AVDTA: Dynamic Traffic Assignment

Michael W. Levin

August 25, 2016

Copyright © 2016 Michael W. Levin

Table of Contents

Chapter 1

Introduction

1.1 Distribution

AVDTA is a research software for studying DTA and mesoscopic simulation models of autonomous vehicle technologies. AVDTA is not available for commercial use. AVDTA may not be used without the permission of the author, Michael W. Levin.

1.2 Support

The author may be contacted at `michaellevin@utexas.edu`.

Chapter 2

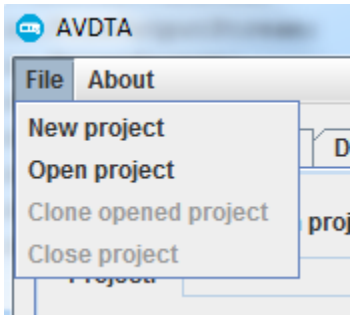
Project structure

2.1 Introduction

AVDTA is organized around *projects*. A project is best thought of as a specific scenario under study. For instance, a project might represent a certain demand scenario or network configuration for a city network or subnetwork. The AVDTA GUI contains methods to modify project options, such as link types or intersection controls. Data files can also be copied to and from Excel for easy modification.

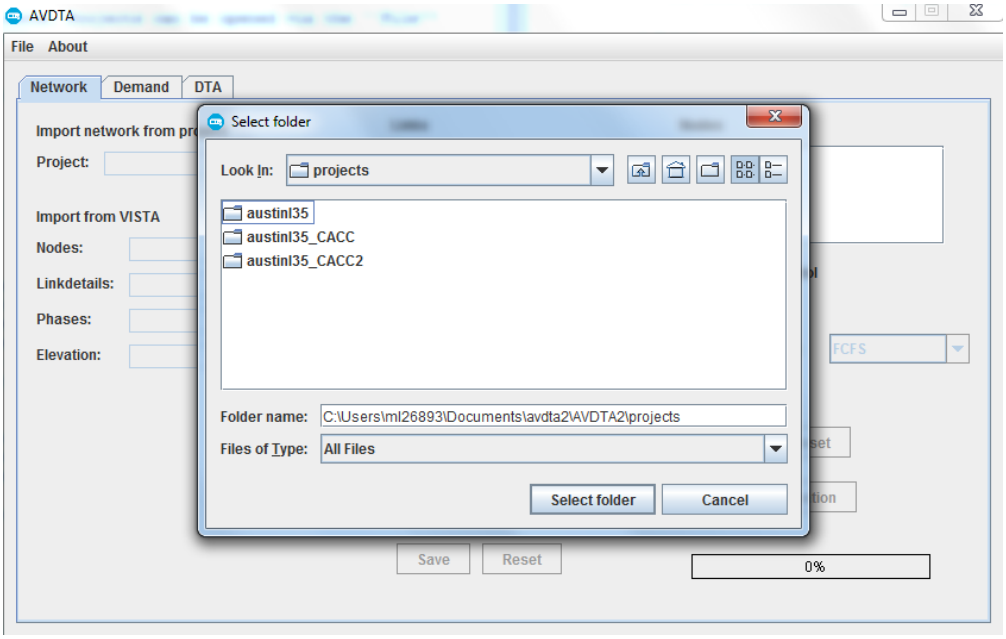
2.2 GUI

The AVDTA GUI is designed to interact with projects. Each instance of AVDTA can have a single project open at a time. Projects can be created and opened via the "File" menu.

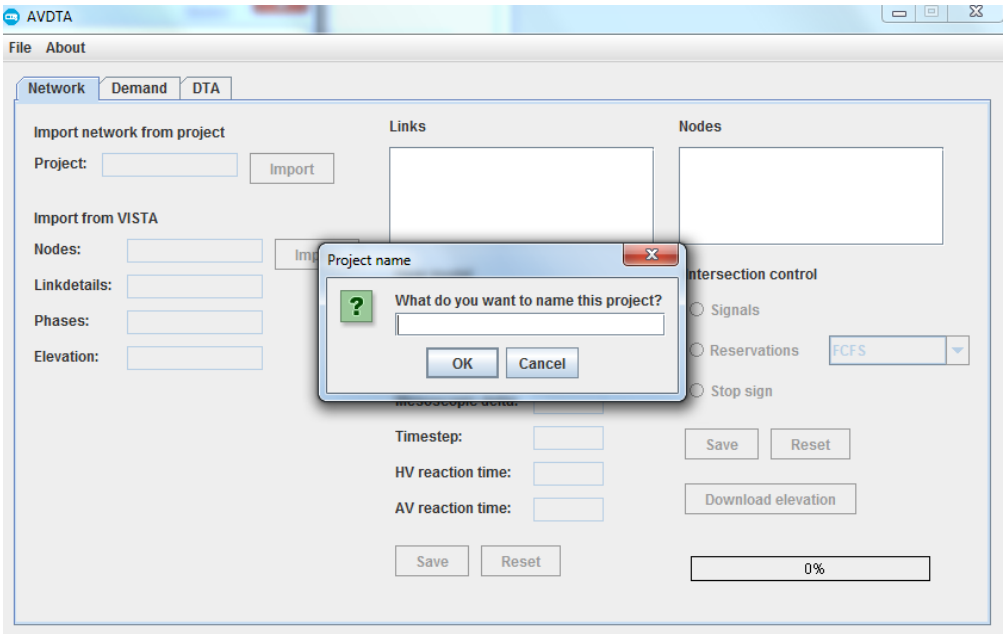


2.2.1 New project

To create a new project, you will first be asked to select the root folder. By default, this folder is `avdta/projects`, but it may be changed.

