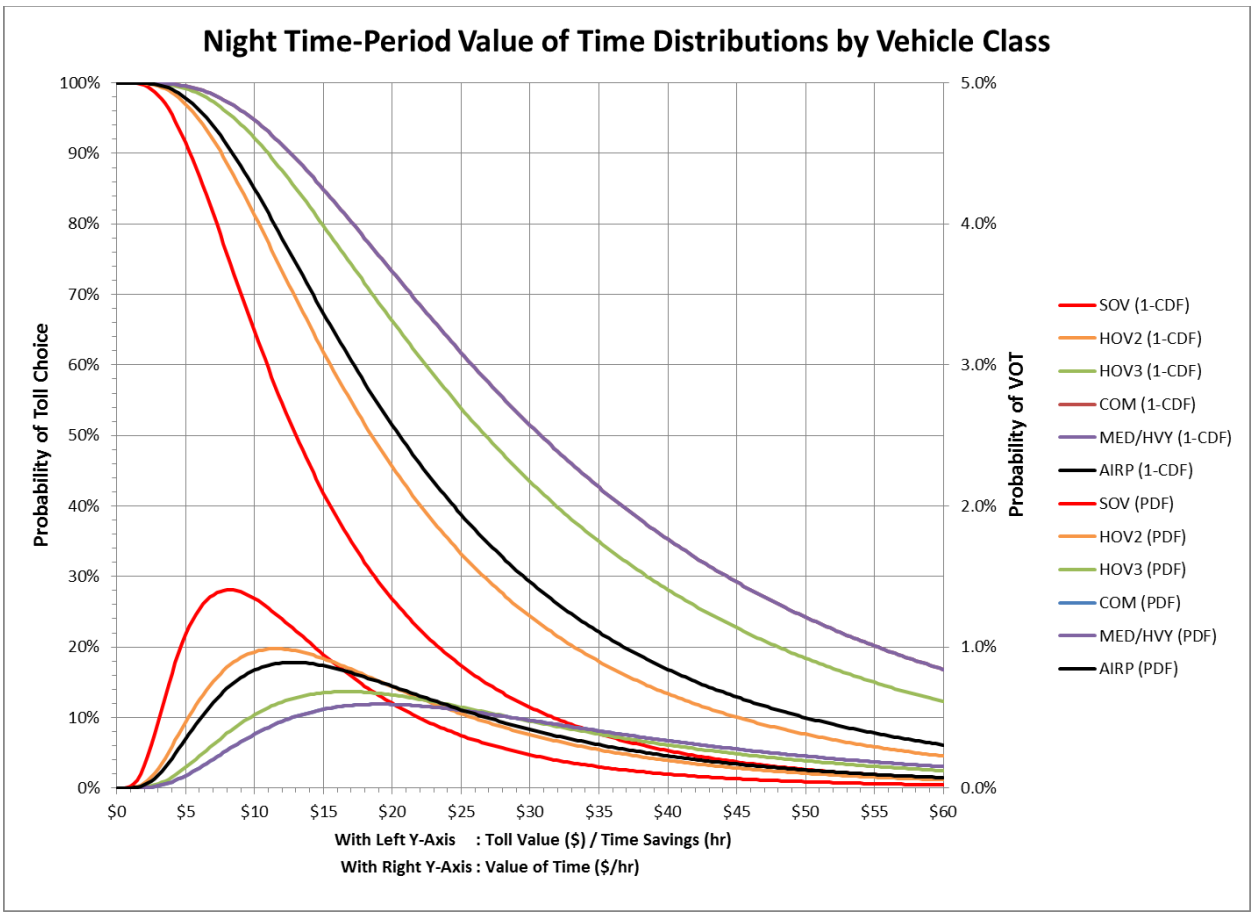


Figure 5-20: Value of Time Distributions - Night Time Period



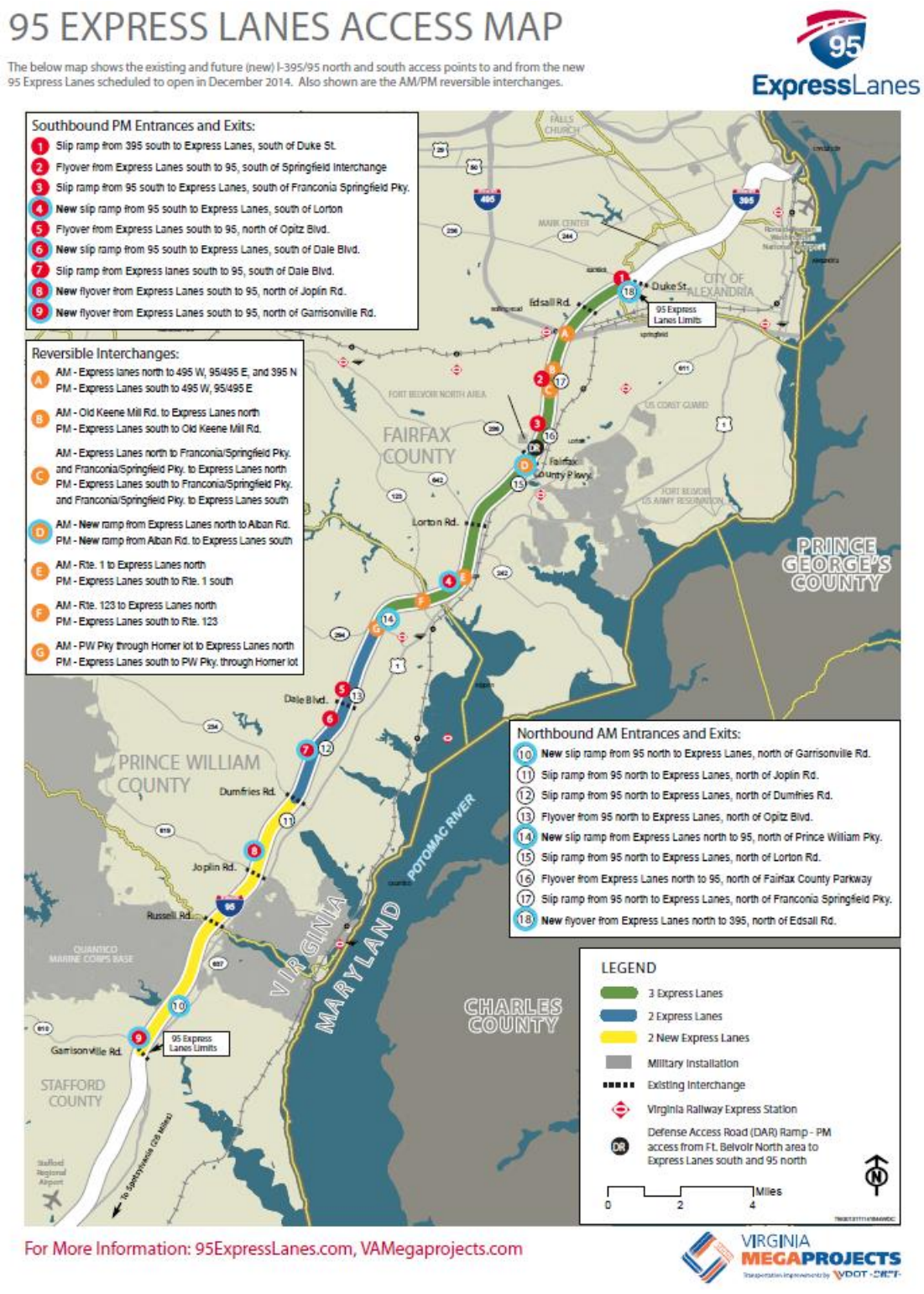
5.5.4 Toll-Group Changes

The COG/TPB toll groups range from 0 to 134 where toll-groups numbered 1 and 2 use static tolls, whereas toll groups numbered 3 and above use dynamic tolls (i.e., HOT lanes). Toll group 1 was designated by MWCOG staff for toll roads that existed at the time of the calibration to year-2007 conditions. Toll group 2 was reserved for the Intercounty Connector (ICC).[31] Following one of the recommendations of Task Order 8, these toll-groups were re-evaluated for possible consolidation, based on ground conditions, i.e., the real-world entries and exits on the express-lanes on I-95 and I-495. Consolidating toll-groups helps improve the processing times for toll-setting by reducing the potential number of independent combinations of toll-rates. A reduction in the number of toll groups, results in a reduction in the number of different tolls to evaluate.

In examination of the real-world access points, as shown in Figure 5-21 and Figure 5-22, the toll groups were reduced from a maximum of 134 to a maximum of 91. During this consolidation, the "Toll_Esc.dbf" file was also modified so as to effectively retain the original toll on links unaltered.

[31] Milone et al., *User's Guide for the MWCOG/NCRTPB Travel Forecasting Model, Version 2.3, Build 52*, 91.

Figure 5-21: I-95 Express Lanes Access Map



Source: 95expresslanes.com (VDOT)

Figure 5-22: I-495 Express Lanes Access Map

Source: 495expresslanes.com (VDOT)

Figure 5-23 shows the location of the original 134 toll groups (above toll group 3), using colors to indicate each group.

Figure 5-23: Location of Original Variably Priced Toll Groups

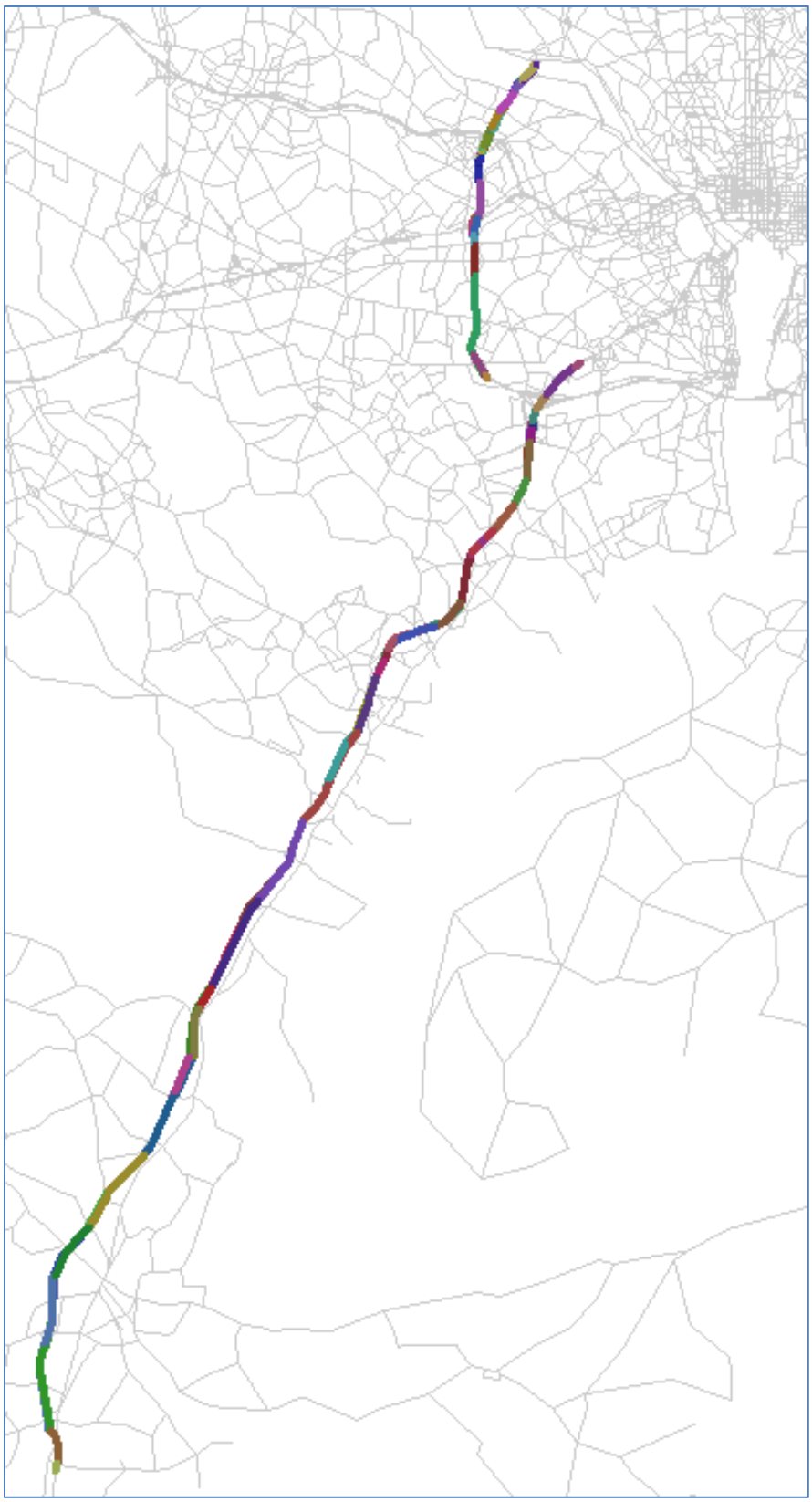


Figure 5-24, Figure 5-25 and Figure 5-26 show the locations where the toll groups were consolidated, using thick and dark colored links to represent places where consolidation occurred

Figure 5-24: Location of Collapsed Toll Groups on I-495

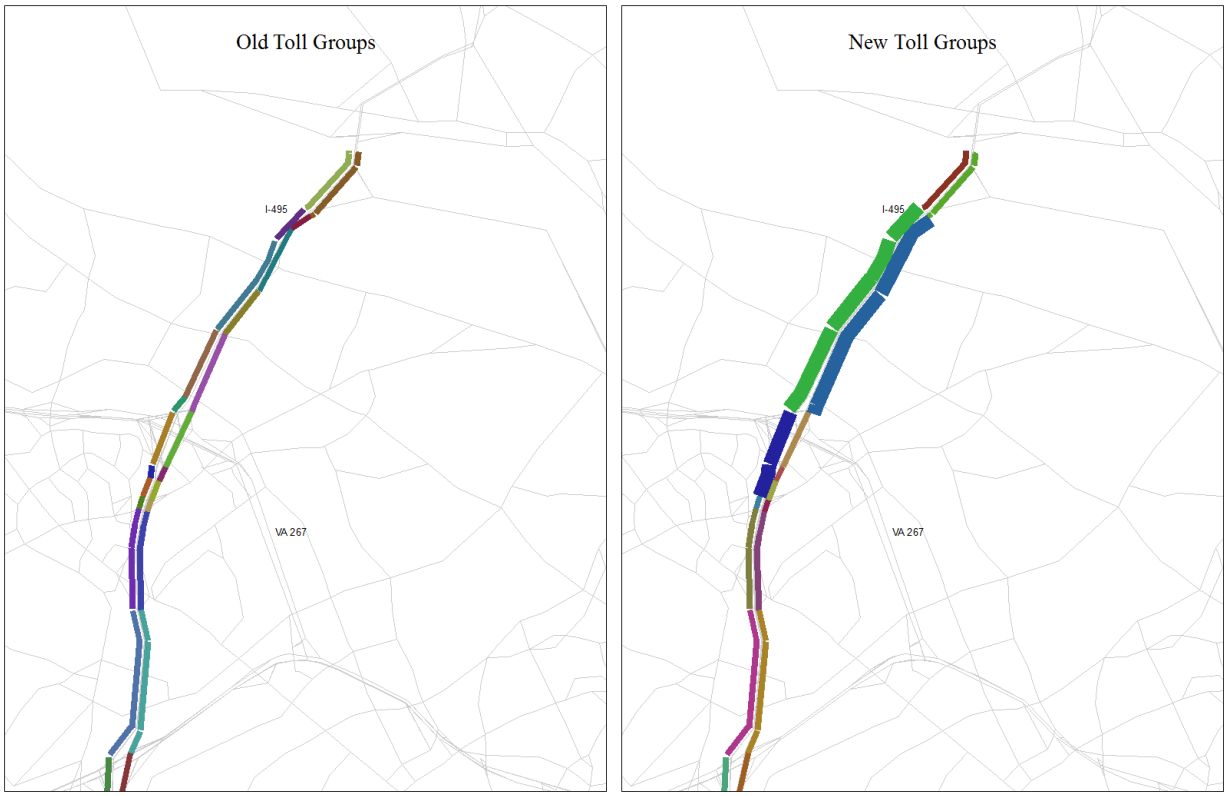


Figure 5-25: Location of Collapsed Toll Groups on I-95 (1/2)

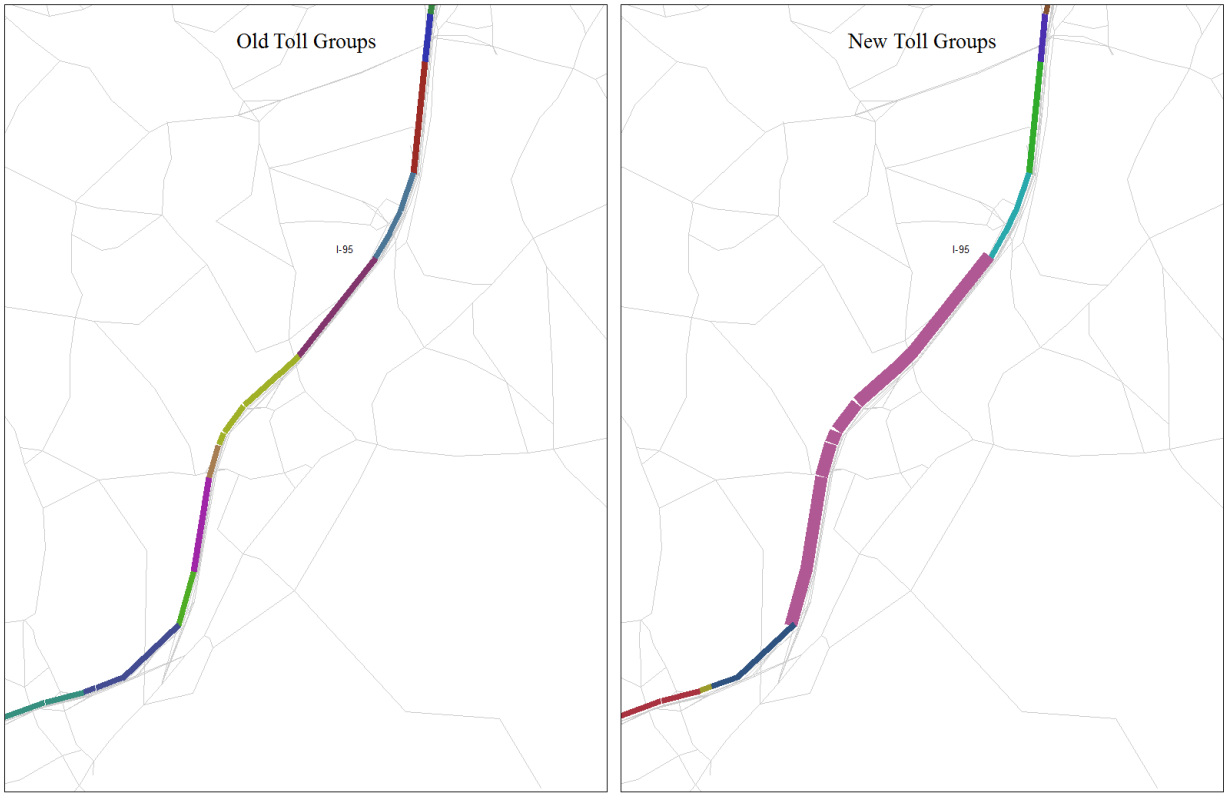


Figure 5-26: Location of Collapsed Toll Groups on I-95 (2/2)

Table 5-5 provides equivalence between the original[32] and proposed toll-groups.

[32] Section 3.2.2, Ron Milone et al., *Highway and Transit Networks for the Version 2.3.52 Travel Model, based on the 2013 CLRP and FY 2013-2018 TIP,* Draft Report, February 14, 2014

Table 5-5: Equivalence between COG and Proposed Toll Groups

HOT Lane Segment	Original Toll Group	Proposed Toll Group
American Legion Brdg. to GW Pkwy	4	5
GW Pkwy to N. of VA 193(Georgetown Pike)	6	6
VA 193 to VA 738 (old Dominion Dr)	8	6
VA 738 to VA 694 (Lewinsville Rd)	10	6
VA 694 to N of VA 267 (Dulles Toll Rd)	12	6
At VA 267	14	7
S. of VA 267	16	7
VA 267 to VA 123 (Chain Bridge Rd)	18	7
At VA 123	20	8
VA 123 to VA 7 (Leesburg Pike)	22	9
VA 7 to N. of I-66	24	10
At I-66	26	11
I-66 to US 29 (Lee Highway)	28	12
US 29 to VA 650 (Gallows Rd.)	30	13
VA 650 to N. of VA 620 (Braddock Rd)	32	14
At VA 620	34	15
VA 620 to Hemming Ave.	36	16
Hemming Ave. to VA 620 (Braddock Rd)	7	19
At VA 620	9	20
VA 620 to VA 650 (Gallows Rd.)	11	21
VA 650 to US 29 (Lee Highway)	13	22
US 29 to I-66	15	23
At I-66	17	24
N. of I-66 to VA 7 (Leesburg Pike)	19	25
VA 7 to S. of VA 123 (Chain Bridge Rd)	21	26
At VA 123	23	27
VA 123 to S. of VA 267 (Dulles Toll Rd)	25	28
S. of VA 267	27	29
At VA 267	29	30
N of VA 267 to VA 738 (old Dominion Dr)	31	31
VA 738 to VA 193 (Georgetown Pike)	33	31
At VA 193	35	31
N of VA 193 to GW Pkwy	37	31
GW Pkwy to American Legion Brdg.	39	32
VA 236 ~ VA 648	76	36
VA648 ~ N. of I-495	78	37
At I-495	80	38
I-495 ~ VA 644	82	39

At VA 644	84	40
VA 644 ~ VA 289	86	41
At VA 289	88	42
VA 613 ~ VA 7100	90	43
At VA 7100	92	44
VA 7100 ~ VA 638	94	45
VA 638 ~ N. of VA 611	96	45
At VA 611	98	45
VA 611 ~ N. of US 1	100	45
At US 1	102	45
US 1 ~ N of VA 123	104	46
US 1 ~ N of VA 123	104	47
VA 123 ~ N. of VA 294	106	48
VA294 ~ N. of VA 642	108	49
At VA 642	110	50
VA 642 ~ Cardinal Dr	112	51
S. of Cardinal Dr ~ S. of VA 234	114	52
S. of VA 234 ~ N. of VA 619	116	52
VA 619 ~ S. of Russell Rd	118	53
S. of Russell Rd ~ N. of Garrisonville Rd	120	53
N. of Garrisonville Rd ~ N. of Courthouse Rd	122	54
N. of Courthouse Rd ~ S. of Ramoth Church Rd	124	55
S. of Ramoth Church Rd ~ N. of US 17	126	56
N. of US 17 ~ S. of US 17	128	57
S. of US 17 ~ N. of US 1	130	58
At US 1	132	59
S. of US 1	134	60
S. of US 1	61	64
At US 1	63	65
N. of US 1 ~ VA 3	65	66
N. of VA 3	67	67
N. of VA 3 ~ N. of US 17	69	68
S. of US 17 ~ S. of Ramoth Church Rd	71	69
S. of Ramoth Church Rd ~ N. of Courthouse Rd	73	70
N. of Courthouse Rd ~ Garrisonville Rd	75	71
N. of Garrisonville Rd	77	72
N. of Garrisonville Rd ~ Russell Rd	79	73
N. of Russell Rd ~ S. of VA 234	81	74
S. of VA 234 ~ Cardinal Dr	83	74
Cardinal Dr ~ VA 642	85	75
N. of VA 642 ~ VA 294	87	76

At VA 294	89	77
N. of VA 294 ~ VA 123	91	78
N. of VA 123 ~ US 1	93	79
N. of US 1 ~ N. of VA 611	95	80
N. of VA 611 ~ VA 638	97	81
N. of VA 611 ~ VA 638	97	82
VA 638 ~ VA 7100	99	83
At VA 7100	101	84
VA 7100 ~ VA 289	103	85
At VA 289	105	86
VA 613 ~ VA 644	107	87
At VA 644	109	88
VA 644 ~ S. of I-495	111	89
At I-495	113	90
N. of I-495	115	91

5.5.5 Toll Setting Algorithm

As part of the toll setting process, the first step is to read a loaded highway network to compute average V/C ratios along with the weighted speed and VMT for each link in a toll group. The next step involves processing every toll group to compare the V/C ratios against the upper threshold V/C ratio of 1.01. For toll groups with V/C ratios above the upper threshold (1.01), the toll is raised, whereas, for toll groups with V/C ratios below the lower threshold (0.95), the toll is lowered.

To avoid oscillation and reduce the number of toll setting iterations, the toll increase is coded as a non-linear function, whereas the toll decrease is coded as a linear function, as shown below. These toll estimation functions define the relationship between current tolls and current V/C ratios used to adjust the tolls for the next iteration. While linear functions with differing slopes could also be used to represent the toll increase and toll decrease functions, progression of toll-rates across toll-setting iterations in the initial test runs revealed that certain toll-groups changed their tolls constantly and ended up with relatively very high toll rates compared to most others. These toll groups corresponded to the very congested freeway segments (such as the links near the I-95/I-395/I-495 Springfield "mixing-bowl" interchange) in the region. Such toll-groups essentially determine the eventual number of toll-setting iterations. So, for computational efficiency, the "step-size" for toll increases should be such that when V/C ratios are far removed from the upper threshold (1.01), large steps or large increases in toll-rates can be made and when the V/C ratios are closer to the lower threshold (0.95), the toll increases should be smaller. This approach would help prevent over-shooting the target V/C ratio range while reducing the number of potential toll-setting iterations especially in presence of toll-groups that require very high toll rates.

Non-Linear Toll Increase Function (applied beyond the upper V/C ratio threshold of 1.01):

$$New\ (higher)\ Toll\ Rate = Old\ Toll\ Rate + \left(\lambda_1 * \ln\left(abs\left(VC_{Avg} - 1.01\right)\right) + \lambda_2 \right)$$

Where,

$$\lambda_1 = 18.0$$

$$\lambda_2 = 80.0$$

Linear Toll Decrease Function (applied below the lower V/C ratio threshold of 0.95)

$$New\ (lower)\ Toll\ Rate = Old\ Toll\ Rate * \left(VC_{Avg}/1.01\right)$$

The non-linear toll increase function was formulated by fitting curves to the data points gathered from the initial and preliminary toll setting iterations. During the initial estimation, values of 12.691 and 68.891 for the λ_1 and λ_2, respectively, were found to correspond to the toll-increase function in MWCOG's offline toll setting procedure. These parameters were later updated to their current values to minimize toll setting iterations.

5.6 Application Considerations

There are two types of applications of the existing COG/TPB model – one where HOT-lane highway assignments are of interest and the other where they are not, all depending on the location and type of outputs the user is interested in. Sometimes, the processing time also plays an important role in deciding amongst these two types of applications. The former type of application requires the multi-run application whereas the latter can be done by running only the "base" scenarios. A third type of application, toll-setting, also exists but is primarily done by COG/TPB staff as an offline process to generate input tolls for variably priced facilities. Other users of the COG/TPB model do not use this type of application due to lack of availability of the procedures or simply because of prohibitive processing times. It may be noted that the first two types of applications use the input tolls and therefore are not sensitive to any network coding changes by the user. Only the toll-setting, the third type of application, is responsive to such network coding changes.

The changes done to highway assignment procedures as part of this effort combine the first two typical applications by the user into a single type while using half of the processing time required for a multi-step run, and also extend the option to perform integrated toll-setting which would be responsive to the user's network coding. Even with the ability to perform integrated toll-setting, the application without toll-setting is anticipated to be the predominant type of application due to the processing time considerations.

With toll-setting enabled, obviously the processing times are significantly extended, in most cases to multi-day runs. As noted in the description of the new procedures, there are several different possible configurations for toll-setting, depending on the precision and method of application. Following are some considerations that the user should keep in mind during a toll-setting application to better understand the expected processing times and results:

1. The toll-setting feature is completely disabled by setting the key "_hwy_HOT_perform_toll_setting_" to '0'. In such a case, a (single) multi-class assignment is performed for each time period similar to the existing off-peak period highway assignments using original tolls and without using the value-of-time distributions or other toll-setting related keys. In other words, this is the "master" key that controls the overall assignment process to be very similar to the way it is currently done in the current COG/TPB model version 2.3.52. Note that, in all cases, the HOV choice model is not optional because it helps compensate for removal of the "two-step" and "multi-run" assignments.

2. Each combination of the controls involved in toll-setting may result in a different solution. For example, if one user chooses to lower/relax the relative gap during toll-search portion of toll-setting by the key "_hwy_HOT_lower_relgap_for_toll_setting_" to '1', he or she may end with a different answer than when this key is set to '0'. Note that some tests suggest a relatively lower value of toll-rates when using lowered/relaxed relative gap. However, this observation is not necessarily always true.

3. Seed toll-rate files are critical to reducing the number of toll-setting iterations. The seed toll-rate files provide a "warm start" to the toll-search algorithm, hence good ones will quicken the search for optimal tolls, whereas the bad ones may either delay or result in an alternate solution.

4. If the option is used to carry forward the tolls across speed-feedback iterations by setting "_hwy_HOT_use_last_tolls_" set to '1', the progressive convergence criterion which lowers the relative gap threshold may result in toll-rates that may be worse starting points than the seed toll-rates. Nonetheless, it may potentially finish quicker.

5. The results from toll-setting enabled runs are un-calibrated. Calibration of the highway assignments needs to be performed to observed tolled facilities (Dulles Greenway, Dulles Toll Road, ICC and I-495 Express Lanes which became operational towards the end of 2012) under any chosen configuration of toll-setting prior to its application. Such a calibration should continue to ensure that the impedances used in the toll-choice (splitting trip tables into toll and non-toll users) are consistent with the impedances used in the toll-assignments.

5.7 Conclusions and Recommendations

The goal of this task order was successfully achieved along with additional processing time savings. It is, however, important to do thorough testing with various combinations of the toll-setting parameters to arrive at a configuration that has reliable processing times. Following are some salient recommendations that may be considered for future improvements:

1. Do more testing of the toll-setting parameters for different future years to determine the most reliable configuration, i.e., the most "seasoned" seed tolls. A by-product of this testing would also suggest which time-periods need to be combined to run in parallel – for 2020 it was found

that the AM time-period took as much time as the PM, MD and NT combined. Hence, AM was run in one Cube Cluster and PM, MD and NT were run in sequence in another Cube Cluster.

2. The testing mentioned above may also include determining the impact/effectiveness of using toll-setting only for the final speed-feedback iteration versus all speed-feedback iterations.
3. Use the most reliable configuration of toll-setting and calibrate the highway assignment volumes to observed volumes and tolls on HOT-lane and tolled-facilities.
4. Instead of using a hard-coded relative gap (currently 0.1) cutoff for toll-setting applications using lowered relative gap cutoff threshold for toll-setting, use a function of the full relative gap cutoff for specific speed feedback iterations. For example, instead of using 0.1 for both i1 and i4, use half of the relative gap cutoffs, i.e., half of 0.01 for i1 and half of 0.0001 for i4.
5. Consolidate input toll parameter file (toll_esc.dbf) and seed toll files or create a process to create one from another for user-convenience.
6. Consider extending the continuous VOT distribution approach to separate toll-users from others in toll-setting to assignments performed without toll-setting. This would be a better approach to model toll-choice by using more realistic continuous VOT distributions as opposed to average VOTs. Such an application should be followed by re-calibration of the highway assignments.
7. Consider modifying the toll-setting procedures to contain inflated-dollars as opposed to currently utilized deflated-dollars, if the need for such a change is felt. This modification would need to utilize deflation and inflation factors that consider the model calibration year (2010), the model year for representing monetary values (2007) and the scenario year (2014, 2017, etc.) in order to be consistent through the entire modeling process.

6 PT Network Preparation and Path Building (Task Order 13)

MWCOG is planning to convert its current transit modeling procedure from the Citilabs Cube TRNBUILD (TB) module to the Citilabs Cube Public Transport (PT) module. PT offers a number of advantages and challenges over the current TB procedure.

6.1 Terminology

Transit paths in PT are built using "transit legs" and "non-transit legs". A "transit leg" is a "trip segment, from a boarding point to an alighting point that uses a single transit line." [33] By contrast, a "non-transit leg" or an NT leg, is a minimum-cost segment, traversed by non-motorized ("non-mechanized") modes.[34] In TRNBUILD, non-transit legs (or connectors) are called "support links." Passengers use non-transit legs to:

- *Access the public transport system*
- *Egress from the public transport system*
- *Transfer between lines* [35]

Non-transit legs may traverse zero, one, or multiple physical links. PT path-building alternates between non-transit and transit legs, i.e., paths cannot use consecutive non-transit legs or consecutive transit legs.

6.2 Background

AECOM has previously provided MWCOG with technical support and guidance to efficiently and effectively convert the current TB transit input files and process to PT. The most important part of the conversion process was a network design that enables PT to build high-quality transit paths for different access modes. This task addressed a few remaining questions regarding drive-access legs to bus park-n-ride lots and station-to-station inputs for Metrorail fare calculations. This chapter describes the steps needed to prepare the input files and then focuses on the actual PT transit processing to generate non-transit legs, perform transit skimming for all access modes (walk, park-n-ride, and kiss-n-ride) for bus+Metrorail (BM) line-haul paths during the peak period. It also explains how PT can be used to save the first and last Metrorail, LRT, and commuter rail stations on the path between each origin-destination (actually production-attraction) zone pair. These data are needed for transit fare calculations.

6.3 Drive-Access Legs to Bus Park-n-Ride Lots

In the MWCOG version 2.3.52 model, bus park-n-ride lots are represented by node numbers 13,000 to 13,999.[36] These nodes are introduced to TRNBUILD with their associated support links. Like other park-

[33] Citilabs, Inc., *Cube Voyager Reference Guide, Version 6.0.2* (Citilabs, Inc., July 26, 2012), 853.
[34] Ibid., 856.
[35] Ibid., 795.
[36] Ronald Milone et al., *Highway and Transit Networks for the Version 2.3.52 Travel Model, Based on the 2013 CLRP and FY 2013-2018 TIP*, Final Report (Washington, D.C.: Metropolitan Washington Council of Governments, National Capital Region Transportation Planning Board, March 18, 2014), 2–12.

n-ride lot nodes, bus park-n-ride lots are connected to highway nodes using a “mode 15” walk/transfer link. During transit path building process, the MWCOG TB process creates drive-access links between zone centroids and bus park-n-ride lots and uses the “mode 15” support links to allow transit trips to transfer from the park-n-ride lot to the nearest bus stops on the transit network. Figure 6-1 shows a bus park-n-ride lot and its associated support links (both drive and walk access) in the MWCOG TRNBUILD model.

Figure 6-1: Bus Park-n-Ride Lot and Drive and Walk Access Links in the MWCOG TRNBUILD Network

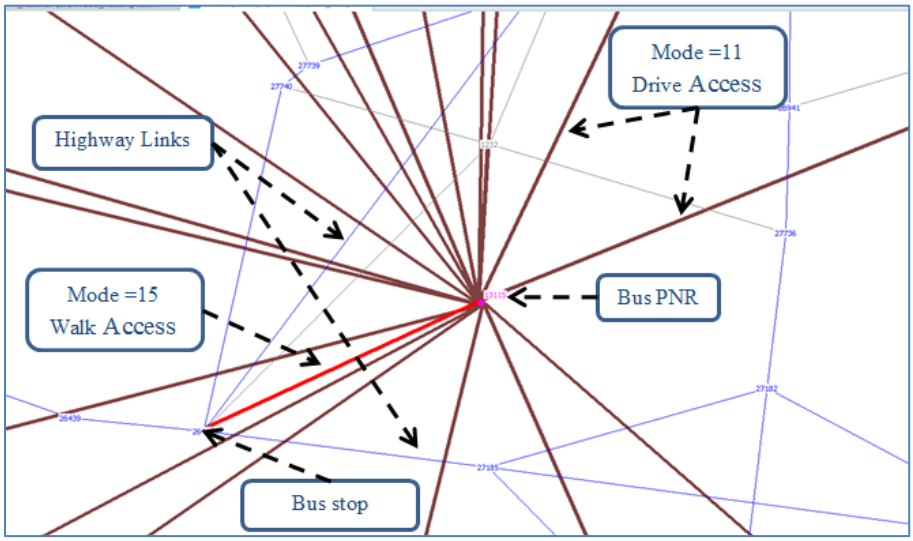
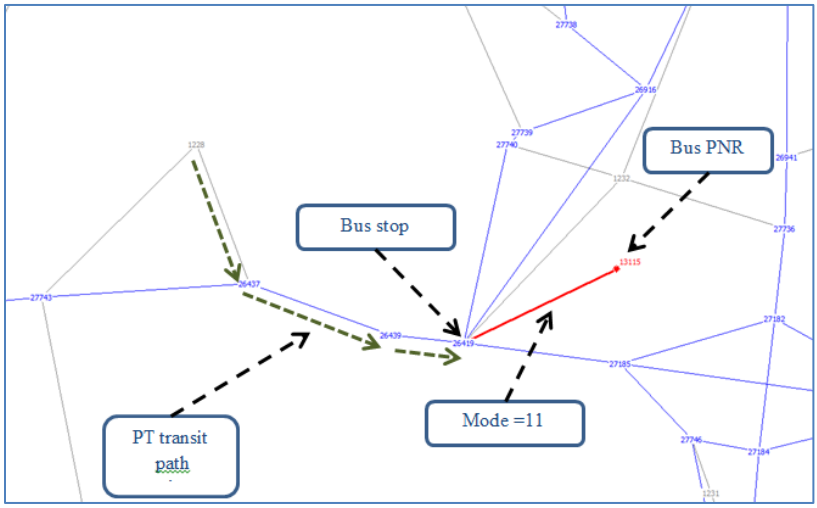


Figure 6-2 shows the PT network with a bus park-n-ride lot, a non-transit connector (the red, mode-11, PNR driveway), and a non-transit, drive-access leg (the three green links and the red link). The initial conversion of the TB-format transit network to the PT format did not function properly for creating drive access legs to bus park-n-ride lots. The reason was that PT finds it unnecessary for a user to drive to a park-n-ride lot and then walk back to a bus stop located on a node that was already part of the transit path (in this case it was part of the drive access path). It should be noted that this same approach does work properly for Metrorail park-n-ride lots in PT because Metrorail stops are not located on the highway network and PT must access the park-n-lot before reaching the Metrorail stop.

Figure 6-2: PT Network Including Bus Park-n-Ride Lots and Access Links



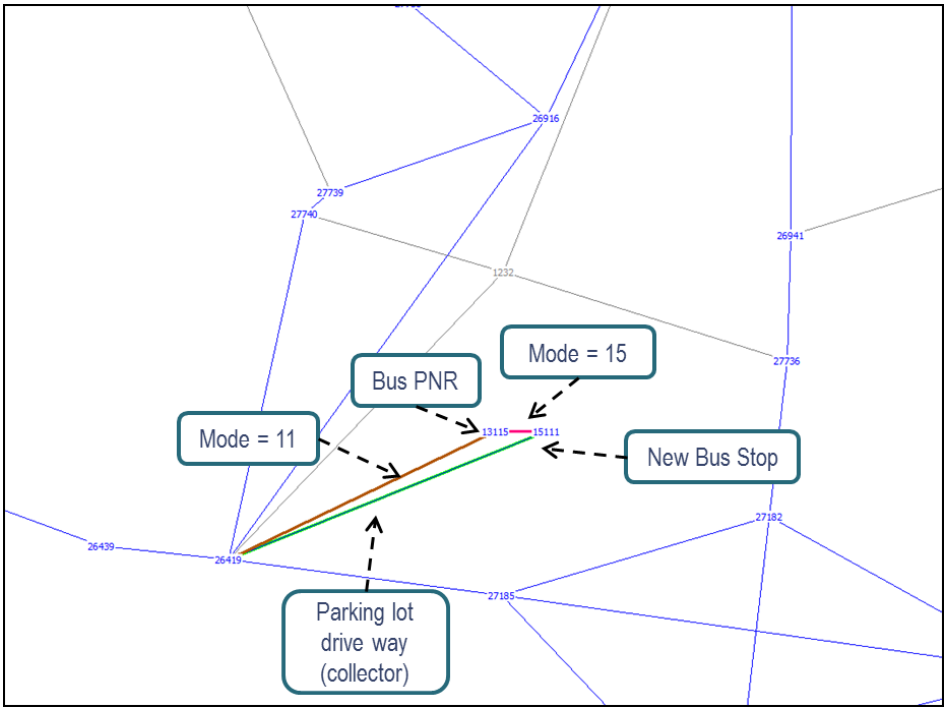
AECOM designed a coding procedure to address the problem by coding bus stops that provide connections to bus park-n-ride lots on transit-only links. Previously, bus stops were located on either the A or B node of highway links. The proposed solution locates the bus stops more accurately, based on the real-world conditions in which most park-n-ride lots are located such that buses must divert from the route on major roadways to access the bus stop. Figure 6-3 shows an example of this common situation at a real-world bus stop serving a bus park-n-ride lot at Swan Creek Road and Gable Lane (west of MD-210) in Prince George's County.

Figure 6-3: An Example of Bus Stop Serving a Park-n-Ride Lot in Prince George's County

Figure 6-4 shows the suggested network coding for a bus park-n-ride lot and its associated support links that would be used to build non-transit legs. The bus stop has been moved to more accurately represent the location of the stop. Compared to Figure 6-2, this figure has two additional links: 1) the green link is a bus-only driveway, which allows the bus to reach a new bus stop; 2) the short horizontal pink link is a

walk/transfer (“mode 15”) link between the bus park-n-ride lot and the new bus stop. The bus park-n-ride lot is reached via the slanted orange/pink link. This “mode 11” link allows drive-access to the bus park-n-ride lot, but is not traversed by any transit routes.

Figure 6-4: Suggested PT Network Representation for the Bus Park-n-Ride Lots



The steps below were used to resolve this issue and create non-transit, drive-access links to bus park-n-ride lots throughout the network.

Create New Bus Stops

Additional bus stops were inserted along routes that serve the bus park-n-ride lots. These stops replace the current bus stops that serve bus park-n-ride lots. Since MWCOG uses a specific node number range (13,000 to 13,999) for the bus park-n-ride lots, new bus stops can be created based on these node numbers. AECOM suggested using a parallel series of node numbers for the new bus stops (15,000 to 15,999). Also, the X/Y coordinates for the new bus stops were initially estimated by adding a fixed value, or offset, to the X or Y coordinates of the bus park-n-ride lots. Figure 6-5 shows the structure of the file including new bus stops.These values can be refined and modified in the future as time permits. AECOM noticed that the X/Y coordinates for some of the bus park-n-ride lots did not seem correct (e.g., 13,085 or 13,053). MWCOG staff reviewed and corrected these two park-n-ride lot locations.

Figure 6-5: Structure of the file including new bus stops

```
XY NODE= 15001 X=   1406620 Y=    409905    ;;Bristol
XY NODE= 15002 X=   1355264 Y=    548984    ;;Broken Land Pkwy
XY NODE= 15003 X=   1332437 Y=    526737    ;;Burtonsville Crossing
```

Create Support Links for New Bus Stops

New bus stops were connected to the highway network using highway support links (green link in Figure 6-4). Figure 6-6 shows a sample file including these support links. These are two-way links that will be used by buses to access the new bus stops. Therefore, care should be taken to ensure that their distance and speed are assigned properly. It is required that a "LIMIT" code is assigned to these links in order for the PT GENERATE statement to exclude them while building drive-walk paths to the new bus stops. This file was created based on the current support link file that connects bus park-n-ride lots to highway nodes. Basically, new bus stops replaced their corresponding park-n-ride lots in that file. AECOM suggested that MWCOG revise its highway network such that these bus-only driveway links are part of its highway network. MWCOG has now incorporated these bus-only driveway links in its PT-formatted network.

Figure 6-6: Structure of the File Including Supporting Highway Links for New Bus Stops

```
LINK NODES=15001-45558 FTYPE=4 DIST=0.05 ONEWAY=N SPEED=15.0 LIMIT_AM_PM_OP=9 LANES_AM_PM_OP=1
AMHTIME= 0.12
LINK NODES=15002-44132 FTYPE=4 DIST=0.05 ONEWAY=N SPEED=15.0 LIMIT_AM_PM_OP=9 LANES_AM_PM_OP=1
AMHTIME= 0.12
LINK NODES=15003-22539 FTYPE=4 DIST=0.05 ONEWAY=N SPEED=15.0 LIMIT_AM_PM_OP=9 LANES_AM_PM_OP=1
AMHTIME= 0.12
```

Create Support Links for Bus Park-n-Ride Lots

Park-n-ride lots needs to be connected to the highway network to build drive-access paths. The current support link file that connects bus park-n-ride lots to highway nodes was slightly modified and used for this purpose. Figure 6-7 shows a sample file including these support links. These are one-way links that will be used by autos to access bus park-n-ride lots and their distance and speed must be assigned properly. These links could be referred to as "PNR lot driveways."

Figure 6-7: Structure of the File Including Links Supporting Bus Park-n-Ride Lots

```
LINK NODES= 45558- 13001 MODE= 11 SPEED = 30.0 ONEWAY= T  DIST = 0.22 COST=0.45
LINK NODES= 44132- 13002 MODE= 11 SPEED = 30.0 ONEWAY= T  DIST = 0.22 COST=0.45
LINK NODES= 22539- 13003 MODE= 11 SPEED = 30.0 ONEWAY= T  DIST = 0.22 COST=0.45
```

Create Support Links to Connect Bus Stops and Park-n-Ride Lots

It was also necessary to create a walk connection between the new bus stops and the bus park-n-ride lots. Figure 6-8 shows a sample of the file including these walk links.

Figure 6-8: Structure of the File Including Walk Links Connecting New Bus Stops and Park-n-Ride Lots

```
SUPPLINK N=13001-15001 ONEWAY=Y MODE=15 DIST=     1 TIME=     2.06
SUPPLINK N=13002-15002 ONEWAY=Y MODE=15 DIST=     1 TIME=     2.56
SUPPLINK N=13003-15003 ONEWAY=Y MODE=15 DIST=     1 TIME=     2.06
```

Modify Current Bus Transit Lines

It was necessary to reroute buses serving bus park-n-ride lots to include the new bus stops. Figure 6-9 shows an example of a bus line that has been rerouted in order to make a stop at the new bus stop with the connection to the park-n-ride lot. This task was done by replacing the current bus stop used as a connection to the park-n-ride lot with a sequence of three nodes. For example, if node "A" was originally

the stop used as to the park-n-ride lot and node “B” was the new stop, node “A” was replaced with the following sequence: “A B -A” or “-A B A”, where the minus sign indicates a through node (or non-stop node). The current bus stop “A” can be kept as a stop node on the route either when the bus approaches or leaves node “B”. For a given route, CUBE does not permit a bus route to stop twice at the same node although the bus may pass through the node multiple times. Figure 6-10 shows an example of a node sequence that has been modified for a bus route in order to serve the new bus stop with a connection to the bus park-n-ride lot.

Figure 6-9: An Example of Rerouted Bus Line

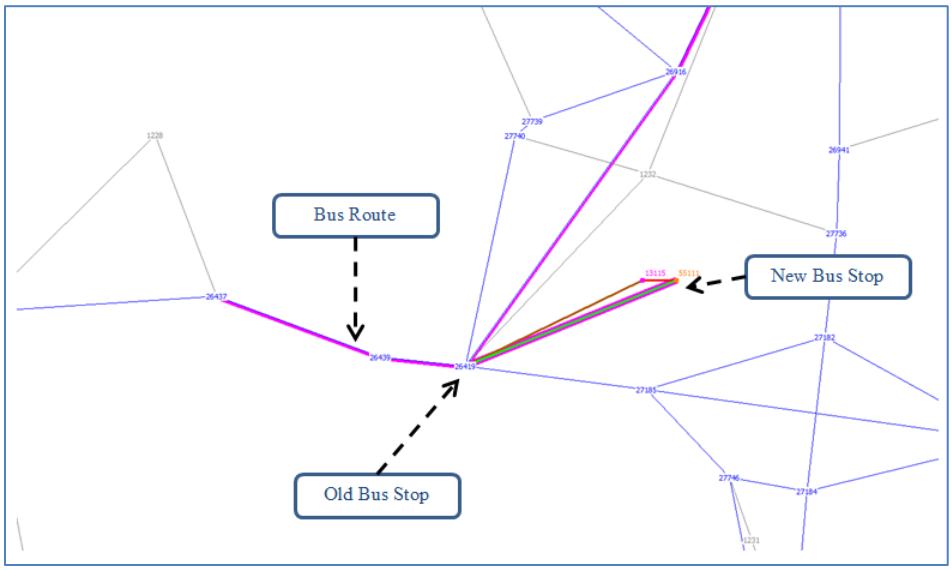


Figure 6-10: An Example of a Rerouted Bus Line Node Sequence

```
LINE NAME="WMC28I",
LONGNAME="Metro Bus;HALL ROAD & POINTER RIDGE DR;NEW CARROLLTON STATION & BUS BAY B;2012;Base",
ONEWAY=T,MODE= 01,HEADWAY[1]= 30,RUNTIME= 36,
          N=  26437  26439  26419  15111 -26419  26916  26254  27737  27741  26410  27742,
              26018 -27742  -26410  27278  26921 -27273  26919  27274 26413  15112 -26413,
             -26825 -26807  26385  15133 -26385 -26807 -26893 -27025 -26398 -27019 -26890,
             -26891 -27011  -26590 -26600 -27004 -27057 -26887 -26889 -26886,
             -26713  26148   27306  26147  27307  26138  26159
```

Since both the bus stop and the sequence that will be replacing that stop are known, it is possible to use a batch file to make a global change in each transit line. AECOM chose to replace the old bus node (“A”) with the new series of nodes (“A B -A”). Figure 6-11 shows an example of a batch file that can be used to replace a bus stop with this new sequence. The batch file makes use of “sed” (stream editor) commands.

Figure 6-11: An Example of a Batch File that Replaces a Bus Stop with a New Sequence

```
sed -i 's/26419/26419 15111 -26419/g' Mode1AM.lin
sed -i 's/44132/44132 15002 -44132/g' Mode1AM.lin
sed -i 's/22539/22539 15003 -22539/g' Mode1AM.lin
sed -i 's/26130/26130 15004 -26130/g' Mode1AM.lin
```

6.4 PT Network Design

The overall PT application process is shown in Figure 6-12. Each of these steps will be described in more detail in the following sections.

Figure 6-12: Overall PT Process Described in this Memorandum

6.4.1 Create PT Network

The PT transit network contains MWCOG's highway links and nodes as well as transit-only links and nodes for Metrorail, LRT, commuter rail and other exclusive guideway routes (e.g., BRT and streetcar lines typically coded as mode 10 in the current modeling process). The PT network should also contain park-n-ride (PNR) lot nodes for all transit modes that permit park-n-ride access. PNR nodes should be connected to the highway network using a PNR driveway link (mode 11) and to the corresponding station or transit stop using a PNR walkway link (mode 15). PNR driveway links and PNR walkway links are used by PT to find park-n-ride access paths to these stations from any location in the PT network.

The PT network should also include kiss-n-ride (KNR) driveway links (mode 14) connecting the highway network to those transit stations which are not located on highway network links (e.g., Metrorail, LRT, commuter rail, and bus stops which support bus PNR locations). PT uses KNR driveway links (mode 14) to build kiss-n-ride and walk-access paths to these stations. Mode 14 links should be coded with appropriate drive and walk travel times to accommodate both access options. (Note: the current modeling process does not build kiss-n-ride access paths to commuter rail stations.) It is also desirable to add special walk access links (mode 12) to the PT network in order to create better walk accessibility for those transit stops located in regions where highway link density is sparse (e.g. suburban areas).

Table 6-1 defines the supplemental highway links that should be added to the basic highway network in order to support PT path building. Since the mode numbers assigned to these links are similar to the current mode numbers used in the TRNBUILD process, Table 6-1 also compares the definition of these links between these two processes. AECOM considered the possibility of removing mode codes from these links in the PT process. It may be desirable to include these links in the highway network as collectors, locals or special driveways, rather than think of them as transit-only links.

Table 6-1: Non-Transit Travel Modes: Current TRNBUILD Convention and Proposed PT Convention

Mode Code	TRNBUILD		PT	
	Non-Transit modes	**Mode**	**Non-Transit modes**	**Mode**
11	Non-transit drive access to transit (PNR or KNR): Each drive-access link represents the complete auto path from TAZ to PNR lot or the rail station	drive	PNR driveway (for drive access to PNR lot): Could be coded as a collector (FTYPE=4)	drive
12	Transfer between rail and non-rail transit (are not used for bus-to-bus transfers)	walk	Special walk-access links connecting zone centroids to transit stops where highway network does not have enough detail.	walk
13	Sidewalks	walk	Special walk links directly connecting transit stops of one mode or different modes together, “direct-walk transfer link,” e.g., pedestrian tunnels.	walk
14	(Unused)		KNR driveway (for drive access to rail stations): Could be coded as a collector (FTYPE=4). Also allows walk access to station.	drive & walk
15	Transfer between PNR lot and transit stop node (rail station or bus stop)	walk	PNR walkway: Transfer between PNR lot and transit stop node (rail station or bus stop)	walk
16	Walk access to/egress from transit	Walk	(Unused)	

Figure 6-13 shows the typical network design for transit stops that are not located on highway network links. It should be noted that PNR driveway (mode 11) and PNR walkway (mode 15) links are only necessary for transit stops that permit park-n-ride access. Figure 6-13 also shows a pedestrian tunnel link (mode 13) connecting two Metrorail stations. The pedestrian tunnel travel time will be compared to the travel time of the alternative walk paths and the best path will be chosen by the path builder. Special walk-access links (mode 12) are also necessary if the station is not reasonably accessible from the highway network.

Figure 6-13: PT Network Design for Transit Stops Off the Highway Network

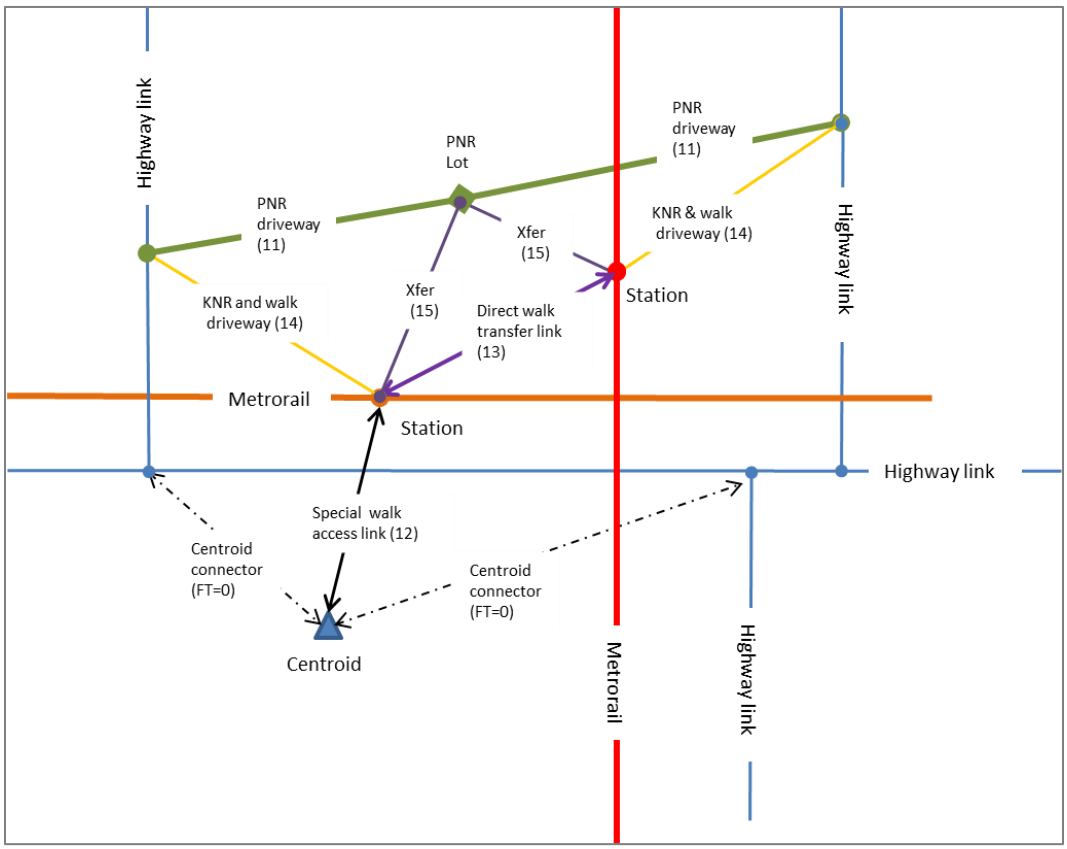


Figure 6-14 shows a sample script that prepares the PT network using input files from MWCOG's highway and transit networks. The process includes three calls to the NETWORK program. The first NETWORK call removes highway links ending with nodes in the 5000 series (Metrorail stations centroids) and 6000 series (commuter rail centroids) from the MWCOG highway network. These links will not be used by the PT network. The second and third calls to the NETWORK program simply merge the required transit links and nodes into the highway network to create a comprehensive PT network. This is done using two NETWORK calls because the program cannot handle more than ten "LINKI" files at once. This script performs the following steps:

1. Remove links ending with nodes in the 5000 series (Metrorail stations centroids) and 6000 series (commuter rail centroids) from the highway network;
2. Merge Metrorail, LRT, and commuter rail links and nodes into the highway network;
3. Merge PNR nodes for all modes into the highway network;
4. Merge new bus stops for bus PNR locations into the highway network;
5. Merge PNR transfer links (mode 15) into the highway network;
6. Merge PNR driveway (mode 11) links connecting the highway network to PNR lots;
7. Merge KNR/walk access (mode 14) links connecting the highway network to transit stops not located on highway links;

8. Calculate walk time, WKTIME, for all links using walk speed of 3 mph. WKTIME is set to 999 for transit-only links, PNR driveway (mode 11) links and those highway links that are not walkable (e.g. freeway, expressway, and ramps); and
9. Calculate transit time, TRNTIME, for Metrorail, LRT, and commuter rail links based on their coded speed and distance. TRNTIME for highway links is calculated based on the AM highway time (AMHTIME in minutes, defined in V2.3_Highway_Build.s as "(DISTANCE/PPAMSPD)*60.00") and, for mode 14 links, is based on their coded distance and speed (25 mph).

Figure 6-14: Create Comprehensive PT Network

```
; Remove links ending at 5000 and 6000 nodes
; Add WKTIME to highway links
; Assign high WKTIME to freeway and expressways
; Calculate TRNTIME for all highway links based on AMHTIME

; First NETWORK Program Call
RUN PGM=NETWORK

  NETI = "COG_Inputs/zonehwy.net"                                          ; MWCOG zonehwy.net
  NETO = "Intermediate/zonehwy Rev.net"                                    ; Processed file1

  IF ((A > 5000 && A < 6000) || (B > 5000 && B < 6000)) delete
  IF ((A > 6000 && A < 7000) || (B > 6000 && B < 7000)) delete

  IF( FTYPE <> 1 && FTYPE <> 5 )                                  ;highway links and station-
highway connector links
    DISTANCE = DISTANCE
    SPEED = 3                          ;Walk link speeds
    TRANTIME = AMHTIME
    WKTIME = DISTANCE * 60.0 / SPEED
  ELSEIF( FTYPE = 1 || FTYPE = 5 || FTYPE = 6)   ;highway links and station-highway connector
links
    TRANTIME = AMHTIME
    WKTIME = 999
  ENDIF

ENDRUN


; Merge Met, LRT, com.rail links and nodes with highway network
; Merge mode 15 links (connecting PNR lots to stations) with highway network
; Merge PNR lot nodes for all modes with highway network
; Merge new bus stations supporting bus PNR lots with highway network
; Calculate TRNTIME for Met, LRT, com.rail links based on coded speed and distance
; Caculate WKTIME as well

; Second NETWORK Program Call
RUN PGM  = NETWORK
   NETI = "Intermediate\zonehwy rev.NET"          ;---- Processed file1 ----
   NETO = "Intermediate\PT_Stage1.NET"            ;---- Processed file2 ----


;---New Hybrid HWY Link for New bus station with PNR lots---
   LINKI[2] = "Inputs\New_Bus_Connection.lin",
        VAR = A,12-17, B,20-24, FTYPE,33-34, AMLIMIT,87-88, PMLIMIT,87-88, OPLIMIT,87-88,
DISTANCE,42-46, SPEED,65-69,
        AMLANE,108-109, PMLANE,108-109, OPLANE,108-109, AMHTIME,123-126, REV=2   ; New highway
links (extensions) to create a bus stop for bus PNR lots

   LINKI[3] = "Inputs\Met Link.lin",
        VAR = A,12-17, B,19-24, MODES,32-34, DISTANCE,41-45, SPEED,64-69, REV=2   ; Metrorail
links

   LINKI[4] = "Inputs\Com_Link.lin",
       VAR = A,12-17, B,19-24, MODES,32-34, DISTANCE,41-53, SPEED,64-69, REV=2   ; Commuter rail
links
```

```
   LINKI[5] = "Inputs\LRT Link.lin",
        VAR = A,12-17, B,19-24, MODES,32-34, DISTANCE,41-45, SPEED,64-69, REV=2   ; LRT  links

   LINKI[6] = "Inputs\metampnr.lin",
        VAR = A,12-16, B,18-22, MODES,38-39, DISTANCE,46-51                        ; Transfer
links between Metrorail stations and parking lots

   LINKI[7] = "Inputs\comampnr.lin",
       VAR = A,12-16, B,18-22, MODES,38-39, DISTANCE,46-51                       ; Transfer links
between Commuter rail stations and parking lots

   LINKI[8] = "Inputs\lrtampnr.lin",
        VAR = A,12-16, B,18-22, MODES,38-39, DISTANCE,46-51                        ; Transfer
links between LRT stations and parking lots

   LINKI[9] = "Inputs\busampnr.lin",
        VAR = A,12-16, B,18-22, MODES,38-39, DISTANCE,46-51                        ; Transfer
links between Bus stations and parking lots


;---- transit only nodes ----
   NODEI[2] = "Inputs\Met_Node.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; Metrorail
station nodes
   NODEI[3] = "Inputs\LRT_Node.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; LRT station
nodes
   NODEI[4] = "Inputs\Com_Node.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; Commuter rail
station nodes

;---- park-n-ride nodes ----
   NODEI[5] = "Inputs\Met_PNRN.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; Metrorail
parking lot nodes
   NODEI[6] = "Inputs\LRT_PNRN.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; LRT parking
lot nodes
   NODEI[7] = "Inputs\Com_PNRN.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; Commuter rail
parking lot nodes
   NODEI[8] = "Inputs\Bus_PNRN.lin",
       VAR = N,9-14, X,18-27, Y,31-40                                              ; Bus parking
lot nodes

;---- New Highway Nodes for bus stops  ----
   NODEI[9] = "Inputs\New_Bus_Nodes.lin",
        VAR = N,9-14, X,18-27, Y,31-40                                             ; New Bus
station nodes

;---- link processing ----

   MERGE RECORD=TRUE

   PHASE = LINKMERGE

         IF ((MODES = 3 || MODES = 4 || MODES = 5 ) && SPEED >0  )            ;Metro/LRT/Commuter
rail links
            DISTANCE = DISTANCE / 100.0
            TRANTIME = DISTANCE * 60.0 / SPEED
            WKTIME = 999
         ELSEIF (MODES=15)                                    ;Transfer links PNR lot to stations
            DISTANCE = DISTANCE / 5280
            SPEED = 3
            TRANTIME = DISTANCE * 60.0 / SPEED
            WKTIME = DISTANCE * 60.0 / SPEED
        ENDIF
    ENDPHASE

ENDRUN
```

```
; Merge PNR transfer links to Bus, Met, LRT, com.rail Parking Lots (mode 11)
; Merge walk and KNR transfer links to Bus, Met, LRT, com.rail stations from highway network
(mode 14)
; Calculate TRNTIME and WKTIME for Mode 11 and 14 links

; Third NETWORK Program Call
RUN PGM  = NETWORK
   NETI = "Intermediate\PT_Stage1.NET"        ;---- Processed file2 ----
   NETO = "Inputs\PT Net.Net"                 ;---- PT network ----


   LINKI[2] = "Inputs\Bus_PNR_DR.lin", VAR = A,12-17, B,19-24, MODES,31-33, DISTANCE,65-69,
SPEED,42-46                                ; PNR transfer links between highway and Bus stations

   LINKI[3] = "Inputs\Met PNR DR.lin",
        VAR = A,12-17, B,19-24, MODES,31-33, DISTANCE,65-69, SPEED,42-46
; PNR transfer links between highway and Metrorail stations
   LINKI[4] = "Inputs\Met_KNR_DR_WK.lin",
        VAR = A,12-17, B,19-23, MODES,30-32, DISTANCE,58-63, SPEED,40-42, REV=2
; KNR transfer links between highway and Metrorail stations


   LINKI[5] = "Inputs\Com_PNR_DR.lin", VAR = A,12-17, B,19-24, MODES,31-33, DISTANCE,65-69,
SPEED,42-46                  ; PNR transfer links between highway and Commuter rail stations
   LINKI[6] = "Inputs\Com_KNR_DR_WK.lin",
        VAR = A,12-17, B,19-23, MODES,30-32, DISTANCE,58-63, SPEED,40-42, REV=2
; KNR transfer links between highway and Commuter rail stations

   LINKI[7] = "Inputs\LRT_PNR_DR.lin",
        VAR = A,12-17, B,19-24, MODES,31-33, DISTANCE,65-69, SPEED,42-46
; PNR transfer links between highway and LRT stations
   LINKI[8] = "Inputs\LRT KNR DR WK.lin",
        VAR = A,12-17, B,19-23, MODES,30-32, DISTANCE,58-63, SPEED,40-42, REV=2
; KNR transfer links between highway and LRT stations

;---- link processing ----

   MERGE RECORD=TRUE

   PHASE = LINKMERGE

        IF (MODES=14)                                    ;Metrorail station-PNR connector links
            DISTANCE = DISTANCE
            SPEED = 25
            TRANTIME = DISTANCE * 60.0 / SPEED
            WKTIME = DISTANCE * 60.0 / 3.0
        ELSEIF (MODES=11)                                ;Metrorail station-PNR connector links
            DISTANCE = DISTANCE
            SPEED = 25
            TRANTIME = DISTANCE * 60.0 / SPEED
            WKTIME = 999
        ENDIF
   ENDPHASE

ENDRUN
```

6.4.2 Combine Transit Lines

For some alternatives or analysis years, MWCOG's transit network does not include all 10 transit modes. For example, there are no LRT or streetcar transit lines in the 2010 network. Unlike TRNBUILD, the PT program does not accept transit line files with no records. In order to address this issue, transit lines are combined by mode using a pilot program script which uses the Windows system command 'COPY'. Figure 6-15 shows a sample script that combines transit lines by mode for the peak period. The commuter rail mode (mode 4) was not combined with any other modes because the MWCOG model

splits transit trips into four line-haul modes and three of these modes (bus only, Metrorail only and bus+Metrorail) do not include commuter rail. Therefore, the commuter rail line files should be kept separate and excluded from the processing for those line-haul modes. Because the Mode4AM.TB and Mode4OP.TB files are not empty for regular MWCOG runs, this will not cause a problem for the PT processing.

Figure 6-15: Combine Transit Lines by Mode

```
;; ************* Combine Transit Lines ***************

; combine bus/streetcar lines 1,2,6,7,8,9,10 (non-rail modes) to AM_Bus_Lines.lin

*copy "Inputs\Mode1AM.lin" + "Inputs\Mode2AM.lin" + "Inputs\Mode6AM.lin" + "Inputs\Mode7AM.lin" +
"Inputs\Mode8AM.lin" + "Inputs\Mode9AM.lin" + "Inputs\Mode10AM.lin"  "Inputs\AM_Bus_Lines.lin"

; combine Metro and LRT 3,5 to AM_MR_LRT_Lines.lin

*copy "Inputs\Mode3AM.lin" + "Inputs\Mode5AM.lin"   "Inputs\AM_MR_LRT_Lines.lin"
```

6.4.3 Generate Non-Transit Legs

PT creates transit paths between origins and destinations by alternating between non-transit and transit legs. PT's non-transit legs are virtual links connecting zone centroids to transit stops (access legs), transit stops to other transit stops (transfer legs), and transit stops to zone centroids (egress legs). PT's GENERATE statement builds actual paths between origin nodes defined by "FROMNODE" and transit stops or destination nodes defined by "TONODE", and saves virtual links connecting the origin and destination nodes with a user-defined mode and the actual path's cost. The total number of virtual legs created by PT is a function of the maximum number of paths allowed ("MAXNTLEGS"), the maximum path cost ("MAXCOST"), and the "SLACK" value. PT takes the SLACK value and adds it to the cost of the best path found for each interchange and drops paths with higher cost.

Figure 6-16 shows the actual path and PT's virtual non-transit legs from a zone centroid to a Metrorail station for the walk access mode. The virtual non-transit leg can be composed of one or more underlying links or connectors. Figure 6-16 shows two virtual, non-transit (NT), walk-access legs (dashed black lines): one to an Orange Line station and one to a Red Line station. The first (to the Orange Line station) is composed of only one underlying connector (the special walk-access link, mode 12, shown with a green dashed line). The second (to the Red Line station) is composed of four underlying links/connectors (also shown with green dashed lines): a centroid connector, two highway links, and a KNR walkway (mode 14). Both of these walk-access legs (black, dashed lines) are assigned the mode code of 20 (see Table 6-2).

Figure 6-16: Virtual PT Non-Transit Walk-Access Leg

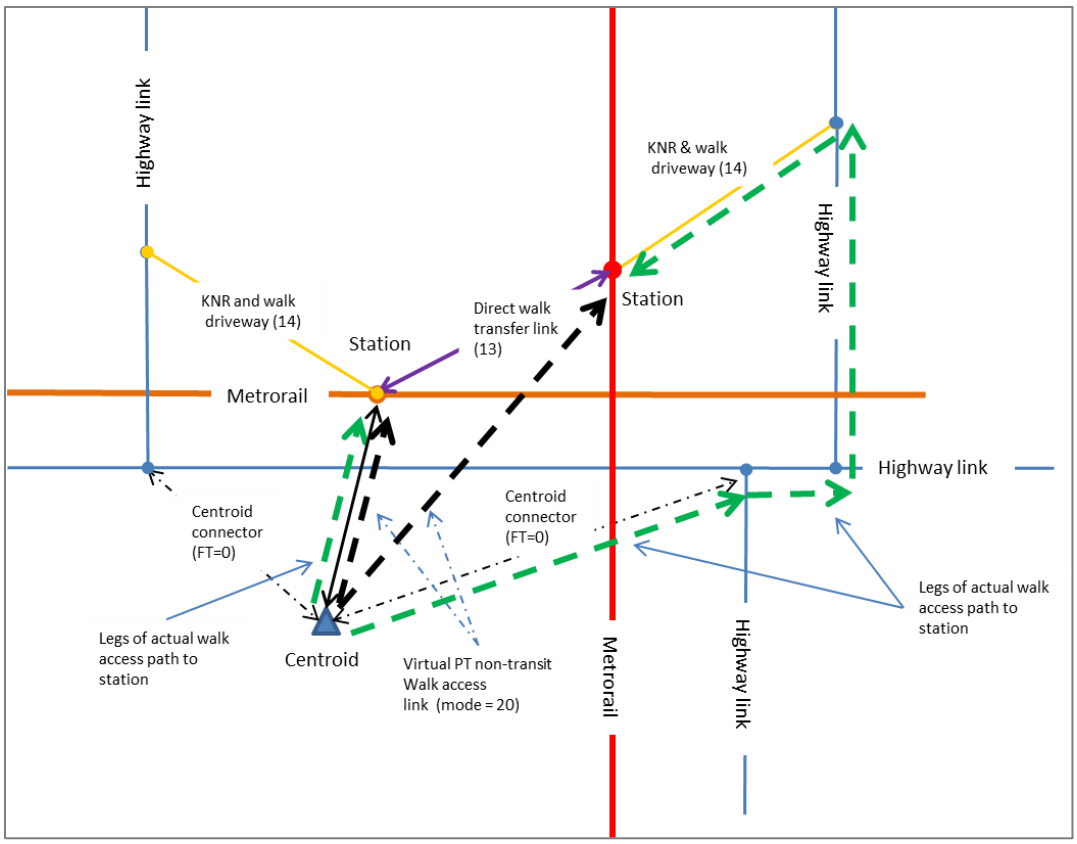
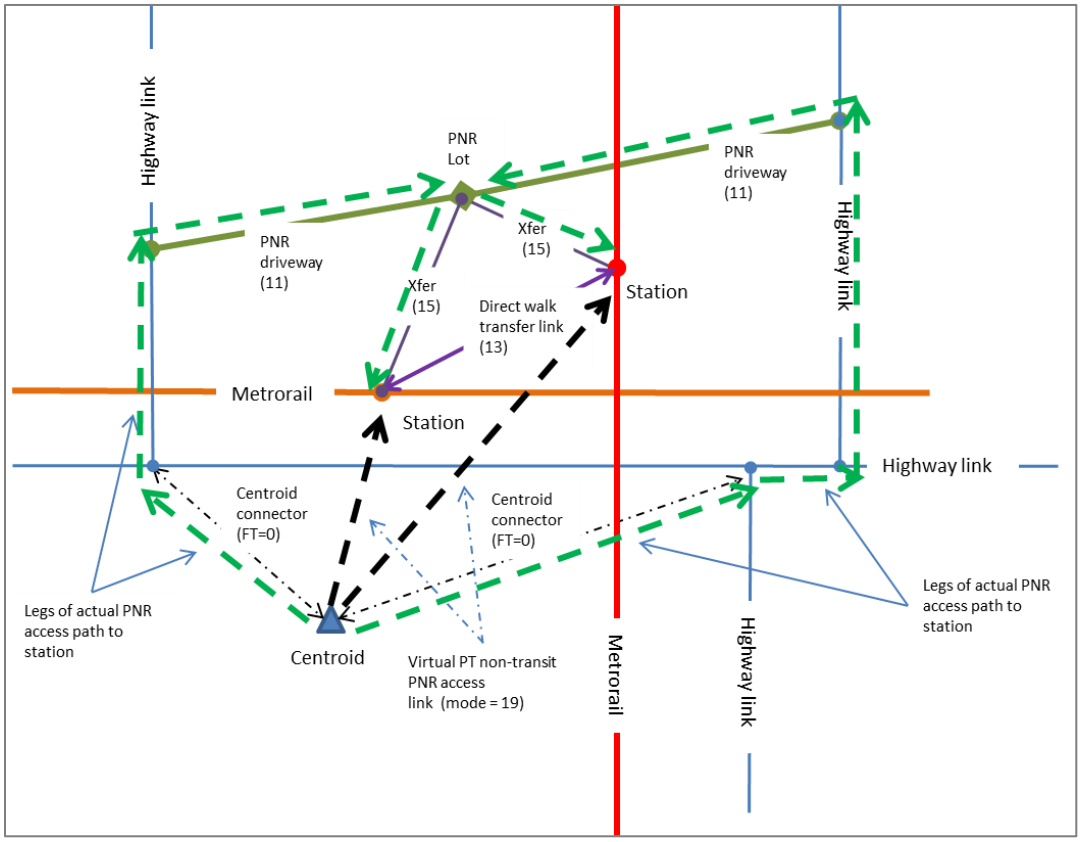


Figure 6-17 shows two virtual, non-transit (NT), drive-access legs (dashed black lines): one to an Orange Line station and one to a Red Line station. The first (to the Orange Line station) is composed of four underlying connectors/links (green dashed lines): a centroid connector, a highway link, a PNR driveway (mode 11), and a PNR walkway (mode 15). The second (to the Red Line station) is composed of five underlying links/connectors (green, dashed lines): a centroid connector, two highway links, a PNR driveway (mode 11), and a PNR walkway (mode 15). Both of these drive-access legs (black, dashed lines) are assigned the mode code of 19 (see Table 6-3).

Figure 6-17: Virtual PT Non-Transit PNR-Access Leg



AECOM proposed using 20 different mode codes for non-transit legs to fully capture how transit users move from one point to another. Only two of the 20 are used for drive-access legs and the rest are used for walk-access/egress legs. Table 6-2 mode codes generated by the PT conversion process based on the origin (production) and destination (attraction) of non-transit, walk-access/egress links. All of the walk-access/egress legs shown in Table 6-2, except mode 20, are two-way legs and connect transit stops. Mode 20 connects zone centroids to transit stops of different transit modes. Two sets of mode-20 legs are generated; one is walk-access legs and the other is walk-egress legs. This distinction will facilitate the removal of walk-access legs during kiss-n-ride and park-n-ride transit skimming or assignment procedures. AECOM investigated the possibility of generating one set of walk access and egress legs and removing walk-access legs during kiss-n-ride and park-n-ride transit skimming or assignment procedures, but felt the added complexity was not warranted.

Modes 22, 33, 44, and 55 in Table 6-2 represent direct walk transfers between transit stops of the same mode when stops are not coterminous (stops do not share a common stop node). It should be noted that if stops are coterminous, then a transfer can be made, but PT does not assign any mode to that transfer movement. In other cases, the first digit represents the mode number the trip alighted from and the second digit represents the mode number the trip is accessing (i.e. bus is mode 2, Metrorail is mode 3, etc.). The mode numbers are automatically assigned to PT non-transit legs by PT's GENERATE command.

Table 6-2: Mode Codes Associated with Virtual PT Non-Transit Walk Access/Egress and Transfer Legs

From	To				
	Zone Centroid	Bus	Metrorail	Commuter Rail	LRT
Zone Centroid	-	20	20	20	20
Bus	20	22	23	24	25
Metro	20	32	33	34	35
Commuter Rail	20	42	43	44	45
LRT	20	52	53	54	55

Table 6-3 shows non-transit, drive-access legs generated by the PT conversion process. Two sets of drive access-legs are created; one for kiss-n-ride and another for park-n-ride. Drive-access legs are one-way and connect zone centroids to transit stops. These legs cannot be used as egress legs. This implies that all transit path building and assignment routings will model transit trips in production-attraction format. The mode codes indicated in Table 6-2 and Table 6-3 are specified in the Cube Voyager script using the statement "NTLEGMODE =".

Table 6-3: Mode Codes Associated with Virtual PT Non-Transit Drive-Access Legs

From	Mode	To			
		Bus	Metrorail	Commuter Rail	LRT
Zone Centroid	Drive (Kiss-n-Ride)	18	18	18	18
Zone Centroid	Drive (Park-n-Ride)	19	19	19	19

6.4.4 PT Path Building Inputs

PT uses a system file and a factors file to manage different modes. The system file contains public transport system data including mode numbers, operators, and wait curves. This file is introduced to the model using the SYSTEMI key. Figure 6-18 shows the system file used in this study. The factors file specifies the generalized cost factors and control information for the route-enumeration, route-evaluation, loading, and loading-analyses processes. The factors file is introduced to the model using the FACTORI key. It is possible to define different factor files for different user classes.

Figure 6-19 shows the factors file used in this study. Since these two files are input to the modeling process, they are usually placed in the input folder.

Figure 6-18: PT System File - TSYSD.PTS

```
;;************<PT><SYSTEM> file **************;;

;;********** Transit Lines ******************
MODE NUMBER=1 LONGNAME="Local Metrobus" NAME="BUS"
MODE NUMBER=2 LONGNAME="Express Metrobus" NAME="XBUS"
MODE NUMBER=3 LONGNAME="Metrorail" NAME="METRO"
MODE NUMBER=4 LONGNAME="Commuter Rail" NAME="RAIL"
MODE NUMBER=5 LONGNAME="Light Rail Transit" NAME="LRT"
MODE NUMBER=6 LONGNAME="Other Local Bus in WMATA Service Area" NAME="BUS1"
MODE NUMBER=7 LONGNAME="Other Express Bus in WMATA Service Area" NAME=XBUS1"
MODE NUMBER=8 LONGNAME="Other Local Bus beyond WMATA Service Area" NAME="BUS2"
MODE NUMBER=9 LONGNAME="Other Express Bus beyond WMATA Service Area" NAME="XBUS2"
MODE NUMBER=10 LONGNAME="Bus Rapid Transit and Streetcar" NAME="BRT"
```

```
; Physical links added to the highway network
MODE NUMBER=11 LONGNAME="PNR Driveway" NAME="PNR DRIVEWAY"
MODE NUMBER=12 LONGNAME="Special Walk Connector Links" NAME="WKCON"
MODE NUMBER=13 LONGNAME="Special Walk Transfer Links" NAME="WKXFERCON"
MODE NUMBER=14 LONGNAME="KNR Driveway" NAME="KNR_DRIVEWAY"
MODE NUMBER=15 LONGNAME="Walk Transfer Link from PNR Lot to Transit Station" NAME="PNR_WALKWAY"
MODE NUMBER=16  ;Unused
MODE NUMBER=17  ;Unused

; Non-Transit legs generated by PT

; Non-Transit KNR and PNR legs generated by PT
MODE NUMBER=18  LONGNAME="KNR Drive Access Connectors" NAME="KNRCON"
MODE NUMBER=19  LONGNAME="PNR Drive Access Connectors" NAME="PNRCON"

; Non-Transit Walk legs generated by PT
MODE NUMBER=20 LONGNAME="Walk access from zone centroid to bus_Met_Com_Lrt stations"
MODE NUMBER=22 LONGNAME="Walk transfer link between bus and bus"
MODE NUMBER=23 LONGNAME="Walk transfer link between bus and Metro"
MODE NUMBER=24 LONGNAME="Walk transfer link between bus and commuter rail"
MODE NUMBER=25 LONGNAME="Walk transfer link between bus and LRT"
MODE NUMBER=33 LONGNAME="Walk transfer link between Metro stations"
MODE NUMBER=34 LONGNAME="Walk transfer link between Metro and commuter rail"
MODE NUMBER=35 LONGNAME="Walk transfer link between Metro and LRT"
MODE NUMBER=44 LONGNAME="Walk transfer link between commuter rail stations"
MODE NUMBER=45 LONGNAME="Walk transfer link between Commuter and LRT"
MODE NUMBER=55 LONGNAME="Walk transfer link between LRT stations"

; Operator numbers
OPERATOR NUMBER=1 LONGNAME="WMATA" NAME="WMATA"

; Wait curve definitions
WAITCRVDEF NUMBER=1 LONGNAME="Initial and Transfer Wait" NAME="InitXferWait" ,
           CURVE=0-0,1-0.5,4-2,60-30,120-60
```

Figure 6-19: PT Factors File – AM_TRN.FAC

```
;;************ PT Factors File for Peak Period *************;;

;Global Settings
 BESTPATHONLY=T  ;Evaluation process identifies a single best path
 MAXFERS=3
 SERVICEMODEL=FREQUENCY
 RECOSTMAX=650.0  ;Routes with costs above $6.50 are not enumerated or evaluated
 FREQBYMODE=T

;Fare and Wait Times
 FARESYSTEM=1, MODE=1
 FARESYSTEM=2, MODE=2

;Runfactors
 RUNFACTOR[1]   = 1.00 ;--- in-vehicle time ----
 RUNFACTOR[2]   = 1.00
 RUNFACTOR[3]   = 1.00
 RUNFACTOR[4]   = 1.00
 RUNFACTOR[5]   = 1.00
 RUNFACTOR[6]   = 1.00
 RUNFACTOR[7]   = 1.00
 RUNFACTOR[8]   = 1.00
 RUNFACTOR[9]   = 1.00
 RUNFACTOR[10]  = 1.00
 RUNFACTOR[11]  = 1.00 ;--- park-&-ride transfer time ----
 RUNFACTOR[12]  = 1.00  ;--- walk access time  ----
 RUNFACTOR[13]  = 1.00  ;--- Walk transfer time  ----
 RUNFACTOR[14]  = 1.00 ;--- KNR Access time  ----
 RUNFACTOR[15]  = 1.00 ;--- park-&-ride transfer time ----
 ;RUNFACTOR[16]  ;Unused
 ;RUNFACTOR[17]  ;Unused
```

```
 RUNFACTOR[18]  = 1.00 ;--- kiss-&-ride non-transit leg time  ----
 RUNFACTOR[19]  = 1.00 ;--- park-&-ride non-transit leg time ----

 RUNFACTOR[20]  = 1.50 ;--- walk access from zone centroid to bus Met Com Lrt stations----

 RUNFACTOR[22]  = 1.00 ;--- walk transfer link between bus and bus ----
 RUNFACTOR[23]  = 1.00 ;--- walk transfer link between bus and Metro ----
 RUNFACTOR[24]  = 1.00 ;--- walk transfer link between bus and commuter rail ----
 RUNFACTOR[25]  = 1.00 ;--- walk transfer link between bus and LRT ----

 RUNFACTOR[33]  = 1.00 ;--- walk transfer link between Metro stations ----
 RUNFACTOR[34]  = 1.00 ;--- walk transfer link between Metro and commuter rail ----
 RUNFACTOR[35]  = 1.00 ;--- walk transfer link between Metro and LRT ----

 RUNFACTOR[44]  = 1.00 ;--- walk transfer link between commuter rail stations  ----
 RUNFACTOR[45]  = 1.00 ;--- walk transfer link between Commuter and LRT ----

 RUNFACTOR[55]  = 1.00 ;--- walk transfer link between LRT stations ----


;Boarding and Transfer Penalties
 BRDPEN=10*5.3
 XFERPEN=1.6,  from=1-10, to=1-10
```

Figure 9-17 in the Appendix shows the script used to generate non-transit walk access/egress legs between zone centroids and transit stops. Figure 9-18 shows the script used to generate non-transit walk transfer legs between transit stops of the same or different modes.

Figure 6-20 and Figure 6-21 show the script used to generate non-transit drive access legs from zone centroids to transit stops for kiss-n-ride and park-n-ride paths respectively.

Figure 6-20: PT Script to Create Non-Transit Kiss-n-Ride Legs from Zone Centroids to Transit Stops

```
;; ********************************
;; ***** Create kiss-n-Ride links
;; ********************************

;; Step 1 **** Create KNR links from zone Centroids to transit stops except commuter rail ******

RUN PGM=PUBLIC TRANSPORT  PRNFILE="Create PT KNR Acc.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI LINEI[2] = "Inputs\AM MR LRT Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "KNR.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.TRANTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 14,
      COST=li.TRANTIME,
      MAXCOST=10*15,
      FROMNODE=1-3675,
      TONODE=8001-8098, 10001-10098, 15001-15135,
      DIRECTION = 1,
      EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 10,
      SLACK = 10*15,
```

```
      LIST = Y
  ENDPROCESS

ENDRUN
```

Figure 6-21: PT Script to Create Non-Transit Park-n-n--Ride Legs from Zone Centroids to Transit Stops

```
;; ********************************
;; ***** Create Park-n-Ride links
;; ********************************
 ;; Step 1 **** Create PNR links from zone Centroids to transit stops with parking lots ********

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Dr_Acc_PNR.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
  FILEI LINEI[3] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "PNR.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.TRANTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 11,
      COST=li.TRANTIME,
      MAXCOST=10*30,
      FROMNODE=1-3675,
      TONODE=8001-8098, 9001-9098, 10001-10098, 15001-15135,
          DIRECTION = 1,
          EXCLUDELINK = (li.MODES = 1-10, 14)|| (li.AMLIMIT = 9)||(li.PMLIMIT = 9)||(li.OPLIMIT =
9),
      MAXNTLEGS = 10,
      SLACK = 10*15,
      LIST = Y
  ENDPROCESS

ENDRUN
```

6.4.5 Combine Non-Transit Legs

There is a high probability that PT will not generate any non-transit walk legs for some of the combinations mentioned in Figure 6-16 and Figure 9-18. There are two common reasons for not having any non-transit walk access, egress or transfer legs for some of these combinations. The first is that the network does not include the specific mode (e.g., LRT). The second is that the cost of non-transit walk legs is higher than the MAXCOST defined in the GENERATE statement.

Since the PT program does not accept non-transit leg files with no records, it is necessary to combine these leg files for transit skimming and assignment. Figure 6-22 shows a sample script that combines non-transit legs for the peak period using a Pilot program script.

Figure 6-22. Combine Non-Transit Walk Legs

```
;; ************* Combine Non-Transit Legs ***************

;combine walk access legs (Walk_Acc_Legs.LEG is the merged file output)
```

```
*copy "Acc_WK2Bus.LEG" + "Acc_WK2MetLRT.LEG" + "Acc_WK2Com.LEG" "Walk_Acc_Legs.LEG"

;combine walk Egress legs (Walk_Egr_Legs.LEG is the merged file output)
*copy "Egr WK2Bus.LEG" + "Egr WK2MetLRT.LEG" + "Egr WK2Com.LEG" "Walk Egr Legs.LEG"

;combine walk transfer legs (Walk_Transfer_Legs.LEG is the merged file output)
*copy "WK_Bus2Bus.LEG" + "WK_Bus2Met.LEG" + "WK_Bus2LRT.LEG" + "WK_Bus2Com.LEG" +
"WK Met2Met.LEG" + "WK Met2LRT.LEG"+ "WK Met2Com.LEG" + "WK LRT2LRT.LEG" + "WK LRT2Com.LEG" +
"WK_Com2Com.LEG" "Walk_Transfer_Legs.LEG"
```

6.5 PT Transit Skimming

The PT transit skimming script is very similar for all access modes (walk, kiss-n-ride, and park-n-ride) given a line-haul mode (bus-only, Metrorail-only, commuter rail, and bus-Metrorail) and modeling time period (peak or off-peak). The main difference between the PT transit skimming scripts for various access modes is the inclusion or exclusion of non-transit access legs. For example, walk access legs are not included in the kiss-n-ride and park-n-ride transit skims. Also, park-n-ride drive access legs replace kiss-n-ride drive access legs in the park-n-ride transit skim script.

The input files to the PT transit skim script are the PT network, factors file, system file, transit route files, and PT non-transit leg files. The main output of the program is the transit skim matrix saving parameters defined under PHASE=SKIMIJ. These parameters will be used by the mode choice model in later steps to estimate mode shares. The PT skimming process also outputs a network file (*.NET) and a route file (*.RTE) that are used by Cube to build and display transit paths on screen. This feature is of interest to MWCOG and is not available in TRNBUILD. Figure 6-23 shows the PT transit skimming script for peak period bus-Metrorail line-haul paths using the walk access mode. This script includes a “TRACE” statement that saves the sequence of transit and non-transit legs between origins and destinations included in a “TRACEI” and “TRACEJ” statement.

Figure 6-23: PT Transit Skim for Walk to Bus-Metrorail during Peak Period

```
RUN PGM = PUBLIC TRANSPORT; MSG = ' Bus_Metrorail AM WK transit paths and skims'

FILEI NETI = "Inputs\PT_NET.NET"

FILEO MATO[1] = "AM_BM_WK.SKM", MO=1-18,
  NAME = IVLB, IVXB, IVMT, IVCR, IVNRM, IVNBM, INWT, XFERWT, WACCT, OWLKT, XFERP, BRDS,
  DACCT, DACCD, PRKT, PRKC, MRD, LRTD

FILEO ROUTEO[1] = "AM_BM_WK.RTE", TRACEI = 1450-1455, 1800-1805,  TRACEJ = 1450-1455
FILEO REPORTO = "PT_Skims_AM_BM_WK.PRN"
FILEO NETO = "AM_BM_WK.NET"
FILEO NTLEGO ="NTLEG AM BM WK.NT"

;Input ModeAccess specific factors
FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
FILEI SYSTEMI = "Inputs\TSYSD.PTS"

;---- Non-Transit legs  ----
FILEI NTLEGI[1] = "Walk_Acc_Legs.LEG"
FILEI NTLEGI[2] = "Walk_Egr_Legs.LEG"
FILEI NTLEGI[3] = "Walk_Transfer_Legs.LEG"
;FILEI NTLEGI[4] = "KNR.LEG"
;FILEI NTLEGI[5] = "PNR.LEG"

;---- Transit modes (1 t0 10)  ----
FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
;FILEI LINEI[3] = "Inputs\Mode4AM.lin"
```

```
zonemsg=50
; OVERALL PARAMETERS OF RUN
PARAMETERS FARE=N, HDWAYPERIOD=1,NOROUTEERRS=999999,
           NOROUTEMSGS=999999,SKIPBADLINES=Y,
          TRANTIME=LI.TRANTIME

REPORT LINES=T
PROCESS PHASE=LINKREAD
 ;LW.AMHTIME=LI.AMHTIME
 ;LW.WALKTIME=(LI.DISTANCE/3*60)
 ;LW.WALKDISTANCE=LI.DISTANCE
 ;LW.DISTANCE=LI.DISTANCE
ENDPROCESS

PROCESS PHASE=DATAPREP
    GENERATE READNTLEGI=1
        GENERATE READNTLEGI=2
        GENERATE READNTLEGI=3
        ;GENERATE READNTLEGI=4
        ;GENERATE READNTLEGI=5
ENDPROCESS

PROCESS PHASE=SKIMIJ
;---- specify output skims ----

 MW[1] = TIMEA(0,1,6,8),                          ;---- ivt-local bus    (0.01 min)
 MW[2] = TIMEA(0,2,7,9),                         ;---- ivt-exp bus      (0.01 min)
 MW[3] = TIMEA(0,3),                             ;---- ivt-metrorail    (0.01 min)
 MW[4] = TIMEA(0,4),                             ;---- ivt-commuter rail(0.01 min)
 MW[5] = TIMEA(0,5),                             ;---- ivt-new rail mode(0.01 min)
 MW[6] = TIMEA(0,10),                            ;---- ivt-new bus mode (0.01 min)
 MW[7] = IWAITA(0),                              ;---- ini.wait time    (0.01 min)
 MW[8] = XWAITA(0),                              ;---- xfr wait time    (0.01 min)
 MW[9] = TIMEA(0,20)/2.0,         ;---- walk acc time    (0.01 min)
 MW[10] = (TIMEA(0,22) + TIMEA(0,23) + TIMEA(0,24) + TIMEA(0,25) +TIMEA(0,33) + TIMEA(0,34) +
TIMEA(0,35) + TIMEA(0,44) + TIMEA(0,45) + TIMEA(0,55) )/2 ,        ;---- other walk time  (0.01
min)
 MW[11] = XFERPENA(0),                           ;---- added xfer time  (0.01 min)
 MW[12] = BRDINGS(0),                            ;---- boardings        (1+)
 MW[13] = TIMEA(0,19),                           ;---- drv acc time     (0.01 min)
 MW[14] = DIST(0,19),                            ;---- drv acc distance (0.01 mile)
 MW[15] = TIMEA(0,15),                           ;---- pnr impedance    (0.01 min)
 MW[16] = DIST(0,15),                            ;---- pnr cost         (cents)
 MW[17] = DIST(0,3),                             ;---- metrorail distance
 MW[18] = DIST(0,5)                              ;---- light rail distance

ENDPROCESS

ENDRUN
```

Figure 6-24 shows a section of a trace report generated by the transit skim process for transit paths between zones 1451 and1450 and zones 1801 and 1450. For each pair of origin and destination zones, the report lists transit or non-transit modes taken for traveling between consecutive nodes, wait time, actual travel time, boarding and transfer penalties, perceived time, distance and total travel time. It also shows the names of the transit lines included in each transit leg of a transit path. A summary of total travel time, distance traveled, wait time, and transfer wait time for all transit and non-transit modes shaping the transit path is reported at the bottom of each trace report for a given origin and destination pair. The print file of transit skimming process (e.g. “PT_Skims_AM_BM_WK.PRN”) includes a complete list of trace results.