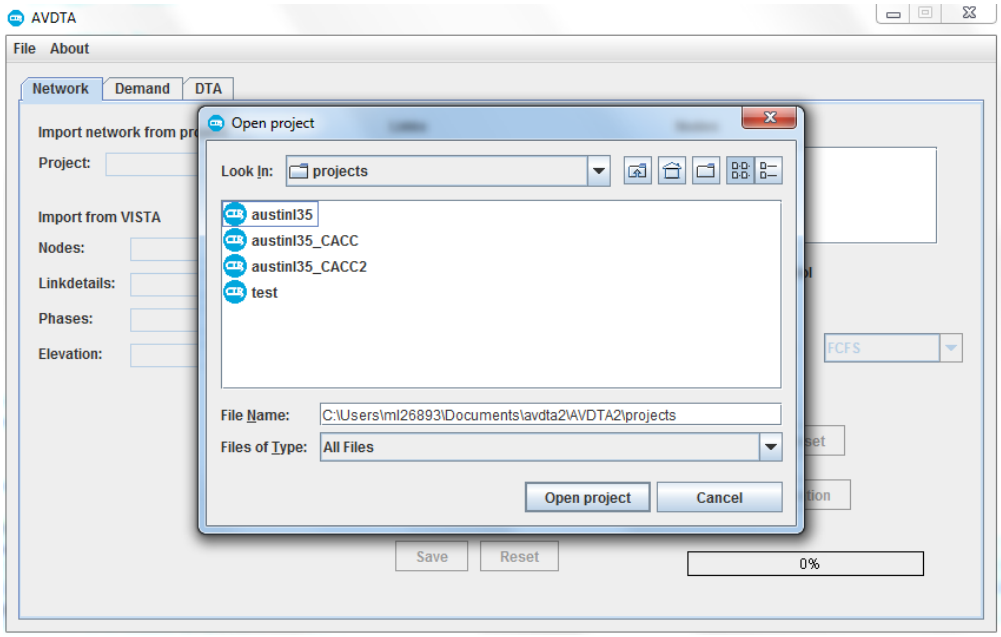


You will then be asked to enter the name of the new project. A project folder with the entered name will be created in the selected root folder. New projects are created with empty data files.



2.2.2 Open project

AVDTA project folders are shown with a special icon. Select a project folder and click "open project" to open it.



2.2.3 Clone opened project

If a project is open, the AVDTA GUI enables the option of creating a clone. Follow the instructions in Section 2.2.1 to choose the location and name of the clone. You can also create a copy of a project folder within the file system.

2.3 Files and folders

A project consists of a file folder containing several specific files and subfolders. The project folder, along with many of its files and subfolders, is automatically generated when a new project is created. However, many files will be empty, requiring the import of data from other projects or other sources. Note that removal or modification of these files outside of the AVDTA GUI could result in errors when loading the project. However, adding files or folders will not affect the project.

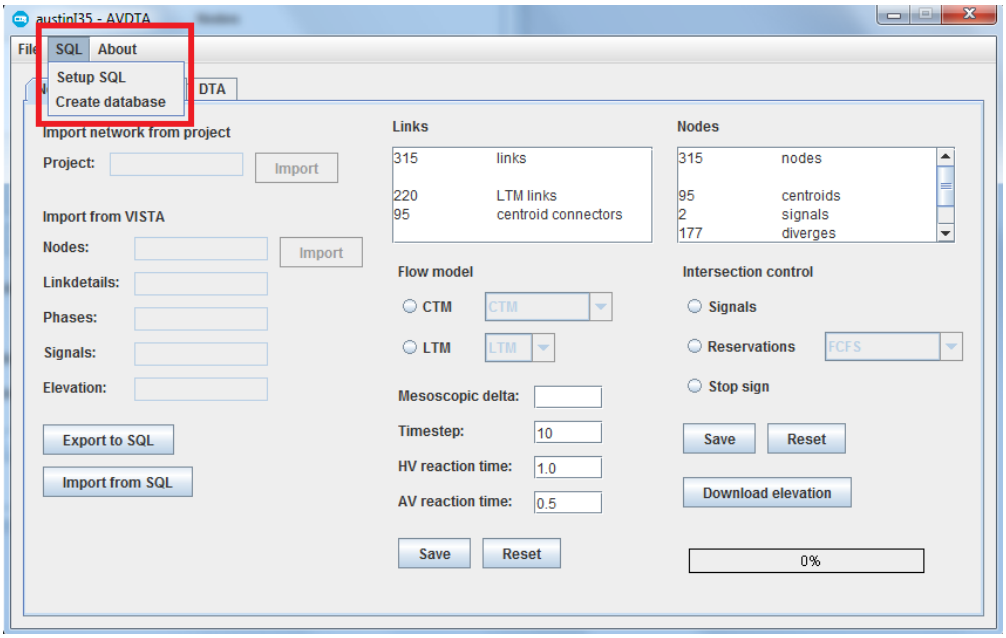
Projects contain several types of files. Text files (`.txt`) contain project inputs or outputs and are intended to be read or modified. However, if modifying these files, ensure that the format and units are correct. Text files are tab-delimited, and have a header indicating the data in each column. Text files may be copied to and from Excel. In addition, AVDTA contains SQL support. A separate SQL installation is required to use SQL.

Data files (`.dat`) are used to load the project and are not intended to be opened or modified. Similarly, files with other unknown extensions are not intended to be opened.

2.3.1 Working with SQL

AVDTA uses an internal file structure that does not require the use of SQL. The purpose of the internal file structure is to allow easy installation and modification of files through other programs (such as Excel). However, if SQL is installed, AVDTA can export data as SQL tables and import data from SQL tables, which allows the use of SQL to work with input data and results.

To use SQL in a project requires two initialization steps. First, set up the SQL connection. AVDTA requires the connection information (IP, port — default 3306) and the username and password of an user account with the permissions to create databases. To set up the SQL connection, select the "Setup SQL" menu item. Note that if SQL is not set up, interface options for SQL support will not be enabled



Next, the project must have an associated SQL database. To create a database, select the "Create database" option. The created database will have the same name as the project, except that all spaces will be replaced with underscores.