For the first example shown in Figure 6-24 (zone 1451 to zone 1450), the first leg of the transit path is a walk access leg (mode 20) followed by a bus leg (mode 6 bus line ART53AW), and the last leg is a walk egress leg (mode 20).

The second example shown (zone 1801 to zone 1450) also includes a walk access leg (mode 20), a bus leg (mode 6), a walk transfer leg between bus and Metrorail routes (mode 23), a Metrorail leg (mode 3), a walk transfer leg between Metrorail and bus routes (mode 32), a bus leg (mode 6), and the last leg is a walk egress leg (mode 20).

Figure 6-24: Sample of PT's Trace Report for Walk to Bus-Metrorail during Peak Period

```
REval Route(s) from Origin 1451 to Destination 1450

N:  1451 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
-> 30479   20      -   3.80   3.80      -   5.70   0.19   0.19
-> 30028    6   7.50   0.24  11.54   5.30  18.74   0.07   0.26  ART53AW(1.000)
->  1450   20      -   5.20  16.74      -  26.54   0.26   0.52
Mode  TimeA   Dist IWaitA XWaitA
   6   0.24   0.07   7.50   0.00
  20   9.00   0.45

REval Route(s) from Origin 1801 to Destination 1450

N:  1801 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
-> 36293   20      -   2.00   2.00      -   3.00   0.10   0.10
-> 35062    6   7.50   6.82  16.32   5.30  22.62   1.38   1.48  SGRN1(1.000)
->  8057   23      -  10.60  26.92      -  33.22   0.53   2.01
->  8061    3   3.00  14.57  44.49   6.90  57.69   9.48  11.49  WMORNA(1.000)
-> 30093   32      -   3.00  47.49      -  60.69   0.15  11.64
-> 30028    6  15.00   8.26  70.76   6.90  90.86   1.51  13.15  ART53AW(1.000)
->  1450   20      -   5.20  75.96      -  98.66   0.26  13.41
Mode  TimeA   Dist IWaitA XWaitA
   3  14.57   9.48   0.00   3.00
   6  15.08   2.89   7.50  15.00
  20   7.20   0.36
  23  13.60   0.68
```

Figure 6-25 shows the PT transit skimming script for peak period Bus-Metrorail line-haul paths using the kiss-n-ride access mode. Compared to the script shown in Figure 6-23, this script excludes walk access non-transit legs and includes kiss-n-ride drive access legs. This change requires changes in the "PHASE=DATAPREP" step as well. Apparently, the input and output file names are different.

Figure 6-25: PT Transit Skim for Kiss-n-Ride to Bus-Metrorail during Peak Period

```
RUN PGM = PUBLIC TRANSPORT; MSG = 'Bus_Metrorail AM KNR transit paths and skims'

FILEI NETI = "Inputs\PT_NET.NET"

FILEO MATO[1] = "AM BM KNR.SKM", MO=1-18,
  NAME = IVLB, IVXB, IVMT, IVCR, IVNRM, IVNBM, INWT, XFERWT, WACCT, OWLKT, XFERP, BRDS,
  DACCT, DACCD, PRKT, PRKC, MRD, LRTD

FILEO ROUTEO[1] = "AM_BM_KNR.RTE", TRACEI = 1300-1400,  TRACEJ = 20-500
FILEO REPORTO = "PT Skims AM BM KNR.PRN"
FILEO NETO = "AM_BM_KNR.NET"
FILEO NTLEGO ="NTLEG_AM_BM_KNR.NT"

;Input ModeAccess specific factors
FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
FILEI SYSTEMI = "Inputs\TSYSD.PTS"

;---- Non-Transit legs  ----
```

```
;FILEI NTLEGI[1] = "Walk_Acc_Legs.LEG"
FILEI NTLEGI[2] = "Walk Egr Legs.LEG"
FILEI NTLEGI[3] = "Walk_Transfer_Legs.LEG"
FILEI NTLEGI[4] = "KNR.LEG"
;FILEI NTLEGI[5] = "PNR.LEG"

;---- Transit modes (1 to 10 except commuter rail)  ----
 FILEI LINEI[1] = "Inputs\AM Bus Lines.lin"
 FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
;FILEI LINEI[3] = "Inputs\Mode4AM.lin"

zonemsg=50
; OVERALL PARAMETERS OF RUN
PARAMETERS FARE=N, HDWAYPERIOD=1,NOROUTEERRS=999999,
           NOROUTEMSGS=999999,SKIPBADLINES=Y,
          TRANTIME=LI.TRANTIME

REPORT LINES=T
PROCESS PHASE=LINKREAD
 ;LW.AMHTIME=LI.AMHTIME
 ;LW.WALKTIME=(LI.DISTANCE/3*60)
 ;LW.WALKDISTANCE=LI.DISTANCE
 ;LW.DISTANCE=LI.DISTANCE
ENDPROCESS

PROCESS PHASE=DATAPREP
    ;GENERATE READNTLEGI=1
       GENERATE READNTLEGI=2
       GENERATE READNTLEGI=3
       GENERATE READNTLEGI=4
       ;GENERATE READNTLEGI=5
ENDPROCESS

PROCESS PHASE=SKIMIJ
;---- specify output skims ----

 MW[1] = TIMEA(0,1,6,8),                          ;---- ivt-local bus    (0.01 min)
 MW[2] = TIMEA(0,2,7,9),                         ;---- ivt-exp bus      (0.01 min)
 MW[3] = TIMEA(0,3),                             ;---- ivt-metrorail    (0.01 min)
 MW[4] = TIMEA(0,4),                             ;---- ivt-commuter rail(0.01 min)
 MW[5] = TIMEA(0,5),                             ;---- ivt-new rail mode(0.01 min)
 MW[6] = TIMEA(0,10),                            ;---- ivt-new bus mode (0.01 min)
 MW[7] = IWAITA(0),                              ;---- ini.wait time    (0.01 min)
 MW[8] = XWAITA(0),                              ;---- xfr wait time    (0.01 min)
 MW[9] = TIMEA(0,20)/2.0,         ;---- walk acc time    (0.01 min)
 MW[10] = (TIMEA(0,22) + TIMEA(0,23) + TIMEA(0,24) + TIMEA(0,25) +TIMEA(0,33) + TIMEA(0,34) +
TIMEA(0,35) + TIMEA(0,44) + TIMEA(0,45) + TIMEA(0,55) )/2 ,        ;---- other walk time  (0.01
min)
 MW[11] = XFERPENA(0),                           ;---- added xfer time  (0.01 min)
 MW[12] = BRDINGS(0),                            ;---- boardings        (1+)
 MW[13] = TIMEA(0,18),                           ;---- drv acc time     (0.01 min)
 MW[14] = DIST(0,18),                            ;---- drv acc distance (0.01 mile)
 MW[15] = TIMEA(0,15),                           ;---- pnr impedance    (0.01 min)
 MW[16] = DIST(0,15),                            ;---- pnr cost         (cents)
 MW[17] = DIST(0,3),                             ;---- metrorail distance
 MW[18] = DIST(0,5)                              ;---- light rail distance

ENDPROCESS

ENDRUN
```

Figure 6-26 shows the trace report for the transit paths between zones 1336 and 68 and zones 1336 and 216. The first example shown includes a kiss-n-ride access leg (mode 18) followed by two bus legs (mode 1 and 9), a walk transfer leg between bus routes (mode 22), a bus leg (mode 1) and a walk egress leg (mode 20). Since PT transit paths alternate between non-transit and transit legs, it is not possible to have two non-transit legs follow each other. However, the example shown below shows that a bus leg

(mode 1) is taken between nodes 15007 and 27951 and followed by a bus leg (mode 9) between nodes 27951 and 20024. In this case, PT allows passengers to transfer between transit modes at a common node (e.g., 27951) without using a non-transit leg. Also note that PT combines transit lines that share paths to calculate combined headways for their shared segment of a transit path. In this case two bus lines have been reported (MT05BI (0.667), MT09AI(0.333)) for travel between nodes 27951 and 20024. The numbers reported in parentheses are the shares of each transit line used in calculating the combined headway and distributing transit trips assigned to this path.

The second example shown below follows the same pattern as the first example but includes a non-transit walk leg between bus and Metrorail routes (mode 23).

Figure 6-26: Sample of PT's Trace Report for Kiss-n-Ride to Bus-Metrorail during Peak Period

```
REval Route(s) from Origin 1336 to Destination 68

N:  1336 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
-> 15007   18     -  10.29  10.29      -  10.29   4.82   4.82
-> 27951    1  7.50   2.12  19.91   5.30  25.21   0.72   5.54  WMC11I(1.000)
-> 27951    -     -   0.00  19.91      -  25.21   0.00   5.54
-> 20024    9  5.00  41.12  66.03   6.90  78.23  14.49  20.03  MT05BI(0.667) MT09AI(0.333)
-> 20112   22     -   1.60  67.63      -  79.83   0.08  20.11
-> 20631    1  7.50  11.76  86.88   6.90 105.98   2.25  22.36  WM32I(0.500) WM36I(0.500)
->    68   20     -   4.40  91.28      - 112.58   0.22  22.58
Mode  TimeA   Dist IWaitA XWaitA
   1  13.88   2.97   7.50   7.50
   9  41.12  14.49   0.00   5.00
  18  10.29   4.82
  20   4.40   0.22
  22   1.60   0.08

REval Route(s) from Origin 1336 to Destination 216

N:  1336 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
-> 15007   18     -  10.29  10.29      -  10.29   4.82   4.82
-> 27951    1  7.50   2.12  19.91   5.30  25.21   0.72   5.54  WMC11I(1.000)
-> 27951    -     -   0.00  19.91      -  25.21   0.00   5.54
-> 20042    9  5.00  40.23  65.14   6.90  77.34  14.28  19.82  MT05BI(0.667) MT09AI(0.333)
->  8014   23     -   0.80  65.94      -  78.14   0.04  19.86
->  8019    3  1.00   9.53  76.47   6.90  95.57   3.81  23.67  WMREDA(0.667) WMREDB(0.333)
->   216   20     -  12.80  89.27      - 114.77   0.64  24.31
Mode  TimeA   Dist IWaitA XWaitA
   1   2.12   0.72   7.50   0.00
   3   9.53   3.81   0.00   1.00
   9  40.23  14.28   0.00   5.00
  18  10.29   4.82
  20  12.80   0.64
  23   0.80   0.04
```

Figure 6-27 shows a PT transit skimming script for peak period bus-Metrorail line-haul paths using the park-n-ride access mode. This script replaces kiss-n-ride drive access legs in Figure 6-25 with park-n-ride drive access legs. The "PHASE=DATAPREP" is also slightly different to reflect the change.

Figure 6-27: PT Transit Skim for Park-n-Ride to Bus-Metrorail during Peak Period

```
 RUN PGM = PUBLIC TRANSPORT; MSG = 'Bus_Metrorail AM PNR transit paths and skims'

FILEI NETI = "Inputs\PT NET.NET"

FILEO MATO[1] = "AM BM PNR.SKM", MO=1-18,
  NAME = IVLB, IVXB, IVMT, IVCR, IVNRM, IVNBM, INWT, XFERWT, WACCT, OWLKT, XFERP, BRDS,
```

```
  DACCT, DACCD, PRKT, PRKC, MRD, LRTD

FILEO ROUTEO[1] = "AM_BM_PNR.RTE", TRACEI = 1300-1455, 1700-1805,  TRACEJ = 20-500
FILEO REPORTO = "PT Skims AM BM PNR.PRN"
FILEO NETO = "AM_BM_PNR.NET"
FILEO NTLEGO ="NTLEG_AM_BM_PNR.NT"

;Input ModeAccess specific factors
FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
FILEI SYSTEMI = "Inputs\TSYSD.PTS"

;---- Non-Transit legs  ----
;FILEI NTLEGI[1] = "Walk_Acc_Legs.LEG"
FILEI NTLEGI[2] = "Walk Egr Legs.LEG"
FILEI NTLEGI[3] = "Walk_Transfer_Legs.LEG"
;FILEI NTLEGI[4] = "KNR.LEG"
FILEI NTLEGI[5] = "PNR.LEG"

;---- Transit modes (1 t0 10)  ----
FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
;FILEI LINEI[3] = "Inputs\Mode4AM.lin"


zonemsg=50
; OVERALL PARAMETERS OF RUN
PARAMETERS FARE=N, HDWAYPERIOD=1,NOROUTEERRS=999999,
           NOROUTEMSGS=999999,SKIPBADLINES=Y,
          TRANTIME=LI.TRANTIME

REPORT LINES=T
PROCESS PHASE=LINKREAD
 ;LW.AMHTIME=LI.AMHTIME
 ;LW.WALKTIME=(LI.DISTANCE/3*60)
 ;LW.WALKDISTANCE=LI.DISTANCE
 ;LW.DISTANCE=LI.DISTANCE
ENDPROCESS

PROCESS PHASE=DATAPREP
       ;GENERATE READNTLEGI=1
       GENERATE READNTLEGI=2
       GENERATE READNTLEGI=3
       ;GENERATE READNTLEGI=4
       GENERATE READNTLEGI=5
ENDPROCESS

PROCESS PHASE=SKIMIJ
;---- specify output skims ----

 MW[1] = TIMEA(0,1,6,8),                          ;---- ivt-local bus    (0.01 min)
 MW[2] = TIMEA(0,2,7,9),                        ;---- ivt-exp bus      (0.01 min)
 MW[3] = TIMEA(0,3),                            ;---- ivt-metrorail    (0.01 min)
 MW[4] = TIMEA(0,4),                            ;---- ivt-commuter rail(0.01 min)
 MW[5] = TIMEA(0,5),                            ;---- ivt-new rail mode(0.01 min)
 MW[6] = TIMEA(0,10),                           ;---- ivt-new bus mode (0.01 min)
 MW[7] = IWAITA(0),                             ;---- ini.wait time    (0.01 min)
 MW[8] = XWAITA(0),                             ;---- xfr wait time    (0.01 min)
 MW[9] = TIMEA(0,20)/2.0,         ;---- walk acc time    (0.01 min)
 MW[10] = (TIMEA(0,22) + TIMEA(0,23) + TIMEA(0,24) + TIMEA(0,25) +TIMEA(0,33) + TIMEA(0,34) +
TIMEA(0,35) + TIMEA(0,44) + TIMEA(0,45) + TIMEA(0,55) )/2 ,        ;---- other walk time  (0.01
min)
 MW[11] = XFERPENA(0),                          ;---- added xfer time  (0.01 min)
 MW[12] = BRDINGS(0),                           ;---- boardings        (1+)
 MW[13] = TIMEA(0,19),                          ;---- drv acc time     (0.01 min)
 MW[14] = DIST(0,19),                           ;---- drv acc distance (0.01 mile)
 MW[15] = TIMEA(0,15),                          ;---- pnr impedance    (0.01 min)
 MW[16] = DIST(0,15),                           ;---- pnr cost         (cents)
 MW[17] = DIST(0,3),                            ;---- metrorail distance
 MW[18] = DIST(0,5)                             ;---- light rail distance

ENDPROCESS
```

```
ENDRUN
```

Figure 6-28 shows the trace report for transit paths between zones 1340 and 103 and zones 1340 and 105. The first example shown includes a park-n-ride access leg (mode 19) followed by a Metrorail leg (mode 3), and a walk egress leg (mode 20). It also reports a transfer at node 8016 from the Green Metrorail line to the Red Metrorail line. The second example shown below follows the same pattern as the first example but includes a non-transit walk leg between Metrorail and bus routes (mode 32).

Figure 6-28: Sample of PT's Trace Report for Park-n-Ride to Bus-Metrorail during Peak Period

```
REval Route(s) from Origin 1340 to Destination 103

N:  1340 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
->  8043   19      -  18.02  18.02      -  18.02   7.95   7.95
->  8016    3   3.00  17.19  38.21   5.30  43.51   7.11  15.06  WMGRNA-(1.000)
->  8016    -      -   0.00  38.21      -  43.51   0.00  15.06
->  8008    3   1.00  16.53  55.74   6.90  67.94   6.18  21.24  WMREDA-(0.667) WMREDB-(0.333)
->   103   20      -  14.20  69.94      -  89.24   0.71  21.95
Mode  TimeA   Dist IWaitA XWaitA
   3  33.72  13.29   3.00   1.00
  19  18.02   7.95
  20  14.20   0.71

REval Route(s) from Origin 1340 to Destination 105

N:  1340 Mode  WaitA  TimeA Actual B/XPen Percvd   Dist  Total  Lines(weight)
->  8043   19      -  18.02  18.02      -  18.02   7.95   7.95
->  8016    3   3.00  17.19  38.21   5.30  43.51   7.11  15.06  WMGRNA-(1.000)
->  8016    -      -   0.00  38.21      -  43.51   0.00  15.06
->  8008    3   1.00  16.53  55.74   6.90  67.94   6.18  21.24  WMREDA-(0.667) WMREDB-(0.333)
-> 22864   32      -   0.40  56.14      -  68.34   0.02  21.26
-> 20743    1   3.00   3.68  62.82   6.90  81.92   0.67  21.93  WME02O(0.300) WME04O(0.300)
WME06O(0.400)
->   105   20      -   5.60  68.42      -  90.32   0.28  22.21
Mode  TimeA   Dist IWaitA XWaitA
   1   3.68   0.67   0.00   3.00
   3  33.72  13.29   3.00   1.00
  19  18.02   7.95
  20   5.60   0.28
  23   0.40   0.02
```

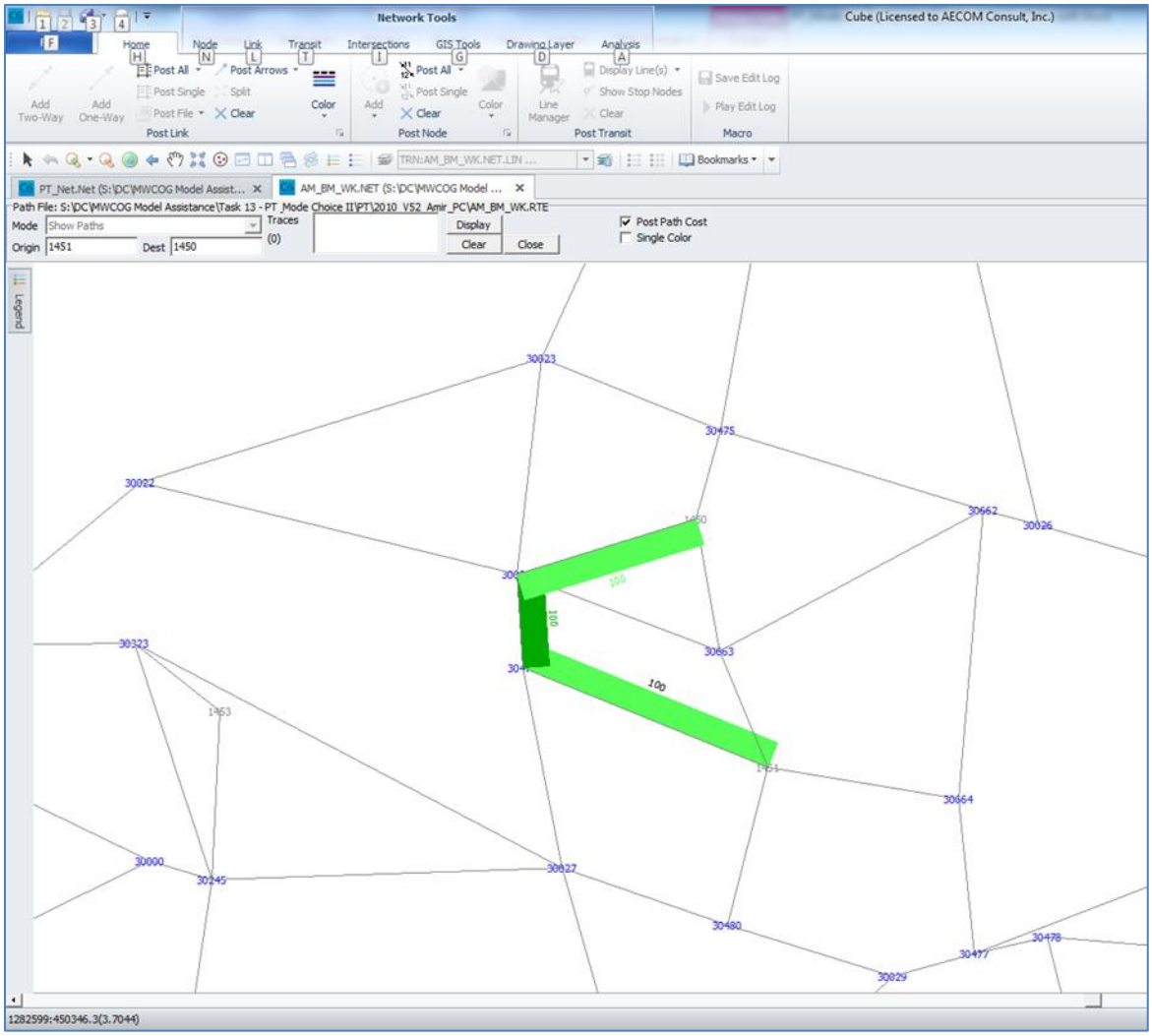
6.5.1 Transit Path Visualization

One of the important features that PT provides to users compared to TRNBUILD is the ability to build and display transit paths interactively on screen (Figure 6-29). The transit path is extracted from a route (*.rte) file generated by the PT skim script. The following steps show how users can build and display transit paths on screen.

To initiate this feature in a Network Window follow the steps below:

1- Open a highway network such as "AM_BM_WK .net", generated as a NETO from a PT skimming process,
2- Select its associated transit (TRN) layer as the top layer,
3- From the analysis tab, click on the Add Path File and select a route file associated with the highway network that is open, for example "AM_BM_WK.RTE",
4- Provide origin and destination zone numbers and click the display button.

Figure 6-29: Visualizing Transit Paths in Cube



6.6 Assessment of PT Path Building and Skims

The paths generated by the PT skimming process were compared to those from the TRNBUILD process and assessed in relation to transit path suggested by Google Maps for a given origin and destination pair. As part of the future calibration effort, the PT transit paths should be compared with observed transit paths reported in the on-board transit survey. This was not completed as part of this task as it requires network files for 2007 rather than the 2010 networks used in the latest version of the MWCOG model. In addition, AECOM's goal was to test the sensitivity of the factors and other parameters affecting the path building process. For that purpose, Google transit paths were helpful in choosing between paths generated by PT and TRNBUILD.

In the testing process, different time and penalty factors in the factors file (e.g. AM_TRN.FAC contains PT factors for the morning peak period) were tested. The comparison showed that the PT and TRNBUILD path building processes are not identical, and hence, the factors and other parameters cannot be exactly

replicated between the two systems. However, the PT process can be calibrated to develop similar paths and skims as TRNBUILD. In the path building and skimming tests performed to date, AECOM attempted to set the PT factors similar to TRNBUILD skimming factors. The resulting PT paths were compared to TRNBUILD paths and the results were reasonable. In cases where PT and TRNBUILD produced different paths, PT factors and non-transit leg generation were adjusted to construct paths that more closely match the TRNBUILD paths. This requires adjustments to the GENERATE logic, mode-to-mode transfer penalties, and input network links, among other PT components.

As a first step, an attempt was made to align PT factors closely with factors used in the TRNBUILD skimming process:

- Link RUNFACTOR values for all transit modes were set at 1.00.[37]
- Walk access and Drive access RUNFACTOR values were set at 1.50 and 2.00, respectively.
- Walk transfer link RUNFACTOR was set at 2.00.
- Boarding and Transfer penalties: TRNBUILD boarding and transfer penalties are specified in a combined format, by including modes 11-16 (in addition to transit modes 1-10) in a 16x16 penalty matrix. In PT, boarding and transfer penalties are specified separately as BRDPEN and XPEN, respectively. So the transfer penalties in PT have to be specified appropriately in order to model similar penalties as in TRNBUILD. For example, in the current MWCOG TRNBUILD model, there is no penalty for a Metrorail-to-Metrorail transfer at the same station. In order to replicate this attribute within PT, the general Metrorail boarding penalty of 2 minutes must be adjusted for Metrorail-to-Metrorail boardings by adding an XPEN value of -2 minutes to remove the boarding penalty.
- A wait factor of 2.50 is specified at nodes served by all transit line-haul modes.
- A factor of 2.50 is specified as the transfer penalty for transfers from modes 1-10 to modes 1-10. The transfers from and to non-transit modes, and between transit modes are ignored by PT in assessing transfer penalty factors.

An excerpt of the PT factors file is shown in Figure 6-30.

Figure 6-30: Excerpt from PT Factors File for Peak Period

```
;Runfactors
 RUNFACTOR[1]   = 1.00 ;--- in-vehicle time ----
 RUNFACTOR[2]   = 1.00
 RUNFACTOR[3]   = 1.00
 RUNFACTOR[4]   = 1.00
 RUNFACTOR[5]   = 1.00
 RUNFACTOR[6]   = 1.00
 RUNFACTOR[7]   = 1.00
 RUNFACTOR[8]   = 1.00
 RUNFACTOR[9]   = 1.00
 RUNFACTOR[10]  = 1.00
 RUNFACTOR[11]  = 1.50 ;--- drive access time
 RUNFACTOR[12]  = 2.00  ;--- walk access time  ----
 RUNFACTOR[13]  = 2.00  ;--- Walk transfer time  ----
 RUNFACTOR[14]  = 1.50 ;--- KNR Access time  ----
 RUNFACTOR[15]  = 2.50 ;--- park-&-ride transfer time
```

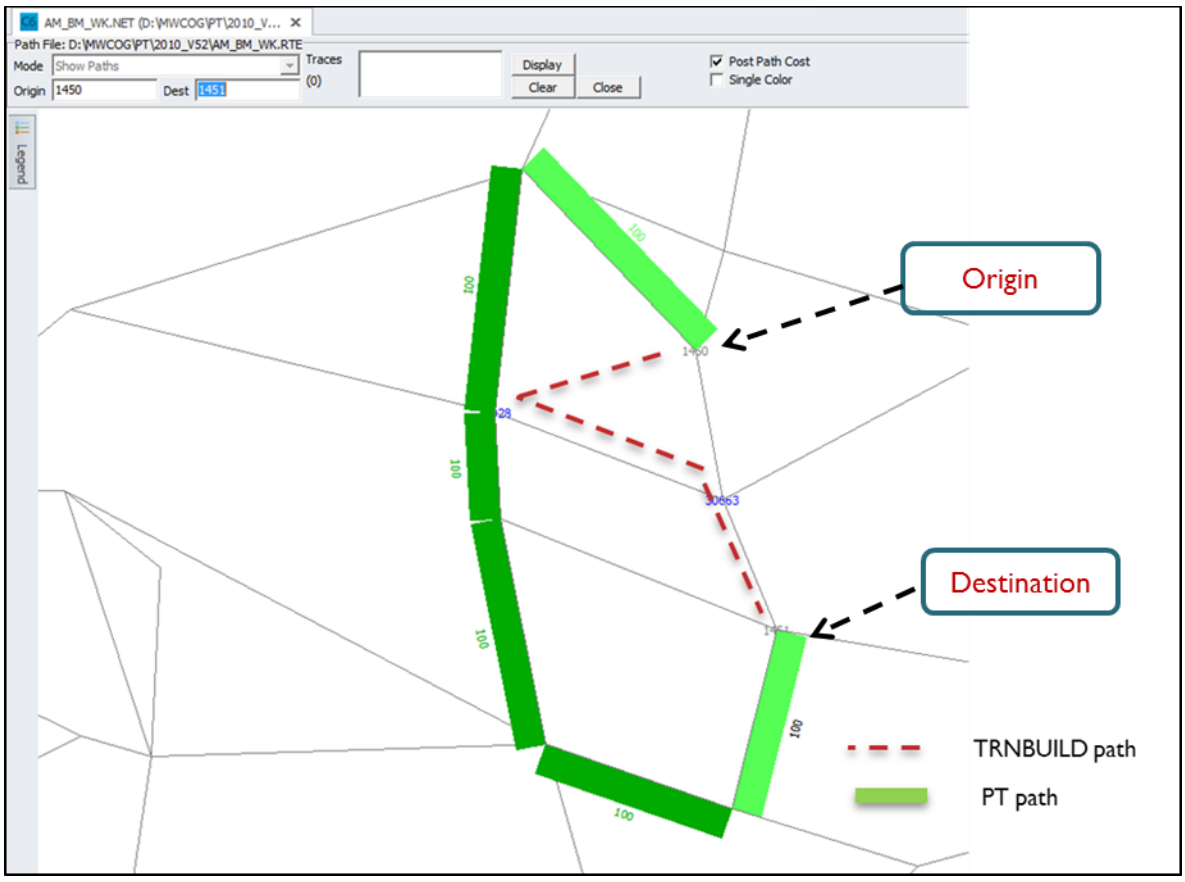
[37] RUNFACTOR is the weighting factor applied to transit in-vehicle times and to non-transit leg times.

```
RUNFACTOR[18]  = 1.50 ;--- kiss-&-ride non-transit leg time
RUNFACTOR[19]  = 1.50 ;--- park-&-ride non-transit leg time

RUNFACTOR[20]  = 2.00 ;--- walk access from zone centroid to bus_Met_Com_Lrt stations----

RUNFACTOR[22]  = 2.00 ;--- walk transfer link between bus and bus ----
RUNFACTOR[23]  = 2.00 ;--- walk transfer link between bus and Metro ----
RUNFACTOR[24]  = 2.00 ;--- walk transfer link between bus and commuter rail ----
RUNFACTOR[25]  = 2.00 ;--- walk transfer link between bus and LRT ----
RUNFACTOR[32]  = 2.00 ;--- walk transfer link between Metro and bus ----
RUNFACTOR[42]  = 2.00 ;--- walk transfer link between bus and commuter rail ----
RUNFACTOR[52]  = 2.00 ;--- walk transfer link between bus and LRT ----

RUNFACTOR[33]  = 2.00 ;--- walk transfer link between Metro stations ----
RUNFACTOR[34]  = 2.00 ;--- walk transfer link between Metro and commuter rail ----
RUNFACTOR[35]  = 2.00 ;--- walk transfer link between Metro and LRT ----
RUNFACTOR[43]  = 2.00 ;--- walk transfer link between Metro and commuter rail ----
RUNFACTOR[53]  = 2.00 ;--- walk transfer link between Metro and LRT ----

RUNFACTOR[44]  = 2.00 ;--- walk transfer link between commuter rails  ----
RUNFACTOR[45]  = 2.00 ;--- walk transfer link between Commuter and LRT ----
RUNFACTOR[54]  = 2.00 ;--- walk transfer link between Commuter and LRT ----

RUNFACTOR[55]  = 2.00 ;--- walk transfer link between LRT stations ----


;Boarding and Transfer Penalties
 BRDPEN= 2*5.0, 3*2.0, 5*5.0

 XFERPEN=3.0,  from=1-10, to=1-2, 6-9
 XFERPEN=-2.0, from=3, to=3

 ;xfer wait time is based on default wait curves defined by PT.
 WAITFACTOR=2.5, n=8000-60000

 XFERFACTOR=2.5, from=1-10, to=1-10,
```

The paths generated for the Bus-Metrorail line-haul mode in the AM period were compared between TRNBUILD and PT. The paths were found to be different in terms of initial walk access to bus or Metrorail, number of transfers between bus and Metrorail, among other trip characteristics. A few of the possible issues in PT path building, particularly as it compares to TRNBUILD, are discussed below.

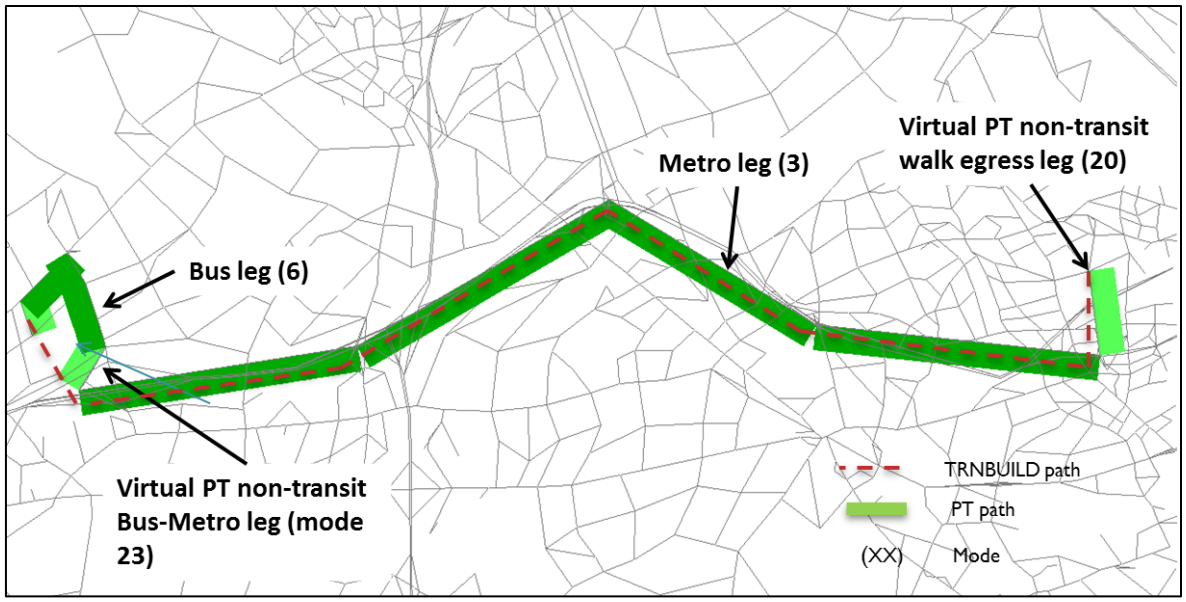
- Due to the nature of modes available during skimming in TRNBUILD, walk-only paths may be built. However, in PT, walk-only paths are not possible while skimming for Bus-Metrorail line-haul paths. PT ensures that both transit and non-transit legs created using the PT GENERATE statement are included in the path. Figure 6-31 provides an illustration of this point. In the figure, zone 1450 is the origin and zone 1451 is the destination. The PT path involves walk access to a bus (light green), a bus route (dark green), and walk egress (light green). By contrast, the TRNBUILD path (shown with dashed red lines) involves walking along zone connectors and highway links. PT will not generate a shorter path that involves only the non-transit legs generated by PT GENERATE because a PT path cannot have two or more non-transit legs in succession. The walk-only path produced by TRNBUILD will eventually be rejected as a bus-Metrorail path option in the mode choice step since a bus-Metrorail path requires in-vehicle travel time for both bus and Metrorail modes. Hence, in assessing the differences in paths between TRNBUILD and PT, implications to mode choice need to be considered.

Figure 6-31: Difference in PT and TRNBUILD Paths: No Walk-Only PT Paths



- Unlike TRNBUILD, the PT setup prepared by AECOM does not allow long walk access legs to Metrorail stations. As a result, PT sometimes adds a bus access leg to paths that TRNBUILD models as a long walk. Figure 6-32 compares a TRNBUILD and PT transit path structure for walk to bus-Metrorail line-haul modes for the same origin and destination pair. It can be seen that the TRNBUILD path includes a direct walk access leg between the zone centroid and the Metrorail station. However, the PT paths includes a non-transit walk leg to a bus stop, a bus leg (mode 6), and a non-transit walk transfer leg (mode 23) from bus stop to the Metrorail station. The rest of the path structure is similar between the TRNBUILD and PT path. A path conditioning step could be added to the model process to eliminate transit paths between zone pairs that are longer than walking paths.

Figure 6-32: Difference in PT and TRNBUILD Paths: PT Bus to Metrorail Transfer



- Differences in the locations of zone connectors between the tested TRNBUILD and PT networks resulted in some cases where the walk access paths to transit were different between TRNBUILD and PT. By addressing the differences in inputs to the two processes, similar paths may be more likely to be produced.

Average in-vehicle time skims were compared between TRNBUILD and PT using the current PT setup. The comparison is shown in Table 6-4. It is to be noted that the PT skim values may change considerably with appropriate changes to the PT steps prior to skimming, such as GENERATE, and system and factor setups.

Table 6-4: Average AM Peak Bus-Metrorail Skim Values by Mode of Access in TRNBUILD and PT

Measure	Mode	TRNBUILD			PT		
		PNR	KNR	WK	PNR	KNR	WK
In-Vehicle Time (Minutes)	Local Bus	24	24	27	47	51	51
	Express Bus	45	46	43	55	54	45
	Metrorail	32	32	32	20	21	22

With the current PT setup, average local and express bus in-vehicle times for bus-Metrorail line-haul paths in PT are higher than those in TRNBUILD. Further review of the access and transfer legs and transfer factors may be needed to investigate the possible causes of the differences in skims.

6.6.1 First and Last Boarding and Alighting Stations

In accordance with WMATA's tariff structure, MWCOG uses first and last transit stations to calculate transit fares. Transit fares are used by mode choice to calculate mode shares. The PT transit skimming process does not have the capability of saving first and last boarding and alighting stations. However, this information can be produced during a PT transit assignment. In order to generate this data, a dummy matrix filled with ones for all origins and destinations that have non-zero trips can be created and used as input to the PT transit assignment procedure to save the first and last boarding and alighting transit stations. Figure 6-33 shows a sample script that can be used to create such a dummy matrix.

Figure 6-33: Create Dummy Origin-Destination Matrix Filled with Ones

```
; Create dummy matrix filled with ones for transit assignment
RUN PGM = MATRIX

FILEI MATI[1] = "Trips\HBW AM MS.tpp"

FILEO MATO[1] = "Trips\Ones HBW AM BM.TRP" , MO=1-3,
NAME="HBW_AM_BM_WK","HBW_AM_BM_PNR","HBW_AM_BM_KNR"

MW[1] = MI.1.6  ; Walk to Bus/Metro
MW[2] = MI.1.11 ; PNR to Bus/Metro
MW[3] = MI.1.12 ; KNR to Bus/Metro

        JLOOP
        IF ( MW[1] = 0 || I > 3675 || J > 3675)
                MW[1] = 0
        ELSE
                MW[1] = 1
        ENDIF

        IF ( MW[2] = 0 || I > 3675 || J > 3675)
                MW[2] = 0
        ELSE
                MW[2] = 1
        ENDIF

        IF ( MW[3] = 0 || I > 3675 || J > 3675)
                MW[3] = 0
        ELSE
                MW[3] = 1
        ENDIF
        ENDJLOOP

ENDRUN
```

6.7 PT Transit Assignment

The PT transit assignment script is very similar to the PT transit skimming script. The main difference is the introduction of a demand matrix (in production/attraction format) and instructions to the program to load the matrix to the PT network. In addition, as mentioned above, the transit assignment procedure has the capability of saving first and last boarding and alighting stations. PT uses the "STOP2STOP" keyword during the assignment step to save this information. The program writes output records according to the statement assigned to the sub keyword "ACCUMULATE". There are multiple options that can be assigned to this sub keyword, but "ADJACENTBYMODE" is the best match for the MWCOG model data requirements for calculating transit fares. Table 6-5 shows an example of a transit path with three transit legs, composed of five segments, and two modes. In this table, "1-2 on mode 1" means that mode 1 is used to travel from node 1 to node 2.

Table 6-6 shows the results of the "STOP2STOP" keyword using "ADJACENTBYMODE" sub keyword compared to other options available. Basically, it shows the information that will be saved using the STOP2STOP keyword depending on the value assigned to the ACCUMULATE keyword. For example, if ACCUMULATE is equal to FIRSTLAST, then the program will saves the first (1) and last node (10) of the transit path.

Table 6-5: An Example of Five-Segment Trip using Two Modes

Segment	Mode
1	1-2 on mode 1
2	3-4 on mode 1
3	5-6 on mode 2
4	7-8 on mode 1
5	9-10 on mode 1

Table 6-6: Results of STOP2STOP Keyword using Different Options for Sub Keyword ACCUMULATE

Value of ACCUMULATE	Results of STOP2STOP
FIRSTLAST	1-10
ADJACENT	1-2, 3-4, 5-6, 7-8, 9-10
ALLONOFFS	1-2, 1-4, 1-6, 1-8, 1-10, 3-4, 3-6, 3-8, 3-10, 5-6, 5-8, 5-10, 7-8, 7-10, 9-10
FIRSLASTBYMODE	1-10 mode 1 5-6 mode 2
ADJACENTBYMODE	1-4 mode 1 5-6 mode 2 7-10 mode 1

Figure 6-34 shows the PT transit assignment script that loads the dummy matrix filled with ones for saving the first and last boarding and alighting stations. This script assigns walk-access trips during the peak period using the bus-Metrorail line-haul mode.

Table 6-7 shows a sample of the data saved in the "AM_BM_WK.dbf" file created by the "STOP2STOP" statement.

Figure 6-34: PT Transit Assignment to Save First and Last Boarding and Alighting Stations by Mode

```
RUN PGM = PUBLIC TRANSPORT; MSG = 'Bus-Metrorail AM WK transit S2S'

FILEI NETI = "Inputs\PT_NET.NET"
FILEI MATI[1] = "Trips\Ones_HBW_AM_BM.TRP"

FILEO ROUTEO[1] = "AM BM WK TEMP.RTE"
FILEO REPORTO = "PT S2S AM BM WK.PRN"
FILEO STOP2STOPO = "AM_BM_WK_S2S.dbf" , ACCUMULATE = ADJACENTBYMODE, NODES= 1-70000, MODES =
3,5, LIST=Y

;Input ModeAccess specific factors
FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
FILEI SYSTEMI = "Inputs\TSYSD.PTS"

;---- Non-Transit legs  ----
FILEI NTLEGI[1] = "Walk_Acc_Legs.LEG"
FILEI NTLEGI[2] = "Walk_Egr_Legs.LEG"
FILEI NTLEGI[3] = "Walk_Transfer_Legs.LEG"
;FILEI NTLEGI[4] = "KNR.LEG"
```

```
;FILEI NTLEGI[5] = "PNR.LEG"

;---- Transit modes (1 to 10)  ----
 FILEI LINEI[1] = "Inputs\AM Bus Lines.lin"
 FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
;FILEI LINEI[3] = "Inputs\Mode4AM.lin"

;---- Access Links (modes 11, 12 and 16) ----
zonemsg=50
; OVERALL PARAMETERS OF RUN
PARAMETERS FARE=N, HDWAYPERIOD=1,NOROUTEERRS=999999,
           NOROUTEMSGS=999999,SKIPBADLINES=Y,
          TRANTIME=LI.TRANTIME, TRIPSIJ[1] = MI.01.1

REPORT LINES=T
PROCESS PHASE=LINKREAD
 ;LW.AMHTIME=LI.AMHTIME
 ;LW.WALKTIME=(LI.DISTANCE/3*60)
 ;LW.WALKDISTANCE=LI.DISTANCE
 ;LW.DISTANCE=LI.DISTANCE
ENDPROCESS

PROCESS PHASE=DATAPREP
       GENERATE READNTLEGI=1
       GENERATE READNTLEGI=2
       GENERATE READNTLEGI=3
       ;GENERATE READNTLEGI=4
       ;GENERATE READNTLEGI=5
ENDPROCESS

ENDRUN
```

Table 6-7: Sample of Data Generated by STOP2STOP Keyword

I	J	FromNode	ToNode	Mode	Accum	VOL
75	1902	8013	8061	3	5	1.00
76	1917	8065	8061	3	5	1.00
77	1902	8013	8061	3	5	1.00
77	1917	8013	8061	3	5	1.00

6.8 Next Steps and Recommendations in PT Conversions

As part of this task order, AECOM successfully developed and finalized the PT network structure, worked with MWCOG to prepare the required input files including conversion of TRNBUILD transit lines to PT transit lines, and developed Cube Voyager scripts to automatically create non-transit legs, perform the PT transit skimming process, save required information for the MWCOG fare process module, and conduct transit assignment. AECOM also provided a solution to effectively rectify the problem that current representations of bus park-n-ride locations creates for the PT process.

AECOM also compared transit paths created by the new PT process with previous TRNBUILD paths as well as transit paths created by Google maps. The comparison between PT and TURNBUILD was mainly performed to help AECOM understands the difference between the PT and TRNBUILD path building methodology and how best to translate factors and calculations implemented in the TRNBUILD scripts to PT scripts and related complementary files such as factor and system files. The comparison of PT transit paths and paths generated by Google Maps was done in order to see how well PT transit paths mimic

the best transit paths suggested by Google Maps. The results of these comparisons were used to adjust the factors and weights used in the PT process.

Despite AECOM efforts to adjust the parameters used by the PT process to this date, the transit paths from the PT process are not considered to be thoroughly validated. it is suggested that a formal calibration process be performed in which the PT transit paths structure, travel time, and ridership are compared with the latest on-board transit survey. The calibration process will help to adjust the parameters used by PT to develop non-transit legs and find the best transit path for a given origin-destination pair. This should include the maximum distance and time that are used by PT's GENERATE statement to create walk and drive access legs as well as walk transfer legs between transit modes. The comparison might suggest that different sets of walk and drive access legs should be created for Metrorail stations depending on the location of the stations (e.g. end of the line stations). Boarding and transfer penalties potentially need to be adjusted further to help the PT transit path component better match the observed paths.

In addition, AECOM's experience using the PT module in other travel demand models suggests that an additional step in the model process may be useful in the MWCOG model. After transit paths are built for all zone to zone interchanges, a path conditioning step could be used to drop transit paths between a pair of origin and destination zones if the total travel time of that path is longer than the walk-only path between the zone pair.

This page is intentionally blank.

7 Mode Choice Modeling (Task Order 13)

The primary focus of this task was to convert the MWCOG Version 2.3 mode choice model from AEMS to ModeChoice software.

7.1 Mode Choice Background

AEMS is a FORTRAN program for applying mode choice models.[38] It reads and writes trip tables and path skim matrix files in the native format of several travel demand forecasting software vendors, including MINUTP, TP+, Cube, TransCAD, and Emme2. It was first developed in the early 1990s by Bill Woodford at KPMG. AEMS (AECOM Mode Split) was the name assigned to the program when the transportation group at KPMG was purchased by AECOM.[39] The software includes a companion program called CALIBMS for assisting with mode choice calibration efforts. The software is still in use in numerous travel demand forecasting systems throughout the country. ModeChoice is a C++ program for applying mode choice models.[40] In 2010, David Roden wrote the ModeChoice program by migrating the AEMS algorithms from FORTRAN to C++, and by making use of TRANSIMS open-source software libraries, which he had helped write in earlier work. The program has been used to implement mode choice models for WMATA and the Central Florida Regional Planning Model (CFRPM) in Orlando, Florida.

The following bullets describe many of benefits of migrating from AEMS to ModeChoice in more detail.

- AEMS is an old FORTRAN program that is not easily compiled using currently available FORTRAN compilers. The ModeChoice program is a C++ program that is easily compiled using the latest C++ compliers for 32-bit and 64-bit Windows and Linux operating systems.

- The AEMS code is also poorly organized and not written with the expectation that a programmer will be able to understand or update it. The ModeChoice code is specifically designed for distribution through the TRANSIMS Open-Source website and, as such, is structured and written to be understood and potentially improved by other programmers.

- There is no active maintenance or support for the AEMS program. The ModeChoice program is supported and maintained through the TRANSIMS e-mail distribution list and SourceForge version control system. David Roden continues to improve and expand the capabilities of the software as new applications, compiler options and 64-bit versions of TransCAD or Cube are released.

- AEMS is compiled separately for Cube, TransCAD, and EMME2. ModeChoice can be compiled to work with Cube and TransCAD interchangeably, plus a generic TRANSIMS matrix format.

[38] AECOM Consult, Inc., *AECOM Consult Mode Choice Computation Programs, AEMS, Users Guide*, Draft report (Fairfax, Virginia: AECOM Consult, Inc., April 5, 2005).
[39] David Roden to Mark S. Moran, "Task Order 9 of COG Contract 12-006 – Public Transport Conversion: Advantages of Migrating from the AEMS Mode Choice Application Software to the ModeChoice Mode Choice Application Software," Memorandum, (February 1, 2013).
[40] AECOM, *ModeChoice, Version 5.1.18* (Arlington, Virginia: AECOM, June 10, 2013).

- In order to improve processing time, AEMS internally converts the log functions to a piece-wise lookup table with straight-line interpolation between data points. The ModeChoice program uses the log functions directly. This makes ModeChoice slightly more accurate and generates small differences in the resulting mode shares.

- The user interface for the AEMS program is quite complex and rigid in its column format. If an extra space is added or missing, the program executes without detecting an error and generates erroneous results. The organization of elements and variables is not very logical or sequential and it requires a considerable number of records to simply initialize variables or fill out unused elements of the control file. A separate control file with the full model specification is required for each trip purpose.

- The ModeChoice program uses the standard TRANSIMS control file structure that includes control keys and values that can be ordered and formatted in many ways. It also includes an integrated help system and data valuation checks. The control file is used to define the input and output file names, reports, and overall model structure. The model utility functions and conditional overrides are defined in a user script file that can be common for all trip purposes. This script uses a programming syntax similar to FORTRAN and C with a full range of variable names, lookup tables, and logical syntax statements (e.g., if-then-else-endif) that provide exceptional flexibility and computational clarity at the same time.

- The AEMS and ModeChoice programs convert the model structure and computations defined in the control file and user script to internal processing commands. The syntax compiler included in the ModeChoice program generates more computationally efficient processing procedures than the AEMS counterpart. Side-by-side performance tests suggest the ModeChoice program reduces processing time by at least 40 percent.

- The ModeChoice program includes a number of report and output file options that are not available in AEMS.

- The ModeChoice program has a built-in calibration capability that iteratively adjusts the mode constants to match a set of mode share targets. This is similar to the CALIBMS program that is available for AEMS. The output file of the ModeChoice calibration application is in the exact format needed for production applications of the program. The output of the CALIBMS program needs to be integrated into the control file for AEMS.

- In addition to calibrating the mode constants, the ModeChoice program includes a secondary calibration adjustment to the demographic subdivision constants within a mode (e.g., income or auto occupancy categories). CALIBMS does not adjust these constants.

- The calibration target file used by ModeChoice includes the option to specify minimum and maximum values for each mode constant. This helps to keep the constant values within a reasonable range. One of the concerns in using CALIBMS is that it does not constrain the

constant values and often generates values that are illogical and need to be adjusted manually for input to the AEMS program.

- The ModeChoice program includes an RMSE convergence parameter to stop the calibration progress at the required level of precision or when the maximum number of iterations is reached. CALIBMS uses a fixed number of iterations

Discrete-choice models, such as logit mode choice models, are typically developed using three steps: estimation, calibration, and validation. Estimation has traditionally been done at the disaggregate level, whereas calibration and validation can be done at either the disaggregate level, the aggregate level, or both. Ultimately, most logit mode choice models are applied at the aggregate level, typically at the level of zone-to-zone person-trip flows. Some models user market segmentation (discussed below), which subdivides the travel market into homogeneous groups.

The Federal Transit Administration (FTA), which reviews many travel demand models as part of the federal government's Section 5309 New Starts program, has produced a series of discussion papers. One of these papers noted that the many agencies spend too much time on model estimation and too little time on model calibration and validation. For example, the FTA stated

> *Misallocation of effort. Many model-development efforts reserve relatively few resources for calibration and validation – and those resources are often eroded by higher-than-anticipated efforts in data assembly and model estimation. Model estimation is an important step: it provides the basic parameters for all components of the model set, it has provided all of the insights into travel behavior available to the practice of travel forecasting, and it is the avenue to all future improvements in travel models. However, model estimation can easily consume all resources available for model development. The pursuit of clean estimation results (with all parameters estimates statistically significant and with the correct sign, for example) can extend the effort beyond allocated resources.*[41]

Due to this and perhaps other factors, many agencies are spending less time on model estimation. In fact, some agencies have taken FTA's advise and forgone model estimation altogether. Instead, these agencies will set time and cost coefficients by fiat (using previous results and a number of rules-of-thumb about the relative magnitudes of various coefficients), and then calibrate the model, using a process that adjusts one or more alternative-specific constants (ASCs). This is the process that has been used recently in the Washington, D.C. area, both by consultants and by MWCOG. The number of ASCs is a function of the number of choices in the model and the amount of market segmentation used in the model. The MWCOG Version 2.3 Travel Model uses four types of market segmentation:

- Household income (four income quartiles)

[41] Federal Transit Administration, "Discussion Piece #16: Calibration and Validation of Travel Models for New Starts Forecasting" (presented at the Workshop on Travel Forecasting for New Starts Proposals, Minneapolis, Minnesota, 2006), 1, http://www.fta.dot.gov/planning/newstarts/planning_environment_5402.html.

- Geography (20 district-to-district interchanges, based on seven superdistricts)
- Transit access mode (walk, park-n-ride, kiss-n-ride), and
- Primary transit mode (all-bus, all-Metrorail, bus plus Metrorail, and commuter rail).

The geographic market segmentation scheme was first developed by AECOM in work done in 2004/2005.[42] COG retained the same scheme for its models development work, from 2008 to the present. In general, the maximum number of alternative-specific constants in a logit model is equal to the number of alternatives minus one (N - 1). In the case of the MWCOG Version 2.3 mode choice model there are 20 geographic market areas, 15 mode options and 6 nested modes. This means the maximum number of mode and nesting constants per trip purpose is 20 * (15 + 6 – 1) = 400. Additionally, the current MWCOG model uses 1200 income constants (20 market segments * 15 modes * 4 income level). However, the income constants were assumed to be equal for all geographic market segments and only have non-zero values for walk to transit modes. Calibrating the nested-logit mode choice model essentially consists of estimating the values of the nesting constants and the income constants.

One of COG's past consultants, Cambridge Systematics, Inc. (CS), discussed the pros and cons of three approaches to developing a mode choice model: the estimation approach, the assertion approach, and the hybrid approach.[43] The hybrid approach combines both the estimation and assertion techniques. CS recommended that COG use a hybrid approach for its next mode choice model calibration effort.[44] However, due to limited resources for the current effort, and due to the fact that the goal of the current effort was a migration of the application software, not a full recalibration of the mode choice model, AECOM decided, in concert with COG staff and FTA guidance, to use the assertion approach for this current effort. In other words, one can think of this mode choice calibration effort as a "proof of concept," not a full-scale calibration to observed data. Thus, the time and cost coefficients are asserted (i.e., the same values used in the existing COG mode choice model), but the nesting constants and income constants are re-estimated using appropriate mode choice targets (i.e., the same observed data COG used to calibrate the existing COG mode choice model).

7.2 Mode Choice Calibration

AECOM replaced the existing AEMS software used in the current MWCOG model for mode choice analysis with the ModeChoice software. This task assumes that the existing MWCOG Version 2.3 mode choice structure will be maintained, but the application software will change. The conversion offers benefits for software maintenance, documentation, run time, model calibration, and batch processing. The ModeChoice program is also used by the WMATA post-processing tool. The WMATA tool reads the

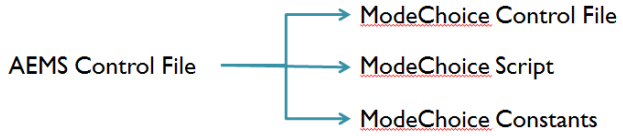
[42] Bill Woodford, "Development of Revised Transit Components of Washington Regional Demand Forecasting Model" (presented at the Transit Modeling Meeting, held at the Metropolitan Washington Council of Governments, Washington, D.C., December 1, 2004).
[43] Cambridge Systematics, Inc., *Fiscal Year 2010 Task Reports*, Final Report (Washington, D.C.: National Capital Region Transportation Planning Board, November 16, 2010), 4–21 to 4–23, http://www.mwcog.org/transportation/activities/models/review.asp.
[44] Ibid., 4–26.

MWCOG trip distribution tables and highway skims. It splits the person trips by trip purpose into peak and offpeak periods and applies a different mode choice model for each trip purpose and time period. The transit path options and skims are similar to the MWCOG process. The original WMATA post-processing tool included the same market segments as the current COG model, but has subsequently been re-calibrated to utilize a pedestrian environment factor rather than 20 geographic market segments. Additional factors are included in the WMATA model to consider parking capacity constraints at park-n-ride lots by time of day. The MWCOG highway and transit network are slightly modified in the WMATA tool mainly to provide a more accurate representation of access to Metrorail stations and park-n-ride locations.

7.2.1 Validate ModeChoice Software

The initial focus of this task was to reproduce the existing mode choice results using the existing inputs. AECOM converted the AEMS control file to a ModeChoice control file for each trip purpose and created the required support files and modeling scripts. For AEMS, there was one control file per trip purpose. By contrast, for ModeChoice, this same information is stored in three files (see Figure 7-1). Figure 7-2 to Figure 7-7 show examples of converting the information in one AEMS control file to the contents of the three ModeChoice files (i.e., the control file, the constant file, and the script file). The ModeChoice control file is used to define the input and output file names, mode choice model nesting structure, market segments, utility function parameter values and selected reports. The ModeChoice constant file stores mode-specific constants as well as income-level adjustment constants for each trip purpose. The ModeChoice script file is common for all trip purposes and it includes utility functions and conditional overrides. Therefore, five AEMS control files were converted to eleven files in the ModeChoice setup (5 control files (one for each trip purpose), 5 constant files (one for each trip purpose) and one common script file for all trip purposes).

Figure 7-1: Conversion of AEMS Control File to ModeChoice Control File, Constant File, and Script



The first conversion task is to translate the nested logit or multi-nomial logit model structure from AEMS to ModeChoice conventions. Figure 7-2 shows the section of the AEMS control file that defines the nesting structure. The matrix structure used in AEMS is defined using a user-defined list of mode and mode-nest names in the ModeChoice control file. The PRIMARY_MODE_CHOICE key defines the mode labels assigned to the primary mode choice. It includes a list of two or more user-defined text strings separated by a comma or space (e.g., AUTO, TRANSIT). The MODE_CHOICE_NEST_# keys define the hierarchy of a nested logit model structure. Each key associates a mode label defined in a higher level key with a comma or space-delimited list of user-defined text strings representing the nested mode names. For example, if the primary mode choice includes AUTO as one of its mode labels, the key

MODE_CHOICE_NEST_1 AUTO = SOV, HOV

subdivides the AUTO mode into two submodes labeled SOV and HOV.

Figure 7-2: Section of AEMS and ModeChoice Control Files Defining Nesting Structure

AEMS Nesting Structure

```
*NEST DEFINITIONS BY CHOICE
*CHOICE          1>DR ALONE  SR2      SR3+     WK-CR    WK-BUS   WK-BU/MR  WK-MR    PNR-CR   KNR-CR   PNR-BUS  KNR-BUS  PNR-BU/MR KNR-BU/MR PNR-MR   KNR-MR
NEST 1,1=        1>Y         Y        Y
NEST 1,2=        1>                             Y        Y        Y         Y        Y        Y        Y        Y        Y         Y         Y        Y
NEST 2,1=        1>                             Y        Y        Y         Y
NEST 2,2=        1>                                                                 Y                 Y                 Y                   Y
NEST 2,3=        1>                                                                          Y                 Y                   Y                  Y
NEST 3,1=        1>                             Y
NEST 3,2=        1>                                      Y
NEST 3,3=        1>                                               Y
NEST 3,4=        1>                                                        Y
NEST 4,1         1>                                                                 Y
NEST 4,2         1>                                                                          Y
NEST 4,3         1>                                                                                   Y
NEST 4,4         1>                                                                                             Y
NEST 5,1         1>                                                                          Y
NEST 5,2         1>                                                                                   Y
NEST 5,3         1>                                                                                             Y
NEST 5,4         1>                                                                                                       Y
NEST 6,1         1>Y
NEST 6,2         1>          Y        Y
NEST 7,1         1>          Y
NEST 7,2         1>                   Y
```

ModeChoice Nesting Structure

```
Primary Mode Choice = AUTO, TRANSIT
Mode Choice Nest #1 = AUTO = SOV, HOV
Mode Choice Nest #2 = HOV = SR2, SR3
Mode Choice Nest #3 = TRANSIT = WALK, PNR, KNR
Mode Choice Nest #4 = WALK = WK_CR, WK_BUS, WK_BUS_MR, WK_MR
Mode Choice Nest #5 = PNR = PNR_CR, PNR_BUS, PNR_BUS_MR, PNR_MR
Mode Choice Nest #6 = KNR = KNR_CR, KNR_BUS, KNR_BUS_MR, KNR_MR
```

Figure 7-3 shows a section of the AEMS control file that defines parameters values used in the mode choice utility functions. It also shows the equivalent section in the ModeChoice control file. Note that the AEMS control file provides users an opportunity to define different coefficients for each attribute (e.g. IVTT) for each mode. This is a legacy capability to support antiquated model structures that apply unadjusted coefficients to nested values rather than automatically adjusting main-level coefficients based on nesting coefficients. Legacy models make it difficult for the modeler to recognize coefficient biases at the nested level. FTA strongly recommends specifying mode choice models using main-level coefficient values and applying the same coefficient value to all modes. This is the primary reason the ModeChoice software does not allow users to include mode-specific attribute coefficients in their model design. This also helps to simplify the ModeChoice script.

Figure 7-3: Example of Converting AEMS Control File to ModeChoice Control File

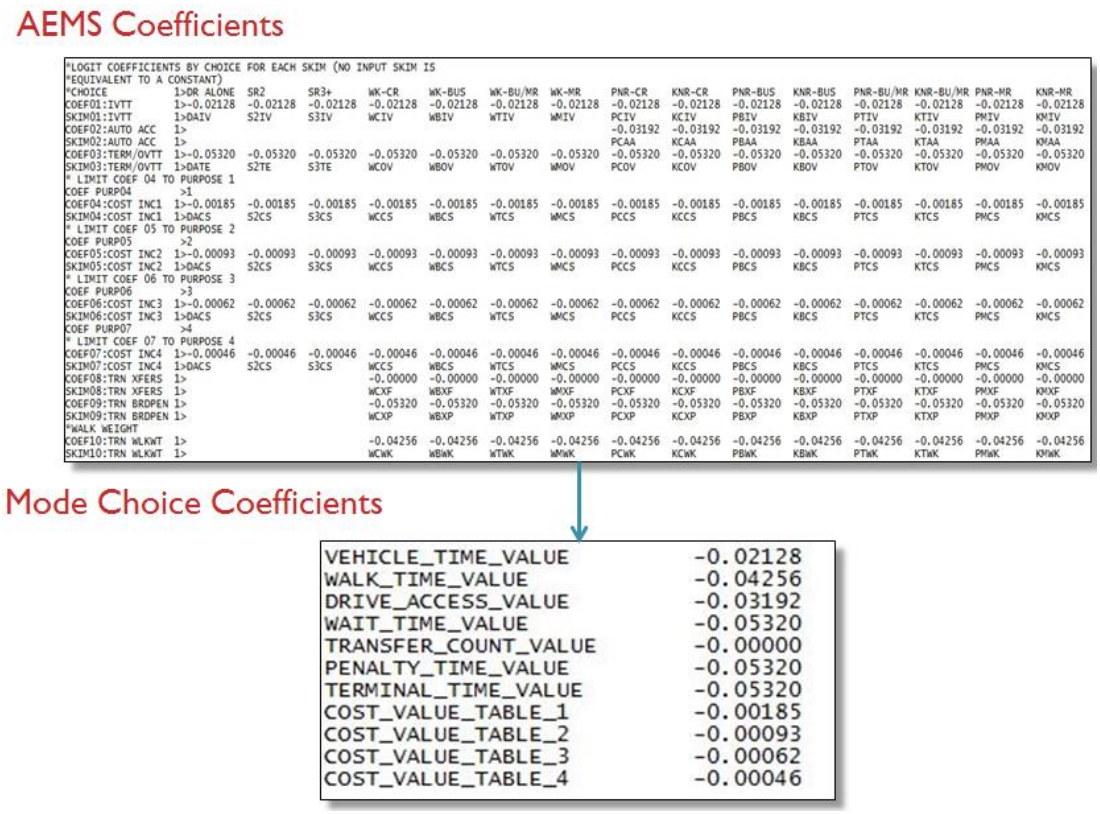
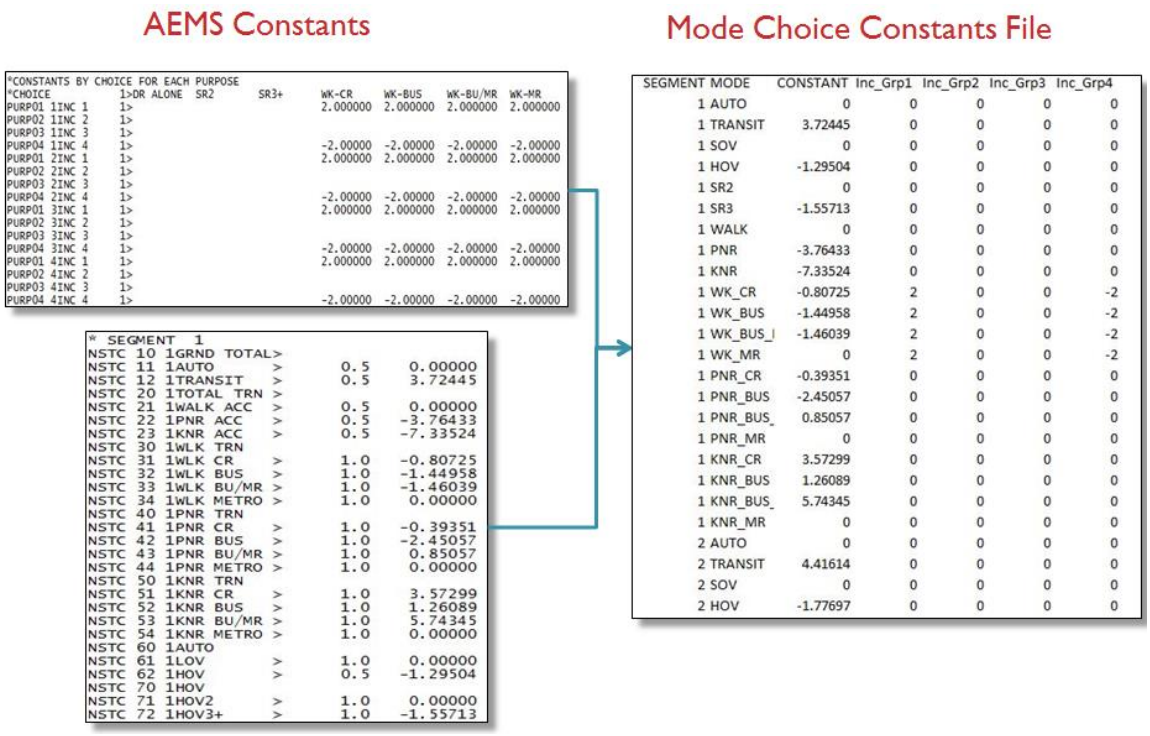


Figure 7-4 shows two sections of the AEMS control file that are used to specify mode-specific constants and income constants. These constants (nesting constants as well as income constants) are stored in a constant file in the ModeChoice setup. Figure 7-4 also shows a section of the ModeChoice constant file.

Figure 7-4: Example of Converting AEMS Control File to ModeChoice Constant File



In addition to mode and nesting constants, a nested logit model uses nesting coefficients to weight the contribution of the nested logsums in the higher level mode choice. In the AEMS control file, these nesting coefficients are coded alongside the mode constants. In the ModeChoice software, the nesting coefficients are coded in the control file for each nesting level using the control key NESTING_COEFFICIENT. The key can be defined as a single value for all nesting level or as separate values for each nesting level using the level extension (i.e., NESTING_COEFFICIENT_1). Figure 7-5 shows the location of nesting coefficients in the AEMS control file and the corresponding coefficients in the ModeChoice control file.

Figure 7-5: Sections of AEMS and ModeChoice Control files Storing Nesting Level Coefficients

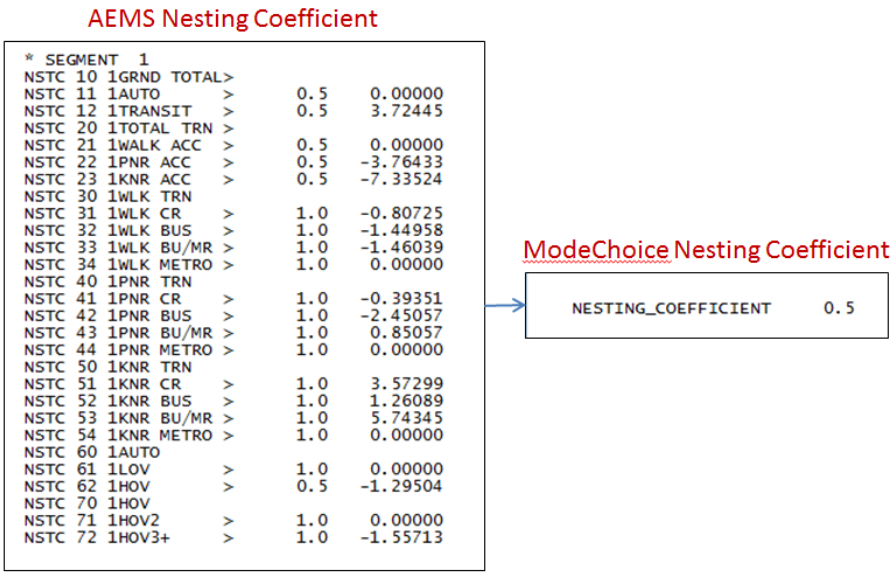
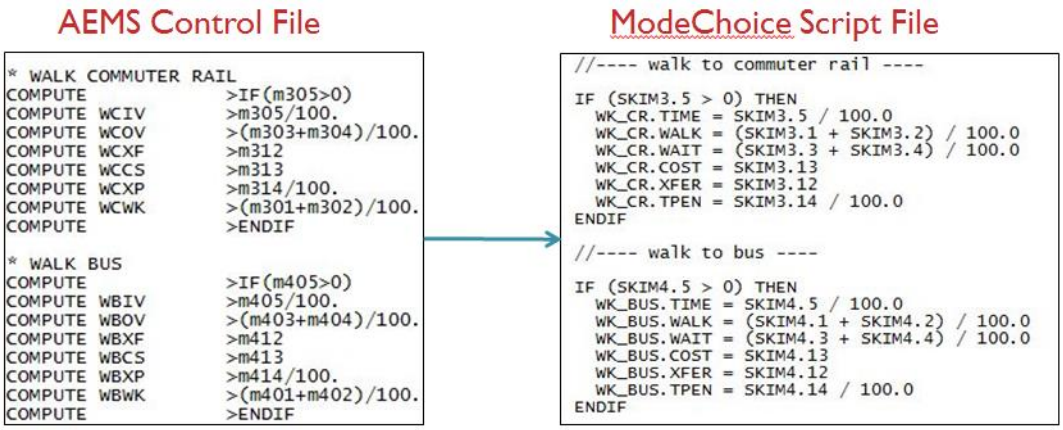


Figure 7-6 shows a section of the AEMS control file as an example that is used to perform conditional calculations. It also shows how that computation has been converted to a more user friendly format in a ModeChoice script file. Similarly, AECOM reformatted all the sections of the AEMS control files that are used to do computational tasks and moved that logic to the ModeChoice script file. If the AEMS control file for a trip purpose included specific calculations for that trip purpose, the ModeChoice script uses that trip purpose as a conditional variable to perform the same calculation only for that purpose.

Figure 7-6: Example of Converting AEMS Control File to ModeChoice Script



7.2.1.1 AEMS Model Conversion Results

AECOM tests demonstrate that the ModeChoice software can accurately reproduce the results of the AEMS software for all trip purposes in significantly less computer processing time. Table 7-1 shows a comparison of auto and transit mode shares generated by AEMS and extracted from the print file of the AEMS program, e.g. HBW_NL_MC.PRN, and ModeChoice software. In addition to these high-level

mode choice results, the ModeChoice program was also able to replicate the submode choices with near-perfect accuracy.

Table 7-1: Auto and Transit Mode Share Generated by AEMS and ModeChoice

	AEMS		ModeChoice		Difference	
Purpose	Auto	Transit	Auto	Transit	Auto	Transit
HBW	2,974,260	788,720	2,974,260	788,719	0	0
HBO	6,658,699	199,628	6,658,699	199,628	0	0
HBS	2,929,669	20,075	2,929,669	20,075	0	0
NHW	1,480,717	75,107	1,480,717	75,107	0	0
NHO	3,129,409	31,418	3,129,409	31,418	0	0

Table 7-2 shows a comparison of AEMS and ModeChoice model run times for all trip purposes. The run-time savings of moving to the ModeChoice program is between 57% and 74%, depending on the trip purpose.

Table 7-2: AEMS and ModeChoice Runtime by Trip Purpose

Purpose	AEMS	ModeChoice	Time Saving	Time Saving %
HBW	0:13:30	0:05:46	0:07:44	57%
HBO	0:12:30	0:05:13	0:07:17	58%
HBS	0:10:00	0:03:38	0:06:22	64%
NHW	0:12:45	0:03:55	0:08:50	69%
NHO	0:14:45	0:03:47	0:10:58	74%

7.2.2 Development of Mode Choice Targets

The ModeChoice software includes the option of calibrating model constants based on mode-specific and market-specific trip targets. This tool differs from the AEMS calibration tool (CALIBMS) in its ability to constrain the constants calculated for a given mode and market to a "reasonable" range. The ModeChoice software also adjusts the income constants as part of the calibration process. These two features help avoid an overemphasis of the constant in the model's predictive capabilities. As part of this task, AECOM re-calibrated the mode choice constants using the ModeChoice software to identify how constraining the mode constants and adjusting the income constants would impact the overall mode choice results.

The first step in re-calibrating the mode choice models was to develop reliable mode specific targets for all trip purposes. MWCOG processed the 2007/08 Household Travel Survey (HTS) and transit trips from a series of transit on-board surveys, including the 2008 Metrorail Passenger Survey, and transmitted a set of initial target files to AECOM on 10/23/2013. The target files included trips for 13 modes and four household income levels for HBW, HBO, and HBS. NHW and NHO trips were not split into different income groups. Table 7-3 shows HBW target files provided by MWCOG as a sample.

Table 7-3: Initial HBW Targets Based on 2007/08 HTS for Four Income Levels

HBW	Segment	Inc_1	Inc_2	Inc_3	Inc_4	Total	Inc_1	Inc_2	Inc_3	Inc_4	Total
	SOV ADR	315,506	857,778	883,147	545,181	2,601,612	12.1%	33.0%	33.9%	21.0%	100.0%
	HOV2 ADR	16,393	49,327	46,158	30,430	142,307	11.5%	34.7%	32.4%	21.4%	100.0%
	HOV3+ADR	5,527	16,358	13,270	7,900	43,057	12.8%	38.0%	30.8%	18.3%	100.0%
	Comm_WlkAcc	197	1,646	1,959	109	3,911	5.0%	42.1%	50.1%	2.8%	100.0%
	BusOnly_WlkAcc	29,028	24,352	16,949	4,456	74,786	38.8%	32.6%	22.7%	6.0%	100.0%
	MetBus_AnyAcc	3,974	2,392	2,370	817	9,553	41.6%	25.0%	24.8%	8.6%	100.0%
	MetOnly_WlkAcc	40,435	91,641	83,195	42,671	257,943	15.7%	35.5%	32.3%	16.5%	100.0%
	Comm_PnrKnr	1,990	7,603	14,850	6,607	31,051	6.4%	24.5%	47.8%	21.3%	100.0%
	BusOnly_PnrAcc	1,463	7,852	12,398	9,194	30,906	4.7%	25.4%	40.1%	29.7%	100.0%
	BusOnly_KnrAcc	787	1,183	740	1,463	4,173	18.9%	28.3%	17.7%	35.1%	100.0%
	MetOnly_PnrAcc	3,547	32,610	53,867	32,275	122,300	2.9%	26.7%	44.0%	26.4%	100.0%
	MetOnly_KnrAcc	2,166	8,994	9,824	8,312	29,295	7.4%	30.7%	33.5%	28.4%	100.0%
	Other	57,169	96,948	80,793	52,455	287,365	19.9%	33.7%	28.1%	18.3%	100.0%
	SUM	478,182	1,198,685	1,219,521	741,871	3,638,258	13.1%	32.9%	33.5%	20.4%	100.0%

The MWCOG transmittal also provided the number of weighted trips by primary mode and income level. Table 7-4 shows the total number of HBW trips summarized by primary mode and income level. AECOM used this information, after consulting with MWCOG, to split trips categorized as "Other" in Table 7-3 to "HOV2 ADR", "HOV3 ADR", and "Non-Motorized" trips and to adjust the mode choice targets for all trip purposes.

Table 7-4: Weighted HBW Internal Trips by "Primary" Mode and Income Level

HBW	Income Level>	1	2	3	4	Total	1	2	3	4	Total
	Transit	84,443	181,611	199,065	106,767	571,886	14.8%	31.8%	34.8%	18.7%	100.0%
	AutoDriv	337,426	923,463	942,575	583,511	2,786,976	12.1%	33.1%	33.8%	20.9%	100.0%
	AutoPax	34,301	56,559	41,880	29,867	162,607	21.1%	34.8%	25.8%	18.4%	100.0%
	Walk	19,058	28,951	24,650	11,617	84,275	22.6%	34.4%	29.2%	13.8%	100.0%
	Bike	2,954	8,101	11,351	10,108	32,514	9.1%	24.9%	34.9%	31.1%	100.0%
	SOVAdr	315,506	857,778	883,147	545,181	2,601,612	12.1%	33.0%	33.9%	21.0%	100.0%
	HOV2Adr	16,393	49,327	46,158	30,430	142,307	11.5%	34.7%	32.4%	21.4%	100.0%
	HOV3+Adr	5,527	16,358	13,270	7,900	43,057	12.8%	38.0%	30.8%	18.3%	100.0%
	Total	478,182	1,198,685	1,219,521	741,871	3,638,258	13.1%	32.9%	33.5%	20.4%	100.0%

AECOM's methodology for splitting "Other" trips to other modes revealed that there is a small portion of trips that cannot be categorized into existing modes. Basically the "Other" component did not sum to Auto Passengers, Bike and Walk trips. Table 7-5 shows the unaccounted number of trips for each trip purpose.

Table 7-5: Unaccounted Number of Trips by Trip Purpose

	Other	Auto Pax	Bike	Walk	Difference	Difference Pct
HBW	287,365	162,607	32,514	84,275	7,969	2.77%
HBO	3,013,800	2,396,515	36,399	576,456	4,430	0.15%
HBS	1,067,152	877,567	8,798	180,660	127	0.01%
NHW	551,812	145,469	7,379	396,435	2,529	0.46%
NHO	1,247,404	947,355	6,608	293,130	311	0.02%

Table 7-6 shows the processed HBW mode choice target file. It should be noted that the total number of trips is lower than the total number of HBW trips shown in Table 7-3 because this summary does not

include “Bike”, “Walk” and the difference shown in Table 7-5. Similar tables were produced for other trip purposes.

Table 7-6: HBW Person Trips after Distributing Auto Passenger Trips

HBW	Segment	Inc_1	Inc_2	Inc_3	Inc_4	Total
	SOV ADR	315,506	857,778	883,147	545,181	2,601,612
	HOV2 ADR	32,786	98,654	92,316	60,860	284,616
	HOV3+ADR	23,435	23,590	8,992	7,337	63,354
	Comm_WlkAcc	197	1,646	1,959	109	3,911
	BusOnly_WlkAcc	29,028	24,352	16,949	4,456	74,785
	MetBus_AnyAcc	3,974	2,392	2,370	817	9,553
	MetOnly_WlkAcc	40,435	91,641	83,195	42,671	257,942
	Comm_PnrKnr	1,990	7,603	14,850	6,607	31,050
	BusOnly_PnrAcc	1,463	7,852	12,398	9,194	30,907
	BusOnly_KnrAcc	787	1,183	740	1,463	4,173
	MetOnly_PnrAcc	3,547	32,610	53,867	32,275	122,299
	MetOnly_KnrAcc	2,166	8,994	9,824	8,312	29,296
	SUM	***455,314***	***1,158,295***	***1,180,607***	***719,282***	***3,513,498***

AECOM compared the processed mode choice targets with the input person trip tables to the mode choice model for each trip purpose by income level, except NHW and NHO where there are no income levels, and applied adjustment factors to the targets to ensure input person trip tables are consistent with the mode choice targets. Table 7-7 shows a summary of the comparison and adjustment factors that were applied to the targets. Table 7-8 shows the adjusted HBW target file as an example.

Table 7-7: Summary of Mode Choice Targets vs Input Trip Table by Trip Purpose and Income Level

Purpose		Inc_1	Inc_2	Inc_3	Inc_4	Total
HBW	Input Trip Table	634,125	1,111,528	907,913	1,109,413	3,762,978
	Target	455,314	1,158,295	1,180,607	719,282	3,513,498
	Ratio	1.39	0.96	0.77	1.54	1.07
HBS	Input Trip Table	549,712	871,438	707,461	821,131	2,949,743
	Target	441,532	999,342	984,940	456,151	2,881,965
	Ratio	1.25	0.87	0.72	1.80	1.02
HBO	Input Trip Table	1,041,523	1,994,036	1,705,448	2,117,320	6,858,327
	Target	847,629	2,158,608	2,187,745	1,222,492	6,416,474
	Ratio	1.23	0.92	0.78	1.73	1.07
NHW	Input Trip Table	1,555,825	-	-	-	1,555,825
	Target	1,608,589	-	-	-	1,608,589
	Ratio	0.97	-	-	-	0.97
NHO	Input Trip Table	3,160,826	-	-	-	3,160,826
	Target	2,916,721	-	-	-	2,916,721
	Ratio	1.08	-	-	-	1.08

Table 7-8: Adjusted HBW Internal Trips by Income Level

HBW	Segment	Inc_1	Inc_2	Inc_3	Inc_4	Total
	SOV ADR	439,411	823,145	679,159	840,881	2,782,597
	HOV2 ADR	45,662	94,671	70,993	93,870	305,195
	HOV3+ADR	32,638	22,638	6,915	11,317	73,507
	Comm_WlkAcc	274	1,580	1,507	168	3,529
	BusOnly_WlkAcc	40,428	23,369	13,034	6,873	83,704
	MetBus_AnyAcc	5,535	2,295	1,823	1,260	10,913
	MetOnly_WlkAcc	56,315	87,941	63,979	65,815	274,050
	Comm_PnrKnr	2,772	7,296	11,420	10,191	31,678
	BusOnly_PnrAcc	2,038	7,535	9,534	14,181	33,288
	BusOnly_KnrAcc	1,096	1,135	569	2,257	5,057
	MetOnly_PnrAcc	4,940	31,293	41,425	49,781	127,439
	MetOnly_KnrAcc	3,017	8,631	7,555	12,820	32,023
	SUM	*634,125*	*1,111,528*	*907,913*	*1,109,413*	*3,762,978*

Since the MWCOG mode choice model includes geographic market segments, MWCOG provided mode-specific total trip targets for each of the twenty (20) market segments estimated from the 2007/2008 survey data. The data did not, however, distribute the mode-specific trips to income groups. AECOM used the income distribution generated by the existing mode choice model to estimate the market segment share for each income level and trip purpose. The sum of the market segment trips by mode and income group were compared to the regional total distribution by mode and income group to confirm the overall accuracy of the market segment distributions. Table 7-9 shows a partial sample of HBW targets by income level and geographic market segment. As mentioned earlier, this calibration effort was viewed as a "proof of concept," not a traditional calibration based on complete data from observed sources.

Table 7-9: Partial Sample of HBW Targets by Income Level and Geographic Market Segment

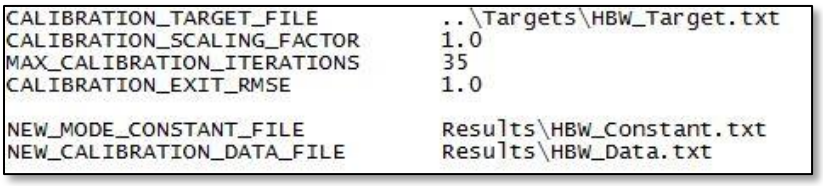
Segment	Mode	Target	HBWI1Psn	HBWI2Psn	HBWI3Psn	HBWI4Psn	Min_Const	Max_Const
1	AUTO	15069	1780	4129	2568	6591	-8	8
1	TRANSIT	135305	39664	45536	26877	23228	-8	8
1	SOV	11111	1039	2958	1989	5125	-8	8
1	HOV	3957	741	1171	579	1466	-8	8
1	SR2	3236	401	961	536	1338	-8	8
1	SR3	721	340	210	43	128	-8	8
1	WALK	126698	38983	43463	25561	18690	-8	8
1	PNR	6806	321	1764	1138	3584	-8	8
1	KNR	1801	360	308	179	954	-8	8
1	WK_CR	0	0	0	0	0	-8	8
1	WK_BUS	26021	9790	8702	4328	3201	-8	8
1	WK_BUS_MR	1078	559	284	166	69	-8	8
1	WK_MR	99599	28634	34477	21067	15421	-8	8
1	PNR_CR	16	0	1	1	14	-8	8
1	PNR_BUS	1957	37	229	191	1500	-8	8
1	PNR_BUS_MR	53	1	8	5	38	-8	8
1	PNR_MR	4780	283	1526	940	2031	-8	8
1	KNR_CR	15	0	1	1	13	-8	8
1	KNR_BUS	252	15	41	13	182	-8	8
1	KNR_BUS_MR	17	1	2	1	14	-8	8
1	KNR_MR	1517	344	265	163	746	-8	8
2	AUTO	1408	303	450	252	402	-8	8
2	TRANSIT	9100	3605	2830	1519	1147	-8	8
2	SOV	1138	211	366	217	345	-8	8
2	HOV	269	92	84	36	57	-8	8
2	SR2	182	40	62	31	49	-8	8
2	SR3	87	53	22	4	8	-8	8

Table 7-9 also includes arbitrary upper and lower bounds on the values of the estimated constants (in this case, -8 and +8). These bounds help to keep the constant values within a reasonable range. One of the concerns in using CALIBMS was that it does not constrain the constant values and often generates values that are illogical and need to be adjusted manually for input to the AEMS program.

7.2.3 Preparation of Mode Choice Control Files

The ModeChoice software implements the calibration process by adding a few control keys to the ModeChoice control file. These control keys identify the target file and specify the maximum number of calibration iterations and the convergence percent RMSE. Figure 7-7 shows the required control keys and their sample values to perform mode choice calibration using ModeChoice.

Figure 7-7: Sample Control Keys in the ModeChoice Control File to Perform Calibration



The calibration process terminates either when it reaches to the maximum number of iterations or the user-defined convergence percent RMSE. The new mode choice constant file includes the resulting constants.

The existing AEMS model includes 400 mode and nesting constants and 1200 income constants. The existing AEMS income constants were asserted and are equal for all geographic market segments and only have values for walk to transit modes (+2 for low income households, 0 for middle income households, and -2 for high income households). These income constants have the effect of increasing the probability (due to the +2.0) that low income travelers will choose walk to transit and decreasing the probability (due to the -2.0) that high income travelers will choose walk to transit.

The ModeChoice model includes the same number of potential constants, but provides the option to calibrate the income constants along with the mode and nesting constants. The ModeChoice income constants are calibrated based on the mode choice targets and can be different for geographic market segments. These income constants have the effect of adjusting the likelihood that travelers of a given income group will choose a given mode. These effects may be different for different origin-destination combinations (i.e., market segments).

The new mode choice constants file generated by the calibration process is used to applying the new model or as input to additional calibration iterations. Table 7-10 shows the HBW constants resulting from the calibration process. This process also generates a data file that compares the estimated trips with targets. The data file mainly shows how closely mode choice results replicate the targets.

Table 7-10: Partial ModeChoice Constant File for HBW by Geographic Market Segment and Income Level

SEGMENT	MODE	CONSTANT	HBWI1Psn	HBWI2Psn	HBWI3Psn	HBWI4Psn
1	AUTO	-4.10187	0	0	0	0
1	TRANSIT	0.457475	0	0	0	0
1	SOV	0.452156	0.086808	-0.047207	-0.115911	-0.021701
1	HOV	-1.26907	0	0	0	0
1	SR2	0.327719	0.019066	-0.002239	-0.052639	0.007548
1	SR3	-1.472459	0.132683	-0.0077	-0.063468	-0.071511
1	WALK	0.559373	0	0	0	0
1	PNR	-6.163959	0	0	0	0
1	KNR	-8	0	0	0	0
1	WK_CR	-8	0	0	0	0
1	WK_BUS	-8	2.050827	-0.111487	-0.388347	-1.546896
1	WK_BUS_MR	-8	2.141045	-0.05426	-0.109891	-1.986867
1	WK_MR	4.355141	0.539211	0.398683	1.380123	-2.715556
1	PNR_CR	0.330064	-0.000004	-0.000019	-0.000834	0.000813
1	PNR_BUS	-0.88912	0.00784	-0.073179	-0.050764	0.147235
1	PNR_BUS_MR	-3.396381	0.000355	0.000255	-0.000725	0.000142
1	PNR_MR	0.402574	0.083632	-0.078456	-0.072367	0.054999
1	KNR_CR	2.842118	-0.000005	0.000056	-0.000606	0.000691
1	KNR_BUS	0.565216	0.006458	-0.012631	-0.022639	0.036693
1	KNR_BUS_MR	-1.731248	0.00036	0.000176	-0.000359	-0.00043
1	KNR_MR	-0.101797	0.100862	-0.018457	-0.043119	-0.126873
2	AUTO	-3.362189	0	0	0	0
2	TRANSIT	0.52012	0	0	0	0
2	SOV	0.451049	0.617618	-0.155096	-0.165504	-0.751373
2	HOV	-1.906077	0	0	0	0
2	SR2	0.275734	0.205458	0.002103	-0.030392	-0.463965
2	SR3	-0.578636	0.331371	-0.057415	-0.142665	-0.48458
2	WALK	0.587861	0	0	0	0
2	PNR	-4.792535	0	0	0	0
2	KNR	-8	0	0	0	0

As can be seen in Table 7-10 some of the estimated constant values have attained the lower-limit value of -8.

Although the mode choice model includes 15 travel mode choices, the top branch of the nesting structure splits daily person trips into two broad modes: auto person trips and transit person trips. Table 7-11 compares estimated person trips (from the calibration) and target person trips for the HBW purpose for the auto mode (shown in blue), the transit mode (shown in green), and the sum of the two modes (shown in gray). The last three columns in the table (shown in gray) show that the target and estimated total person trips are quite close at the geographic segment level. By contrast, the segment-level differences for auto person trips and transit person trips are not as closely matched. This could be due to the way the trips are distributed between the zones and the transit and highway access available to the zones.

Table 7-11: Comparison of HBW Estimated Auto and Transit Trips vs. Targets

Geographic Segment	Auto Target	Auto Trips	Auto Difference	Transit Target	Transit Trips	Transit Difference	Total Target	Total Trips	Total Difference
1	15,068	5,068	(10,000)	135,305	145,306	10,001	150,373	150,374	1
2	1,408	1,813	405	9,097	8,694	(403)	10,505	10,507	2
3	12,219	12,570	351	64,008	63,659	(349)	76,227	76,229	2
4	34,426	37,648	3,222	9,403	6,185	(3,218)	43,829	43,833	4
5	7,826	7,899	73	23,678	23,607	(71)	31,504	31,506	2
6	1,031	1,171	140	1,955	1,816	(139)	2,986	2,987	1
7	22,326	23,447	1,121	12,845	11,725	(1,120)	35,171	35,172	1
8	23,501	23,826	325	3,441	3,117	(324)	26,942	26,943	1
9	5,216	6,064	848	38,422	37,576	(846)	43,638	43,640	2
10	15,703	17,396	1,693	11,470	9,779	(1,691)	27,173	27,175	2
11	24,503	27,147	2,644	19,157	16,510	(2,647)	43,660	43,657	(3)
12	37,263	35,916	(1,347)	2,899	4,248	1,349	40,162	40,164	2
13	177,452	178,547	1,095	120,364	119,270	(1,094)	297,816	297,817	1
14	29,012	29,535	523	16,728	16,205	(523)	45,740	45,740	-
15	226,826	209,538	(17,288)	51,325	68,611	17,286	278,151	278,149	(2)
16	1,196,552	1,196,782	230	21,859	21,629	(230)	1,218,411	1,218,411	-
17	110,368	98,934	(11,434)	55,865	67,300	11,435	166,233	166,234	1
18	50,533	49,644	(889)	27,004	27,893	889	77,537	77,537	-
19	137,427	125,526	(11,901)	25,637	37,537	11,900	163,064	163,063	(1)
20	972,742	973,404	662	11,100	10,438	(662)	983,842	983,842	-
Total	**3,101,402**	**3,061,875**	**(39,527)**	**661,562**	**701,105**	**39,543**	**3,762,964**	**3,762,980**	**16**

It is necessary to maintain the total number of trips in each geographic market segment and income group. The segmented total person trips between the model run and calibration for HBW are shown in Table 7-12. They are segmented by geographic market segments (20) and household income level (4). The total person trips match the targets by income group, but individual geographic segments totals could not be matched. This discrepancy was corrected by scaling the mode targets to the model results at the segment level (e.g. targets for geographic segment 1 and income group 1 were multiplied by a factor 1.24). The mode choice calibration was performed one more time using the new targets.