2.3.2 Files

2.3.2.1 `project.txt`

The `project.txt` file contains project properties used to load the project within AVDTA. The file consists of two columns: The file consists of two columns:

key	value

name This is the project name that is displayed when AVDTA loads a project. This does not have to correspond to the project folder name.

seed This is the random number generator seed. If two projects have matching seeds, actions involving random numbers performed in the same order should have identical outputs.

type This denotes the project type.

2.3.2.2 `options.txt`

The `options.txt` file contains project parameters that define the loading and simulation of the project. The file consists of two columns:

key	value

Keys are case insensitive. Values are a string that could represent multiple data types, such as integers, floating-point numbers, or booleans ("true" or "false"). The options file is automatically generated with default values when a project is created.

av-reaction-time This is used to determine the capacity and congested wave speed increase due to autonomous vehicles when using the multiclass CTM [9]. A typical value is 0.5 (s). Note that capacity and congested wave speed are scaled from the values in the `networks/links.txt` file.

hv-reaction-time This is used to determine the baseline capacity and congested wave speed when using the multiclass CTM [9]. A typical value is 1 (s). Note that capacity and congested wave speed are scaled from the values in the `networks/links.txt` file.

dynamic-lane-reversal If set to "true", DLR will be activated on DLR links. DLR is a specific type of CTM link, and the type of links can be set in the `networks/links.txt` file.

hvs-use-reservations If set to "true", HVs will not avoid reservation-controlled intersections in their route choices. Reservations will use the legacy early method for intelligent traffic management [2], adapted to DNL [9] for HVs. Otherwise, HVs will avoid reservations in their route choices, passing through reservations only if no other route is available.

simulation-duration This is the duration of the simulation, in seconds. The duration should be sufficiently longer than the demand departure times interval to allow all vehicles to exit the network.

simulation-mesoscopic-step This is the time step used in simulation. For CTM, a typical value is 6 (s). For LTM, a typical value is 10 (s).

ast-duration This the interval for averaging travel times for calculating shortest paths. A typical value is 900 (s).

2.3.2.3 `dta.dat`

The `dta.dat` file marks the project as a DTA project, and should not be modified.

2.3.2.4 `paths.dat`

The `paths.dat` file contains a list of all known paths in the network. AVDTA does not generate a complete list of paths, but all paths that are created during DTA are saved here. The file is used when loading and saving assignments.

2.3.3 Folders

`network` folder The `network` folder contains all network data files, discussed in Chapter 3.

`results` folder The `results` folder is used to store and organize results files.

`demand` folder The `demand` folder contains all demand data files, discussed in Chapter 5.

`assignments` folder The `assignments` folder stores previous assignments for the project. Each assignment consists of a subfolder, usually named by the time at which the assignment was created. Previous assignments can be loaded through the AVDTA GUI.

Each assignment subfolder contains several files. The `log.txt` file contains a log of the DTA run that created the assignment. It includes results from each iteration of DTA as well as summary statistics for the assignment. The `vehicles.dat` file contains summary statistics in a more readable form for AVDTA, and then a list of vehicles and the path they are assigned to. The path ids correspond to a path in the `paths.dat` file. Do not modify the `vehicles.dat` file.

Chapter 3

Traffic network

3.1 Introduction

A *network* in AVDTA defines the road structure through which vehicles can flow. It does not include vehicular demand or trips because these depend on the application. For instance, the vehicle trips for DTA models is different from vehicle trips for SAV models. The network in AVDTA compartmentalizes the road network, with vehicle trips handled separately depending on the application.

A network consists of *nodes* and (directed) *links* connecting nodes. Nodes are further divided into *intersections*, which connect roads, and *zones*, where demand and vehicles can enter or exit the network. AVDTA includes several options for intersection controls and link flow models, which are discussed in Sections 3.2 and 3.3, respectively. The discussion in this document pertains to using and modifying node and link data within AVDTA. References to the theoretical developments of flow models are included.

Network data files are contained in the `network` subfolder within a project folder. Network files include `nodes.txt`, `links.txt`, `link_coordinates.txt`. `signals.txt`, and `phases.txt` (which defines signal phases for nodes). All data files include a header as the first line.

3.2 Nodes

Nodes are either intersections or zones. Intersections represent physical intersections in the road network, and zones represent locations where vehicles or travelers enter and exit. It is typical for zones to double intersections. Each zone should be either a source or a sink.

Each node has sets of incoming and outgoing links. During simulation, nodes determine vehicle flow from incoming links to outgoing links. Incoming links define *sending flows* as a set of vehicles, and outgoing links define *receiving flows*. Vehicle flow is constrained by sending and receiving flows, and may be additionally constrained by intersection conflict constraints.

3.2.1 `nodes.txt`

The `nodes.txt` file contains the data to describe intersections and zones, and consists of the following:

id	type	longitude	latitude	elevation

Columns are tab-delimited, and each line corresponds to a new node. Additional columns will be ignored, and may be deleted if the GUI is used to modify the file.

id Each node must be given an unique id, which is used to identify the node in other data files. Ids should be positive, but do not need to be consecutive.

type The type specifies whether the node is a zone or an intersection. If an intersection, the type also identifies the intersection control. Type options are given below.

Category	Type	Description
Reservations	301	FCFS
	332	Pressure-based
	333	P_0
	334	Q^2 [11]
	335	DE^4 [11]
	321	PHASED
	322	WEIGHTED
Traffic signal	100	Standard traffic signal
Stop sign	200	4-way stop
Centroid	1000	Centroid

Note that merges and diverges will be automatically used. An intersection with only one incoming link will become a diverge, and an intersection with only one outgoing link will become a merge. This is because merges/diverges avoid the need for intersection controls, and also performed better than some reservation policies [10].

longitude/latitude Longitude and latitudes are specified in decimal format, with the sign indicating west/east and north/south, respectively. The longitude/latitude data is used only for display purposes. False data will not affect vehicle loading or simulation. Zones doubling an intersection typically have the same coordinates.

elevation Elevation data is used to determine link grade for estimating vehicle fuel consumptions [7]. False data will not affect vehicle loading or simulation. The GUI contains a method to download elevation data from Google Maps, given longitudes and latitudes.