Table 7-12: Comparison of HBW Targets and ModeChoice Results by Geographic Segments and Income Level

Geographic Segment	Initial Targets				Trips from the Calibration Run				Ratios			
	Total HBWI1Psn	Total HBWI2Psn	Total HBWI3Psn	Total HBWI4Psn	Total HBWI1Psn	Total HBWI2Psn	Total HBWI3Psn	Total HBWI4Psn	HBWI1Psn	HBWI2Psn	HBWI3Psn	HBWI4Psn
1	33,513	44,391	26,130	32,935	41,444	49,664	29,446	29,820	1.24	1.12	1.13	0.91
2	3,422	3,204	1,754	1,875	3,908	3,279	1,771	1,549	1.14	1.02	1.01	0.83
3	16,318	18,560	10,173	13,655	22,928	25,187	14,099	14,015	1.41	1.36	1.39	1.03
4	13,367	13,246	6,287	5,440	17,513	14,188	6,861	5,270	1.31	1.07	1.09	0.97
5	4,121	10,027	7,938	10,434	5,558	9,810	7,378	8,760	1.35	0.98	0.93	0.84
6	690	1,037	722	593	753	967	639	629	1.09	0.93	0.88	1.06
7	5,840	11,001	7,082	8,516	7,694	11,715	7,569	8,194	1.32	1.06	1.07	0.96
8	6,937	9,259	5,318	4,723	7,709	9,247	5,392	4,596	1.11	1.00	1.01	0.97
9	2,848	10,451	10,650	20,373	4,364	12,345	11,948	14,984	1.53	1.18	1.12	0.74
10	3,553	8,814	6,826	8,721	4,320	8,574	6,558	7,724	1.22	0.97	0.96	0.89
11	4,939	12,639	9,939	15,022	6,688	13,709	10,719	12,541	1.35	1.08	1.08	0.83
12	8,726	13,836	9,251	8,840	8,938	13,579	9,217	8,431	1.02	0.98	1.00	0.95
13	21,110	70,064	76,476	100,371	28,074	75,557	77,597	116,589	1.33	1.08	1.01	1.16
14	4,918	11,477	11,587	13,533	5,502	12,316	11,688	16,233	1.12	1.07	1.01	1.20
15	29,365	77,564	70,710	88,258	35,706	79,745	71,493	91,206	1.22	1.03	1.01	1.03
16	267,519	410,101	306,509	305,388	238,084	391,248	294,863	294,217	0.89	0.95	0.96	0.96
17	3,812	24,933	35,238	75,343	7,048	30,704	42,070	86,412	1.85	1.23	1.19	1.15
18	4,379	16,213	18,117	30,445	6,780	18,138	19,446	33,174	1.55	1.12	1.07	1.09
19	8,989	34,643	39,374	69,189	12,496	36,996	41,678	71,893	1.39	1.07	1.06	1.04
20	189,759	310,067	247,833	295,759	168,618	294,565	237,483	283,177	0.89	0.95	0.96	0.96
	634,125	1,111,528	907,913	1,109,413	634,125	1,111,529	907,913	1,109,413	1.00	1.00	1.00	1.00

7.2.4 Mode Choice Calibration Results

AECOM performed the mode choice calibration using the final targets as explained in the previous section. Table 7-13 summarizes recalibrated mode choice model results and compares the results with the existing MWCOG mode choice model (AEMS). The new mode choice model estimates more HBW drive alone trips and fewer walk to bus and bus+Metrorail trips. The park-n-ride and kiss-n ride trips to bus+Metrorail are also reduced significantly. For the other trip purposes, the split between drive alone and shared-ride 3+ trips show the most significant differences.

Table 7-13: Comparison of Recalibrated Mode Choice Model vs. Existing MWCOG Mode Choice Model

	HBW		HBO		HBS		NHW		NHO	
	AEMS	ModeChoice	AEMS	ModeChoice	AEMS	ModeChoice	AEMS	ModeChoice	AEMS	ModeChoice
DR ALONE	67.17%	72.32%	38.41%	37.45%	43.20%	46.55%	68.83%	69.79%	44.62%	46.41%
SR2	8.89%	8.09%	30.96%	29.34%	30.23%	30.34%	13.95%	13.44%	30.90%	30.92%
SR3+	2.98%	2.01%	27.71%	29.97%	25.89%	20.88%	12.39%	7.02%	23.49%	16.90%
WK-CR	0.08%	0.08%	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.00%	0.00%
WK-BUS	5.29%	3.59%	1.29%	1.53%	0.48%	1.33%	1.43%	4.62%	0.42%	4.50%
WK-BU/MR	3.50%	2.11%	0.39%	0.10%	0.07%	0.12%	0.56%	0.40%	0.09%	0.09%
WK-MR	4.46%	4.98%	0.74%	1.03%	0.09%	0.55%	2.08%	4.00%	0.28%	1.00%
PNR-CR	0.78%	0.84%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
KNR-CR	0.07%	0.08%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%
PNR-BUS	0.44%	0.94%	0.04%	0.02%	0.00%	0.02%	0.03%	0.02%	0.05%	0.02%
KNR-BUS	0.12%	0.15%	0.03%	0.04%	0.01%	0.01%	0.10%	0.00%	0.02%	0.03%
PNR-BU/MR	0.83%	0.05%	0.04%	0.00%	0.00%	0.00%	0.06%	0.04%	0.01%	0.01%
KNR-BU/MR	0.26%	0.02%	0.03%	0.00%	0.01%	0.00%	0.07%	0.04%	0.02%	0.02%
PNR-MR	3.89%	3.68%	0.27%	0.29%	0.01%	0.13%	0.26%	0.32%	0.03%	0.05%
KNR-MR	1.24%	1.06%	0.07%	0.21%	0.01%	0.07%	0.22%	0.29%	0.04%	0.06%
TOTAL	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

7.3 Conclusions and Recommendations

The ModeChoice software was able to replicate the results of the AEMS software near perfectly while reducing run times by 50 percent or more. The ModeChoice software offers many additional advantages that recommended it as well. The user interface approach and the expanded calibration features are among the more important.

The initial attempt to re-calibrate the mode choice model using the 2007/2008 on-board transit and Metrorail surveys raised a number of data preparation challenges that should be revisited and researched in more detail. In particular the distribution of incomes by mode and geographic market segment was difficult to estimate accurately. One possible approach to addressing this complication may be to reduce or replace the number of geographic market segments. Migrating to a Pedestrian Environment Factor (PEF) concept may be helpful in this regard. The PEF approach can capture some of the sensitivities represented by the geographic market segments, but is also able to consider how future changes in land-use patterns and development densities impact transit demand.

This page is intentionally blank.

8 Summary of Recommendations

Throughout this document, AECOM identified places where MWCOG may wish to consider focusing additional attention to improve the regional modeling process. This chapter consolidates many of these suggestions for easier reference.

1. AECOM demonstrated that the percent walk to transit process using ArcPy and Cube was successfully integrated with the TPB modeling process while removing the dependency on a full implementation of ArcGIS, but still maintaining compatibility with it.
2. Preparation of other GIS-based transit network inputs to the TPB model, such as PEF (pedestrian environment factor), could be implemented using ArcPy as well. Further investigation into improving the ArcPy processing times may also be worth considering.
3. The HOV choice model proved effective in replicating observed HOV demand for HOV facilities; ensuring that travel speeds for the HOV 3+ traffic on HOT lanes are not degraded by the other traffic using the HOT lanes; and eliminating the need for "multi-run" and "two-step" assignments. This modeling approach is worth serious consideration by COG.
4. The HOT lane analysis used observed data (counts) for the year 2010, which does not have any HOT-lane facilities and therefore is not modeled with a "multi-run" in the COG model. It will be important to re-calibrate the HOV choice model once data from the newly opened HOT lanes becomes available.
5. For application purposes, the HOV choice model and the multi-class assignment procedures were integrated into the current mode choice and assignment procedures. Restructuring of the overall mode choice model within a common software platform (i.e., AEMS → ModeChoice) is needed to integrate the HOV/HOT lane and PT/mode choice enhancements into the COG process.
6. Additional testing of the toll-setting parameters for different future years is recommended to determine the most reliable configuration, most "seasoned" seed tolls. This testing may also include noting the impact/effectiveness of using toll-setting only for the final speed-feedback iteration versus all speed-feedback iterations and/or the impact of using lowered relative gap cutoff thresholds for toll-setting in different speed feedback iterations.
7. The conversion of the COG transit network from TRNBUILD to PT proved successful. Changes to the way COG prepares network and support links and codes some transit routes is necessary, but worth implementing. Improvements to path building and transit access and transfer details will be very useful for detailed transit applications.
8. Despite AECOM efforts to adjust the parameters used by the PT process to date, an extensive calibration effort is recommended where the PT transit path structures, travel times and ridership are compared with the latest on-board transit survey. The calibration process will help to adjust the parameters used by PT to develop non-transit legs and find the best transit path for a given origin-destination pair.

9. AECOM's experience with using the PT process in other travel demand models suggests that an additional step is helpful after transit paths are built for all zone to zone interchanges. A path conditioning step drops transit paths between a pair of origin and destination zones if the total travel time of that path is longer than the walk-only path between the zone pair.
10. The ModeChoice software was able to replicate the results of the AEMS software near perfectly while reducing run times by 50 percent or more. The ModeChoice software offers many additional advantages that recommended it as well. The user interface approach and the expanded calibration features are among the more important.
11. The initial attempt to re-calibrate the mode choice model raised a number of data preparation challenges that should be revisited and researched in more detail. In particular the distribution of incomes by mode and geographic market segment was difficult to estimate accurately. One possible approach to addressing this complication may be to reduce or replace the number of geographic market segments. Migrating to a Pedestrian Environment Factor (PEF) concept may be helpful in this regard. The PEF approach can capture some of the sensitivities represented by the geographic market segments, but is also able to consider how future changes in land-use patterns and development densities impact transit demand.

9 Appendix

The appendix includes longer script files from each of the modeling chapters.

9.1 Cube-Based Transit Walk-Shed Scripts

Figure 9-1: Contents of 'ArcPy_Walkshed_Process.bat' Batch File

```
< Contents of ArcPy_Walkshed_Process.bat >

@ECHO OFF
IF [%1] == [] goto usage
CD %1

SET orig_dir=%CD%
SET WalkshedDirName=Transit_Walksheds_GIS

:: Python Directory
::        Cube 6.1.0 SP1 comes with ArcGIS 10.1 runtime
::        Cube 6.0.2     comes with ArcGIS 10.0 runtime and is not supported here
::        Python 2.6 will support ArcPy upto ArcGIS 10.0. Python 2.7 is needed to support ArcPy
with ArcGIS 10.1.

SET python bindir=0

:: Look in C drive
IF %python_bindir%==0  CALL:CheckPythonPath  C:\Python27\ArcGIS10.2  python_bindir
IF %python_bindir%==0  CALL:CheckPythonPath  C:\Python27\ArcGIS10.1  python_bindir
IF %python_bindir%==0  CALL:CheckPythonPath  C:\Python26\ArcGIS10.0  python_bindir

:: Look in D drive
IF %python_bindir%==0  CALL:CheckPythonPath  D:\Python27\ArcGIS10.2  python_bindir
IF %python_bindir%==0  CALL:CheckPythonPath  D:\Python27\ArcGIS10.1  python_bindir
IF %python_bindir%==0  CALL:CheckPythonPath  D:\Python26\ArcGIS10.0  python_bindir

:: Look in E drive
IF %python bindir%==0  CALL:CheckPythonPath  E:\Python27\ArcGIS10.2  python bindir
IF %python_bindir%==0  CALL:CheckPythonPath  E:\Python27\ArcGIS10.1  python_bindir
IF %python_bindir%==0  CALL:CheckPythonPath  E:\Python26\ArcGIS10.0  python_bindir

:: Python should be found by now
IF %python_bindir%==0  GOTO:error-dependency

:DoSteps
ECHO.
ECHO.
ECHO. Using Python from Directory = %python_bindir%
CD /D %orig_dir%
ECHO.
ECHO.

:: Create directories

ECHO.
ECHO.1) Creating Subdirectories ...
ECHO.

IF NOT EXIST  "inputs\%WalkshedDirName%\input"   MKDIR  "inputs\%WalkshedDirName%\input"
IF NOT EXIST  "inputs\%WalkshedDirName%\output"  MKDIR  "inputs\%WalkshedDirName%\output"

:: Change Working Directory

DEL /F /Q /S  inputs\%WalkshedDirName%\*
CD  /D        inputs\%WalkshedDirName%

:: Prepare Inputs (if *.Lin files exist then use PT else use TRNBUILD)
```

```
ECHO.
ECHO.2) Preparing Inputs ...
ECHO.

if exist "..\Mode1AM.Lin" (
       ECHO.    using PT line files
       start /w Voyager.exe
..\..\..\scripts\MWCOG Prepare Inputs to Walkshed Process PT.s        /start -Pvoya -
S"%orig_dir%\inputs\%WalkshedDirName%"
       if errorlevel 2 goto error
) else (
       ECHO.    using TRNBUILD line files
       start /w Voyager.exe
..\..\..\scripts\MWCOG Prepare Inputs to Walkshed Process TRNBUILD.s  /start -Pvoya -
S"%orig_dir%\inputs\%WalkshedDirName%"
       if errorlevel 2 goto error
)

:: Create Walksheds

ECHO.
ECHO.3) Launching ArcPy-based Walkshed Process ...
ECHO.

%python bindir%\python.exe    ..\..\..\scripts\MWCOG ArcPy Walkshed Process.py
if errorlevel 1 goto error

:: Copy AreaWalk.txt file

ECHO.
ECHO.4) Copying AreaWalk.txt / PercentWalk.txt File(s) ...
ECHO.

:: Backup existing copies as "Old"

IF EXIST ..\AreaWalk.txt        COPY /Y  ..\AreaWalk.txt          ..\AreaWalk_Old.txt
IF EXIST ..\PercentWalk.txt       COPY /Y  ..\PercentWalk.txt       ..\PercentWalk_Old.txt

:: Now install and overwrite existing copies

IF EXIST output\AreaWalk.txt    COPY /Y  output\AreaWalk.txt      ..\.
if errorlevel 1 goto error

IF EXIST output\PercentWalk.txt      COPY /Y  output\PercentWalk.txt   ..\.
if errorlevel 1 goto error

:: Copy Walkshed MXD

ECHO.
ECHO.5) Copying ArcGIS MXD File ...
ECHO.

COPY /Y  ..\..\..\scripts\MWCOG_ArcPy_Walkshed_Process_TEMPLATE.mxd
MWCOG_ArcPy_Walksheds_%1.mxd
if errorlevel 1 goto error

:: Change to Original Directory
CD /D %orig dir%
ECHO.
ECHO.

:: Done

ECHO. Process Complete!
GOTO end

::-------------------------------------------------------------------------------------------------
::-------------------------------------------------------------------------------------------------

:error
ECHO.
```

```
ECHO.
ECHO. ERROR: Error in Walkshed Process
ECHO.
ECHO.
PAUSE
:end
CD..
GOTO:EOF

:usage
ECHO.
ECHO. Error: No Folder Name Provided
ECHO. This batch file requires a folder name.
ECHO.
PAUSE
GOTO:EOF

:error-dependency
ECHO.
ECHO.
ECHO ERROR: DEPENDENCIES NOT SATISFIED (Do you have Cube 6.1.0 SP1 or above (with ArcGIS
runtime)?)
ECHO.
ECHO.
goto error

::-----------------------------------------------------------
::-- BEGIN Function CheckPythonPath
::-----------------------------------------------------------

:CheckPythonPath          -- This function checks and sets python path
::                        -- %~1: Path containing Python.exe
::                        -- %~2: Variable containing status
SET myPath=0

ECHO     Searching for Python in Path %~1
IF EXIST %~1 (
       CD /D %~1
       IF EXIST python.exe (
              SET myPath=%~1
              ECHO     Found Python in Path %~1
       ) ELSE ( SET myPath=0 )
) ELSE ( SET myPath=0 )

SET "%~2=%myPath%"

GOTO:EOF
::-----------------------------------------------------------
::-- END Function CheckPythonPath
::-----------------------------------------------------------
```

Figure 9-2: Cube Script to Process TRNBUILD Line Files to Create Shapefile Inputs to Walkshed Process

```
< Contents of MWCOG_Prepare_Inputs_to_Walkshed_Process.s >

;
; This Voyager script creates shapefile inputs for Walkshed Process
; Outputs are:
;    1) ALLStops_PK.shp         (.dbf, .shx)
;    2) ALLStops_OP.shp         (.dbf, .shx)
;    3) MetroandLRT_AllDay.shp  (.dbf, .shx)
;
; 10/17/2013, Author: KCP with support from Citilabs
;-------------------------------------------------------------------------

; Reference: V2.3.52_Users_Guide_v2_w_appA.pdf Page 113
; Mode 3= Metrorail
```

```
; Mode 4 = Commuter rail
; Mode 5 = light rail
; Mode 10= BRT, Streetcar

;
; Step 1) Create network with all nodes
;
RUN PGM=HWYNET

        FILEI NODEI[1] = "..\Node.dbf"
        FILEI NODEI[2] = "..\Met_Node.tb",
                                        VAR=N,9-14,
                                        VAR=X,18-27,
                                        VAR=Y,31-40
        FILEI NODEI[3] = "..\Com_Node.tb",
                                        VAR=N,9-14,
                                        VAR=X,18-27,
                                        VAR=Y,31-40
        FILEI NODEI[4] = "..\LRT_Node.tb",
                                        VAR=N,9-14,
                                        VAR=X,18-27,
                                        VAR=Y,31-40
        FILEI NODEI[5] = "..\NEW_Node.tb",
                                        VAR=N,9-14,
                                        VAR=X,18-27,
                                        VAR=Y,31-40

        FILEI LINKI[1] = "..\Link.dbf"
        FILEI LINKI[2] = "..\Met_Link.tb",
                                            VAR=A,12-17,
                                            VAR=B,19-24
    FILEI LINKI[3] = "..\Com_Link.tb",
                                            VAR=A,12-17,
                                            VAR=B,19-24
        FILEI LINKI[4] = "..\LRT_Link.tb",
                                            VAR=A,12-17,
                                            VAR=B,19-24
    FILEI LINKI[5] = "..\NEW_Link.tb",
                                            VAR=A,12-17,
                                            VAR=B,19-24

        ZONES          = 3722
        NETO           = "input\temp_HwyNetWithTrnNodes.NET"

ENDRUN

;
; Step 2a) Read TRNBUILD line files and export links (ALLStops_PK)
;
RUN PGM=TRNBUILD

        FILEI NETI     = "input\temp HwyNetWithTrnNodes.NET"
        FILEO LINKO    = "input\temp_LinkALLPK.DBF"

        parameters buildpaths=false
        phase1 hwytime=DISTANCE                                         ;give a link variable
for path cost. This has no effect on the stop node shape file output
        support modes=200 dist=100 N=1-2                                ;generate a dummy
support link. Use a mode which is not used in your transit line file

        READ FILE      = "..\MODE1AM.TB"
        READ FILE      = "..\MODE2AM.TB"
        READ FILE      = "..\MODE3AM.TB"
        READ FILE      = "..\MODE4AM.TB"
        READ FILE      = "..\MODE5AM.TB"
        READ FILE      = "..\MODE6AM.TB"
        READ FILE      = "..\MODE7AM.TB"
        READ FILE      = "..\MODE8AM.TB"
        READ FILE      = "..\MODE9AM.TB"
        READ FILE      = "..\MODE10AM.TB"
```

```
ENDRUN

;
; Step 2b) Read TRNBUILD line files and export links (ALLStops OP)
;
RUN PGM=TRNBUILD

       FILEI NETI     = "input\temp HwyNetWithTrnNodes.NET"
       FILEO LINKO    = "input\temp_LinkALLOP.DBF"

       parameters buildpaths=false
       phase1 hwytime=DISTANCE                                          ;give a link variable
for path cost. This has no effect on the stop node shape file output
       support modes=200 dist=100 N=1-2                                 ;generate a dummy
support link. Use a mode which is not used in your transit line file

       READ FILE      = "..\MODE1OP.TB"
       READ FILE      = "..\MODE2OP.TB"
       READ FILE      = "..\MODE3OP.TB"
       READ FILE      = "..\MODE4OP.TB"
       READ FILE      = "..\MODE5OP.TB"
       READ FILE      = "..\MODE6OP.TB"
       READ FILE      = "..\MODE7OP.TB"
       READ FILE      = "..\MODE8OP.TB"
       READ FILE      = "..\MODE9OP.TB"
       READ FILE      = "..\MODE10OP.TB"

ENDRUN

;
; Step 2c) Read TRNBUILD line files and export links (MetroandLRT_AllDay)
;
RUN PGM=TRNBUILD

       FILEI NETI     = "input\temp_HwyNetWithTrnNodes.NET"
       FILEO LINKO    = "input\temp_LinkMetroLRT_PK.DBF"

       parameters buildpaths=false
       phase1 hwytime=DISTANCE                                          ;give a link variable
for path cost. This has no effect on the stop node shape file output
       support modes=200 dist=100 N=1-2                                 ;generate a dummy
support link. Use a mode which is not used in your transit line file

       READ FILE      = "..\MODE3AM.TB"
       READ FILE      = "..\MODE5AM.TB"

ENDRUN
RUN PGM=TRNBUILD

       FILEI NETI     = "input\temp HwyNetWithTrnNodes.NET"
       FILEO LINKO    = "input\temp_LinkMetroLRT_OP.DBF"

       parameters buildpaths=false
       phase1 hwytime=DISTANCE                                          ;give a link variable
for path cost. This has no effect on the stop node shape file output
       support modes=200 dist=100 N=1-2                                 ;generate a dummy
support link. Use a mode which is not used in your transit line file

       READ FILE      = "..\MODE3OP.TB"
       READ FILE      = "..\MODE5OP.TB"

ENDRUN

;
; Step 3a) Export nodes (ALLStops_PK)
;
RUN PGM=MATRIX

       FILEI RECI     = "input\temp_LinkALLPK.DBF"                      ;LINKO file from
TRNBUILD
       FILEO RECO[1]  = "input\temp_NodeALLPK.DBF" FIELDS=N,STOP
```

```
       STOP=1
       N=RI.A
       IF (RI.MODE>0 & RI.STOP A=1) WRITE RECO=1                          ;Select Mode
       N=RI.B
       IF (RI.MODE>0 & RI.STOP_B=1) WRITE RECO=1

ENDRUN

;
; Step 3b) Export nodes (ALLStops_OP)
;
RUN PGM=MATRIX

       FILEI RECI     = "input\temp_LinkALLOP.DBF"                        ;LINKO file from
TRNBUILD
       FILEO RECO[1]  = "input\temp_NodeALLOP.DBF" FIELDS=N,STOP

       STOP=1
       N=RI.A
       IF (RI.MODE>0 & RI.STOP_A=1) WRITE RECO=1                          ;Select Mode
       N=RI.B
       IF (RI.MODE>0 & RI.STOP_B=1) WRITE RECO=1

ENDRUN

;
; Step 3c) Export nodes (MetroandLRT_AllDay)
;
RUN PGM=MATRIX

       FILEI RECI     = "input\temp_LinkMetroLRT_PK.DBF"                  ;LINKO file from
TRNBUILD
       FILEO RECO[1]  = "input\temp_NodeMetroLRT_PK.DBF" FIELDS=N,STOP

       STOP=1
       N=RI.A
       IF (RI.MODE>0 & RI.STOP_A=1) WRITE RECO=1  ;Select Mode
       N=RI.B
       IF (RI.MODE>0 & RI.STOP_B=1) WRITE RECO=1  ;Select Mode

ENDRUN
RUN PGM=MATRIX

       FILEI RECI     = "input\temp LinkMetroLRT OP.DBF"                  ;LINKO file from
TRNBUILD
       FILEO RECO[1]  = "input\temp_NodeMetroLRT_OP.DBF" FIELDS=N,STOP

       STOP=1
       N=RI.A
       IF (RI.MODE>0 & RI.STOP_A=1) WRITE RECO=1  ;Select Mode
       N=RI.B
       IF (RI.MODE>0 & RI.STOP_B=1) WRITE RECO=1  ;Select Mode

ENDRUN

;
; Step 4a) Use nodes and network to export stop-shapefile (ALLStops PK)
;
RUN PGM=NETWORK

       FILEI LINKI[1] = "input\temp HwyNetWithTrnNodes.NET"               ;Input highway network
       FILEI NODEI[2] = "input\temp_NodeALLPK.DBF"  COMBINE=T             ;Stop node file from
previous MATRIX step
       FILEO NODEO    = "input\ALLStops_PK.shp" FORMAT=SHP                ;stop nodes shape
file.

       MERGE RECORD=T

       PHASE=NODEMERGE
         IF (ni.2.stop=0) delete
```

```
        ENDPHASE

ENDRUN

;
; Cube does not create a projection (.prj) file, so copy from template
;
*COPY ..\..\..\Scripts\Maryland1900Ft ShapefileProjection TEMPLATE.prj    input\ALL Stops PK.prj

;
; Step 4b) Use nodes and network to export stop-shapefile (ALLStops_OP)
;
RUN PGM=NETWORK

        FILEI LINKI[1] = "input\temp_HwyNetWithTrnNodes.NET"              ;Input highway network
        FILEI NODEI[2] = "input\temp NodeALLOP.DBF"  COMBINE=T            ;Stop node file from
previous MATRIX step
        FILEO NODEO    = "input\ALLStops_OP.shp" FORMAT=SHP               ;stop nodes shape
file.

        MERGE RECORD=T

        PHASE=NODEMERGE
          IF (ni.2.stop=0) delete
        ENDPHASE

ENDRUN

;
; Cube does not create a projection (.prj) file, so copy from template
;
*COPY ..\..\..\Scripts\Maryland1900Ft_ShapefileProjection_TEMPLATE.prj    input\ALLStops_OP.prj


;
; Step 4c) Use nodes and network to export stop-shapefile (MetroandLRT_AllDay)
;
RUN PGM=NETWORK

        FILEI LINKI[1] = "input\temp_HwyNetWithTrnNodes.NET"              ;Input highway network
        FILEI NODEI[2] = "input\temp_NodeMetroLRT_PK.DBF"  COMBINE=T      ;Stop node file from
previous MATRIX step
        FILEI NODEI[3] = "input\temp_NodeMetroLRT_OP.DBF"  COMBINE=T      ;Stop node file from
previous MATRIX step
        FILEO NODEO    = "input\MetroandLRT AllDay.shp" FORMAT=SHP        ;stop nodes shape
file.

        MERGE RECORD=T

        PHASE=NODEMERGE
          IF (ni.2.stop=0) delete
          IF (ni.3.stop=0) delete
        ENDPHASE

ENDRUN

;
; Cube does not create a projection (.prj) file, so copy from template
;
*COPY ..\..\..\Scripts\Maryland1900Ft_ShapefileProjection_TEMPLATE.prj
input\MetroandLRT_AllDay.prj
```

This page is intentionally blank.

Figure 9-3: ArcPy Based Python Script for Walkshed Process

```
< Contents of MWCOG_ArcPy_Walkshed_Process.py >

# -*- coding: utf-8 -*-
'''

See Supported Configurations

       ESRI ArcGIS             Citilabs CUBE                                ArcPySupported?      Result | Machine
       ------------            ---------------------------------------      ---------------      ----------------
    1) ArcGIS 10.1             CUBE 6.1.0 SP1 (without ArcGIS runtime)       YES                 Tested OK
    2) ArcGIS 10.0             CUBE 6.1.0 SP1 (without ArcGIS runtime)       *NO*                BUG in ArcPy with ArcGIS 10.0
(Join doesn't work)
    3) ArcGIS  9.3             CUBE 6.1.0 SP1 (without ArcGIS runtime)       *NO*

    4) ArcGIS 10.1             CUBE 6.0.2     (without ArcGIS runtime)       YES                 Tested OK
    5) ArcGIS 10.0             CUBE 6.0.2     (without ArcGIS runtime)       *NO*                BUG in ArcPy with ArcGIS 10.0
(Join doesn't work)
    6) ArcGIS  9.3             CUBE 6.0.2     (without ArcGIS runtime)       *NO*

    7) No ArcGIS               CUBE 6.1.0 SP1 (with ArcGIS runtime)          YES                 Tested OK
    8) No ArcGIS               CUBE 6.0.2     (with ArcGIS runtime 9.3.1 SP2)  *NO*

'''

__version__ = '2.3.52-2'      ## Does not impact execution. Used for reference only. Version is a suffix to TPB Model Version

  all   = [
    ]

#
# Revisions:
#       2014-03-24 - Created Script - KCP
#       2014-06-30 - Added TAZ sort - KCP
#       ..........
#

#
# import 'sys' to update search path
#
import sys

#
# include paths for ArcGIS 10.2 full installation (use forward slashes '/' in place of Windows default backward slashes '\', otherwise
use two backward slahes '\\')
#
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Desktop10.2\\arcpy")
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Desktop10.2\\bin")
```

```
#
# include paths for ArcGIS 10.1 full installation (use forward slashes '/' in place of Windows default backward slashes '\', otherwise
use two backward slahes '\\')
#
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Desktop10.1\\arcpy")
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Desktop10.1\\bin")

#
# include paths for ArcGIS 10.2 runtime installation (use forward slashes '/' in place of Windows default backward slashes '\',
otherwise use two backward slahes '\\')
#
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Engine10.2\\arcpy")
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Engine10.2\\bin")

#
# include paths for ArcGIS 10.1 runtime installation (use forward slashes '/' in place of Windows default backward slashes '\',
otherwise use two backward slahes '\\')
#
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Engine10.1\\arcpy")
sys.path.append("C:\\Program Files (x86)\\ArcGIS\\Engine10.1\\bin")

#
# now import arcpy (with updated path variables) and other modules
#
try:
    import arcpy
    from arcpy import env
    import os
    import datetime
    import traceback
except ImportError:
    MWCOG_PrintWriter( "ERROR: unable to import required modules!" )
    sys.exit(1)

#
# Global variables:
#

calc type                                   = "AreaWalk"             # Options are 'PercentWalk' or 'AreaWalk'
                                                                           # 'PercentWalk' will result in integer percent-of-zone-
that-is-walkable-to-transit
                                                                           # 'AreaWalk'    will result in decimal square-miles-of-
zone-that-is-walkable-to-transit

dist short                                  = "2640 Feet"
dist_long                                   = "5280 Feet"

my_TAZ_name                                 = "tazID"                    # MUST Exist in Input TAZ,            INTEGER
my_TAZ_area_name                            = "TAZ_Area"                 # Used if exists otherwise created,   DOUBLE in Square
Feet   (Do not set to 'Shape Area')
User_Input_TAZLandArea_Shapefile            = os.path.abspath("..\\..\\..\\TPBTAZ3722_TPBMod\\TAZLand3722.shp")
```

```
my_join_uniq_id                              = 1                           # If Input TAZ shapefile has 'Shape_Area' already in it,
make this '1', else '0'

my GDB type                                  = "PERSONAL"                  # Options are 'FILE' or 'PERSONAL'
my_GDB_name                                  = "Walkshed_Geodatabase.mdb"  # for FILE use '.gdb' for PERSONAL use '.mdb'

# NOTE: Currently there is an issue with ArcGIS 10.1 which reports 'workspace or data source is read only' after creating a file geo-
database on a network drive.
#       So use only 'PERSONAL' geo-database option.
# my GDB type                                  = "FILE"                      # Options are 'FILE' or 'PERSONAL',
# my_GDB_name                                  = "Walkshed_Geodatabase.gdb"  # for FILE use '.gdb' for PERSONAL use '.mdb'

my_PWT_folder                                = os.path.abspath(".")
my inp folder                                = my PWT folder + "\\input"
my_out_folder                                = my_PWT_folder + "\\output"
my tmp folder                                = my PWT folder + "\\output"
my_GDB                                       = my_tmp_folder + "\\" + my_GDB_name

User_Input_All_PK_All_Stops_Shapefile        = my_inp_folder + "\\" + "ALLStops_PK.shp"
User Input All OP All Stops Shapefile        = my inp folder + "\\" + "ALLStops OP.shp"
User_Input_All_DY_MetroLRT_Stops_Shapefile   = my_inp_folder + "\\" + "MetroandLRT_AllDay.shp"
User Output Walkshed CSV File                = my out folder + "\\" + calc type + ".csv"
User_Output_Walkshed_TXT_File                = my_out_folder + "\\" + calc_type + ".txt"
User_Output_Walkshed_Report_File             = my_out_folder + "\\" + calc_type + "_REPORT.prn"

my_out_fields                                = [my_TAZ_name, my_TAZ_area_name]  # What fields to carry over to final output
my dissolve Fields                           = [my TAZ name]                    # What field to perform dissolve
my_join_Fieldlist                            = [ "Shape_Area" ]                 # This is automatic geometry field created for every
shapefile in geodatabase

my buffer sideType                           = "FULL"
my_buffer_endType                            = "ROUND"
my buffer dissolveType                       = "NONE"
my_buffer_dissolveField                      = ""

my_temp_buffer_dissolve_field_name           = "Buf_Dis"

my_pctwlk_calcfield_codeblock                = """def CalcAttribute(tazarea, shedarea):
                                                   if (shedarea):
                                                       shedarea = (shedarea/(5280*5280))
                                                       if (shedarea > tazarea):
                                                           return round(100,8)
                                                       else:
                                                           return round((shedarea/tazarea)*100,8)
                                                   else:
                                                       return round(0,8)"""

my_arawlk_calcfield_codeblock                = """def CalcAttribute(tazarea, shedarea):
                                                   if (shedarea):
                                                       shedarea = (shedarea/(5280*5280))
                                                       if (shedarea > tazarea):
```

```
                                                    return round(tazarea,8)
                                                else:
                                                    return round(shedarea,8)
                                            else:
                                                return round(0,8)"""

my_rightstringlength                        = 75  # Maximum length of the string for displaying full paths

##########################################################################################################################
## Sub: MWCOG BufferAndExport
##########################################################################################################################

def MWCOG_BufferAndExport(point_shape, temp_prefix_short, temp_prefix_long, startfresh, export):

    global my_out_fields
    global my join uniq id

    #
    # Step 1: Message
    #

    mystarttime   = datetime.datetime.now()
    myendtime     = None
    myelapsedtime = None
    MWCOG PrintWriter( "\n\tMWCOG BufferAndExport (Started: " + str(mystarttime).split(".")[0] + ")" )
    MWCOG_PrintWriter( "\tShapefile  : " + str( ('..' + point_shape[(len(point_shape)-my_rightstringlength):]) if len(point_shape) >
my rightstringlength else point shape) )
    MWCOG_PrintWriter( "\tShort Walk : " + str(dist_short) + ", Field = " + str(temp_prefix_short) )
    MWCOG_PrintWriter( "\tLong Walk  : " + str(dist_long)  + ", Field = " + str(temp_prefix_long) )

    #
    # Step 2: Create temporary geodatabase
    #

    if (startfresh == 1):
        if (arcpy.Exists(my_GDB)) :
            MWCOG PrintWriter( "\tdeleting geo-database ..." )
            arcpy.Delete_management(my_tmp_folder + "\\" + my_GDB_name)
        if (my GDB type == "PERSONAL"):
                MWCOG_PrintWriter( "\tcreating personal geo-database ..." )
                arcpy.CreatePersonalGDB_management(my_tmp_folder, my_GDB_name, "10.0")
        else:
                MWCOG PrintWriter( "\tcreating file geo-database ..." )
                arcpy.CreateFileGDB_management(my_tmp_folder, my_GDB_name, "10.0")
    else:
        if (arcpy.Exists(my_GDB)) : pass
        else:
                if (my_GDB_type == "PERSONAL"):
                        MWCOG_PrintWriter( "\tcreating personal geo-database ..." )
                        arcpy.CreatePersonalGDB management(my tmp folder, my GDB name, "10.0")
                else:
```

```
                        MWCOG_PrintWriter( "\tcreating file geo-database ..." )
                        arcpy.CreateFileGDB management(my tmp folder, my GDB name, "10.0")

    #
    # Step 3: Set Workspace(s)
    #

    arcpy.env.scratchWorkspace = my GDB

    ## if (startfresh == 1):
    ##     MWCOG_PrintWriter( "\tlisting environment variables:" )
    ##     environments = arcpy.ListEnvironments()
    ##     for environment in environments:
    ##             envSetting = getattr(env, environment)
    ##             MWCOG_PrintWriter( "\t\t%-28s: %s" % (environment, envSetting) )

    #
    # Step 4: Work
    #

    try:

        #
        # Construct paths
        #
        temp_TAZ      = my_GDB        + "\\" + "TAZ_with_PercentWalkSheds"
        sort TAZ      = my GDB        + "\\" + "TAZ with PercentWalkSheds sorted"  # This is the TAZ-sorted version of above for
export

        short_Temp1   = my_GDB        + "\\" + str(temp_prefix_short) + "Temp1"
        short Temp2   = my GDB        + "\\" + str(temp prefix short) + "Temp2"
        short_Temp3   = my_GDB        + "\\" + str(temp_prefix_short) + "Temp3"
        short Temp4   = my GDB        + "\\" + str(temp prefix short)
        short_OutFile = my_out_folder + "\\" + str(temp_prefix_short) + ".csv"

        long_Temp1    = my_GDB        + "\\" + str(temp_prefix_long)  + "Temp1"
        long Temp2    = my GDB        + "\\" + str(temp prefix long)  + "Temp2"
        long_Temp3    = my_GDB        + "\\" + str(temp_prefix_long)  + "Temp3"
        long Temp4    = my GDB        + "\\" + str(temp prefix long)
        long_OutFile  = my_out_folder + "\\" + str(temp_prefix_long)  + ".csv"

        #
        # Delete Existing Outputs
        #
        MWCOG PrintWriter( "\tinitializing outputs ..." )

        if (startfresh == 1):

                # if starting afresh, copy TAZ layer into geodatabase and compute area
                if arcpy.Exists(temp TAZ)  : arcpy.Delete management(temp TAZ)
                if arcpy.Exists(sort_TAZ)  : arcpy.Delete_management(sort_TAZ)
```

```
                arcpy.CopyFeatures_management(User_Input_TAZLandArea_Shapefile, temp_TAZ)

                # check if area field exists else compute
                my fieldList = arcpy.ListFields(temp TAZ)

                my_fieldexists = False
                for my_field in my_fieldList:
                        if my field.name == my TAZ area name:
                                my_fieldexists = True
                                MWCOG PrintWriter( "\t\tfound/using existing TAZ area field: " + my TAZ area name )
                                break

                if (my_fieldexists): pass
                else:
                        MWCOG_PrintWriter( "\t\tcreating TAZ area field: " + my_TAZ_area_name )
                        arcpy.AddField management(temp TAZ, my TAZ area name, "DOUBLE", 14, "", "", my TAZ area name, "NULLABLE",
"NON_REQUIRED")
                        arcpy.CalculateField_management( temp_TAZ, my_TAZ_area_name, "!shape.area@SQUAREFEET!", "PYTHON", "")

        # delete old files
        if arcpy.Exists(short_Temp1)   : arcpy.Delete_management(short_Temp1)
        if arcpy.Exists(short Temp2)   : arcpy.Delete management(short Temp2)
        if arcpy.Exists(short_Temp3)   : arcpy.Delete_management(short_Temp3)
        if arcpy.Exists(short_Temp4)   : arcpy.Delete_management(short_Temp4)
        if arcpy.Exists(short OutFile) : arcpy.Delete management(short OutFile)

        if arcpy.Exists(long Temp1)    : arcpy.Delete management(long Temp1)
        if arcpy.Exists(long_Temp2)    : arcpy.Delete_management(long_Temp2)
        if arcpy.Exists(long_Temp3)    : arcpy.Delete_management(long_Temp3)
        if arcpy.Exists(long_Temp4)    : arcpy.Delete_management(long_Temp4)
        if arcpy.Exists(long OutFile)  : arcpy.Delete management(long OutFile)

        #
        # Process: Buffer & Compact database
        #
        MWCOG_PrintWriter( "\tbuffering ..." )

        arcpy.Buffer_analysis(point_shape, short_Temp1, dist_short, my_buffer_sideType, my_buffer_endType, my_buffer_dissolveType,
my buffer dissolveField)
        arcpy.Compact_management( my_GDB )

        arcpy.Buffer_analysis(point_shape,  long_Temp1,  dist_long, my_buffer_sideType, my_buffer_endType, my_buffer_dissolveType,
my buffer dissolveField)
        arcpy.Compact_management( my_GDB )

        #
        # Process: Add a field to dissolve on
        #
        MWCOG_PrintWriter( "\tadding a field for dissolving split buffers ..." )
```

```
        arcpy.AddField_management(short_Temp1, my_temp_buffer_dissolve_field_name, "SHORT", 1, "", "",
my temp buffer dissolve field name, "NULLABLE", "NON REQUIRED")
        arcpy.CalculateField_management(short_Temp1, my_temp_buffer_dissolve_field_name, "0", "PYTHON", "")

        arcpy.AddField_management( long_Temp1, my_temp_buffer_dissolve_field_name, "SHORT", 1, "", "",
my_temp_buffer_dissolve_field_name, "NULLABLE", "NON_REQUIRED")
        arcpy.CalculateField_management( long_Temp1, my_temp_buffer_dissolve_field_name, "0", "PYTHON", "")

        #
        # Process: Dissolve 1
        #
        MWCOG_PrintWriter( "\tdissolving any split buffers ..." )

        arcpy.Dissolve management(short Temp1, short Temp2, "Buf Dis", "", "MULTI PART", "DISSOLVE LINES")
        arcpy.Compact_management( my_GDB )

        arcpy.Dissolve_management( long_Temp1,  long_Temp2, "Buf_Dis", "", "MULTI_PART", "DISSOLVE_LINES")
        arcpy.Compact_management( my_GDB )

        #
        # Process: Intersect
        #
        MWCOG_PrintWriter( "\tintersecting ..." )

        arcpy.Intersect analysis("'" + short Temp2 + "'" + " #;" + "'" + temp TAZ + "'" + " #", short Temp3, "NO FID", "", "INPUT")
        arcpy.Compact_management( my_GDB )

        arcpy.Intersect_analysis("'" + long_Temp2 + "'" + " #;" + "'" + temp_TAZ + "'" + " #", long_Temp3, "NO_FID", "", "INPUT")
        arcpy.Compact_management( my_GDB )

        #
        # Process: Dissolve 2
        #
        MWCOG_PrintWriter( "\tdissolving ..." )

        arcpy.Dissolve_management(short_Temp3, short_Temp4, my_dissolve_Fields, "", "MULTI_PART", "DISSOLVE_LINES")
        arcpy.Compact management( my GDB )

        arcpy.Dissolve management( long Temp3,  long Temp4, my dissolve Fields, "", "MULTI PART", "DISSOLVE LINES")
        arcpy.Compact_management( my_GDB )

        #
        # Process: Join Short-Walk to Zone Layer
        #
        MWCOG PrintWriter( "\tcomputing short-walk (" + calc type + ") ..." )

        # join
        arcpy.JoinField_management(temp_TAZ, my_TAZ_name, short_Temp4, my_TAZ_name, my_join_Fieldlist) # Help:
http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//001700000065000000

        # construct shape_area name
```

```
        if   my_join_uniq_id == 0:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape Area"                              # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 1:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape Area 1"                            # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 2:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12"                           # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 3:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape Area 12 13"                        # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 4:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape Area 12 13 14"                     # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 5:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12_13_14_15"                  # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 6:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape Area 12 13 14 15 16"               # Quirky, but this is how ArcGIS appends fields
        else: my_shape_area_field = "UNDEFINED"

        my_join_uniq_id += 1

        # calculate percent walk
        my calcfield expression = "CalcAttribute( !" + my TAZ area name + "!, !" + my shape area field + "! )"
        arcpy.AddField_management(temp_TAZ, temp_prefix_short, "DOUBLE", "", "", "", "", "NULLABLE", "REQUIRED", "")

        # select based on calculation type
        if   (calc_type == "PercentWalk") :
                arcpy.CalculateField management(temp TAZ, temp prefix short, my calcfield expression, "PYTHON",
my_pctwlk_calcfield_codeblock)
        elif (calc type == "AreaWalk") :
                arcpy.CalculateField_management(temp_TAZ, temp_prefix_short, my_calcfield_expression, "PYTHON",
my_arawlk_calcfield_codeblock)
        else:
                arcpy.AddError("ERROR: Un-recognized calc type specified!")

        # make a note of this field
        my_out_fields.append(temp_prefix_short)

        #
        # Process: Join Long-Walk to Zone Layer
        #
        MWCOG PrintWriter( "\tcomputing long-walk (" + calc type + ") ..." )

        # join
        arcpy.JoinField_management(temp_TAZ, my_TAZ_name, long_Temp4, my_TAZ_name, my_join_Fieldlist) # Help:
http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//001700000065000000

        # construct shape area name
        if   my_join_uniq_id == 0:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area"                              # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 1:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_1"                            # Quirky, but this is how ArcGIS appends fields
        elif my join uniq id == 2:                                          # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12"                           # Quirky, but this is how ArcGIS appends fields
```

```
        elif my_join_uniq_id == 3:                                                  # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12_13"                                # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 4:                                                  # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12_13_14"                             # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 5:                                                  # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12_13_14_15"                          # Quirky, but this is how ArcGIS appends fields
        elif my_join_uniq_id == 6:                                                  # Quirky, but this is how ArcGIS appends fields
            my_shape_area_field = "Shape_Area_12_13_14_15_16"                       # Quirky, but this is how ArcGIS appends fields
        else: my_shape_area_field = "UNDEFINED"

        my_join_uniq_id += 1

        # calculate percent walk
        my_calcfield_expression = "CalcAttribute( !" + my_TAZ_area_name + "!, !" + my_shape_area_field + "! )"
        arcpy.AddField_management(temp_TAZ, temp_prefix_long, "DOUBLE", "", "", "", "", "NULLABLE", "REQUIRED", "")

        # select based on calculation type
        if   (calc_type == "PercentWalk") :
                arcpy.CalculateField_management(temp_TAZ, temp_prefix_long, my_calcfield_expression, "PYTHON",
my_pctwlk_calcfield_codeblock)
        elif (calc_type == "AreaWalk") :
                arcpy.CalculateField_management(temp_TAZ, temp_prefix_long, my_calcfield_expression, "PYTHON",
my_arawlk_calcfield_codeblock)
        else:
                arcpy.AddError("ERROR: Un-recognized calc type specified!")

        # make a note of this field
        my_out_fields.append(temp_prefix_long)

        #
        # Process: Export Feature Attribute to ASCII...
        #
        if (export==1):

                MWCOG_PrintWriter( "\tsorting final output by TAZ ..." )
                arcpy.Sort_management(temp_TAZ, sort_TAZ, [[my_TAZ_name, "ASCENDING"]])

                MWCOG_PrintWriter( "\twriting out " + calc_type + " CSV file ..." )
                if arcpy.Exists(User_Output_Walkshed_CSV_File)  : arcpy.Delete_management(User_Output_Walkshed_CSV_File)
                arcpy.ExportXYv_stats(sort_TAZ, my_out_fields, "COMMA", User_Output_Walkshed_CSV_File, "ADD_FIELD_NAMES")

                MWCOG_PrintWriter( "\twriting out " + calc_type + " TXT file ..." )
                if arcpy.Exists(User_Output_Walkshed_TXT_File)  : arcpy.Delete_management(User_Output_Walkshed_TXT_File)

                # list fields
                fieldList = arcpy.ListFields(User_Output_Walkshed_CSV_File)

                # skip the fields 1,2 & 5  'XCoord, YCoord, TAZ, TAZ_AREA, MTLRTSHR, MTLRTLNG, ALLPKSHR, ALLPKLNG, ALLOPSHR, ALLOPLNG'

                FieldsToSkip = ['XCoord', 'YCoord']
```

```
# open input and process

i = 1
f = open(User Output Walkshed TXT File,'w')

# variables for statistics

my num taz                      = 0
my_total_taz_area               = 0
my total taz walk area          = 0
my_total_taz_walk_pk_area       = 0
my_total_taz_walk_op_area       = 0

my total mtlrt shr area         = 0
my_total_mtlrt_lng_area         = 0
my total allpk shr area         = 0
my_total_allpk_lng_area         = 0
my_total_allop_shr_area         = 0
my_total_allop_lng_area         = 0

my_taz_list_zero_land_area      = []
my taz list zero walk area      = []
my_taz_shr_list_pk_less_than_op = []
my_taz_lng_list_pk_less_than_op = []

# write header
for field in fieldList:
     if field.name in FieldsToSkip: i += 1
     else:
           if field.name == my_TAZ_name.upper() : f.write('%6s'    % field.name)
           elif i < len(fieldList)              : f.write('%10s'   % field.name)
           else                                 : f.write('%10s\n' % field.name)
           i += 1

# write (copy) data
rows = arcpy.SearchCursor(User_Output_Walkshed_CSV_File)
for row in rows:

     i = 1
     my_mtlrt_shr = 0
     my_mtlrt_lng = 0
     my_allpk_shr = 0
     my allpk lng = 0
     my_allop_shr = 0
     my allop lng = 0

     for field in fieldList:
          if field.name in FieldsToSkip: i += 1
          else:
                # write fields
```

```
                                if   field.name == my_TAZ_name.upper()      : f.write('%6d'      %
row.getValue(field.name) )
                                elif field.name == my_TAZ_area_name.upper() : f.write('%10.4f'   % round( (row.getValue(field.name) /
(5280*5280)), 4) )
                                elif i < len(fieldList)                     : f.write('%10.4f'   % round(
row.getValue(field.name), 4) )
                                else                                        : f.write('%10.4f\n' % round(
row.getValue(field.name), 4) )
                                i += 1

                                # save field value for checks
                                if( field.name == "MTLRTSHR" ) : my_mtlrt_shr = round( row.getValue(field.name), 4)
                                if( field.name == "MTLRTLNG" ) : my_mtlrt_lng = round( row.getValue(field.name), 4)
                                if( field.name == "ALLPKSHR" ) : my allpk shr = round( row.getValue(field.name), 4)
                                if( field.name == "ALLPKLNG" ) : my_allpk_lng = round( row.getValue(field.name), 4)
                                if( field.name == "ALLOPSHR" ) : my allop shr = round( row.getValue(field.name), 4)
                                if( field.name == "ALLOPLNG" ) : my_allop_lng = round( row.getValue(field.name), 4)

                        # update stats on fields

                        my_num_taz                                                              += 1
                        my total taz area                                                       += round(
(row.getValue(my_TAZ_area_name.upper()) / (5280*5280)), 4)

                        if( row.getValue(my TAZ area name.upper()) == 0 ):
                           my_taz_list_zero_land_area.append( str( row.getValue(my_TAZ_name.upper()) ) )

                        if( (my_mtlrt_shr + my_mtlrt_lng + my_allpk_shr + my_allpk_lng + my_allop_shr + my_allop_lng) == 0 ):
                           my_taz_list_zero_walk_area.append( str( row.getValue(my_TAZ_name.upper()) ) )

                        my total mtlrt shr area                                                 +=  my mtlrt shr
                        my_total_mtlrt_lng_area                                                 +=  my_mtlrt_lng
                        my total allpk shr area                                                 +=  my allpk shr
                        my_total_allpk_lng_area                                                 +=  my_allpk_lng
                        my_total_allop_shr_area                                                 +=  my_allop_shr
                        my_total_allop_lng_area                                                 +=  my_allop_lng

                        if( my_allpk_shr < my_allop_shr ): my_taz_shr_list_pk_less_than_op.append( str( row.getValue(my_TAZ_name.upper())
) )
                        if( my_allpk_lng < my_allop_lng ): my_taz_lng_list_pk_less_than_op.append( str( row.getValue(my_TAZ_name.upper())
) )

                        my total taz walk area    += ( max( my mtlrt lng, my allpk lng, my allop lng ) )
                        my_total_taz_walk_pk_area += ( max( my_mtlrt_lng, my_allpk_lng               ) )
                        my total taz walk op area += ( max( my mtlrt lng,              my allop lng ) )

                del rows
                f.close()

                # report stats on fields
                if( calc_type == "AreaWalk" ):
```

```
                        MWCOG_PrintWriter( "\nSUMMARY REPORT:"
)
                        MWCOG_PrintWriter( "\n\tNumber of TAZ Records in Output                                    : " +
str('{:,}'.format(my num taz))
)
                        MWCOG_PrintWriter( "\tTotal TAZ LAND Area                                               : " + str('{:9.4f} sq.
mi.'.format(my_total_taz_area))                                                                                    )
                        if( len(my taz list zero land area) == 0 ) :
                                MWCOG_PrintWriter( "\tTAZs with Zero LAND Area                                        : NONE"
)
                        else:
                                MWCOG_PrintWriter( "\tTAZs with Zero LAND Area                                        : " + "(Count="
+ str(len(my_taz_list_zero_land_area)) + ") " + ','.join(sorted(my_taz_list_zero_land_area, key=int))                )
                        MWCOG PrintWriter( "\n\tTotal TAZ Long-Walk Area                                  : " + str('{:9.4f}
sq. mi. ({:6.2%} of TAZ Land Area)'.format(my_total_taz_walk_area, my_total_taz_walk_area/my_total_taz_area))              )
                        MWCOG PrintWriter( "\tTotal TAZ Long-Walk (Peak Period) Area                          : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my_total_taz_walk_pk_area, my_total_taz_walk_pk_area/my_total_taz_area))          )
                        MWCOG_PrintWriter( "\tTotal TAZ Long-Walk (Off-Peak Period) Area                        : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my_total_taz_walk_op_area, my_total_taz_walk_op_area/my_total_taz_area))          )
                        if( len(my taz list zero walk area) == 0 ) :
                                MWCOG_PrintWriter( "\tTAZs with Zero WALK Area                                        : NONE"
)
                        else:
                                MWCOG_PrintWriter( "\tTAZs with Zero WALK Area                                        : " + "(Count="
+ str(len(my taz list zero walk area)) + ") " + ','.join(sorted(my taz list zero walk area, key=int))                )
                        MWCOG_PrintWriter( "\n\tTotal MTLRTSHR Area                                        : " + str('{:9.4f}
sq. mi. ({:6.2%} of TAZ Land Area)'.format(my total mtlrt shr area, my total mtlrt shr area/my total taz area))            )
                        MWCOG_PrintWriter( "\tTotal MTLRTLNG Area                                         : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my_total_mtlrt_lng_area, my_total_mtlrt_lng_area/my_total_taz_area))              )
                        MWCOG_PrintWriter( "\tTotal ALLPKSHR Area                                          : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my total allpk shr area, my total allpk shr area/my total taz area))              )
                        MWCOG_PrintWriter( "\tTotal ALLPKLNG Area                                          : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my total allpk lng area, my total allpk lng area/my total taz area))              )
                        MWCOG_PrintWriter( "\tTotal ALLOPSHR Area                                          : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my_total_allop_shr_area, my_total_allop_shr_area/my_total_taz_area))              )
                        MWCOG_PrintWriter( "\tTotal ALLOPLNG Area                                          : " + str('{:9.4f} sq.
mi. ({:6.2%} of TAZ Land Area)'.format(my total allop lng area, my total allop lng area/my total taz area))              )
                        if( len(my_taz_shr_list_pk_less_than_op) == 0 ) :
                                MWCOG PrintWriter( "\n\tTAZs with Short-Walk Less in Peak Period than in Off-Peak Period : NONE" )
                        else:
                                MWCOG_PrintWriter( "\n\tTAZs with Short-Walk Less in Peak Period than in Off-Peak Period : " +
"(Count=" + str(len(my_taz_shr_list_pk_less_than_op)) + ") " + ','.join(sorted(my_taz_shr_list_pk_less_than_op, key=int))      )
                        if( len(my taz lng list pk less than op) == 0 ) :
                                MWCOG_PrintWriter( "\tTAZs with  Long-Walk Less in Peak Period than in Off-Peak Period : NONE\n" )
                        else:
                                MWCOG_PrintWriter( "\tTAZs with  Long-Walk Less in Peak Period than in Off-Peak Period : " + "(Count="
+ str(len(my_taz_lng_list_pk_less_than_op)) + ") " + ','.join(sorted(my_taz_lng_list_pk_less_than_op, key=int)) + "\n" )
    except:
        #
        # Get the traceback object
        #
```

```
        tb = sys.exc_info()[2]
        tbinfo = traceback.format tb(tb)[0]

        #
        # Concatenate information together concerning the error into a message string
        #
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" + str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n\n\n"

        #
        # Return python error messages for use in script tool or Python Window
        #
        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)

        return 1

    myendtime     = datetime.datetime.now()
    myelapsedtime = myendtime - mystarttime
    MWCOG PrintWriter( "\tMWCOG BufferAndExport (Finished: " + str(myendtime).split(".")[0] + ", Elapsed: " +
str(myelapsedtime).split(".")[0] + ")" )
    return 0

################################################################################################################################
## Sub: MWCOG BufferAndExport END
################################################################################################################################

################################################################################################################################
## Sub: MWCOG_PrintWriter
################################################################################################################################

def MWCOG_PrintWriter( message ):

    # global report file handle
    global my_report_file_handle

    # show message on screen
    print ( str(message) )

    # write message to report file
    my_report_file_handle.write( str(message) + '\n' )

################################################################################################################################
## Sub: MWCOG_PrintWriter END
################################################################################################################################

################################################################################################################################
## Main: Begin
################################################################################################################################

if __name__ == '__main__':
```

```
    mystarttime   = datetime.datetime.now()
    myendtime     = None
    myelapsedtime = None

    my_report_file_handle = open(User_Output_Walkshed_Report_File, 'w', 0)

    MWCOG PrintWriter ("BEGIN ArcPy Walkshed Process (Started: " + str(mystarttime).split(".")[0] + ")")

    # Process the three shapefiles in sequence
    #               Arg1=Shapefile, Arg2=NameforShortWalk, Arg3=NameforLongWalk, Arg4=StartfromScratch, Arg5=WriteOutputFile

    if( MWCOG_BufferAndExport( User_Input_All_DY_MetroLRT_Stops_Shapefile, "MTLRTShr",  "MTLRTLng", 1, 0) == 1 ): sys.exit(1)
    if( MWCOG BufferAndExport(      User Input All PK All Stops Shapefile, "ALLPKShr",  "ALLPKLng", 0, 0) == 1 ): sys.exit(1)
    if( MWCOG_BufferAndExport(      User_Input_All_OP_All_Stops_Shapefile, "ALLOPShr",  "ALLOPLng", 0, 1) == 1 ): sys.exit(1)

    # if( MWCOG_BufferAndExport( User_Input_All_DY_MetroLRT_Stops_Shapefile, "MTLRTShr",  "MTLRTLng", 1, 1) == 1 ): sys.exit(1)    ##
debug individual
    # if( MWCOG_BufferAndExport(      User_Input_All_PK_All_Stops_Shapefile, "ALLPKShr",  "ALLPKLng", 1, 1) == 1 ): sys.exit(1)    ##
debug individual
    # if( MWCOG_BufferAndExport(      User_Input_All_OP_All_Stops_Shapefile, "ALLOPShr",  "ALLOPLng", 1, 1) == 1 ): sys.exit(1)    ##
debug individual

    # if( MWCOG_BufferAndExport( User_Input_All_DY_MetroLRT_Stops_Shapefile, "MTLRTShr",  "MTLRTLng", 1, 0) == 1 ): sys.exit(1)    ##
fast run for debugging
    # if( MWCOG_BufferAndExport( User_Input_All_DY_MetroLRT_Stops_Shapefile, "ALLPKShr",  "ALLPKLng", 0, 0) == 1 ): sys.exit(1)    ##
fast run for debugging
    # if( MWCOG_BufferAndExport( User_Input_All_DY_MetroLRT_Stops_Shapefile, "ALLOPShr",  "ALLOPLng", 0, 1) == 1 ): sys.exit(1)    ##
fast run for debugging

    myendtime     = datetime.datetime.now()
    myelapsedtime = myendtime - mystarttime

    MWCOG_PrintWriter ("\nEND ArcPy Walkshed Process (Finished: " + str(myendtime).split(".")[0] + ", Elapsed: " +
str(myelapsedtime).split(".")[0] + ")")

    my report file handle.close()

##################################################################################################################################
## Main: End
##################################################################################################################################
```

Figure 9-4: Sample Screen Output during Execution of PWT Process

```
    Searching for Python in Path C:\Python27\ArcGIS10.2
    Searching for Python in Path C:\Python27\ArcGIS10.1
    Searching for Python in Path C:\Python26\ArcGIS10.0
    Searching for Python in Path D:\Python27\ArcGIS10.2
```

```
    Searching for Python in Path D:\Python27\ArcGIS10.1
    Searching for Python in Path D:\Python26\ArcGIS10.0
    Searching for Python in Path E:\Python27\ArcGIS10.2
    Searching for Python in Path E:\Python27\ArcGIS10.1
    Found Python in Path E:\Python27\ArcGIS10.1


 Using Python from Directory = E:\Python27\ArcGIS10.1



1) Creating Subdirectories ...


2) Preparing Inputs ...

    using TRNBUILD line files

3) Launching ArcPy-based Walkshed Process ...

BEGIN ArcPy Walkshed Process (Started: 2014-03-24 16:19:23)

        MWCOG_BufferAndExport (Started: 2014-03-24 16:19:23)
        Shapefile  : ..dified\2010_Final\inputs\Transit_Walksheds_GIS\input\MetroandLRT_AllDay.shp
        Short Walk : 2640 Feet, Field = MTLRTShr
        Long Walk  : 5280 Feet, Field = MTLRTLng
        creating personal geo-database ...
        initializing outputs ...
                creating TAZ area field: TAZ_Area
        buffering ...
        adding a field for dissolving split buffers ...
        dissolving any split buffers ...
        intersecting ...
        dissolving ...
        computing short-walk (AreaWalk) ...
        computing long-walk (AreaWalk) ...
        MWCOG BufferAndExport (Finished: 2014-03-24 16:21:25, Elapsed: 0:02:01)

        MWCOG BufferAndExport (Started: 2014-03-24 16:21:25)
        Shapefile  : ..3.52_Modified\2010_Final\inputs\Transit_Walksheds_GIS\input\ALLStops_PK.shp
        Short Walk : 2640 Feet, Field = ALLPKShr
        Long Walk  : 5280 Feet, Field = ALLPKLng
        initializing outputs ...
        buffering ...
        adding a field for dissolving split buffers ...
        dissolving any split buffers ...
        intersecting ...
        dissolving ...
        computing short-walk (AreaWalk) ...
        computing long-walk (AreaWalk) ...
        MWCOG_BufferAndExport (Finished: 2014-03-24 16:39:13, Elapsed: 0:17:48)
```

```
       MWCOG BufferAndExport (Started: 2014-03-24 16:39:13)
       Shapefile  : ..3.52_Modified\2010_Final\inputs\Transit_Walksheds_GIS\input\ALLStops_OP.shp
       Short Walk : 2640 Feet, Field = ALLOPShr
       Long Walk  : 5280 Feet, Field = ALLOPLng
       initializing outputs ...
       buffering ...
       adding a field for dissolving split buffers ...
       dissolving any split buffers ...
       intersecting ...
       dissolving ...
       computing short-walk (AreaWalk) ...
       computing long-walk (AreaWalk) ...
       writing out AreaWalk CSV file ...
       writing out AreaWalk TXT file ...

SUMMARY REPORT:

       Number of TAZ Records in Output                                  : 3,669
       Total TAZ LAND Area                                              : 6818.7936 sq. mi.
       TAZs with Zero LAND Area                                         : NONE

       Total TAZ Long-Walk Area                                         : 1917.7276 sq. mi. (28.12% of TAZ Land Area)
       Total TAZ Long-Walk (Peak Period) Area                           : 1891.6803 sq. mi. (27.74% of TAZ Land Area)
       Total TAZ Long-Walk (Off-Peak Period) Area                       : 1648.5199 sq. mi. (24.18% of TAZ Land Area)
       TAZs with Zero WALK Area                                         : (Count=691)
396,444,445,454,455,495,496,756,760,761,766,767,770,1142,1143,1267,1268,1282,1283,1284,1285,1286,1287,1288,1289,1290,1291,1292,1293,12
94,1295,1296,1299,1300,1301,1302,1303,1304,1305,1306,1307,1308,1309,1310,1311,1312,1313,1319,1320,1324,1325,1326,1327,1328,1329,1330,1
331,1332,1333,1335,1336,1371,1376,1378,1400,1615,1616,1621,1622,1702,1712,1713,1782,1896,1897,1898,1899,2092,2093,2154,2156,2158,2159,
2160,2161,2162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191
,2192,2193,2194,2195,2196,2197,2198,2201,2202,2203,2208,2209,2210,2211,2212,2213,2214,2215,2216,2217,2218,2219,2220,2225,2227,2228,222
9,2230,2232,2233,2235,2236,2237,2238,2239,2240,2241,2243,2244,2245,2246,2247,2248,2251,2252,2253,2254,2257,2283,2306,2307,2308,2312,23
14,2315,2316,2317,2318,2389,2391,2393,2396,2397,2398,2399,2400,2401,2402,2404,2408,2412,2413,2414,2415,2416,2417,2418,2419,2420,2421,2
423,2424,2425,2426,2427,2428,2429,2430,2431,2432,2433,2434,2435,2436,2437,2438,2439,2440,2441,2442,2443,2444,2445,2446,2447,2448,2449,
2450,2451,2452,2453,2454,2455,2456,2457,2458,2459,2460,2461,2462,2463,2465,2466,2467,2468,2469,2470,2471,2472,2473,2474,2475,2476,2477
,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2497,2498,2499,2500,2501,2502,2505,2506,2507,2508,2509,2510,251
1,2512,2513,2514,2515,2516,2517,2518,2519,2520,2522,2523,2579,2584,2586,2587,2597,2600,2606,2610,2611,2612,2613,2614,2615,2616,2618,26
19,2620,2654,2655,2656,2657,2658,2659,2660,2661,2662,2663,2697,2698,2699,2700,2701,2702,2703,2704,2705,2706,2707,2708,2715,2719,2720,2
813,2814,2815,2816,2817,2818,2819,2823,2824,2825,2827,2831,2832,2833,2844,2845,2846,2847,2848,2849,2850,2855,2861,2864,2872,2873,2874,
2875,2876,2881,2882,2883,2884,2885,2886,2887,2888,2889,2890,2891,2892,2893,2894,2895,2896,2902,2903,2904,2905,2906,2907,2908,2910,2942
,2945,2952,2953,2954,2955,2956,2957,2958,2959,2960,2961,2962,2963,2964,2965,2966,2976,3032,3033,3048,3052,3053,3054,3055,3056,3057,305
9,3060,3061,3062,3063,3064,3065,3066,3067,3077,3078,3079,3080,3081,3082,3083,3084,3085,3086,3087,3095,3097,3098,3104,3106,3107,3108,31
09,3110,3111,3114,3121,3128,3139,3140,3160,3200,3202,3203,3213,3214,3215,3217,3220,3223,3230,3231,3232,3233,3234,3235,3236,3237,3238,3
239,3240,3241,3242,3243,3244,3245,3246,3247,3248,3249,3250,3251,3252,3253,3254,3255,3256,3257,3258,3259,3260,3261,3262,3263,3264,3265,
3266,3267,3268,3269,3270,3271,3272,3273,3274,3275,3276,3277,3278,3279,3280,3281,3282,3283,3284,3285,3286,3287,3298,3300,3335,3336,3342
,3347,3403,3404,3409,3417,3418,3421,3424,3425,3426,3427,3428,3429,3430,3431,3432,3434,3436,3437,3449,3450,3451,3452,3453,3454,3455,345
6,3457,3458,3459,3460,3461,3462,3463,3464,3465,3466,3467,3469,3470,3471,3472,3473,3476,3483,3484,3486,3487,3488,3497,3500,3502,3504,35
05,3506,3517,3518,3520,3525,3527,3528,3529,3530,3532,3533,3540,3542,3543,3544,3545,3546,3547,3548,3549,3550,3551,3552,3553,3554,3555,3
556,3557,3558,3559,3560,3561,3562,3563,3564,3567,3581,3582,3583,3584,3586,3587,3588,3589,3590,3591,3592,3593,3594,3595,3596,3597,3598,
3599,3600,3601,3602,3603,3604,3605,3606,3607,3608,3609,3610,3611,3612,3613,3614,3615,3616,3617,3618,3619,3620,3621,3622,3623,3624,3625
```

```
,3626,3627,3628,3629,3630,3631,3632,3633,3634,3635,3636,3637,3638,3639,3640,3641,3642,3643,3644,3645,3646,3647,3648,3649,3650,3651,365
2,3653,3654,3655,3656,3657,3658,3659,3660,3661,3662,3663,3664,3667,3668,3672,3673,3674,3675

       Total MTLRTSHR Area                                          :   54.3779 sq. mi. ( 0.80% of TAZ Land Area)
       Total MTLRTLNG Area                                          :  149.5641 sq. mi. ( 2.19% of TAZ Land Area)
       Total ALLPKSHR Area                                          : 1129.7719 sq. mi. (16.57% of TAZ Land Area)
       Total ALLPKLNG Area                                          : 1891.6803 sq. mi. (27.74% of TAZ Land Area)
       Total ALLOPSHR Area                                          :  970.6922 sq. mi. (14.24% of TAZ Land Area)
       Total ALLOPLNG Area                                          :  970.6922 sq. mi. (14.24% of TAZ Land Area)

       TAZs with Short-Walk Less in Peak Period than in Off-Peak Period : (Count=131)
359,360,389,415,421,429,436,446,448,449,450,451,459,460,461,462,463,464,465,467,590,591,1033,1151,1154,1160,1161,1162,1236,1344,1345,1
346,1653,1656,1657,1658,1693,1734,1788,2177,2223,2266,2287,2288,2289,2290,2330,2331,2354,2357,2367,2368,2369,2374,2376,2377,2378,2379,
2381,2525,2529,2530,2531,2541,2548,2621,2622,2631,2634,2636,2637,2638,2643,2646,2648,2649,2653,2667,2668,2745,2746,2757,2759,2760,2761
,2776,2821,2837,2857,2859,2878,2880,2897,2898,2899,2901,2911,2918,2920,2931,2936,2937,2939,2943,2944,2995,3172,3173,3174,3205,3206,320
7,3209,3210,3211,3212,3218,3219,3344,3345,3346,3348,3349,3350,3351,3355,3356,3392,3393,3395,3396
       TAZs with  Long-Walk Less in Peak Period than in Off-Peak Period : (Count=79)
436,446,447,448,449,450,451,452,453,457,458,459,460,461,462,463,465,1160,1161,1346,1612,1661,1788,2175,2176,2222,2266,2285,2286,2287,2
288,2289,2290,2291,2525,2541,2548,2622,2636,2643,2653,2837,2878,2880,2897,2898,2899,2901,2911,2939,2943,2944,3172,3173,3174,3201,3205,
3206,3207,3208,3209,3210,3211,3212,3218,3219,3224,3344,3345,3346,3348,3349,3350,3351,3355,3356,3393,3395,3396

       MWCOG BufferAndExport (Finished: 2014-03-24 16:50:10, Elapsed: 0:10:56)

END ArcPy Walkshed Process (Finished: 2014-03-24 16:50:10, Elapsed: 0:30:46)

4) Copying AreaWalk.txt / PercentWalk.txt File(s) ...

        1 file(s) copied.
        1 file(s) copied.

5) Copying ArcGIS MXD File ...

        1 file(s) copied.


 Process Complete!
```

9.2 HOT Lane and Toll Road Modeling

Figure 9-5: Contents of Script: Highway_Assignment_Parallel.s

```
< Comments at the beginning are not shown here >


PAGEHEIGHT=32767 ; preclude insertion of page headers

;  useIdp = t (true) or f (false);  this is set in the wrapper batch file
distribute intrastep=%useIdp% multistep=%useMdp%

; Choose traffic assignment type, using "enhance=" keyword
;   enhance=0 Frank-Wolfe
;   enhance=1 Conjugate Frank-Wolfe
;   enhance=2 Bi-conjugate Frank-Wolfe
assignType=2

;;;*************************************************************************
;;; Step 1: Execute peak-period traffic assignments (AM & PM)
;;;         AM nonHOV, HOV and PM nonHOV and HOV Assignemnts
;;;*************************************************************************


iter        = '% iter %'   ;; NOTE that when using DOS environment variables, there should be no spaces before or after the operator
'=' or use single quotes "'"
prev        = '%_prev_%'   ;; NOTE that when using DOS environment variables, there should be no spaces before or after the operator
'=' or use single quotes "'"

INPNET      = 'ZONEHWY.NET'

in_tmin     = '..\support\toll_minutes.txt'                    ;;  read in toll minutes equiv file
in_AMTfac   = 'inputs\AM_Tfac.dbf'                             ;;  AM Toll Factors by Veh. Type
in_PMTfac   = 'inputs\PM_Tfac.dbf'                             ;;  PM Toll Factors by Veh. Type
in_MDTfac   = 'inputs\MD_Tfac.dbf'                             ;;  MD Toll Factors by Veh. Type
in NTTfac   = 'inputs\NT Tfac.dbf'                             ;;  NT Toll Factors by Veh. Type

in_capSpd   = '..\support\hwy_assign_capSpeedLookup.s'         ;;     FT x AT Speed & Capacity lookup
VDF_File    = '..\support\hwy_assign_Conical_VDF.s'            ;;     Volume Delay Functions file

in_VOT      = 'inputs\VOT_by_VehClass_by_TOD.dbf'              ;; Value of Time Distributions by Six Vehicle Classes and Four Times of
Day

;

;
; NEW Parameters
;
;
```

```
; For Starting Speeds in Toll Setting Process
;
IF (iter   == 'pp')
   NON PP   = ';; Disabled for PP ;;'
ELSE
   NON_PP   = ' '
ENDIF
;
; This is set in the wrapper batch file; a value of '1' enables toll setting, any other value disables it
;
DO_TOLL_SETTING                      = '%_hwy_HOT_perform_toll_setting_%'            ; NOTE: that when using DOS environment
variables, there should be no spaces before or after the operator '=' or use single quotes "'"
LOWER_RELGAP_FOR_TOLL_SETTING        = '%_hwy_HOT_lower_relgap_for_toll_setting_%'   ;
USE LAST TOLLS                       = '% hwy HOT use last tolls %'                  ;
APPLY_VOT_TO_SPLIT_VTT               = '%_hwy_HOT_use_vot_distrib_in_toll_setting_%' ;

ST                                   =    3                                          ;  First dynamic-toll toll group
TGRPS                                =  134                                          ;  Highest dynamic-toll toll group
T_NUM                                =  132                                          ;  Total number of toll groups with dynamic tolls

TOLL_CAP                             =  902.8700                                     ; Max Toll in dollars/mile by time-period in
Calib Year $   ($10.00 * delfation factor)
TOLL_FLR_PK                          =   18.0574                                     ; Min Toll in dollars/mile by time-period in
Calib Year $   ($ 0.20 * delfation factor)
TOLL FLR OP                          =   13.5431                                     ; Min Toll in dollars/mile by time-period in
Calib Year $   ($ 0.15 * delfation factor)

READ FILE                            = 'HWY_Deflator.txt'                            ; Defines Variable: DEFLATIONFTR

Min_VC                               =    0.9500                                     ; Threshold V/Cs for variably priced facilities
Max VC                               =    1.0100                                     ; Threshold V/Cs for variably priced facilities

TOLL RES                             =    0.0001                                     ; Resolution of Tolls During Evaluation; Enter in
cents/mile; ForRealism keep in integers; ForComputationalEfficiency keep at 4 decimals.
VC_RES                               =    2.0                                        ; Resolution of V/C During Evaluation; Enter in
number of decimals; ForComputationalEfficiency & Realism Round to 2 decimals

HOT_LAMBDA1                          =   18.0                                        ; multiplier  other values 12.691
HOT LAMBDA2                          =   80.0                                        ; additive    other values 68.891

SOV_TollTimeReliability              =    0.0                                        ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile
HV2 TollTimeReliability              =    0.0                                        ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile
HV3 TollTimeReliability              =    0.0                                        ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile
 CV_TollTimeReliability              =    0.0                                        ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile
TRK_TollTimeReliability              =    0.0                                        ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile
```

```
APX_TollTimeReliability              =    0.0                                          ; TollPath Time Decrease - A +ve value specified
as minutes per mile; a calibration parameter typically 0.1 - 0.5 min/mile

;
; Set lower relgap targets for toll-setting if that optin is enabled
;
low_rel_gap                          =    0.1
low mxIters                          =   10
POST_TOLL_SETTING_ASSIGNMENT_ACTIVE  =    0

ALLOW_EXCLUSIVE_TOLL_REDUCTION_LOOPS =    0                                          ; 1 = Allow; 0 = Don't Allow Exclusive Lowered-
Toll Loops. For computational efficiency set to '0' or disable exclusive toll-reduction toll-setting loops.
                                                                                     ; As part of toll adjustment lowered tolls do not
exclusively trigger toll-setting loops. Enabling this key would allow toll-setting loops when the tolls only reduce.

                ; Warning: Enabling this key can significantly increase the model processing time!
;
; BEGIN ASSIGNMENTS
;
DistributeMULTISTEP ProcessID=%DP A ProcessID%, ProcessNum=1

    ;;;*************************************************************************
    ;;; Assign AM trip tables
    ;;;           (SOV, HOV2, HOV3+, CV, TRUCK & AIRPORT PASSENGER TRIPS)
    ;;;*************************************************************************

    PRD    =  'AM'            ;
    PCTADT =   41.7           ; (% of traffic in pk hr of period)
    CAPFAC=1/(PCTADT/100)     ; Capacity Factor = 1/(PCTADT/100)

    READ FILE=..\Scripts\Highway Assignment Parallel part1 Main.s

ENDDistributeMULTISTEP

DistributeMULTISTEP ProcessID=%DP_B_ProcessID%, ProcessNum=1

    ;;;*************************************************************************
    ;;; Assign PM trip tables
    ;;;           (SOV, HOV2, HOV3+, CV, TRUCK & AIRPORT PASSENGER TRIPS)
    ;;;*************************************************************************

    PRD    =  'PM'            ;
    PCTADT =   29.4           ; (% of traffic in pk hr of period)
    CAPFAC=1/(PCTADT/100)     ; Capacity Factor = 1/(PCTADT/100)

    READ FILE=..\Scripts\Highway_Assignment_Parallel_part1_Main.s

    ;;;*************************************************************************
    ;;; Assign MD trip tables
    ;;;           (SOV, HOV2, HOV3+, CV, TRUCK & AIRPORT PASSENGER TRIPS)
    ;;;*************************************************************************
```

```
    PRD    =  'MD'            ;
    PCTADT =   17.7           ; (% of traffic in pk hr of period)
    CAPFAC=1/(PCTADT/100)     ; Capacity Factor = 1/(PCTADT/100)

    READ FILE=..\Scripts\Highway_Assignment_Parallel_part1_Main.s

    ;;;*************************************************************************
    ;;; Assign NT trip tables
    ;;;           (SOV, HOV2, HOV3+, CV, TRUCK & AIRPORT PASSENGER TRIPS)
    ;;;*************************************************************************

    PRD    =  'NT'            ;
    PCTADT =   15.0           ; (% of traffic in pk hr of period)
    CAPFAC=1/(PCTADT/100)     ; Capacity Factor = 1/(PCTADT/100)

    READ FILE=..\Scripts\Highway_Assignment_Parallel_part1_Main.s

ENDDistributeMULTISTEP


Wait4Files Files=%DP A ProcessID%1.script.end, %DP B ProcessID%1.script.end, CheckReturnCode=T, PrintFiles=MergeSave,
DelDistribFiles=F

;
; END ALL ASSIGNMENTS
;

;;;*************************************************************************
;;; Step 4: Summarize 24-hour VMT of current AM, PM, MD & NT assignments
;;;*************************************************************************

RUN PGM=HWYNET     ; Summarize 24-hour VMT of current AM, PM, MD & OP assignments

    FILEI NETI[1]=temp_AM.net
    FILEI NETI[2]=temp_MD.net
    FILEI NETI[3]=temp_PM.net
    FILEI NETI[4]=temp_NT.net
    FILEO NETO   =% iter % HWY.NET,
    EXCLUDE=OLDVOL1,NEWVOL1,OLDVOL2,NEWVOL2,OLDVOL3,NEWVOL3,
    OLDVOL4,NEWVOL4,OLDVOL5,NEWVOL5,
    OLDSPD1,OLDSPD2,OLDSPD3,OLDSPD4,OLDSPD5,%_iter_%24VMT,
    CSPD 2,VDT 2,VHT 2

    % iter %amspd =  LI.1.% iter %amspd
    %_iter_%mdspd =  LI.2.%_iter_%mdspd
    %_iter_%pmspd =  LI.3.%_iter_%pmspd
    %_iter_%ntspd =  LI.4.%_iter_%ntspd
    ;
    ;
    _VOLAM =  LI.1.%_iter_%AMVOL
```

```
    _VOLMD =  LI.2.%_iter_%MDVOL
     VOLPM =  LI.3.% iter %PMVOL
    _VOLNT =  LI.4.%_iter_%NTVOL

    ; COMPUTE FINAL DAILY VOLUME  ON ALL LINKS
    %_iter_%24VOL =  _VOLAM + _VOLMD + _VOLPM + _VOLNT   ; Total Daily Volume

    ; COMPUTE FINAL DAILY VMT ON ALL NON-CENTROID LINKS
    IF (FTYPE = 0)
        % iter %24VMT =  0
        ELSE
        %_iter_%24VMT =  %_iter_%24VOL * DISTANCE  ; Total Daily VMT
    ENDIF

    ;
    ;
    IF (FTYPE=1-6)
        TVOL00=ROUND((_VOLAM + _VOLMD + _VOLPM + _VOLNT)/1000.0) ; total hwy vol in 000s
        TVMT00=TVOL00*DISTANCE                                   ; total hwy VMT in 000s
        ELSE
        TVOL00=0
        TVMT00=0                                        ;
    ENDIF
    ;

    comp atype=spdclass%10       ; area type code     1-7

    ;;============================================================================
    ;; DAILY X-Tabs
    ;;============================================================================

    ;;  Crosstab DAILY VMT by ATYPE and FTYPE
    CROSSTAB VAR=% iter %24VMT, FORM=12cs,
    ROW=ATYPE, RANGE=1-7-1,,1-7,
    COL=FTYPE, RANGE=1-6-1,1-6

    ; Crosstab Total VMT by Jurisdiction and FTYPE
    CROSSTAB VAR=%_iter_%24VMT, FORM=12cs,
    ROW=JUR,   RANGE=0-23-1,,0-23,
    COL=FTYPE, RANGE=0-6-1,0-6

ENDRUN
```

Figure 9-6: Contents of Script: Highway_Assignment_Parallel_part1_Main.s

```
;
; IDP Command
;
IF( PRD == 'AM' ) idp string =  'distributeIntrastep ProcessID=%DP A ProcessID%, ProcessList=%DP A ProcessList%'

IF( PRD == 'PM' ) idp_string =  'distributeIntrastep ProcessID=%DP_B_ProcessID%, ProcessList=%DP_B_ProcessList%'
IF( PRD == 'MD' ) idp string =  'distributeIntrastep ProcessID=%DP B ProcessID%, ProcessList=%DP B ProcessList%'
IF( PRD == 'NT' ) idp_string =  'distributeIntrastep ProcessID=%DP_B_ProcessID%, ProcessList=%DP_B_ProcessList%'
;
; Time Period Number
;
IF( PRD == 'AM' ) period_num =  '1'
IF( PRD == 'MD' ) period num =  '2'
IF( PRD == 'PM' ) period_num =  '3'
IF( PRD == 'NT' ) period num =  '4'
;
; Highway Assignment with or without Toll-Setting
;
IF(DO TOLL SETTING == '0')

    ts string = ';; Disabled for REG ;;'    ; Disable Toll Setting Relevant Commands

    rel_gap = %_relGap_%
    mxIters = %_maxUeIter_%

    RUN PGM=HIGHWAY  MSG='Export a Copy of Coded Tolls for Review : @PRD@ @TSitr@'
        ZONES=3722
        ZONEMSG=10
        FILEI NETI =@INPNET@
        READ FILE=..\Scripts\Highway Assignment Parallel part3 SummarizeTolls.s
    ENDRUN

    RUN PGM=HIGHWAY  MSG='Highway Assignment with Toll-Setting DISABLED : @PRD@'

        @idp_string@

        ZONES=3722
        ZONEMSG=10
        FILEI NETI  =  @INPNET@
        FILEI MATI  =  @iter@_@prd@.VTT
        FILEO NETO  =  temp2_@PRD@.NET        ; Output loaded network of current iter/time prd.

        READ FILE=..\Scripts\Highway_Assignment_Parallel_part5_BuildPaths.s
    ENDRUN

    RUN PGM=NETWORK   MSG='Calculate restrained speed/perform MSA volume averaging : @PRD@'
        ZONES=3722
```

```
        FILEI NETI=temp2_@PRD@.net             ; input network from highway assignment
        FILEO NETO=temp @PRD@.net,             ; output/@PRD@ network with updated speeds
        EXCLUDE=V_1,TIME_1,VC_1,V1_1, V2_1, V3_1, V4_1,V5_1,V6_1,
                  VT 1,V1T 1,V2T 1,V3T 1,V4T 1,V5T 1,V6T 1,
                  CSPD_1,VDT_1,VHT_1,WRSPD,WFFSPD

        ; Code Stored in External File
        READ FILE=..\Scripts\Highway Assignment Parallel part6 CalculateRestrainedFinalVolSpdVC.s
    ENDRUN

ELSE

    ts_string = ' '                 ; Enable Toll Setting Relevant Commands

    ;
    ; Set Source Toll File
    ;
    IF (USE_LAST_TOLLS == '1')
        IF (iter == 'pp')
            SourceTollsFile = 'inputs\SEED TOLLS ' + PRD + '.TXT'
        ELSE
            SourceTollsFile = 'LATEST TOLLS '      + PRD + '.TXT'
        ENDIF
    ELSE
            SourceTollsFile = 'inputs\SEED TOLLS ' + PRD + '.TXT'
    ENDIF

    ;
    ;Step A2: Summarize network outputs and adjust toll rate based on HOT lane speed
    ;
    LOOP  TSLoop = 0, 99      ; big outer loop for iteration; MAX 100 ITERATION TO FIND TOLL RATES

        TSprv =  TSLoop
        TSitr = _TSLoop + 1

        IF(TSprv == 0)

            ;
            ;Step A3: Read Seed Tolls (Seed tolls & final tolls are in base/calibration year dollars; no deflation applied)
            ;
            RUN PGM=NETWORK  MSG='Toll-Setting Process: Initialize : @PRD@ @TSitr@'
                ZONES=3722
                FILEI NETI=@prev@ HWY.NET ;;  @INPNET@
                FILEO NETO=@iter@_NewTollsToEval_@PRD@_@TSitr@.NET           ; Output network in TP+ format
                READ FILE=..\Scripts\Highway Assignment Parallel part2 Initialize.s
            ENDRUN

        ELSE

            ;
            ;Step A4: Update TOLLs in Highway Network
```

```
        ;
        RUN PGM=NETWORK  MSG='Toll-Setting Process: Load Last Tolls : @PRD@ @TSitr@'
            ZONES=3722
            FILEI NETI=@iter@ NewTollsAssignedMod @PRD@ @TSprv@.NET
            FILEO NETO=@iter@_NewTollsToEval_@PRD@_@TSitr@.NET        ; Output network in TP+ format

            ; READ TOLL RATE BY TOLLGRP
            lookup name = TOLL@PRD@,
                lookup[1]  = 1,result=1,   ; Toll Group
                lookup[2]  = 1,result=2,   ; Num Links
                lookup[3]  = 1,result=3,   ; Average FTYPE
                lookup[4]  = 1,result=4,   ; Old Toll Rate
                lookup[5]  = 1,result=5,   ; New Toll Rate
                lookup[6]  = 1,result=6,   ; Weighted Speed
                lookup[7]  = 1,result=7,   ; Average VC Ratio
                lookup[8]  = 1,result=8,   ; VMT
                lookup[9]  = 1,result=9,   ; Difference in VC Ratio
                lookup[10] = 1,result=10,  ; Toggle Switch (1=Done, 0=NotDone)
                interpolate=N,fail=0,0,0,file=@iter@_NEW_@TSprv@_TOLLS_@PRD@.TXT

            ;
            ; Replace @PRD@TOLL
            ;
            ; ALL TOLLS ARE UPDATED REGARDLESS OF FLAGS
            ;
            LOOP _IDX=@ST@,@TGRPS@          ;
                IF(LI.1.tollgrp = TOLL@PRD@(1, IDX))
                    ; Toll deflation is not required here as it is only applied to input/seed tolls once   (TOLL)
                    @PRD@TOLL    = TOLL@PRD@(5,_IDX) * LI.1.DISTANCE
                    @PRD@TOLL_VP = TOLL@PRD@(5,_IDX) * LI.1.DISTANCE
                ELSE
                    @PRD@TOLL    = @PRD@TOLL   ; (TOLL)
                    @PRD@TOLL VP = @PRD@TOLL   ; (TOLL VP)
                ENDIF
            ENDLOOP
        ENDRUN

    ENDIF

    ;
    ;Step A5.1: Echo coded tolls
    ;
    RUN PGM=HIGHWAY  MSG='Toll-Setting Process: Summarize Tolls : @PRD@ @TSitr@'
        ZONES=3722
        ZONEMSG=10
        FILEI NETI =@iter@_NewTollsToEval_@PRD@_@TSitr@.NET
        READ FILE=..\Scripts\Highway_Assignment_Parallel_part3_SummarizeTolls.s
    ENDRUN

    ;
    ;Step A5: Split trip tables into toll & non-toll
```

```
;
RUN PGM=HIGHWAY  MSG='Toll-Setting Process: Split Trip Tables : @PRD@ @TSitr@'

    @idp string@

    ZONES=3722
    ZONEMSG=10
    FILEI NETI     =  @iter@ NewTollsToEval @PRD@ @TSitr@.NET
    READ FILE=..\Scripts\Highway_Assignment_Parallel_part4_ApplyVOTSplitTollNonToll.s
ENDRUN

;
; Set relative gap
;
IF ( LOWER_RELGAP_FOR_TOLL_SETTING  == '1' )
    rel gap = low rel gap
    mxIters = low_mxIters
    IF (%_maxUeIter_% < mxIters) mxIters = %_maxUeIter_%   ; in case user-setting is lower
ELSE
    rel gap = % relGap %
    mxIters = %_maxUeIter_%
ENDIF

;
;Step A6: Perform Highway Assignment
;
RUN PGM=HIGHWAY  MSG='Highway Assignment with Toll-Setting ENABLED : @PRD@ @TSitr@'

    @idp_string@

    ZONES=3722
    ZONEMSG=10
    FILEI NETI  =  @iter@ NewTollsToEval @PRD@ @TSitr@.NET
    FILEI MATI  =  @iter@_@prd@_Split_Toll_NonToll.VTT
    ; Output loaded network of current iter/time prd.
    FILEO NETO  =  @iter@_NewTollsAssigned_@PRD@_@TSitr@.NET

    READ FILE=..\Scripts\Highway_Assignment_Parallel_part5_BuildPaths.s
ENDRUN

;
;Step A7: Calculate restrained final Volumes, speeds, V/Cs (No MSA)
;

RUN PGM=NETWORK  MSG='Calculate restrained speed/perform MSA volume averaging : @PRD@ @TSitr@'
    ZONES=3722

    ; input network from highway assignment
    FILEI NETI=@iter@_NewTollsAssigned_@PRD@_@TSitr@.NET
    ; output/@PRD@ network with updated speeds
    FILEO NETO=@iter@_NewTollsAssignedMod_@PRD@_@TSitr@.NET,
```

```
          EXCLUDE=V_1,TIME_1,VC_1,V1_1, V2_1, V3_1, V4_1,V5_1,V6_1,
                  VT 1,           V1T 1,V2T 1,V3T 1,V4T 1,V5T 1,V6T 1,
                                  V11_1, V12_1, V13_1, V14_1, V15_1, V16_1,
                                  V11T 1,V12T 1,V13T 1,V14T 1,V15T 1,V16T 1,
                              CSPD_1,VDT_1,VHT_1,WRSPD,WFFSPD

    ; Code Stored in External File
    READ FILE=..\Scripts\Highway Assignment Parallel part6 CalculateRestrainedFinalVolSpdVC.s
ENDRUN

;
;Step A9: Summarize network outputs and adjust toll rate based on HOT lane speed
;
RUN PGM=HIGHWAY  MSG='Toll-Setting Process: Summarize and Adjust Tolls : @PRD@ @TSitr@'
    ZONES=3722
    ZONEMSG=10
    FILEI NETI =@iter@_NewTollsAssignedMod_@PRD@_@TSitr@.NET
    READ FILE=..\Scripts\Highway_Assignment_Parallel_part7_SummarizeAndAdjustTolls.s
ENDRUN

;
;Step A10: Terminate the process upon conditions
;
RUN PGM=NETWORK  MSG='Toll-Setting Process: Check Termination Criterion : @PRD@ @TSitr@'
    ZONES=3722
    FILEI NETI=@iter@_NewTollsAssignedMod_@PRD@_@TSitr@.NET             ; Input network in TP+ format
    READ FILE=..\Scripts\Highway Assignment Parallel part8 TollEvalTerminationCheck.s
ENDRUN

;
; Now Check Whether to Continue Toll Setting Loop
;
IF( PRD == 'AM' )
    IF ( LOWER_RELGAP_FOR_TOLL_SETTING == '1' )
        IF( TSitr == 100 )
            POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 1
            BREAK
        ELSE
            IF( TOLL SETTING CLOSURE.FLAG AM == T NUM )
                POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 1
                BREAK
            ELSE
                POST TOLL SETTING ASSIGNMENT ACTIVE = 0
            ENDIF
        ENDIF
    ELSE
        POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
        IF( TOLL_SETTING_CLOSURE.FLAG_AM == T_NUM )
            BREAK
        ENDIF
    ENDIF
```

```
        ELSEIF ( PRD == 'MD' )
            IF ( LOWER RELGAP FOR TOLL SETTING == '1' )
                IF( TSitr == 100 )
                    POST TOLL SETTING ASSIGNMENT ACTIVE = 1
                    BREAK
                ELSE
                    IF( TOLL_SETTING_CLOSURE.FLAG_MD == T_NUM )
                        POST TOLL SETTING ASSIGNMENT ACTIVE = 1
                        BREAK
                    ELSE
                        POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
                    ENDIF
                ENDIF
            ELSE
                POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
                IF( TOLL SETTING CLOSURE.FLAG MD == T NUM )
                    BREAK
                ENDIF
            ENDIF
        ELSEIF ( PRD == 'PM' )
            IF ( LOWER_RELGAP_FOR_TOLL_SETTING == '1' )
                IF( TSitr == 100 )
                    POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 1
                    BREAK
                ELSE
                    IF( TOLL_SETTING_CLOSURE.FLAG_PM == T_NUM )
                        POST TOLL SETTING ASSIGNMENT ACTIVE = 1
                        BREAK
                    ELSE
                        POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
                    ENDIF
                ENDIF
            ELSE
                POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
                IF( TOLL_SETTING_CLOSURE.FLAG_PM == T_NUM )
                    BREAK
                ENDIF
            ENDIF
        ELSE
            IF ( LOWER_RELGAP_FOR_TOLL_SETTING == '1' )
                IF( TSitr == 100 )
                    POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 1
                    BREAK
                ELSE
                    IF( TOLL SETTING CLOSURE.FLAG NT == T NUM )
                        POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 1
                        BREAK
                    ELSE
                        POST_TOLL_SETTING_ASSIGNMENT_ACTIVE = 0
                    ENDIF
                ENDIF
```

```
            ELSE
                POST TOLL SETTING ASSIGNMENT ACTIVE = 0
                IF( TOLL_SETTING_CLOSURE.FLAG_NT == T_NUM )
                    BREAK
                ENDIF
            ENDIF
        ENDIF

        ;
        ; If post-toll-setting assignment is to be performed
        ;
        IF ( POST_TOLL_SETTING_ASSIGNMENT_ACTIVE == 1 )

            rel gap = % relGap %
            mxIters = %_maxUeIter_%

            ;
            ;Step A6: Perform Highway Assignment
            ;
            RUN PGM=HIGHWAY  MSG='FINAL Highway Assignment with Toll-Setting ENABLED : @PRD@ @TSitr@'

                @idp string@

                ZONES=3722
                ZONEMSG=10
                FILEI NETI  =  @iter@_NewTollsToEval_@PRD@_@TSitr@.NET
                FILEI MATI  =  @iter@ @prd@ Split Toll NonToll.VTT
                ; Output loaded network of current iter/time prd.
                FILEO NETO  =  @iter@_NewTollsAssigned_@PRD@_@TSitr@.NET

                READ FILE=..\Scripts\Highway Assignment Parallel part5 BuildPaths.s
            ENDRUN

            ;
            ;Step A7: Calculate restrained final Volumes, speeds, V/Cs (No MSA)
            ;

            RUN PGM=NETWORK  MSG='Calculate restrained speed/perform MSA volume averaging : @PRD@ @TSitr@'
                ZONES=3722

                ; input network from highway assignment
                FILEI NETI=@iter@_NewTollsAssigned_@PRD@_@TSitr@.NET
                ; output/@PRD@ network with updated speeds
                FILEO NETO=@iter@_NewTollsAssignedMod_@PRD@_@TSitr@.NET,
                      EXCLUDE=V 1,TIME 1,VC 1,V1 1, V2 1, V3 1, V4 1,V5 1,V6 1,
                             VT_1,           V1T_1,V2T_1,V3T_1,V4T_1,V5T_1,V6T_1,
                                             V11_1, V12_1, V13_1, V14_1, V15_1, V16_1,
                                             V11T_1,V12T_1,V13T_1,V14T_1,V15T_1,V16T_1,
                                    CSPD_1,VDT_1,VHT_1,WRSPD,WFFSPD

                ; Code Stored in External File
```

```
             READ FILE = ..\Scripts\Highway_Assignment_Parallel_part6_CalculateRestrainedFinalVolSpdVC.s
            ENDRUN

            ;
            ; Done with this @PRD@
            ;
            BREAK

        ENDIF

    ENDLOOP

    ;
    ; Toll Setting is Done, Finally Make a copy of the Network
    ;
    RUN PGM=NETWORK  MSG='Toll-Setting Process: Make a Copy of the Loaded Network : @PRD@ @TSitr@'
        ZONES=3722
        FILEI NETI = @iter@_NewTollsAssignedMod_@PRD@_@TSitr@.NET
        FILEO NETO = temp_@PRD@.NET                          ; Copy of Input
    ENDRUN

ENDIF
```