3.2.2 `phases.txt`

The `phases.txt` file contains information about signal phases. Note that the order in which the phases appear is the order in which they will be created for the signal cycle. Each line in the `phases.txt` file is a separate phase, and the columns are

nodeid	type	sequence	time_red	time_yellow	time_green	num_moves	link_from
link_to							

nodeid The node id for which the phase is used.

type denotes the type of the phase. Currently 1 for all phases.

sequence the order in which the phase is used in the signal cycle. The sequence count starts at 1.

time_red, time_yellow Clearance interval times for this phase.

time_green Green time for this phase.

num_moves The number of protected turning movements, also the size of the "link_from" and "link_to" arrays.

link_from, link_to Protected turning movements for this phase. These consist of two equal-length arrays, one of incoming links and one of outgoing links. A valid turning movement is a pair of an incoming link and an outgoing link. Consequently, links may appear multiple times in the arrays to enumerate all protected turning movements.

3.2.3 `signals.txt`

The `signals.txt` file contains signal offsets for each node. Each node may have one entry. Duplicate entries will overwrite previous ones. A missing entry indicates an offset of 0. The `signals.txt` file columns are

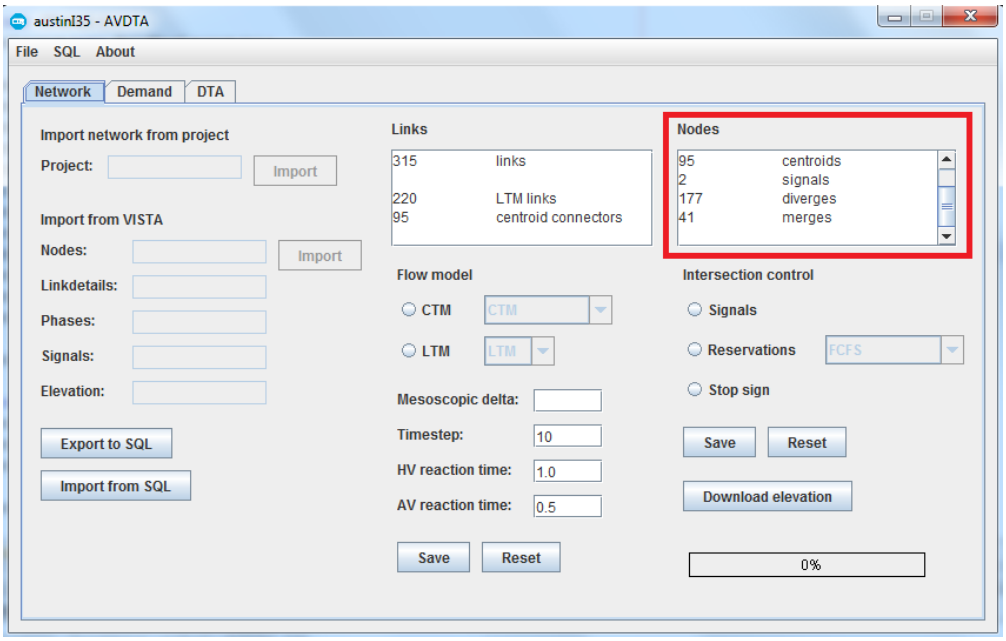
nodeid	offset

nodeid the node id for this signal

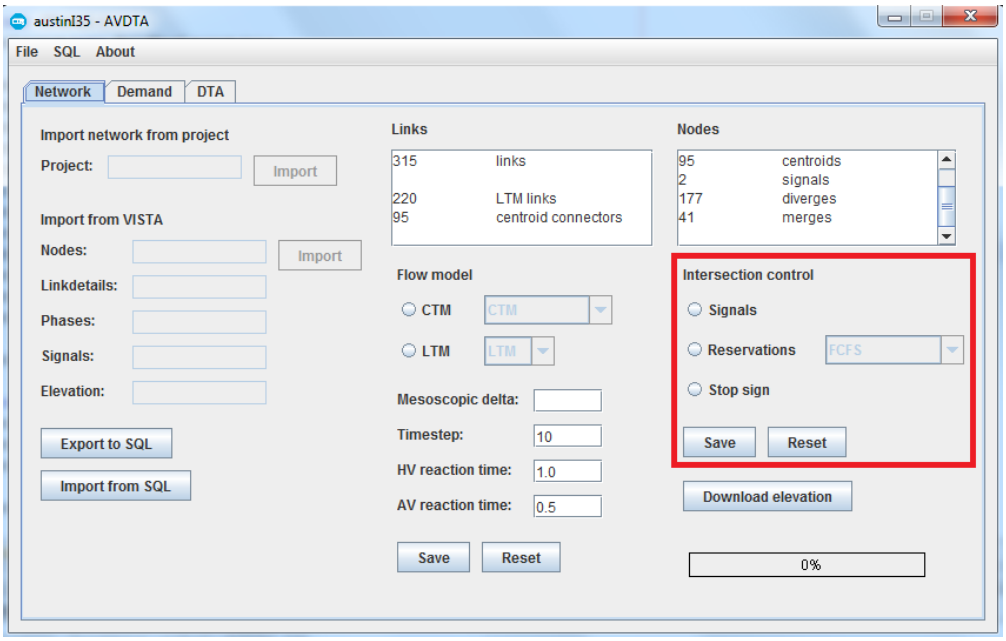
offset the time offset (in seconds). This can be a floating point number.