Figure 9-7: Contents of Script: Highway_Assignment_Parallel_part2_Initialize.s

```
; READ TOLL RATE BY TOLLGRP
lookup name = TOLL@PRD@,
       lookup[1]  = 1,result=1,   ; Toll Group
       lookup[2]  = 1,result=2,   ; Num Links
       lookup[3]  = 1,result=3,   ; Average FTYPE
       lookup[4]  = 1,result=4,   ; Old Toll Rate
       lookup[5]  = 1,result=5,   ; New Toll Rate
       lookup[6]  = 1,result=6,   ; Weighted Speed
       lookup[7]  = 1,result=7,   ; Average VC Ratio
       lookup[8]  = 1,result=8,   ; VMT
       lookup[9]  = 1,result=9,   ; Difference in VC Ratio
       lookup[10] = 1,result=10,  ; Toggle Switch (1=Done, 0=NotDone)
       interpolate=N,fail=0,0,0,file=@SourceTollsFile@
;
; Replace @PRD@TOLL
;
; ALL TOLLS ARE UPDATED REGARDLESS OF FLAGS
;
LOOP  IDX=@ST@,@TGRPS@          ;
    IF(LI.1.tollgrp = TOLL@PRD@(1,_IDX))
       ; Toll deflation is not required here as it is only applied to input/seed tolls once
        @PRD@TOLL    = TOLL@PRD@(5,_IDX) * LI.1.DISTANCE
        @PRD@TOLL_VP = TOLL@PRD@(5,_IDX) * LI.1.DISTANCE
    ELSE
        @PRD@TOLL    = @PRD@TOLL   ; (TOLL)
        @PRD@TOLL_VP = @PRD@TOLL   ; (TOLL_VP)
    ENDIF
ENDLOOP
;
; Add and Initialize Toll Variables
;
Tolls@PRD@spd = 0
Tolls@PRD@vol = 0
Tolls@prd@vc  = 0
;
; print debug list
;
PRINT LIST = 'A=', A(10),
             ', B=', B(10),
             ', TOLLGRP=', LI.1.TOLLGRP(10),
             ', @PRD@TOLL=', @PRD@TOLL(10.2),
             ', LI.1.@PRD@TOLL=', LI.1.@PRD@TOLL(10.2),
             ', DISTANCE=', LI.1.DISTANCE(10.2),
             ', DEFLATIONFTR=', @DEFLATIONFTR@(10.2),
             FILE=Debug_SeedTollsEncoding_@PRD@.txt
```

Figure 9-8: Contents of Script: Highway_Assignment_Parallel_part3_SummarizeTolls.s

```
                ARRAY TG_TVMT_@PRD@  = @TGRPS@, ; Total VMT
                      TG_ASP_@PRD@   = @TGRPS@, ; Average Speed
                      TG_SWSP_@PRD@  = @TGRPS@, ; Sum of VMT weighted Speed
                      TG_AFT_@PRD@   = @TGRPS@, ; Average FType
                      TG_AVC_@PRD@   = @TGRPS@, ; Average VC-Ratio
                      TG_SWVC_@PRD@  = @TGRPS@, ; Sum of VMT Weighted VC-Ratio
                      TG_TDST_@PRD@  = @TGRPS@, ; Total Distance
                      TG_ORATE_@PRD@ = @TGRPS@, ; Old Toll RATE (cents/mile - deflated)
                      TG_NRATE_@PRD@ = @TGRPS@, ; NEW Toll RATE (cents/mile - deflated
                      TG_DVC_@PRD@   = @TGRPS@, ; Average VC-Ratio MINUS Max VC-Ratio
                      TG_CNT_@PRD@   = @TGRPS@, ; Count of Links
                      TG_TTOLL_@PRD@ = @TGRPS@, ; Total Toll (NOT rate) in cents - deflated
                      TG_DONE_@PRD@  = @TGRPS@  ; Toggle Switch (1=Done, 0=NotDone)

                ZONES = 1

                ; Set up arrays
                PHASE = LINKREAD

                     LW.tollgrp          = LI.tollgrp
                     LW.ftype            = LI.FTYPE
                     LW.distance         = LI.distance
                     LW.@PRD@toll        = LI.@PRD@toll

                     LW.Tolls@PRD@spd    = 0
                     LW.Tolls@PRD@vol    = 0
                     LW.Tolls@PRD@vc     = 0

@ts_string@          LW.Tolls@PRD@spd    = LI.Tolls@PRD@spd
@ts string@          LW.Tolls@PRD@vol    = LI.Tolls@PRD@vol
@ts string@          LW.Tolls@PRD@vc     = LI.Tolls@PRD@vc

                ENDPHASE

                ;*********************************************************************************************
                ; estimate average V/C ratio in each tollgroup by time-period
                ;
                ;                  Sum of VMT Weighted VC Ratios
                ;  average V/C = ---------------------------------
                ;                  Sum of VMT
                ;
                ;*********************************************************************************************

                PHASE = ILOOP

                    ;
                    ; Loop through Toll-Groups to initialize Arrays
                    ;
```

```
   LOOP k=@ST@,@TGRPS@
       TG_TVMT_@PRD@[k]   = 0
       TG_ASP_@PRD@[k]    = 0
       TG_SWSP_@PRD@[k]   = 0
       TG_AFT_@PRD@[k]    = 0
       TG_AVC_@PRD@[k]    = 0
       TG_SWVC_@PRD@[k]   = 0
       TG_TDST_@PRD@[k]   = 0
       TG_ORATE_@PRD@[k]  = 0
       TG_NRATE_@PRD@[k]  = 0
       TG_DVC_@PRD@[k]    = 0
       TG_CNT_@PRD@[k]    = 0
       TG_TTOLL_@PRD@[k]  = 0
       TG_DONE_@PRD@[k]   = 0
   ENDLOOP

   ;
   ; Loop through links to populate Toll-Group Arrays
   ;
LINKLOOP
    IF(LW.tollgrp > 1)
        TG_CNT_@PRD@[LW.tollgrp]   = TG_CNT_@PRD@[LW.tollgrp]   +  1
        TG_AFT_@PRD@[LW.tollgrp]   = TG_AFT_@PRD@[LW.tollgrp]   +  LW.ftype
        TG_TDST_@PRD@[LW.tollgrp]  = TG_TDST_@PRD@[LW.tollgrp]  +  LW.distance
        TG_TTOLL_@PRD@[LW.tollgrp] = TG_TTOLL_@PRD@[LW.tollgrp] +  LW.@PRD@toll
        TG_TVMT_@PRD@[LW.tollgrp]  = TG_TVMT_@PRD@[LW.tollgrp]  +  (LW.Tolls@PRD@vol  * LW.distance)
        TG_SWVC_@PRD@[LW.tollgrp]  = TG_SWVC_@PRD@[LW.tollgrp]  +  ((LW.Tolls@PRD@vol * LW.distance) * LW.Tolls@PRD@VC)
        TG_SWSP_@PRD@[LW.tollgrp]  = TG_SWSP_@PRD@[LW.tollgrp]  +  ((LW.Tolls@PRD@vol * LW.distance) * LW.Tolls@PRD@spd)
    ENDIF
ENDLINKLOOP

   ;
   ; Loop through Toll-Groups to Estimate Average V/C Ratio & Average Weighted Speed
   ;
   LOOP k=@ST@,@TGRPS@

    ; Toll Rate (cents/mile)  - No need to deflate tolls here as tolls are only deflated from input/seed tolls once
       IF( TG_TDST_@PRD@[k] > 0 )
           TG_ORATE_@PRD@[k] = TG_TTOLL_@PRD@[k] /  TG_TDST_@PRD@[k]
       ELSE
           TG_ORATE_@PRD@[k] = 0
       ENDIF

       IF(TG_DONE_@PRD@[K]=0)
           IF (TG_TVMT_@PRD@[k] == 0.0)
               TG_AVC_@PRD@[k]  =  0.0
               TG_ASP_@PRD@[k]  =  0.0
           ELSE
               TG_AVC_@PRD@[k]  =  TG_SWVC_@PRD@[k] / TG_TVMT_@PRD@[k]  ; average VC-Ratio
               TG_ASP_@PRD@[k]  =  TG_SWSP_@PRD@[k] / TG_TVMT_@PRD@[k]  ; average speed
           ENDIF
       ELSE
           TG_AVC_@PRD@[k]  =  0
```

```
                        TG_ASP_@PRD@[k]  =  0
                    ENDIF

                    IF( TG_AVC_@PRD@[k] > 0 )
                        TG_DVC_@PRD@[k] = TG_AVC_@PRD@[k] - @Max_VC@
                    ELSE
                        TG_DVC_@PRD@[k] = 0
                    ENDIF

                    IF(TG_CNT_@PRD@[k]>0) TG_AFT_@PRD@[k] = TG_AFT_@PRD@[k] / TG_CNT_@PRD@[k] ; average FTYPE (for toll-groups
with heterogeneous FTYPEs the answer will be a fraction)

                ENDLOOP

              ;*************************************************************************************************************
                ; SUMMARIZE
                ; (this file can be renamed and used a seed-toll file)

                LOOP k=@ST@,@TGRPS@
                  Print form=13.0 list = k(10),                                          ; Toll Group
                                         TG_CNT_@PRD@[k](10),                            ; Num Links
                                         TG_AFT_@PRD@[k](10.2),                          ; Average FTYPE
                                         TG_ORATE_@PRD@[k](10.4),                        ; Old Toll Rate
                                         TG_ORATE_@PRD@[K](10.4), ;; TG_ORATE_@PRD@[K](10.4),  ; New Toll Rate
                                         TG_ASP_@PRD@[k](10.4),                          ; Average Speed
                                         TG_AVC_@PRD@[k](10.4),                          ; Average VC Ratio
                                         TG_TVMT_@PRD@[k](10.2),                         ; Total VMT
                                         TG_DVC_@PRD@[K](10.4),                          ; Difference in VC Ratio
                                         TG_DONE_@PRD@[K](10),                           ; Toggle Switch (1=Done, 0=NotDone)
                                         file=@iter@_CHECK_@TSitr@_TOLLS_@PRD@.TXT
                ENDLOOP

            ENDPHASE            ; End of Phase
```

Figure 9-9: Contents of Script: Highway_Assignment_Parallel_part4_ApplyVOTSplitTollNonToll.s

```
                  ;
                ;  The input trip table has 6 Vehicle Tables:
                ;     1 - 1-Occ Auto Drivers
                ;     2 - 2-Occ Auto Drivers
                ;     3 - 3+Occ Auto Drivers
                ;     4 - Commercial Vehicles
                ;     5 - Trucks
                ;     6 - Airport Pass. Auto Driver Trips

                FILEI MATI=@iter@_@prd@.VTT ;

                FILEO MATO[1]=@iter@_@prd@_Split_Toll_NonToll.VTT, MO = 901-906,911-916,701-706 NAME = sov_toll_@prd@,
hov2 toll @prd@,    hov3 toll @prd@,    com toll @prd@,    trk toll @prd@,    apx toll @prd@,
                                                                                                  sov_nontoll_@prd@,
hov2 nontoll @prd@, hov3 nontoll @prd@, com nontoll @prd@, trk nontoll @prd@, apx nontoll @prd@,
                                                                                                  sov_tsave_@prd@,
hov2_tsave_@prd@,   hov3_tsave_@prd@,   com_tsave_@prd@,   trk_tsave_@prd@,   apx_tsave_@prd@

                FILEO MATO[2]=@iter@ @prd@ TollPathSkims.MAT, MO = 101-106,201-206,301-306      NAME = sov dist @prd@,
hov2_dist_@prd@,    hov3_dist_@prd@,    com_dist_@prd@,    trk_dist_@prd@,    apx_dist_@prd@,
                                                                                                  sov tolls @prd@,
hov2_tolls_@prd@,   hov3_tolls_@prd@,   com_tolls_@prd@,   trk_tolls_@prd@,   apx_tolls_@prd@,
                                                                                                  sov_time_@prd@,
hov2_time_@prd@,    hov3_time_@prd@,    com_time_@prd@,    trk_time_@prd@,    apx_time_@prd@

                FILEO MATO[3]=@iter@_@prd@_NonTollPathSkims.MAT, MO = 401-406,501-506,601-606,  NAME = sov_dist_@prd@,
hov2 dist @prd@,    hov3 dist @prd@,    com dist @prd@,    trk dist @prd@,    apx dist @prd@,
                                                                                                  sov_tolls_@prd@,
hov2_tolls_@prd@,   hov3_tolls_@prd@,   com_tolls_@prd@,   trk_tolls_@prd@,   apx_tolls_@prd@,
                                                                                                  sov time @prd@,
hov2 time @prd@,    hov3 time @prd@,    com time @prd@,    trk time @prd@,    apx time @prd@

                ;-----------------------------------------------------$
                ;    Read in LOS'E' Capacities and Freeflow Speeds     $
                ;-----------------------------------------------------$
                READ FILE = @in capSpd@
                ;
                ;-----------------------------------------------------$
                ;    Read in Toll Parameters:                         $
                ;-----------------------------------------------------$
                READ FILE = @in_tmin@

                FileI  LOOKUPI[1] =         "@in_AMtfac@"
                LOOKUP LOOKUPI=1,           NAME=AM Tfac,
                        LOOKUP[1]= TOLLGrp, result=AMSOVTFTR,    ;
                        LOOKUP[2]= TOLLGrp, result=AMHV2TFTR,    ;
                        LOOKUP[3]= TOLLGrp, result=AMHV3TFTR,    ;
                        LOOKUP[4]= TOLLGrp, result=AMCOMTFTR,    ;
```

```
        LOOKUP[5]= TOLLGrp, result=AMTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=AMAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

FileI  LOOKUPI[2] =         "@in_PMtfac@"
LOOKUP LOOKUPI=2,           NAME=PM_Tfac,
        LOOKUP[1]= TOLLGrp, result=PMSOVTFTR,    ;
        LOOKUP[2]= TOLLGrp, result=PMHV2TFTR,    ;
        LOOKUP[3]= TOLLGrp, result=PMHV3TFTR,    ;
        LOOKUP[4]= TOLLGrp, result=PMCOMTFTR,    ;
        LOOKUP[5]= TOLLGrp, result=PMTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=PMAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

FileI  LOOKUPI[3] =         "@in_MDtfac@"
        LOOKUP LOOKUPI=3,     NAME=MD Tfac,
        LOOKUP[1]= TOLLGrp, result=MDSOVTFTR,    ;
        LOOKUP[2]= TOLLGrp, result=MDHV2TFTR,    ;
        LOOKUP[3]= TOLLGrp, result=MDHV3TFTR,    ;
        LOOKUP[4]= TOLLGrp, result=MDCOMTFTR,    ;
        LOOKUP[5]= TOLLGrp, result=MDTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=MDAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

FileI  LOOKUPI[4] =         "@in NTtfac@"
        LOOKUP LOOKUPI=4,     NAME=NT_Tfac,
        LOOKUP[1]= TOLLGrp, result=NTSOVTFTR,    ;
        LOOKUP[2]= TOLLGrp, result=NTHV2TFTR,    ;
        LOOKUP[3]= TOLLGrp, result=NTHV3TFTR,    ;
        LOOKUP[4]= TOLLGrp, result=NTCOMTFTR,    ;
        LOOKUP[5]= TOLLGrp, result=NTTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=NTAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

;
;
;-------------------------------------------------------$
;    Toll Curves (Temporarily Using Tampa VOTs          $
;    These need to be updated for MWCOG region          $
;-------------------------------------------------------$
;
;

FileI  LOOKUPI[5] =         "@in_VOT@"
        LOOKUP LOOKUPI=5,     NAME=AM VOT,
        LOOKUP[1]= TOLLFORHR result=AMSOVPROB,
        LOOKUP[2]= TOLLFORHR result=AMHV2PROB,
        LOOKUP[3]= TOLLFORHR result=AMHV3PROB,
        LOOKUP[4]= TOLLFORHR result=AMCOMPROB,
        LOOKUP[5]= TOLLFORHR result=AMTRKPROB,
        LOOKUP[6]= TOLLFORHR result=AMAPXPROB,
```

```
                INTERPOLATE=Y, FAIL= 0,0,0, LIST=N

                FileI  LOOKUPI[6] =         "@in_VOT@"
                        LOOKUP LOOKUPI=6,     NAME=MD VOT,
                        LOOKUP[1]= TOLLFORHR result=MDSOVPROB,
                        LOOKUP[2]= TOLLFORHR result=MDHV2PROB,
                        LOOKUP[3]= TOLLFORHR result=MDHV3PROB,
                        LOOKUP[4]= TOLLFORHR result=MDCOMPROB,
                        LOOKUP[5]= TOLLFORHR result=MDTRKPROB,
                        LOOKUP[6]= TOLLFORHR result=MDAPXPROB,
                INTERPOLATE=Y, FAIL= 0,0,0, LIST=N

                FileI  LOOKUPI[7] =         "@in_VOT@"
                        LOOKUP LOOKUPI=7,     NAME=PM VOT,
                        LOOKUP[1]= TOLLFORHR result=PMSOVPROB,
                        LOOKUP[2]= TOLLFORHR result=PMHV2PROB,
                        LOOKUP[3]= TOLLFORHR result=PMHV3PROB,
                        LOOKUP[4]= TOLLFORHR result=PMCOMPROB,
                        LOOKUP[5]= TOLLFORHR result=PMTRKPROB,
                        LOOKUP[6]= TOLLFORHR result=PMAPXPROB,
                INTERPOLATE=Y, FAIL= 0,0,0, LIST=N

                FileI  LOOKUPI[8] =         "@in_VOT@"
                        LOOKUP LOOKUPI=8,     NAME=NT_VOT,
                        LOOKUP[1]= TOLLFORHR result=NTSOVPROB,
                        LOOKUP[2]= TOLLFORHR result=NTHV2PROB,
                        LOOKUP[3]= TOLLFORHR result=NTHV3PROB,
                        LOOKUP[4]= TOLLFORHR result=NTCOMPROB,
                        LOOKUP[5]= TOLLFORHR result=NTTRKPROB,
                        LOOKUP[6]= TOLLFORHR result=NTAPXPROB,
                INTERPOLATE=Y, FAIL= 0,0,0, LIST=N

                PHASE=LINKREAD

                    ;
                    ; Use appropriate loaded speeds for skimming
                    ;
                    SPEED = SPEEDFOR(LI.@PRD@LANE,LI.SPDCLASS)    ; Restrained speed (min) in pp iter

@NON_PP@            IF( @TSitr@ == 1 )
@NON_PP@                SPEED = LI.@prev@@PRD@SPD                 ; Restrained speed (min) in non-pp iter
@NON_PP@            ELSE
@NON_PP@                SPEED = LI.Tolls@prd@SPD                  ; Restrained speed (min) in non-pp iter
@NON_PP@            ENDIF

                    ;
                    ; Compute loaded times
                    ;
                    IF (SPEED = 0)
                        T1 = 9999                                       ; outliers
                    ELSE
```

```
                    T1 = (LI.DISTANCE / SPEED) * 60.0  +  LI.TIMEPEN ; (miles / mph) * 60 = minutes
                ENDIF
                ;
                ; Define        link level tolls by vehicle type here:
                LW.SOV@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(1,LI.TOLLGRP)  ; SOV       TOLLS in 2007 cents
                LW.HV2@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(2,LI.TOLLGRP)  ; HOV 2 occ TOLLS in 2007 cents
                LW.HV3@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(3,LI.TOLLGRP)  ; HOV 3+occ TOLLS in 2007 cents
                LW.CV@PRD@TOLL  = LI.@PRD@TOLL * @PRD@ TFAC(4,LI.TOLLGRP)  ; CV        TOLLS in 2007 cents
                LW.TRK@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(5,LI.TOLLGRP)  ; Truck     TOLLS in 2007 cents
                LW.APX@PRD@TOLL = LI.@PRD@TOLL * @PRD@ TFAC(6,LI.TOLLGRP)  ; AP Pax    TOLLS in 2007 cents
                ;$
                ;
                ;   The highway network is coded with limit codes from 1 to 9
                ;     LimitCode addGrp  Definition
                ;     --------  -----   -----------------------------------------------------------
                ;        1        1     All vehicles accepted
                ;        2        2     Only HOV2 (or greater) vehicles accepted only
                ;        3        3     Only HOV3 vehicles accepted only
                ;        4        4     Med,Hvy Trks not accepted, all other traffic is accepted
                ;        5        5     Airport Passenger Veh. Trips
                ;        6-8      6     (Unused)
                ;        9        7     No vehicles are accepted at all
                ;                 9     TollGrp > 0
                ;
                IF (LI.@PRD@LIMIT==1)
                    ADDTOGROUP=1
                ELSEIF (LI.@PRD@LIMIT==2)
                    ADDTOGROUP=2
                ELSEIF (LI.@PRD@LIMIT==3)
                    ADDTOGROUP=3
                ELSEIF (LI.@PRD@LIMIT==4)
                    ADDTOGROUP=4
                ELSEIF (LI.@PRD@LIMIT==5)
                    ADDTOGROUP=5
                ELSEIF (LI.@PRD@LIMIT==6-8)
                    ADDTOGROUP=6
                ELSEIF (LI.@PRD@LIMIT==9)
                    ADDTOGROUP=7
                ENDIF
                ;
                ;
                IF (LI.TOLLGRP > 0) ADDTOGROUP=9  ;; NEW   1-2 = static tolls, >3 = dynamic tolls
                ;
                ;
                ;
                ;
                IF (LI.@PRD@LANE==0) ADDTOGROUP=32 ;; Group to trap and exclude ZERO lane links; Using highest allowed PathGroup
in Cube (1-32)
                ;
                ;
                IF (LI.FTYPE = 0)      ;  LinkClass related to TC[?] above
```

```
                        LINKCLASS = 1      ;
                    ELSEIF (LI.FTYPE = 1)  ;
                        LINKCLASS= 2       ;
                    ELSEIF (LI.FTYPE = 2)  ;
                        LINKCLASS= 3       ;
                    ELSEIF (LI.FTYPE = 3)  ;
                        LINKCLASS= 4       ;
                    ELSEIF (LI.FTYPE = 4)  ;
                        LINKCLASS= 5       ;
                    ELSEIF (LI.FTYPE = 5)  ;
                        LINKCLASS= 6       ;
                    ELSEIF (LI.FTYPE = 6)  ;
                        LINKCLASS= 7       ;
                    ENDIF

                ENDPHASE

                PHASE=ILOOP

                    ;
                    ;
                    ;-----------------------------------------------------$
                    ;    Tolled Network Costs (Including Entire Network)   $
                    ;-----------------------------------------------------$
                    ;
                    ;
                    ;        --toll paths--                                                -- skim dist --
-- skim tolls --                              -- skim time --
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  2,3,5,6,7,   MW[101]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[201]=PATHTRACE(LW.SOV@PRD@TOLL), NOACCESS = 0, MW[301]=PATHTRACE(TIME), NOACCESS = 0 ;  SOV veh
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  3,5,6,7,     MW[102]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[202]=PATHTRACE(LW.HV2@PRD@TOLL), NOACCESS = 0, MW[302]=PATHTRACE(TIME), NOACCESS = 0 ;  HOV 2
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  5,6,7,       MW[103]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[203]=PATHTRACE(LW.HV3@PRD@TOLL), NOACCESS = 0, MW[303]=PATHTRACE(TIME), NOACCESS = 0 ;  HOV 3
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  2,3,5,6,7,   MW[104]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[204]=PATHTRACE(LW.CV@PRD@TOLL ), NOACCESS = 0, MW[304]=PATHTRACE(TIME), NOACCESS = 0 ;  CVs
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  2,3,4,5,6,7, MW[105]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[205]=PATHTRACE(LW.TRK@PRD@TOLL), NOACCESS = 0, MW[305]=PATHTRACE(TIME), NOACCESS = 0 ;  Trucks
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,  6,7,         MW[106]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[206]=PATHTRACE(LW.APX@PRD@TOLL), NOACCESS = 0, MW[306]=PATHTRACE(TIME), NOACCESS = 0 ;  Airport
                    ;
                    ;
                    ;-----------------------------------------------------$
                    ;    Free Network Costs (Excluding Toll Links)         $
                    ;-----------------------------------------------------$
                    ;
                    ;
                    ;        --non-toll paths--                                            -- skim dist --
-- skim tolls (should be zero) --                        -- skim time --
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,2,3,5,6,7,   MW[401]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[501]=PATHTRACE(LW.SOV@PRD@TOLL), NOACCESS = 0, MW[601]=PATHTRACE(TIME), NOACCESS = 0 ;  SOV veh
```

```
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,3,5,6,7,     MW[402]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[502]=PATHTRACE(LW.HV2@PRD@TOLL), NOACCESS = 0, MW[602]=PATHTRACE(TIME), NOACCESS = 0 ;  HOV 2
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,5,6,7,       MW[403]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[503]=PATHTRACE(LW.HV3@PRD@TOLL), NOACCESS = 0, MW[603]=PATHTRACE(TIME), NOACCESS = 0 ;  HOV 3
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,2,3,5,6,7,   MW[404]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[504]=PATHTRACE(LW.CV@PRD@TOLL ), NOACCESS = 0, MW[604]=PATHTRACE(TIME), NOACCESS = 0 ;  CVs
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,2,3,4,5,6,7, MW[405]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[505]=PATHTRACE(LW.TRK@PRD@TOLL), NOACCESS = 0, MW[605]=PATHTRACE(TIME), NOACCESS = 0 ;  Trucks
                    PATHLOAD PATH=TIME, EXCLUDEGROUP=32,9,6,7,         MW[406]=PATHTRACE(LI.DISTANCE), NOACCESS = 0,
MW[506]=PATHTRACE(LW.APX@PRD@TOLL), NOACCESS = 0, MW[606]=PATHTRACE(TIME), NOACCESS = 0 ;  Airport
                    ;
                    ;
                    ;-----------------------------------------------------$
                    ;    Identify trips that have positive tolled time savings & Compute VOT
                    ;-----------------------------------------------------$
                    ;
                    ;

                    JLOOP

                        MW[701] = 0
                        MW[702] = 0
                        MW[703] = 0
                        MW[704] = 0
                        MW[705] = 0
                        MW[706] = 0

                        MW[711] = 0
                        MW[712] = 0
                        MW[713] = 0
                        MW[714] = 0
                        MW[715] = 0
                        MW[716] = 0

                        MW[801] = 0
                        MW[802] = 0
                        MW[803] = 0
                        MW[804] = 0
                        MW[805] = 0
                        MW[806] = 0

                        ; -- reduce TollPathTime (perceived tollpath time) if at all
                        ;  min  = MIN( 0, min     - (min/mile              * mile       ) )
                        MW[301] = MIN( 0, MW[301] - (@SOV_TollTimeReliability@ * MW[101]) )
                        MW[302] = MIN( 0, MW[302] - (@HV2_TollTimeReliability@ * MW[102]) )
                        MW[303] = MIN( 0, MW[303] - (@HV3_TollTimeReliability@ * MW[103]) )
                        MW[304] = MIN( 0, MW[304] - ( @CV_TollTimeReliability@ * MW[104]) )
                        MW[305] = MIN( 0, MW[305] - (@TRK_TollTimeReliability@ * MW[105]) )
                        MW[306] = MIN( 0, MW[306] - (@APX_TollTimeReliability@ * MW[106]) )

                        ; -- time difference
```

```
                        MW[701] = ROUND((MW[601] - MW[301]) * 100) / 100   ; SOV veh    time-savings rounded to two decimals
                        MW[702] = ROUND((MW[602] - MW[302]) * 100) / 100   ; HOV 2      time-savings rounded to two decimals
                        MW[703] = ROUND((MW[603] - MW[303]) * 100) / 100   ; HOV 3      time-savings rounded to two decimals
                        MW[704] = ROUND((MW[604] - MW[304]) * 100) / 100   ; CVs        time-savings rounded to two decimals
                        MW[705] = ROUND((MW[605] - MW[305]) * 100) / 100   ; Trucks     time-savings rounded to two decimals
                        MW[706] = ROUND((MW[606] - MW[306]) * 100) / 100   ; Airport    time-savings rounded to two decimals

                        ;    -+ve save-  -trips-         -$-     /     -hr-
                        IF (MW[701] > 0) MW[711] = (MW[201]/100) / (MW[701]/60) ; SOV veh
                        IF (MW[702] > 0) MW[712] = (MW[202]/100) / (MW[702]/60) ; HOV 2
                        IF (MW[703] > 0) MW[713] = (MW[203]/100) / (MW[703]/60) ; HOV 3
                        IF (MW[704] > 0) MW[714] = (MW[204]/100) / (MW[704]/60) ; CVs
                        IF (MW[705] > 0) MW[715] = (MW[205]/100) / (MW[705]/60) ; Trucks
                        IF (MW[706] > 0) MW[716] = (MW[206]/100) / (MW[706]/60) ; Airport

                    ENDJLOOP
                    ;
                    ;
                    ;-----------------------------------------------------$
                    ;     Estimate proportion of trips using toll facility
                    ;-----------------------------------------------------$
                    ;
                    ;
                    IF ((@DO_TOLL_SETTING@ == 1) && (@APPLY_VOT_TO_SPLIT_VTT@ == 1)) ;; If toll setting is enabled, compute VOT
distribution if enabled
                        JLOOP ;                                        -- VOT $/hr --
                            IF (MW[701] > 0) MW[801] = @PRD@ VOT(1, MW[711])  ; SOV veh  ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                            IF (MW[702] > 0) MW[802] = @PRD@_VOT(2, MW[712])  ; HOV 2    ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                            IF (MW[703] > 0) MW[803] = @PRD@ VOT(3, MW[713])  ; HOV 3    ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                            IF (MW[704] > 0) MW[804] = @PRD@ VOT(4, MW[714])  ; CVs      ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                            IF (MW[705] > 0) MW[805] = @PRD@_VOT(5, MW[715])  ; Trucks   ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                            IF (MW[706] > 0) MW[806] = @PRD@ VOT(6, MW[716])  ; Airport  ; Picks toll-choice probability from VOT
distribution for each OD ( Toll($)/Time-Savings(hr) )
                        ENDJLOOP
                    ELSEIF ((@DO_TOLL_SETTING@ == 1) && (@APPLY_VOT_TO_SPLIT_VTT@ == 0)) ;; If toll setting is enabled, & VOT
distribution is disabled, include all trips with toll-based time savings in toll trips
                        JLOOP ; -+ve save-   -trips-
                            IF (MW[701] > 0) MW[801] = 1 ; SOV veh
                            IF (MW[702] > 0) MW[802] = 1 ; HOV 2
                            IF (MW[703] > 0) MW[803] = 1 ; HOV 3
                            IF (MW[704] > 0) MW[804] = 1 ; CVs
                            IF (MW[705] > 0) MW[805] = 1 ; Trucks
                            IF (MW[706] > 0) MW[806] = 1 ; Airport
                        ENDJLOOP
                    ELSE                            ;; If toll setting is disabled, allow all trips to take toll paths; ignore VOT
distribution (=zero non-toll trips)
```

```
                JLOOP
                    MW[801] = 1
                    MW[802] = 1
                    MW[803] = 1
                    MW[804] = 1
                    MW[805] = 1
                    MW[806] = 1
                ENDJLOOP
            ENDIF
            ;
            ;
            ;-----------------------------------------------------$
            ;     Calculate toll and non-toll trips
            ;-----------------------------------------------------$
            ;
            ;
            JLOOP
                ; toll trips
                MW[901] = MI.1.1 * MW[801] ; SOV veh    toll trips
                MW[902] = MI.1.2 * MW[802] ; HOV 2      toll trips
                MW[903] = MI.1.3 * MW[803] ; HOV 3      toll trips
                MW[904] = MI.1.4 * MW[804] ; CVs        toll trips
                MW[905] = MI.1.5 * MW[805] ; Trucks     toll trips
                MW[906] = MI.1.6 * MW[806] ; Airport    toll trips
                ;non-toll trips
                MW[911] = MI.1.1 - MW[901] ; SOV veh    non-toll trips
                MW[912] = MI.1.2 - MW[902] ; HOV 2      non-toll trips
                MW[913] = MI.1.3 - MW[903] ; HOV 3      non-toll trips
                MW[914] = MI.1.4 - MW[904] ; CVs        non-toll trips
                MW[915] = MI.1.5 - MW[905] ; Trucks     non-toll trips
                MW[916] = MI.1.6 - MW[906] ; Airport    non-toll trips
            ENDJLOOP
        ENDPHASE
```

Figure 9-10: Contents of Script: Highway_Assignment_Parallel_part5_BuildPaths.s

```
                ;
            ;  The input trip table has 6 Vehicle Tables:
            ;     1 - 1-Occ Auto Drivers
            ;     2 - 2-Occ Auto Drivers
            ;     3 - 3+Occ Auto Drivers
            ;     4 - Commercial Vehicles
            ;     5 - Trucks
            ;     6 - Airport Pass. Auto Driver Trips
            ;
            PARAMETERS COMBINE=EQUI ENHANCE=@assignType@ ; COMBINE=EQUI requires special attention especially when using Cube Cluster;
see Cube's documentation

            PARAMETERS RELATIVEGAP=@rel gap@   ; Set a relative gap tolerance
            PARAMETERS MAXITERS=@mxIters@      ; We control on relative gap.  This is backup criterion

            ;
            ;    Read in LOS'E' Capacities and Freeflow Speeds
            ;
            READ FILE = @in capSpd@
            ;
            ;
            ;    Read in Toll Parameters:
            ;
            READ FILE = @in_tmin@

            FileI  LOOKUPI[1] =         "@in_AMtfac@"
            LOOKUP LOOKUPI=1,           NAME=AM Tfac,
                    LOOKUP[1]= TOLLGrp, result=AMSOVTFTR,    ;
                    LOOKUP[2]= TOLLGrp, result=AMHV2TFTR,    ;
                    LOOKUP[3]= TOLLGrp, result=AMHV3TFTR,    ;
                    LOOKUP[4]= TOLLGrp, result=AMCOMTFTR,    ;
                    LOOKUP[5]= TOLLGrp, result=AMTRKTFTR,    ;
                    LOOKUP[6]= TOLLGrp, result=AMAPXTFTR,    ;
            INTERPOLATE=N, FAIL= 0,0,0, LIST=N

            FileI  LOOKUPI[2] =         "@in PMtfac@"
            LOOKUP LOOKUPI=2,           NAME=PM_Tfac,
                    LOOKUP[1]= TOLLGrp, result=PMSOVTFTR,    ;
                    LOOKUP[2]= TOLLGrp, result=PMHV2TFTR,    ;
                    LOOKUP[3]= TOLLGrp, result=PMHV3TFTR,    ;
                    LOOKUP[4]= TOLLGrp, result=PMCOMTFTR,    ;
                    LOOKUP[5]= TOLLGrp, result=PMTRKTFTR,    ;
                    LOOKUP[6]= TOLLGrp, result=PMAPXTFTR,    ;
            INTERPOLATE=N, FAIL= 0,0,0, LIST=N

            FileI  LOOKUPI[3] =         "@in_MDtfac@"
                    LOOKUP LOOKUPI=3,     NAME=MD_Tfac,
                    LOOKUP[1]= TOLLGrp, result=MDSOVTFTR,    ;
```

```
        LOOKUP[2]= TOLLGrp, result=MDHV2TFTR,    ;
        LOOKUP[3]= TOLLGrp, result=MDHV3TFTR,    ;
        LOOKUP[4]= TOLLGrp, result=MDCOMTFTR,    ;
        LOOKUP[5]= TOLLGrp, result=MDTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=MDAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

FileI  LOOKUPI[4] =         "@in NTtfac@"
        LOOKUP LOOKUPI=4,           NAME=NT_Tfac,
        LOOKUP[1]= TOLLGrp, result=NTSOVTFTR,    ;
        LOOKUP[2]= TOLLGrp, result=NTHV2TFTR,    ;
        LOOKUP[3]= TOLLGrp, result=NTHV3TFTR,    ;
        LOOKUP[4]= TOLLGrp, result=NTCOMTFTR,    ;
        LOOKUP[5]= TOLLGrp, result=NTTRKTFTR,    ;
        LOOKUP[6]= TOLLGrp, result=NTAPXTFTR,    ;
INTERPOLATE=N, FAIL= 0,0,0, LIST=N

;
;
;    VDF (Volume Delay Function) establishment:
;
;
LOOKUP NAME=VCRV,
    lookup[1] = 1,result = 2,  ;Centroids   old VCRV1
    lookup[2] = 1,result = 3,  ;Fwys        old VCRV2
    lookup[3] = 1,result = 4,  ;MajArts     old VCRV3
    lookup[4] = 1,result = 5,  ;MinArts     old VCRV4
    lookup[5] = 1,result = 6,  ;Colls       old VCRV5
    lookup[6] = 1,result = 7,  ;Expways     old VCRV6
    lookup[7] = 1,result = 8,  ;Ramps       old VCRV2
FAIL=0.00,0.00,0.00, INTERPOLATE=T,file=@VDF File@

;
;
CAPFAC=@CAPFAC@

PHASE=LINKREAD

    C     = CAPACITYFOR(LI.@PRD@LANE,LI.CAPCLASS) * @CAPFAC@  ; Convert hourly capacities to period-specific
    SPEED = SPEEDFOR(LI.@PRD@LANE,LI.SPDCLASS)
    T0    = (LI.DISTANCE/SPEED)*60.0
    ;  Since there is no "DISTANCE =" statement, this assumes that DISTANCE is avail. on input network

    IF (ITERATION = 0)
        ; Define        link level tolls by vehicle type here:
        LW.SOV@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(1,LI.TOLLGRP)  ; SOV       TOLLS in 2007 cents
        LW.HV2@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(2,LI.TOLLGRP)  ; HOV 2 occ TOLLS in 2007 cents
        LW.HV3@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(3,LI.TOLLGRP)  ; HOV 3+occ TOLLS in 2007 cents
        LW.CV@PRD@TOLL  = LI.@PRD@TOLL * @PRD@_TFAC(4,LI.TOLLGRP)  ; CV        TOLLS in 2007 cents
        LW.TRK@PRD@TOLL = LI.@PRD@TOLL * @PRD@ TFAC(5,LI.TOLLGRP)  ; Truck     TOLLS in 2007 cents
        LW.APX@PRD@TOLL = LI.@PRD@TOLL * @PRD@_TFAC(6,LI.TOLLGRP)  ; AP Pax    TOLLS in 2007 cents
```

```
                    ; Initial Iteration LINK IMPEDANCE (HIGHWAY TIME + Equiv.Toll/Time) by vehicle type here:
                    LW.SOV@PRD@IMP = T0               +  LI.TIMEPEN +  (LW.SOV@PRD@TOLL/100.0) * SV@PRD@EQM ;SOV   IMP
                    LW.HV2@PRD@IMP = T0               +  LI.TIMEPEN +  (LW.HV2@PRD@TOLL/100.0) * H2@PRD@EQM ;HOV 2 IMP
                    LW.HV3@PRD@IMP = T0               +  LI.TIMEPEN +  (LW.HV3@PRD@TOLL/100.0) * H3@PRD@EQM ;HOV 3+IMP
                    LW.CV@PRD@IMP  = T0               +  LI.TIMEPEN +  ( LW.CV@PRD@TOLL/100.0) * CV@PRD@EQM ;CV    IMP
                    LW.TRK@PRD@IMP = T0               +  LI.TIMEPEN +  (LW.TRK@PRD@TOLL/100.0) * TK@PRD@EQM ;Truck IMP
                    LW.APX@PRD@IMP = T0               +  LI.TIMEPEN +  (LW.APX@PRD@TOLL/100.0) * AP@PRD@EQM ;APAX  IMP


@ts_string@ @NON_PP@  IF( @TSitr@ == 1 )
@ts_string@ @NON_PP@      SPEED = LI.@prev@@PRD@SPD               ; Restrained speed (min) in non-pp iter    ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts string@ @NON PP@  ELSE                                                                                   ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts string@ @NON PP@      SPEED = LI.Tolls@prd@SPD                ; Restrained speed (min) in non-pp iter    ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@ @NON_PP@  ENDIF                                                                                  ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts string@                                                                                                  ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts string@         LW.TollInitTime = (LI.DISTANCE / SPEED)*60.0                                             ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@                                                                                                  ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         ; Subsequent Iteration LINK IMPEDANCE (HIGHWAY TIME Only) by vehicle type here:  Toll-Setting uses toll-cost only
as part of Toll-Choice
@ts_string@         LW.SOV@PRD@IMP = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         LW.HV2@PRD@IMP = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         LW.HV3@PRD@IMP = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         LW.CV@PRD@IMP  = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         LW.TRK@PRD@IMP = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment
@ts_string@         LW.APX@PRD@IMP = LW.TollInitTime  +  LI.TIMEPEN                                          ; Speeds used in Toll-
Choice to start toll-setting assignment


                    IF (LI.@PRD@TOLL > 0)
                        PRINT LIST = 'iteration: ',iteration(3),' A: ',A(7),' B: ',B(7),
                        ' DISTANCE: ',LI.DISTANCE(6.2),
                        ' LI.@PRD@TOLL: ',                        LI.@PRD@TOLL(5.2),
                        ' FFSPEED: ',                                   SPEED(5.2),
                        ' @PRD@_TFAC(1,LI.TOLLGRP): ',@PRD@_TFAC(1,LI.TOLLGRP)(5.1),
                        ' SV@PRD@EQM: ',                          SV@PRD@EQM(5.1),
                        ' LW.SOV@PRD@TOLL: ',                LW.SOV@PRD@TOLL(5.2),
                        ' T0: ',                                        T0(5.2),
                        ' LW.SOV@PRD@IMP',                    LW.SOV@PRD@IMP(5.2),
```

```
                    file = @prd@CHK.LKREAD
                    ENDIF

                ENDIF


                ;$
                ;
                ;   The highway network is coded with limit codes from 1 to 9
                ;     LimitCode addGrp  Definition
                ;     --------  -----   ------------------------------------------------------
                ;        1        1     All vehicles accepted
                ;        2        2     Only HOV2 (or greater) vehicles accepted only
                ;        3        3     Only HOV3 vehicles accepted only
                ;        4        4     Med,Hvy Trks not accepted, all other traffic is accepted
                ;        5        5     Airport Passenger Veh. Trips
                ;        6-8      6     (Unused)
                ;        9        7     No vehicles are accepted at all
                ;                 9     TollGrp > 0
                ;
                IF (LI.@PRD@LIMIT==1)
                    ADDTOGROUP=1
                ELSEIF (LI.@PRD@LIMIT==2)
                    ADDTOGROUP=2
                ELSEIF (LI.@PRD@LIMIT==3)
                    ADDTOGROUP=3
                ELSEIF (LI.@PRD@LIMIT==4)
                    ADDTOGROUP=4
                ELSEIF (LI.@PRD@LIMIT==5)
                    ADDTOGROUP=5
                ELSEIF (LI.@PRD@LIMIT==6-8)
                    ADDTOGROUP=6
                ELSEIF (LI.@PRD@LIMIT==9)
                    ADDTOGROUP=7
                ENDIF
                ;
@ts string@     ;
@ts_string@     ;
@ts string@     IF (LI.TOLLGRP > 0) ADDTOGROUP=9  ;; NEW   1-2 = static tolls, >2 = dynamic tolls
@ts_string@     ;
@ts_string@     ;
                ;
                IF (LI.@PRD@LANE==0) ADDTOGROUP=32 ;; Group to trap and exclude ZERO lane links; Using highest allowed PathGroup in
Cube (1-32)
                ;
                ;
                IF (LI.FTYPE = 0)      ;  LinkClass related to TC[?] above
                    LINKCLASS = 1      ;
                ELSEIF (LI.FTYPE = 1)  ;
                    LINKCLASS= 2       ;
                ELSEIF (LI.FTYPE = 2)  ;
```

```
                    LINKCLASS= 3       ;
                ELSEIF (LI.FTYPE = 3)  ;
                    LINKCLASS= 4       ;
                ELSEIF (LI.FTYPE = 4)  ;
                    LINKCLASS= 5       ;
                ELSEIF (LI.FTYPE = 5)  ;
                    LINKCLASS= 6       ;
                ELSEIF (LI.FTYPE = 6)  ;
                    LINKCLASS= 7       ;
                ENDIF

            ENDPHASE

            PHASE=ILOOP

                ;
                ;
                ;  (TOLL/ALL) Multi-user class or multiclass assignment implemented through volume sets (vol[#])
                ;
                ;
                PATHLOAD PATH=LW.SOV@PRD@IMP,  EXCLUDEGROUP=32,  2,3,5,6,7,   VOL[1]=MI.1.1   ;  SOV veh   (toll path/ALL trips)
                PATHLOAD PATH=LW.HV2@PRD@IMP,  EXCLUDEGROUP=32,  3,5,6,7,     VOL[2]=MI.1.2   ;  HOV 2     (toll path/ALL trips)
                PATHLOAD PATH=LW.HV3@PRD@IMP,  EXCLUDEGROUP=32,  5,6,7,       VOL[3]=MI.1.3   ;  HOV 3     (toll path/ALL trips)
                PATHLOAD PATH=LW.CV@PRD@IMP,   EXCLUDEGROUP=32,  2,3,5,6,7,   VOL[4]=MI.1.4   ;  CVs       (toll path/ALL trips)
                PATHLOAD PATH=LW.TRK@PRD@IMP,  EXCLUDEGROUP=32,  2,3,4,5,6,7, VOL[5]=MI.1.5   ;  Trucks    (toll path/ALL trips)
                PATHLOAD PATH=LW.APX@PRD@IMP,  EXCLUDEGROUP=32,  6,7,         VOL[6]=MI.1.6   ;  Airport   (toll path/ALL trips)
@ts string@     ;
@ts_string@     ;
@ts_string@     ;  (Non-TOLL) Multi-user class or multiclass assignment implemented through volume sets (vol[#])
@ts_string@     ;
@ts string@     ;
@ts_string@     PATHLOAD PATH=LW.SOV@PRD@IMP,  EXCLUDEGROUP=32,9,2,3,5,6,7,   VOL[11]=MI.1.7  ;  SOV veh  (non-toll path trips)
@ts string@     PATHLOAD PATH=LW.HV2@PRD@IMP,  EXCLUDEGROUP=32,9,3,5,6,7,     VOL[12]=MI.1.8  ;  HOV 2    (non-toll path trips)
@ts_string@     PATHLOAD PATH=LW.HV3@PRD@IMP,  EXCLUDEGROUP=32,9,5,6,7,       VOL[13]=MI.1.9  ;  HOV 3    (non-toll path trips)
@ts_string@     PATHLOAD PATH=LW.CV@PRD@IMP,   EXCLUDEGROUP=32,9,2,3,5,6,7,   VOL[14]=MI.1.10 ;  CVs      (non-toll path trips)
@ts_string@     PATHLOAD PATH=LW.TRK@PRD@IMP,  EXCLUDEGROUP=32,9,2,3,4,5,6,7, VOL[15]=MI.1.11 ;  Trucks   (non-toll path trips)
@ts string@     PATHLOAD PATH=LW.APX@PRD@IMP,  EXCLUDEGROUP=32,9,6,7,         VOL[16]=MI.1.12 ;  Airport  (non-toll path trips)

            ENDPHASE

            PHASE=ADJUST
                ;
                ;
                ;    Functions
                FUNCTION {

                            V = VOL[1] + VOL[2] + VOL[3] + VOL[4] + VOL[5] + VOL[6]   @ts_string@   + VOL[11] + VOL[12] + VOL[13] +
VOL[14] + VOL[15] + VOL[16]

                            TC[1]= T0*VCRV(1,V/C)  ; TC(LINKCLASS) =
                            TC[2]= T0*VCRV(2,V/C)  ;   Uncongested Time(T0) *
```

```
                            TC[3]= T0*VCRV(3,V/C)  ;   Volume Delay Funtion(VDF)Value
                            TC[4]= T0*VCRV(4,V/C)  ;   VDF function is based on ((V/C)
                            TC[5]= T0*VCRV(5,V/C)  ; Note: the LINKCLASS is defined
                            TC[6]= T0*VCRV(6,V/C)  ; during the LINKREAD phase below.
                            TC[7]= T0*VCRV(7,V/C)  ; during the LINKREAD phase below.

                }

                ; Subsequent Iteration LINK IMPEDANCE (HIGHWAY TIME + Equiv.Toll/Time) by vehicle type here:
                LW.SOV@PRD@IMP = TIME  +  LI.TIMEPEN  +  (LW.SOV@PRD@TOLL/100.0) * SV@PRD@EQM ;SOV   IMP
                LW.HV2@PRD@IMP = TIME  +  LI.TIMEPEN  +  (LW.HV2@PRD@TOLL/100.0) * H2@PRD@EQM ;HOV 2 IMP
                LW.HV3@PRD@IMP = TIME  +  LI.TIMEPEN  +  (LW.HV3@PRD@TOLL/100.0) * H3@PRD@EQM ;HOV 3+IMP
                LW.CV@PRD@IMP  = TIME  +  LI.TIMEPEN  +  ( LW.CV@PRD@TOLL/100.0) * CV@PRD@EQM ;CV    IMP
                LW.TRK@PRD@IMP = TIME  +  LI.TIMEPEN  +  (LW.TRK@PRD@TOLL/100.0) * TK@PRD@EQM ;Truck IMP
                LW.APX@PRD@IMP = TIME  +  LI.TIMEPEN  +  (LW.APX@PRD@TOLL/100.0) * AP@PRD@EQM ;APAX  IMP

@ts_string@     ; Subsequent Iteration LINK IMPEDANCE (HIGHWAY TIME Only) by vehicle type here:  Toll-Setting uses toll-cost only as
part of Toll-Choice
@ts_string@     LW.SOV@PRD@IMP = TIME  +  LI.TIMEPEN                                          ;SOV   IMP
@ts_string@     LW.HV2@PRD@IMP = TIME  +  LI.TIMEPEN                                          ;HOV 2 IMP
@ts_string@     LW.HV3@PRD@IMP = TIME  +  LI.TIMEPEN                                          ;HOV 3+IMP
@ts_string@     LW.CV@PRD@IMP  = TIME  +  LI.TIMEPEN                                          ;CV    IMP
@ts_string@     LW.TRK@PRD@IMP = TIME  +  LI.TIMEPEN                                          ;Truck IMP
@ts_string@     LW.APX@PRD@IMP = TIME  +  LI.TIMEPEN                                          ;APAX  IMP

                IF (LI.@PRD@TOLL > 0)
                    PRINT LIST = 'iteration: ',iteration(3),' A: ',A(7),' B: ',B(7),
                    ' DISTANCE: ',LI.DISTANCE(6.2),
                    ' LI.@PRD@TOLL: ',                        LI.@PRD@TOLL(5.2),
                    ' FFSPEED: ',                                   SPEED(5.2),
                    ' @PRD@ TFAC(1,LI.TOLLGRP): ',@PRD@ TFAC(1,LI.TOLLGRP)(5.1),
                    ' SV@PRD@EQM: ',                             SV@PRD@EQM(5.1),
                    ' LW.SOV@PRD@TOLL: ',                 LW.SOV@PRD@TOLL(5.2),
                    ' T0: ',                                        T0(5.2),
                    ' TIME: ',                                    TIME(5.2),
                    ' LW.SOV@PRD@IMP',                     LW.SOV@PRD@IMP(5.2),
                    file = @prd@CHK.ADJUST
                ENDIF

            ENDPHASE

            PHASE=CONVERGE

                FILEO PRINTO[1] = "@iter@_ue_iteration_report_@prd@.txt"
                PRINT LIST= "Iter: ", Iteration(3.0),"   Gap: ",GAP(16.15),"   Relative Gap: ",RGAP(16.15), PRINTO=1
                IF (RGAP < RGAPCUTOFF)
                    BALANCE=1
                ENDIF

            ENDPHASE
```

Figure 9-11: Contents of Script: Highway_Assignment_Parallel_part6_CalculateRestrainedFinalVolSpdVC.s

```
                ;
                ;-------------------------------------------------------$
                ;    VDF (Volume Delay Function) establishment:         $
                ;-------------------------------------------------------$
                ; Note:  curves updated 2/16/06 rjm/msm
                ;
                LOOKUP NAME=VCRV,
                lookup[1] = 1,result = 2,  ;Centroids   old VCRV1
                lookup[2] = 1,result = 3,  ;Fwys        old VCRV2
                lookup[3] = 1,result = 4,  ;MajArts     old VCRV3
                lookup[4] = 1,result = 5,  ;MinArts     old VCRV4
                lookup[5] = 1,result = 6,  ;Colls       old VCRV5
                lookup[6] = 1,result = 7,  ;Expways     old VCRV6
                lookup[7] = 1,result = 8,  ;Rmps
                FAIL=0.00,0.00,0.00, INTERPOLATE=T,file=@VDF File@
                ;
                ; to keep stratified vehicular volume
                ; only in Iteration 4
                ;
                IF ('@iter@' = 'i4')
                    @iter@@PRD@SOV = V1_1   @ts string@   + V11_1
                    @iter@@PRD@HV2 = V2_1   @ts_string@   + V12_1
                    @iter@@PRD@HV3 = V3_1   @ts_string@   + V13_1
                    @iter@@PRD@CV  = V4_1   @ts_string@   + V14_1
                    @iter@@PRD@TRK = V5_1   @ts string@   + V15_1
                    @iter@@PRD@APX = V6_1   @ts_string@   + V16_1
                ENDIF
                ;
                ;
                ;
                @iter@@prd@VOL = V_1                                                    ; Final Link Volume
                @iter@@prd@VMT = @iter@@prd@VOL * distance                              ; Final Link VMT
                @iter@@prd@FFSPD  =SPEEDFOR(@prd@LANE,SPDCLASS)                         ; Freeflow speed
                @prd@HRLKCAP=CAPACITYFOR(@prd@LANE,CAPCLASS)                            ; Hrly LINK capacity
                @prd@HRLNCAP=CAPACITYFOR(1,CAPCLASS)                                    ; Hrly LANE capacity
                @iter@@prd@VC=(@iter@@prd@VOL*(@pctadt@/100.0)/@prd@HRLKCAP)            ; Period VC ratio
                @iter@@prd@VDF = VCRV((Ftype + 1), @iter@@prd@VC)                       ; Period VDF value
                ; Restrained Link speed(no Queuing delay)
                if (@iter@@prd@VDF > 0)  @iter@@prd@SPD = @iter@@prd@FFSPD / @iter@@prd@VDF
                ATYPE=SPDCLASS%10                                                       ; area type
                _cnt = 1.0                                                              ; counter


@ts string@     ;
@ts_string@     ; -- Duplicating for Toll Setting --
@ts_string@     Tolls@prd@VOL = V_1                                                     ;  @PRD@ Volume
@ts_string@     Tolls@prd@VC=(@iter@@prd@VOL*(@pctadt@/100.0)/@prd@HRLKCAP)             ;  @PRD@ VC ratio
@ts_string@     if (@iter@@prd@VDF > 0)  Tolls@prd@SPD = @iter@@prd@FFSPD / @iter@@prd@VDF   ;  @PRD@ Speed (No queuing)
```

```
@ts_string@     ; -- End Duplicating for Toll Setting --
@ts string@     ;


                ;;
                ; compute WEIGHTED restrained and freeflow SPEEDS for Aggregate summaries

                WRSPD =ROUND(@iter@@prd@VMT * @iter@@prd@SPD)
                WFfSPD=ROUND(@iter@@prd@VMT * @iter@@prd@FFSPD)

                ; Crosstab VMT,WrSPD,WffSPD, by FTYPE and JUR
                CROSSTAB VAR=@iter@@prd@VMT,WrSPD,WffSPD,_CNT,FORM=12cs,
                ROW=JUR,   RANGE=0-23-1,,0-23,
                COL=FTYPE, RANGE=1-6-1,1-6,
                COMP=WrSPD/@iter@@prd@VMT, FORM=12.2cs,     ; AVG INITIAL SPD
                COMP=WffSPD/@iter@@prd@VMT, FORM=12.2cs     ; AVG FINAL SPD

                ; Crosstab @iter@@prd@VMT,WOSPD,WNSPD,_CNT2 by ATYPE and FTYPE
                CROSSTAB VAR=@iter@@prd@VMT,WrSPD,WffSPD,_CNT, FORM=12cs,
                ROW=ATYPE, RANGE=1-7-1,,1-7,
                COL=FTYPE, RANGE=1-6-1,1-6,
                COMP=WrSPD/@iter@@prd@VMT, FORM=12.2cs,     ; AVG INITIAL SPD
                COMP=WffSPD/@iter@@prd@VMT, FORM=12.2cs     ; AVG FINAL SPD

                ; Crosstab VMT,WOSPD,WNSPD,WFSPD, CNT2 by EVC and FTYPE
                CROSSTAB VAR=@iter@@prd@VMT,WrSPD,WffSPD,_CNT, FORM=12cs,
                ROW=@iter@@prd@VC, RANGE=0-5-0.1,,1-99,
                COL=FTYPE, RANGE=1-6-1,1-6,
                COMP=WrSPD/@iter@@prd@VMT, FORM=12.2cs,    ;  AVG INITIAL SPD
                COMP=WFfSPD/@iter@@prd@VMT, FORM=12.2cs    ; Freeflow Speed

                ; PRINT TO check

                print LIST=A(5),' ',B(5),DISTANCE(7.2),' ',@PCTADT@(4.3),' ',@prd@LANE(2.0),' ',
                @prd@HRLKCAP(5.0),' ',@prd@HRLNCAP(5.0),' ',
                @iter@@prd@VOL(8.2),' ',
                @iter@@prd@ffspd(5.1),' ',@iter@@prd@VC(6.4),' ',@iter@@prd@VDF(6.4),' ',
                ftype(3.0),' ',ATYPE(3.0), ' ',@iter@@prd@SPD(5.1),
                FILE=@iter@ @prd@ load link.asc
```

Figure 9-12: Contents of Script: Highway_Assignment_Parallel_part7_SummarizeAndAdjustTolls.s

```
    ARRAY TG_TVMT_@PRD@  = @TGRPS@, ; Total VMT
      TG_ASP_@PRD@   = @TGRPS@, ; Average Speed
      TG_SWSP_@PRD@  = @TGRPS@, ; Sum of VMT weighted Speed
      TG_AFT_@PRD@   = @TGRPS@, ; Average FType
      TG_AVC_@PRD@   = @TGRPS@, ; Average VC-Ratio
      TG_SWVC_@PRD@  = @TGRPS@, ; Sum of VMT Weighted VC-Ratio
      TG_TDST_@PRD@  = @TGRPS@, ; Total Distance
      TG_ORATE_@PRD@ = @TGRPS@, ; Old Toll RATE (cents/mile - deflated)
      TG_NRATE_@PRD@ = @TGRPS@, ; NEW Toll RATE (cents/mile - deflated
      TG_DVC_@PRD@   = @TGRPS@, ; Average VC-Ratio MINUS Max VC-Ratio
      TG_CNT_@PRD@   = @TGRPS@, ; Count of Links
      TG_TTOLL_@PRD@ = @TGRPS@, ; Total Toll (NOT rate) in cents - deflated
      TG_DONE_@PRD@  = @TGRPS@  ; Toggle Switch (1=Done, 0=NotDone)

ZONES = 1

; Set up arrays
PHASE = LINKREAD

     LW.tollgrp          = LI.tollgrp
     LW.ftype            = LI.FTYPE
     LW.distance         = LI.distance
     LW.@PRD@toll        = LI.@PRD@toll

     LW.Tolls@PRD@spd    = LI.Tolls@PRD@spd
     LW.Tolls@PRD@vol    = LI.Tolls@PRD@vol
     LW.Tolls@PRD@vc     = LI.Tolls@PRD@vc

ENDPHASE

;*********************************************************************************************
; estimate average V/C ratio in each tollgroup by time-period
;
;                  Sum of VMT Weighted VC Ratios
;  average V/C = ---------------------------------
;                  Sum of VMT
;
;*********************************************************************************************

PHASE = ILOOP

    ;
    ; Loop through Toll-Groups to initialize Arrays
    ;
    LOOP k=@ST@,@TGRPS@
        TG_TVMT_@PRD@[k]   = 0
        TG_ASP_@PRD@[k]    = 0
        TG_SWSP_@PRD@[k]   = 0
```

```
                        TG_AFT_@PRD@[k]    = 0
                        TG_AVC_@PRD@[k]    = 0
                        TG_SWVC_@PRD@[k]   = 0
                        TG_TDST_@PRD@[k]   = 0
                        TG_ORATE_@PRD@[k]  = 0
                        TG_NRATE_@PRD@[k]  = 0
                        TG_DVC_@PRD@[k]    = 0
                        TG_CNT_@PRD@[k]    = 0
                        TG_TTOLL_@PRD@[k]  = 0
                        TG_DONE_@PRD@[k]   = 0
                    ENDLOOP

                    ;
                    ; Loop through links to populate Toll-Group Arrays
                    ;
                LINKLOOP
                    IF(LW.tollgrp > 1)
                        TG_CNT_@PRD@[LW.tollgrp]   = TG_CNT_@PRD@[LW.tollgrp]   +  1
                        TG_AFT_@PRD@[LW.tollgrp]   = TG_AFT_@PRD@[LW.tollgrp]   +  LW.ftype
                        TG_TDST_@PRD@[LW.tollgrp]  = TG_TDST_@PRD@[LW.tollgrp]  +  LW.distance
                        TG_TTOLL_@PRD@[LW.tollgrp] = TG_TTOLL_@PRD@[LW.tollgrp] +  LW.@PRD@toll
                        TG_TVMT_@PRD@[LW.tollgrp]  = TG_TVMT_@PRD@[LW.tollgrp]  +  (LW.Tolls@PRD@vol  * LW.distance)
                        TG_SWVC_@PRD@[LW.tollgrp]  = TG_SWVC_@PRD@[LW.tollgrp]  +  ((LW.Tolls@PRD@vol * LW.distance) * LW.Tolls@PRD@VC)
                        TG_SWSP_@PRD@[LW.tollgrp]  = TG_SWSP_@PRD@[LW.tollgrp]  +  ((LW.Tolls@PRD@vol * LW.distance) * LW.Tolls@PRD@spd)
                    ENDIF
                ENDLINKLOOP

                    ;
                    ; Loop through Toll-Groups to Estimate Average V/C Ratio & Average Weighted Speed
                    ;
                    LOOP k=@ST@,@TGRPS@

                        ; Toll Rate (cents/mile)  - No need to deflate tolls here as tolls are only deflated from input/seed tolls
once
                        IF( TG_TDST_@PRD@[k] > 0 )
                            TG_ORATE_@PRD@[k] = TG_TTOLL_@PRD@[k] /  TG_TDST_@PRD@[k]
                        ELSE
                            TG_ORATE_@PRD@[k] = 0
                        ENDIF

                        IF(TG_DONE_@PRD@[K]=0)
                            IF (TG_TVMT_@PRD@[k] == 0.0)
                                TG_AVC_@PRD@[k]  =  0.0
                                TG_ASP_@PRD@[k]  =  0.0
                            ELSE
                                TG_AVC_@PRD@[k]  =  TG_SWVC_@PRD@[k] / TG_TVMT_@PRD@[k]  ; average VC-Ratio
                                TG_ASP_@PRD@[k]  =  TG_SWSP_@PRD@[k] / TG_TVMT_@PRD@[k]  ; average speed
                            ENDIF
                        ELSE
                            TG_AVC_@PRD@[k]  =  0
                            TG_ASP_@PRD@[k]  =  0
                        ENDIF
```

```
                        IF( TG_AVC_@PRD@[k] > 0 )
                            TG DVC @PRD@[k] = TG AVC @PRD@[k] - @Max VC@
                        ELSE
                            TG DVC @PRD@[k] = 0
                        ENDIF

                        IF(TG_CNT_@PRD@[k]>0) TG_AFT_@PRD@[k] = TG_AFT_@PRD@[k] / TG_CNT_@PRD@[k] ; average FTYPE (for toll-groups
with heterogeneous FTYPEs the answer will be a fraction)

                    ENDLOOP

                    ;
                    ; TOLL-SETTING
                    ;
                    LOOP K=@ST@,@TGRPS@

                        ;
                        ; Default -- IMPORTANT -- without this, trials will not be efficient and some un-used toll groups will make
the loop run to the max always !!
                        ;
                        TG_DONE_@PRD@[K] = 1 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;

;

                        ; Round Toll Per Toll-Resolution
;
                        TG ORATE @PRD@[K] = ROUND( TG ORATE @PRD@[K] / @TOLL RES@ ) *  @TOLL RES@
;

;
                        IF ((ROUND( TG AVC @PRD@[k] * 10^@VC RES@ ) / (10^@VC RES@)) > @Max VC@)
; CASE 1: when VC > 1.01, Increase TOLL (use Rounded VCs for comparison only)

;
                            ; This is a Non-Linear Function
;
                            TG NRATE @PRD@[K] = TG ORATE @PRD@[K] + (@HOT LAMBDA1@ * LN(ABS(TG DVC @PRD@[k])) + @HOT LAMBDA2@)
;  Raise tolls

;
                            ; Try again only if the Toll is non-zero
;
                            IF ( TG NRATE @PRD@[K] > 0 )  TG DONE @PRD@[K] = 0 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;  Set flag to try again

;
                        ELSEIF (((ROUND( TG_AVC_@PRD@[k] * 10^@VC_RES@ ) / (10^@VC_RES@)) < @Min_VC@) && (TG_AVC_@PRD@[K] > 0))
; CASE 2: when VC < 0.95, Decrease Toll (use Rounded VCs for comparison only)

;
```

```
                            ; This is a Linear Function (better to dampen tolls-reduction function to minimize oscillation)
;
                            TG_NRATE_@PRD@[K] = TG_ORATE_@PRD@[K] * (TG_AVC_@PRD@[K] / @Max_VC@)
;  Lower tolls

;
                            ; Lowered tolls will trigger trials depending on a user-specified setting
;
                            IF ( @ALLOW_EXCLUSIVE_TOLL_REDUCTION_LOOPS@ == 1 )
;
                                TG_DONE_@PRD@[K] = 0 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;  Set flag to try again
                            ENDIF
;

;
                        ELSE
;

; CASE 3: NO TOLL CHANGE WHEN 0.95 <= VC <= 1.01
                            TG_NRATE_@PRD@[K]  = TG_ORATE_@PRD@[K]
;

;
                        ENDIF
;

;
                        ; Round Toll Per Toll-Resolution
;
                        TG NRATE @PRD@[K] = ROUND( TG NRATE @PRD@[K] / @TOLL RES@ ) *  @TOLL RES@
;

;
                        ; CHECKS, if -ve or if caps/floors are hit, don't try again
;

;
                        IF ( TG NRATE @PRD@[K] < 0 )
;   (protect against -ve tolls)

;
                            TG NRATE @PRD@[K] = 0
;
                            TG DONE @PRD@[K]  = 1 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;

;
                        ELSEIF (TG_NRATE_@PRD@[K] > @TOLL_CAP@)
;
```

```
;
                              TG_NRATE_@PRD@[K] = @TOLL_CAP@
;   (assumed already deflated)
                              TG_DONE_@PRD@[K]  = 1 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;

;
                         ELSEIF ( (@period_num@ == 1 || @period_num@ == 3) && (TG_NRATE_@PRD@[K] < @TOLL_FLR_PK@) && (TG_CNT_@PRD@[K] >
0) ) ;

;
                              TG_NRATE_@PRD@[K] = @TOLL_FLR_PK@
;   (assumed already deflated)
                              TG_DONE_@PRD@[K]  = 1 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;

;
                         ELSEIF ( (@period_num@ == 2 || @period_num@ == 4) && (TG_NRATE_@PRD@[K] < @TOLL_FLR_OP@) && (TG_CNT_@PRD@[K] >
0) ) ;

;
                              TG_NRATE_@PRD@[K] = @TOLL_FLR_OP@
;   (assumed already deflated)
                              TG DONE @PRD@[K]  = 1 ; (0=TryThisGroupAgain, 1=Don'tThisGroupAgain)
;

;
                         ELSE
;
                              ; Else Do Nothing
;
                         ENDIF
;

;
                         ; After rounding & capping if the tolls are not effectively different; don't try any more
;
                         IF ( TG NRATE @PRD@[K] == TG ORATE @PRD@[K] )  TG DONE @PRD@[K] = 1 ; (0=TryThisGroupAgain,
1=Don'tThisGroupAgain)  ;

;
                    ENDLOOP
; loop for toll update


;*************************************************************************************************************************
                    ; SUMMARIZE
                    ; (this file can be renamed and used a seed-toll file)

                    LOOP k=@ST@,@TGRPS@
```

```
                Print form=13.0 list = k(10),                    ; Toll Group
                                       TG CNT @PRD@[k](10),      ; Num Links
                                       TG_AFT_@PRD@[k](10.2),    ; Average FTYPE
                                       TG ORATE @PRD@[k](10.4),  ; Old Toll Rate
                                       TG_NRATE_@PRD@[K](10.4),  ; New Toll Rate
                                       TG_ASP_@PRD@[k](10.4),    ; Average Speed
                                       TG_AVC_@PRD@[k](10.4),    ; Average VC Ratio
                                       TG TVMT @PRD@[k](10.2),   ; Total VMT
                                       TG_DVC_@PRD@[K](10.4),    ; Difference in VC Ratio
                                       TG DONE @PRD@[K](10),     ; Toggle Switch (1=Done, 0=NotDone)
                                       file=@iter@_NEW_@TSitr@_TOLLS_@PRD@.TXT
            ENDLOOP

        ENDPHASE            ; End of Phase
```

Figure 9-13: Contents of Script: Highway_Assignment_Parallel_part8_TollEvalTerminationCheck.s

```
;
; READ TOLL RATE BY TOLLGRP
;
lookup name = TOLL@PRD@,
       lookup[1] = 1,result=2,   ; Num Links
       lookup[2] = 1,result=3,   ; Average FTYPE
       lookup[3] = 1,result=4,   ; Old Toll Rate
       lookup[4] = 1,result=5,   ; New Toll Rate
       lookup[5] = 1,result=6,   ; Weighted Speed
       lookup[6] = 1,result=7,   ; Average VC Ratio
       lookup[7] = 1,result=8,   ; VMT
       lookup[8] = 1,result=9,   ; Difference in VC Ratio
       lookup[9] = 1,result=10,  ; Toggle Switch (1=Done, 0=NotDone)
       interpolate=N,fail=0,0,0,file=@iter@_NEW_@TSitr@_TOLLS_@PRD@.TXT

;
; Mark how many toll groups have finished (count of 'ones')
;
PHASE=SUMMARY

    ;
    ; Check to see if design vc raio is achieved
    ;
    FLAG_@PRD@ = 0

    LOOP _IDX=@ST@,@TGRPS@ ;
        IF (TOLL@PRD@(9, IDX) = 1)
            FLAG_@PRD@ = FLAG_@PRD@ + 1
        ELSE
            FLAG @PRD@ = FLAG @PRD@ + 0
        ENDIF
    ENDLOOP

    ;
    ; Save the number of toll-groups that do not need any further adjustment in the log file
    ;
    LOG PREFIX=TOLL_SETTING_CLOSURE, VAR=FLAG_@PRD@

    ;
    ; Create a copy of the latest tolls with generic name
    ;
    LOOP  IDX=@ST@,@TGRPS@
      Print form=13.0 list = _IDX(10),                    ; Toll Group
                             TOLL@PRD@(1,  IDX)(10),      ; Num Links
                             TOLL@PRD@(2, _IDX)(10.2),    ; Average FTYPE
                             TOLL@PRD@(3, _IDX)(10.4),    ; Old Toll Rate
                             TOLL@PRD@(4, _IDX)(10.4),    ; New Toll Rate
                             TOLL@PRD@(5, _IDX)(10.4),    ; Weighted Speed
```

```
                                TOLL@PRD@(6, _IDX)(10.4),     ; Average VC Ratio
                                TOLL@PRD@(7,  IDX)(10.2),     ; VMT
                                TOLL@PRD@(8, _IDX)(10.4),     ; Difference in VC Ratio
                                TOLL@PRD@(9,  IDX)(10),       ; Toggle Switch (1=Done, 0=NotDone)
                                file=LATEST_TOLLS_@PRD@.TXT
      ENDLOOP

      ;
      ; Record number of toll groups finalized
      ;
      Print form=13.0 list = "Number of Toll Groups Finalized in @PRD@ = ",FLAG_@PRD@(10), " out of @T_NUM@"

   ENDPHASE
```

Figure 9-14: Contents of 2020 seed toll file for revised dynamically priced toll-groups: AM Peak Period

3	0	0	0	0	0	0	0	-1.0100	0
4	0	0	0	0	0	0	0	-1.0100	0
5	0	0	0	0	0	0	0	-1.0100	0
6	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
7	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
8	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
9	4	1.00	18.0574	18.0574	0	0	0	-1.0100	0
10	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
11	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
12	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
13	5	1.00	18.0574	18.0574	0	0	0	-1.0100	0
14	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
15	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
16	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
17	0	0	0	0	0	0	0	-1.0100	0
18	0	0	0	0	0	0	0	-1.0100	0
19	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
20	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
21	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
22	5	1.00	72.2296	72.2296	0	0	0	-1.0100	0
23	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
24	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
25	3	1.00	135.4305	135.4305	0	0	0	-1.0100	0
26	4	1.00	18.0574	18.0574	0	0	0	-1.0100	0
27	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
28	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
29	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
30	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
31	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
32	0	0	0	0	0	0	0	-1.0100	0
33	0	0	0	0	0	0	0	-1.0100	0
34	0	0	0	0	0	0	0	-1.0100	0
35	0	0	0	0	0	0	0	-1.0100	0
36	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
37	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
38	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
39	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
40	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
41	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
42	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
43	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
44	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
45	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
46	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
47	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
48	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
49	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
50	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0

51	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
52	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
53	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
54	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
55	0	0	0	0	0	0	0	-1.0100	0
56	0	0	0	0	0	0	0	-1.0100	0
57	0	0	0	0	0	0	0	-1.0100	0
58	0	0	0	0	0	0	0	-1.0100	0
59	0	0	0	0	0	0	0	-1.0100	0
60	0	0	0	0	0	0	0	-1.0100	0
61	0	0	0	0	0	0	0	-1.0100	0
62	0	0	0	0	0	0	0	-1.0100	0
63	0	0	0	0	0	0	0	-1.0100	0
64	0	0	0	0	0	0	0	-1.0100	0
65	0	0	0	0	0	0	0	-1.0100	0
66	0	0	0	0	0	0	0	-1.0100	0
67	0	0	0	0	0	0	0	-1.0100	0
68	0	0	0	0	0	0	0	-1.0100	0
69	0	0	0	0	0	0	0	-1.0100	0
70	0	0	0	0	0	0	0	-1.0100	0
71	0	0	0	0	0	0	0	-1.0100	0
72	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
73	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
74	9	1.00	18.0574	18.0574	0	0	0	-1.0100	0
75	4	1.00	99.3157	99.3157	0	0	0	-1.0100	0
76	2	1.00	478.5211	478.5211	0	0	0	-1.0100	0
77	1	1.00	81.2583	81.2583	0	0	0	-1.0100	0
78	2	1.00	63.2009	63.2009	0	0	0	-1.0100	0
79	3	1.00	117.3731	117.3731	0	0	0	-1.0100	0
80	4	1.00	117.3731	117.3731	0	0	0	-1.0100	0
81	3	1.00	108.3444	108.3444	0	0	0	-1.0100	0
82	1	1.00	108.3444	108.3444	0	0	0	-1.0100	0
83	1	1.00	117.3731	117.3731	0	0	0	-1.0100	0
84	3	1.00	45.1435	45.1435	0	0	0	-1.0100	0
85	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
86	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
87	1	1.00	126.4018	126.4018	0	0	0	-1.0100	0
88	1	1.00	135.4305	135.4305	0	0	0	-1.0100	0
89	2	1.00	343.0906	343.0906	0	0	0	-1.0100	0
90	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
91	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0

Figure 9-15: Contents of 2020 seed toll file for revised dynamically priced toll-groups: PM Peak Period

3	0	0	0	0	0	0	0	-1.0100	0
4	0	0	0	0	0	0	0	-1.0100	0
5	0	0	0	0	0	0	0	-1.0100	0
6	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
7	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
8	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
9	4	1.00	18.0574	18.0574	0	0	0	-1.0100	0
10	3	1.00	153.4879	153.4879	0	0	0	-1.0100	0
11	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
12	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
13	5	1.00	45.1435	45.1435	0	0	0	-1.0100	0
14	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
15	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
16	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
17	0	0	0	0	0	0	0	-1.0100	0
18	0	0	0	0	0	0	0	-1.0100	0
19	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
20	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
21	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
22	5	1.00	18.0574	18.0574	0	0	0	-1.0100	0
23	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
24	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
25	3	1.00	18.0529	18.0529	0	0	0	-1.0100	0
26	4	1.00	18.0574	18.0574	0	0	0	-1.0100	0
27	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
28	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
29	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
30	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
31	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
32	0	0	0	0	0	0	0	-1.0100	0
33	0	0	0	0	0	0	0	-1.0100	0
34	0	0	0	0	0	0	0	-1.0100	0
35	0	0	0	0	0	0	0	-1.0100	0
36	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
37	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
38	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
39	2	1.00	388.2341	388.2341	0	0	0	-1.0100	0
40	1	1.00	135.4305	135.4305	0	0	0	-1.0100	0
41	1	1.00	99.3157	99.3157	0	0	0	-1.0100	0
42	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
43	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
44	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
45	8	1.00	18.0574	18.0574	0	0	0	-1.0100	0
46	2	1.00	18.0574	18.0574	0	0	0	-1.0100	0
47	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
48	3	1.00	63.2009	63.2009	0	0	0	-1.0100	0
49	2	1.00	54.1722	54.1722	0	0	0	-1.0100	0
50	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0

51	3	1.00	54.1722	54.1722	0	0	0	-1.0100	0
52	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
53	6	1.00	18.0574	18.0574	0	0	0	-1.0100	0
54	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
55	0	0	0	0	0	0	0	-1.0100	0
56	0	0	0	0	0	0	0	-1.0100	0
57	0	0	0	0	0	0	0	-1.0100	0
58	0	0	0	0	0	0	0	-1.0100	0
59	0	0	0	0	0	0	0	-1.0100	0
60	0	0	0	0	0	0	0	-1.0100	0
61	0	0	0	0	0	0	0	-1.0100	0
62	0	0	0	0	0	0	0	-1.0100	0
63	0	0	0	0	0	0	0	-1.0100	0
64	0	0	0	0	0	0	0	-1.0100	0
65	0	0	0	0	0	0	0	-1.0100	0
66	0	0	0	0	0	0	0	-1.0100	0
67	0	0	0	0	0	0	0	-1.0100	0
68	0	0	0	0	0	0	0	-1.0100	0
69	0	0	0	0	0	0	0	-1.0100	0
70	0	0	0	0	0	0	0	-1.0100	0
71	0	0	0	0	0	0	0	-1.0100	0
72	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
73	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
74	9	1.00	18.0574	18.0574	0	0	0	-1.0100	0
75	4	1.00	18.0556	18.0556	0	0	0	-1.0100	0
76	2	1.00	18.0402	18.0402	0	0	0	-1.0100	0
77	1	1.00	18.0556	18.0556	0	0	0	-1.0100	0
78	2	1.00	18.0565	18.0565	0	0	0	-1.0100	0
79	3	1.00	18.0520	18.0520	0	0	0	-1.0100	0
80	4	1.00	18.0520	18.0520	0	0	0	-1.0100	0
81	3	1.00	18.0610	18.0610	0	0	0	-1.0100	0
82	1	1.00	18.0610	18.0610	0	0	0	-1.0100	0
83	1	1.00	18.0520	18.0520	0	0	0	-1.0100	0
84	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0
85	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
86	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
87	1	1.00	18.0628	18.0628	0	0	0	-1.0100	0
88	1	1.00	18.0529	18.0529	0	0	0	-1.0100	0
89	2	1.00	18.0466	18.0466	0	0	0	-1.0100	0
90	1	1.00	18.0574	18.0574	0	0	0	-1.0100	0
91	3	1.00	18.0574	18.0574	0	0	0	-1.0100	0

Figure 9-16: Contents of 2020 seed toll file for revised dynamically priced toll-groups: Mid-day and Night Time Periods

3	0	0	0	0	0	0	0	-1.0100	0
4	0	0	0	0	0	0	0	-1.0100	0
5	0	0	0	0	0	0	0	-1.0100	0
6	6	1.00	13.5430	13.5430	0	0	0	-1.0100	0
7	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0
8	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
9	4	1.00	13.5430	13.5430	0	0	0	-1.0100	0
10	3	1.00	13.5430	13.5430	0	0	0	-1.0100	0
11	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
12	2	1.00	13.5430	13.5430	0	0	0	-1.0100	0
13	5	1.00	13.5430	13.5430	0	0	0	-1.0100	0
14	8	1.00	13.5430	13.5430	0	0	0	-1.0100	0
15	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
16	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
17	0	0	0	0	0	0	0	-1.0100	0
18	0	0	0	0	0	0	0	-1.0100	0
19	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
20	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
21	8	1.00	13.5431	13.5431	0	0	0	-1.0100	0
22	5	1.00	13.5431	13.5431	0	0	0	-1.0100	0
23	2	1.00	13.5430	13.5430	0	0	0	-1.0100	0
24	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
25	3	1.00	13.5430	13.5430	0	0	0	-1.0100	0
26	4	1.00	13.5430	13.5430	0	0	0	-1.0100	0
27	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
28	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
29	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
30	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
31	6	1.00	13.5430	13.5430	0	0	0	-1.0100	0
32	0	0	0	0	0	0	0	-1.0100	0
33	0	0	0	0	0	0	0	-1.0100	0
34	0	0	0	0	0	0	0	-1.0100	0
35	0	0	0	0	0	0	0	-1.0100	0
36	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
37	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0
38	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
39	2	1.00	13.5431	13.5431	0	0	0	-1.0100	0
40	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
41	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
42	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
43	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
44	3	1.00	13.5430	13.5430	0	0	0	-1.0100	0
45	8	1.00	13.5431	13.5431	0	0	0	-1.0100	0
46	2	1.00	13.5430	13.5430	0	0	0	-1.0100	0
47	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
48	3	1.00	13.5430	13.5430	0	0	0	-1.0100	0
49	2	1.00	13.5431	13.5431	0	0	0	-1.0100	0
50	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0

51	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0
52	6	1.00	13.5430	13.5430	0	0	0	-1.0100	0
53	6	1.00	13.5431	13.5431	0	0	0	-1.0100	0
54	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
55	0	0	0	0	0	0	0	-1.0100	0
56	0	0	0	0	0	0	0	-1.0100	0
57	0	0	0	0	0	0	0	-1.0100	0
58	0	0	0	0	0	0	0	-1.0100	0
59	0	0	0	0	0	0	0	-1.0100	0
60	0	0	0	0	0	0	0	-1.0100	0
61	0	0	0	0	0	0	0	-1.0100	0
62	0	0	0	0	0	0	0	-1.0100	0
63	0	0	0	0	0	0	0	-1.0100	0
64	0	0	0	0	0	0	0	-1.0100	0
65	0	0	0	0	0	0	0	-1.0100	0
66	0	0	0	0	0	0	0	-1.0100	0
67	0	0	0	0	0	0	0	-1.0100	0
68	0	0	0	0	0	0	0	-1.0100	0
69	0	0	0	0	0	0	0	-1.0100	0
70	0	0	0	0	0	0	0	-1.0100	0
71	0	0	0	0	0	0	0	-1.0100	0
72	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
73	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0
74	9	1.00	13.5430	13.5430	0	0	0	-1.0100	0
75	4	1.00	13.5467	13.5467	0	0	0	-1.0100	0
76	2	1.00	13.5421	13.5421	0	0	0	-1.0100	0
77	1	1.00	13.5458	13.5458	0	0	0	-1.0100	0
78	2	1.00	13.5440	13.5440	0	0	0	-1.0100	0
79	3	1.00	13.5449	13.5449	0	0	0	-1.0100	0
80	4	1.00	13.5449	13.5449	0	0	0	-1.0100	0
81	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0
82	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
83	1	1.00	13.5449	13.5449	0	0	0	-1.0100	0
84	3	1.00	13.5430	13.5430	0	0	0	-1.0100	0
85	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
86	1	1.00	13.5431	13.5431	0	0	0	-1.0100	0
87	1	1.00	13.5376	13.5376	0	0	0	-1.0100	0
88	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
89	2	1.00	13.5521	13.5521	0	0	0	-1.0100	0
90	1	1.00	13.5430	13.5430	0	0	0	-1.0100	0
91	3	1.00	13.5431	13.5431	0	0	0	-1.0100	0

9.3 PT Network Preparation and Path Building

Figure 9-17: PT Script to Create Non-Transit Walk Access and Egress Legs

```
;; ********************************
;; *****1- Create Walk Acc and Egr links
;; ********************************

;; Step 1 ************* Walk Acc Link from Zone Centroid to Bus stops ***************
RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Acc_WK2Bus_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"
  FILEI LINEI[1] = "Inputs\AM Bus Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"


  FILEO NTLEGO = "Acc_WK2Bus.LEG", XN=N  ;Do not output nodes between the start and end nodes of
                                         ;nontransit legs.
  ;PARAMETERS TRANTIME = GENTRSTIME
  PARAMETERS TRANTIME = li.WKTIME

  ;PROCESS PHASE = LINKREAD
  ;     GENTRSTIME = li.WKTIME
  ;ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=1-3675,
      TONODE=20000-54999,
      DIRECTION = 1,
      EXCLUDELINK = (li.MODES = 1-10),   ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 5,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 2 ************* Walk Egr Link from Bus Stops to zone Centroids ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Egr_WK2Bus_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"
  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "Egr_WK2Bus.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=1-3675,
      TONODE=20000-54999,
      DIRECTION = 2,
      EXCLUDELINK = (li.MODES = 1-10),    ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 5,
```

```
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN


;; Step 3 ************* Walk Acc Link from Zone Centroid  to Metro/LRT stations ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT Acc WK2MetLRT Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"

  FILEI LINEI[1] = "Inputs\AM MR LRT Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "Acc_WK2MetLRT.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*30,
      FROMNODE=1-3675,
      TONODE=8001-8098,10001-10099,
      DIRECTION = 1,
      EXCLUDELINK = (li.MODES = 1-10),  ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 4 ************* Walk Egr Link from Metro/LRT stations to zone Centroids ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Egr_WK2MetLRT_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"

  FILEI LINEI[1] = "Inputs\AM_MR_LRT_Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "Egr WK2MetLRT.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*30,
      FROMNODE=1-3675,
      TONODE=8001-8098,10001-10099,
      DIRECTION = 2,
      EXCLUDELINK = (li.MODES = 1-10),    ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
```

```
  ENDPROCESS

ENDRUN

;; Step 5 ********* Walk Acc Link from Zone Centroid to Commuter rail stations **********

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Acc_WK2Com_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"

  FILEI LINEI[1] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "Acc_WK2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*30,
      FROMNODE=1-3675,
      TONODE=9001-9098,
      DIRECTION = 1,
      EXCLUDELINK = (li.MODES = 1-10),  ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 6 ********* Walk Egr Link from commuter rail stations to zone Centroids *********

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_Egr_WK2Com_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"

  FILEI LINEI[1] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "Egr_WK2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 20,
      COST=li.WKTIME,
      MAXCOST=10*30,
      FROMNODE=1-3675,
      TONODE=9001-9098,
      DIRECTION = 2,
      EXCLUDELINK = (li.MODES = 1-10),    ;Mode 1 to 10 cannot be a part of non-transit path.
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN
```

Figure 9-18: PT Script to Create Non-Transit Walk-Transfer Legs between Transit Stops

```
;; ********************************
;; *****1- Create Walk Transfer links
;; ********************************

;; Step 1 ********** Transfer Walk links between bus stops ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT WK Bus2Bus Links.PRN"

  FILEI NETI = "Inputs\PT NET.NET"
  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_Bus2Bus.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 22,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=20000-54999,
      TONODE=20000-54999,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11,14),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 2 ************* Transfer Walk links between buses and Metro  ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_WK_Bus2Met_Links.PRN"

  FILEI NETI = "Inputs\PT_Net.NET"

  FILEI LINEI[1] = "Inputs\AM Bus Lines.lin"
  FILEI LINEI[2] = "Inputs\AM MR LRT Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK Bus2Met.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 23,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=8001-8098,
      TONODE=20000-54999,
      DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 10,
      SLACK = 10*5,
```

```
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 3 ************* Transfer Walk links between buses and LRT  ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT WK Bus2LRT Links.PRN"

  FILEI NETI = "Inputs\PT Net.NET"

  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI LINEI[2] = "Inputs\AM_MR_LRT_Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_Bus2LRT.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 25,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=10001-10099,
          TONODE=20000-54999,
      DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 4 ************* Transfer Walk links between buses and Commuter rail  ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_WK_Bus2Com_Links.PRN"

  FILEI NETI = "Inputs\PT Net.NET"

  FILEI LINEI[1] = "Inputs\AM_Bus_Lines.lin"
  FILEI LINEI[2] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK Bus2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 24,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=9001-9098,
      TONODE=20000-54999,
      DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 10,
      SLACK = 10*5,
      LIST = Y
```

```
  ENDPROCESS

ENDRUN

;; Step 5 ************* Transfer Walk links between Metro Stations ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_WK_Met2Met_Links.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM MR LRT Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK Met2Met.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 33,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=8001-8098,
      TONODE=8001-8098,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 6************* Transfer Walk links between Metro and LRT ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_WK_Met2LRT_Links.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM_MR_LRT_Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_Met2LRT.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 35,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=8001-8098,
      TONODE=10001-10099,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 7 ************* Transfer Walk links between Metro and Commuter rail ***************
```

```
RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT WK Met2Com Links.PRN"

  FILEI NETI = "Inputs\PT NET.NET"
  FILEI LINEI[1] = "Inputs\AM_MR_LRT_Lines.lin"
  FILEI LINEI[2] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK Met2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 34,
      COST=li.WKTIME,
      MAXCOST=10*30,
      FROMNODE=8001-8098,
      TONODE=9001-9098,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 8 ************* Transfer Walk links between LRT Stations ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create_PT_WK_LRT2LRT_Links.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM MR LRT Lines.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_LRT2LRT.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 55,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=10001-10099,
      TONODE=10001-10099,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN


;; Step 9 ***** Transfer Walk links between LRT and Commuter rail Stations ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT WK LRT2Com Links.PRN"
```

```
  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\AM MR LRT Lines.lin"
  FILEI LINEI[2] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_LRT2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 45,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=10001-10099,
      TONODE=9001-9098,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN

;; Step 10 ********** Transfer Walk links between Commuter rail Stations ***************

RUN PGM=PUBLIC TRANSPORT PRNFILE="Create PT WK Com2Com Links.PRN"

  FILEI NETI = "Inputs\PT_NET.NET"
  FILEI LINEI[1] = "Inputs\Mode4AM.lin"
  FILEI FACTORI[1] = "Inputs\AM_TRN.FAC"
  FILEI SYSTEMI = "Inputs\TSYSD.PTS"

  FILEO NTLEGO = "WK_Com2Com.LEG", XN=N

  PARAMETERS TRANTIME = GENTRSTIME

  PROCESS PHASE = LINKREAD
       GENTRSTIME = li.WKTIME
  ENDPROCESS

  PROCESS PHASE = DATAPREP
    GENERATE,
      NTLEGMODE = 44,
      COST=li.WKTIME,
      MAXCOST=10*15,
      FROMNODE=9001-8098,
      TONODE=9001-9098,
          DIRECTION = 3,
          EXCLUDELINK = (li.MODES = 1-10, 11),
      MAXNTLEGS = 3,
      SLACK = 10*5,
      LIST = Y
  ENDPROCESS

ENDRUN
```