3.2.4 GUI panel

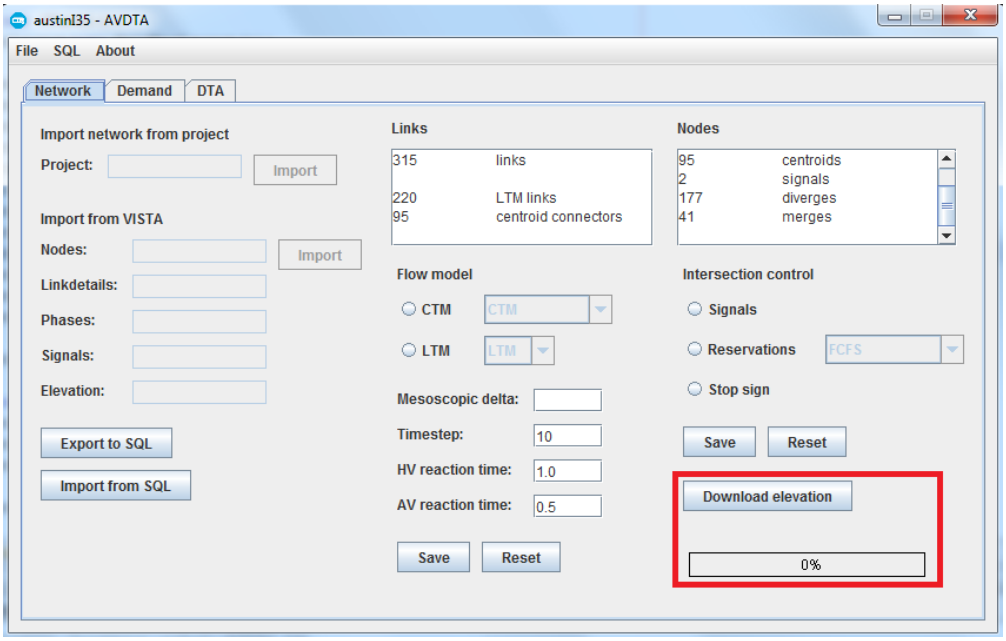
The AVDTA GUI nodes panel is on the right hand side of the "network" tab. The uppermost text area displays information about the network, including the number of total nodes, and nodes of each type.



The next panel controls intersections. Selecting an intersection control, then pressing the "save" button, will result in all intersections converted to the specified intersection control.



The final panel will attempt to download elevation data from the Google Maps API. This will only affect nodes for which the elevation is 0. This requires a connection to the internet. The download may take some time because a delay between requests is necessary to avoid being rejected as a DOS attack. In addition, the Google Maps API may limit the number of requests per day. If the process exits with an error, restart it later. It will save progress and continue from where it left off.



3.3 Links

3.3.1 `links.txt`

Each line in the `links.txt` file corresponds to a link in the network. The `links.txt` file has columns

id	type	source	dest	length	ffspd	w	capacity	num_lanes

id The id of the link. Each link must have an unique, positive id, but the ids do not have to be consecutive.

type This determines the flow model used for the link. The possible types are

Flow model	Type	Description
CTM	100	Multiclass CTM [9]
	102	CTM with DLR [8]
	103	CTM with shared transit lane
LTM	200	Standard LTM [14, 15]
	205	LTM with CACC
Centroid connector	1000	Link between a centroid and a node

Standard CTM is achieved through type 100 with HVs.

source The id of the source node.

dest The id of the destination node.

length The length of the link, in feet.

ffspd The free flow speed of the link, in miles per hour. Note that the free flow travel time is rounded up to the nearest time step.

w The congested wave speed of the link, in miles per hour. For LTM links, the free flow speed, capacity, and congested wave speed must be consistent as the fundamental diagram is over-determined. This is not an issue with CTM because CTM accepts a trapezoidal fundamental diagram. A typical value is half of the free flow speed.

capacity The capacity *per lane* for the link, in vehicles per hour.

num_lanes The number of lanes on the link. This affects the total capacity as well as intersection dynamics.

Note that for centroid connectors, the free flow travel time does not depend on the length and free flow speed, but is a constant 1 time step. Also, centroid connectors are not restricted by wavespeed or capacity limitations.

With CTM, the minimum number of cells per link is two. This means the minimum free flow travel time is two time steps.

3.3.2 `link_coordinates.txt`

The `link_coordinates.txt` file contains an optional list of coordinates for each link used for display purposes only. The contents of this file will not affect simulation results. The columns are

id	coordinates

id The id of the link. This corresponds to the id used in the `links.txt` file.

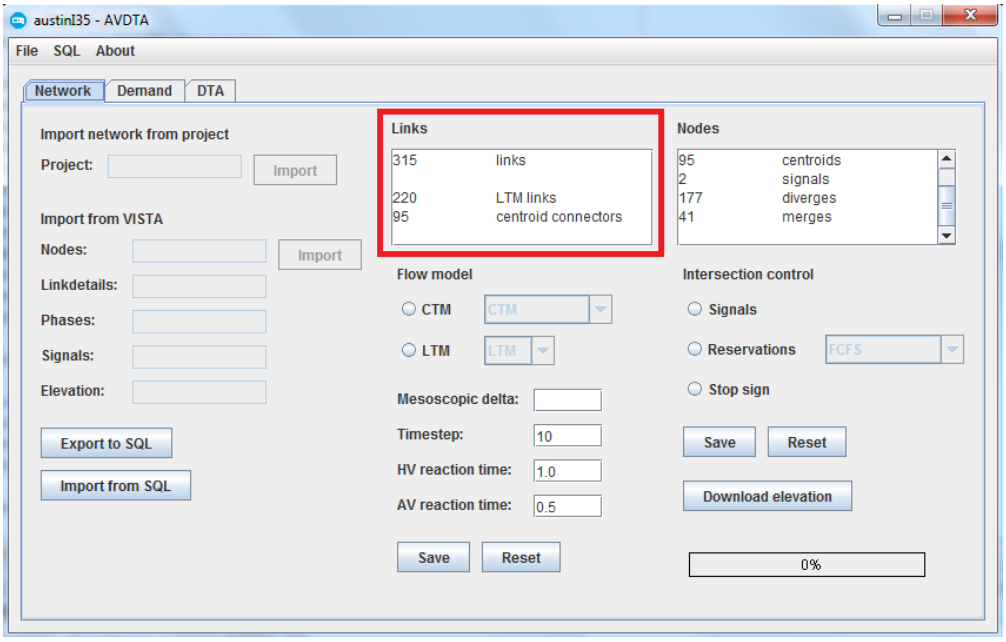
coordinates The list of coordinates used to display the link. Each coordinate should be surrounded by parentheses, with the x and y components separated by a comma. The list of coordinates should be delimited by commas. An example is

```
[(-97.7388,30.2788),(-97.739,30.2783)]
```

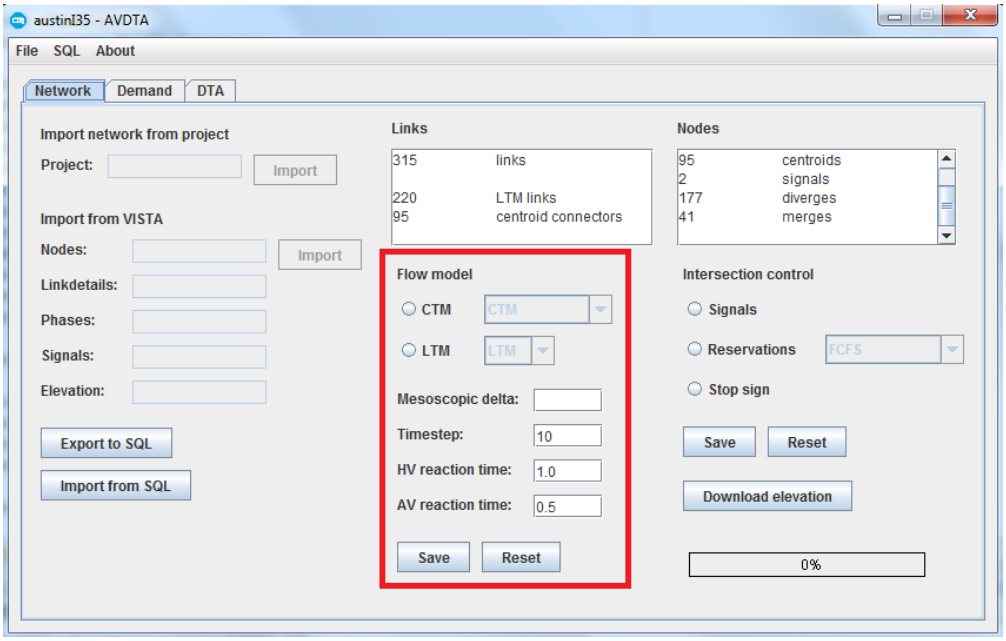
The default link visualization is a straight line from the source node to the destination node. If used, these coordinates will replace the link visualization, so it is advisable to include the source and destination node coordinates in the list.

3.3.3 GUI

The center panel of the "networks" tab is used to interact with links. The uppermost text area displays information about the network, including the number of total links, and the number of links of various types.

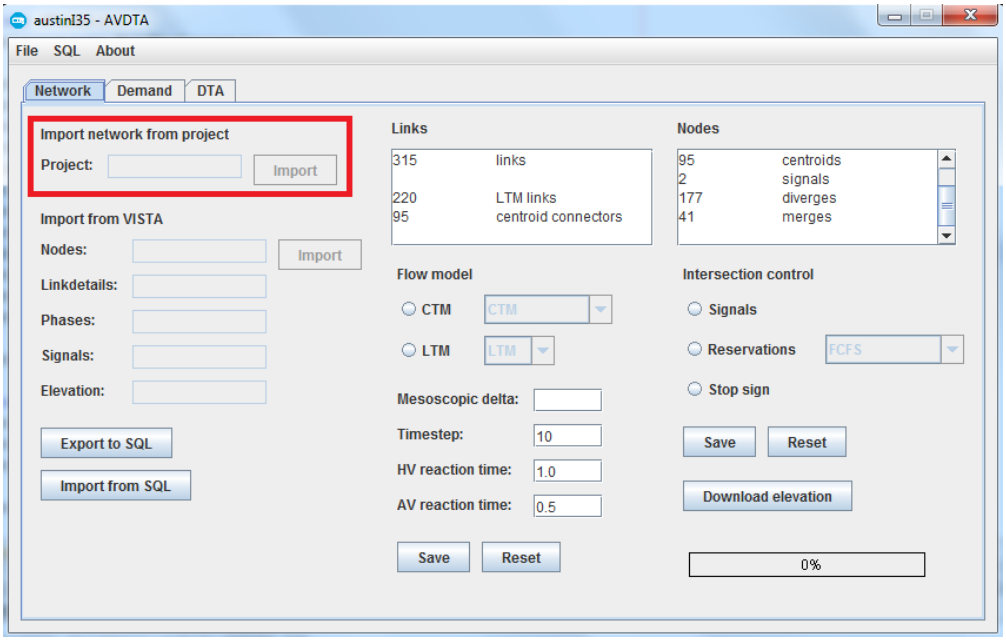


The lower panel provides an interface for modifying the link flow model and network options. Selecting the "CTM" or "LTM" radio buttons, then pressing "save", will change all non-centroid connector links to the selected type. The "mesoscopic delta" field will set the congested wave speed to the specified fraction of the free flow speed for each link. The remaining text fields are network options (Section 2.3.2.2).



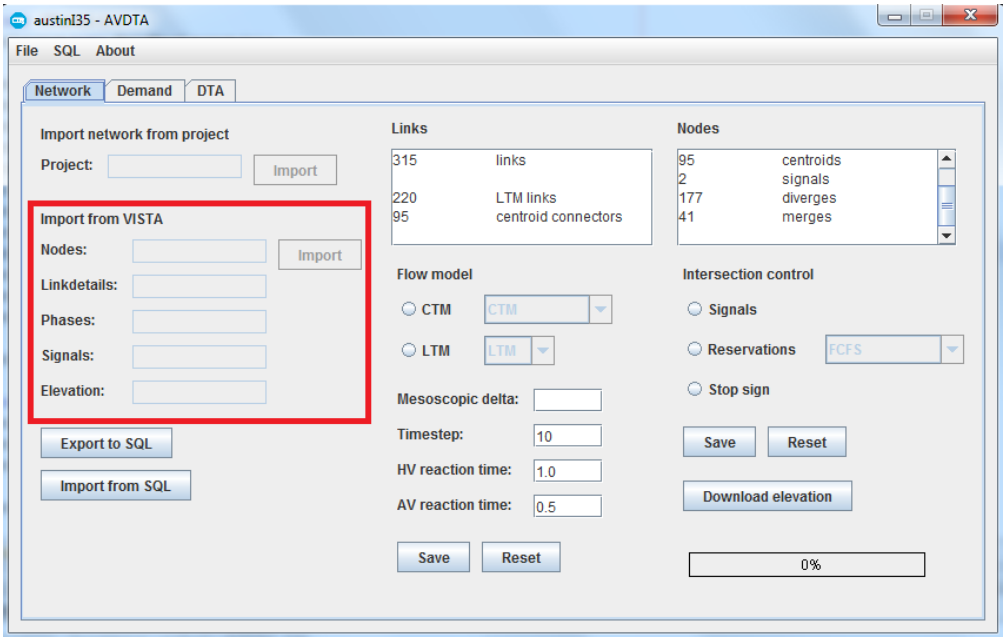
3.4 Import network

The left hand side of the "network tab" contains an interface to import data from other sources. Note that using this interface will overwrite the network data of the project. There are two options. First, the network can be imported from any AVDTA project. This includes DTA projects and projects of other types.



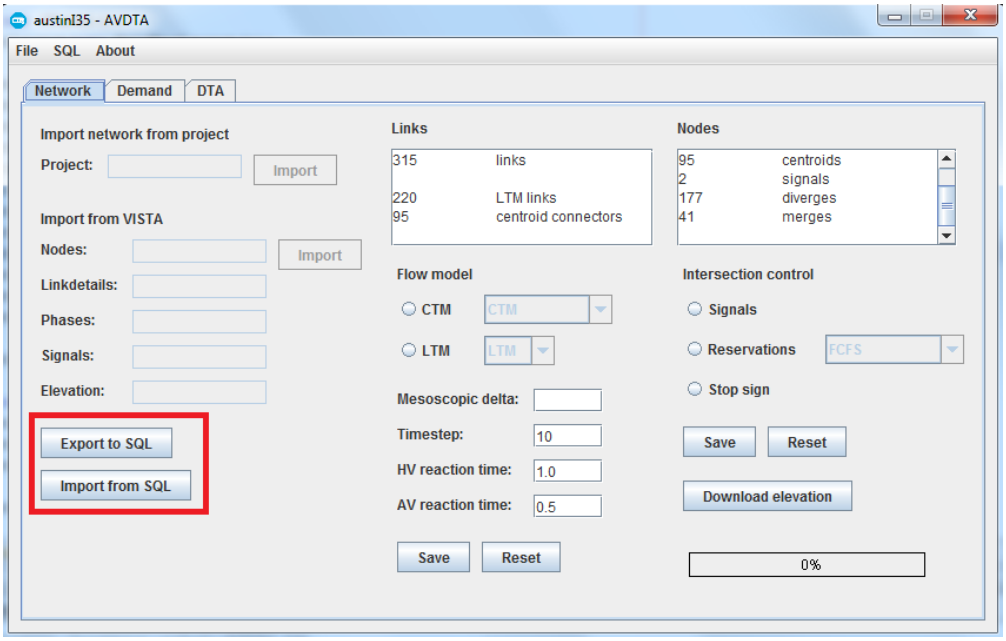
Click on the text field to select a project file.

The second option is to import from VISTA. Copy the `nodes`, `linkdetails`, `phases`, and `signals` tables into text files, and select them by clicking on the text field. An optional elevation file can be used to fill node elevations. It is not required to import the network; if left blank, elevations of 0 will be used.



AVDTA and VISTA use different file formats, so it is not recommended to copy VISTA tables into AVDTA directly.

The third option is to export or import data to/from SQL. This allows data to be manipulated within SQL, then returned to AVDTA in the appropriate format. To use SQL, the connection must first be set up (Section 2.3.1). Exporting will create and populate `nodes`, `linkdetails`, `phases`, `signals`, and `options` tables in the SQL database. Importing will copy the contents of these tables into AVDTA.


austinI35 - AVDTA
File SQL About
Network Demand DTA
Import network from project
Project:
Import
Import from VISTA
Nodes:
Import
Linkdetails:
Phases:
Signals:
Elevation:
Export to SQL
Import from SQL
Links
315 links
220 LTM links
95 centroid connectors
Nodes
95 centroids
2 signals
177 diverges
41 merges
Flow model
CTM
CTM
LTM
LTM
Mesoscopic delta:
Timestep: 10
HV reaction time: 1.0
AV reaction time: 0.5
Save
Reset
Intersection control
Signals
Reservations
FCFS
Stop sign
Save
Reset
Download elevation
0%

Chapter 4

Transit

4.1 Introduction

4.2 Files

Transit is specified by four files, located in the `transit` subfolder within a project folder. The `bus_period.txt` and `bus_frequency.txt` files are used to generate buses. The `bus.txt` and `bus_route_link.txt` files actually specify the buses themselves. To add transit demand, define the routes in the `bus_route_link.txt` file. Define the frequency of each route in the `bus_frequency.txt` file, and the period for which the frequency applies in the `bus_period.txt` file. Then, use AVDTA to generate the `bus.txt` file.

4.2.1 `bus.txt`

The `bus.txt` file contains a list of all buses. These are added as vehicles to the simulation. As buses travel from stop to stop, time-dependent transit travel times between pairs of stops are recorded. The `bus.txt` file consists of the following columns:

id	type	route	dtime

id The unique id for the bus. These ids should not overlap with vehicle ids in demand. Ids must be positive and unique but do not have to be consecutive.

type

type The type indicates the type of vehicle, including the driver, engine, and vehicle behavior. The options for types are indicated below:

Category	type	description
Driver	10	HV
	20	AV
Engine	1	ICV
	2	BEV
Behavior	500	transit (fixed route)
hline		

For instance, the type 511 indicates a bus that is human-driven and uses an internal combustion engine vehicle.

route The id of the bus route, which corresponds to the id in the `bus_route_links.txt` file. This links an individual bus to a specific route.

dtime The departure time of the bus.

4.2.2 bus_route_link.txt

The `bus_route_link.txt` file specifies the routes that buses follow. Each route is an ordered set of connected links. Buses stop at some of the links. The columns are as follows:

route	sequence	link	stop	dwelltime

route This is the route id that connects this bus route with individual buses in the `bus.txt` file.

sequence The order in which the bus visits links. The sequence must start at 1 and be consecutive. Consecutive links must be connected.

stop This indicates whether a stop exists on this link. The values are either 1 (a stop exists) or 0 (no stop exists). The stop, if it exists, is set to the link's downstream node.

dwelltime The time spent waiting at the stop on this link. This is currently not yet implemented.

4.2.3 bus_frequency.txt

The `bus_frequency.txt` file specifies the frequency at which buses operate on each route. (Routes are specified in the `bus_route_link.txt` file.) Note that each route may have multiple entries in this file if it operates at different frequencies in different periods. Each (route, period) combination is unique. The columns are as follows:

route	period	frequency	offset

route This is the route id used in the `bus_route_link.txt` file.

period This references a period in the `bus_period.txt` file.

frequency The time between individual buses on the specified route during the period.

offset The time of the first bus during the period.

4.2.4 `bus_period.txt`

The `bus_period.txt` file specifies bus periods. Routes may have a different frequency for each period (specified in `bus_frequency.txt`) which is used for creating individual buses. Bus periods can overlap. The columns are as follows:

id	starttime	endtime

id The unique id for this bus period. Ids do not have to be consecutive, but they must be positive and unique.

starttime, endtime The start and end time for this bus period.

4.3 GUI

Chapter 5

Demand

5.1 Introduction

The demand specifies personal vehicle trips in terms of origins, destinations, and departure times. This chapter discusses the organization of the demand inputs to DTA. All related data files are contained within the `demand` subfolder of a project. The demand is specified through four files. The `static_od.txt` file is a static (time-invariant) trip table that is representative of the static data available to many planning organizations. The `dynamic_od.txt` file is a dynamic trip table that specifies the number of trips per *assignment interval* (AST). The `demand_profile.txt` file specifies the weight, start time, and duration of each AST. The `demand.txt` file contains a list of discrete vehicles, each with a specific origin, destination, and departure time. The `dynamic_od.txt` file can be generated from the `static_od.txt` and the `demand_profile.txt` files. The `demand.txt` file can be generated from the `dynamic_od.txt` file and the `demand_profile.txt` file. Vehicle types (i.e. HV, AV, etc.) are specified in the `static_od.txt`, `dynamic_od.txt`, and `demand.txt` files. The AVDTA GUI includes methods to generate the `dynamic_od.txt` and `demand.txt` files given appropriate inputs.

5.2 Files

5.2.1 `static_od.txt`

The `static_od.txt` file is a time-invariant trip table. Each line indicates the number of trips of some type between some origin and destination. The columns are

id	type	origin	destination	demand

id An unique id for the trip table entry. Ids do not have to be consecutive, but must be unique and positive.

type The type indicates the type of vehicle, including the driver, engine, and vehicle behavior. The options for types are indicated below:

Category	type	description
Driver	10	HV
	20	AV
Engine	1	ICV
	2	BEV
Behavior	100	personal vehicle (UE routing)
	500	transit (fixed route)

A valid type is the sum of a driver type, an engine type, and a vehicle behavior. Typical types are 111 for HVs and 121 for AVs.

origin, destination These are ids of origin and destination zones in the network (see Section 3.2).

demand A floating point number indicating the number of trips of the specified type from the origin to the destination. Duplicate entries (in terms of type, origin, and destination) are accepted.

5.2.2 `dynamic_od.txt`

The `dynamic_od.txt` file is like the `static_od.txt` file, but with the addition of an AST. Each line The columns are

id	type	origin	destination	ast	demand

ast This is the id of an AST from the `demand_profile.txt` file (Section 5.2.3).

The remainder of the columns are the same as in the `static_od.txt` file (Section 5.2.1). The ids in `dynamic_od.txt` do not need to correspond to the ids in `static_od.txt`. The `dynamic_od.txt` file can be generated within AVDTA from the `static_od.txt` and `demand_profile.txt` files. The distribution of flow over ASTs is determined by the weights in the `demand_profile.txt` file.

5.2.3 `demand_profile.txt`

The `demand_profile.txt` file specifies the ASTs. Each line is an individual AST. The columns are

id	weight	start	duration

id The id must be positive and unique, but does not need to be consecutive.

weight This is the proportion of total flow departing within this AST. Proportions will be scaled to sum to 1 if necessary.

start This is the starting time of the AST, measured in seconds from the beginning of the simulation period.

duration This is the duration of the AST. A typical value is 900 (s).

5.2.4 `demand.txt`

Each line in the `demand.txt` file is an individual vehicle trip. The columns are

id	type	origin	dest	dtime	vot

id The unique vehicle id. Ids must be positive and unique but do not have to be consecutive.

type The vehicle type (see Section 5.2.1 for a list of types).

origin, dest The ids of the origin and destination zones for the vehicle (see Section 3.2).

dtime The departure time of the vehicle, measured in seconds from the start of the simulation period.

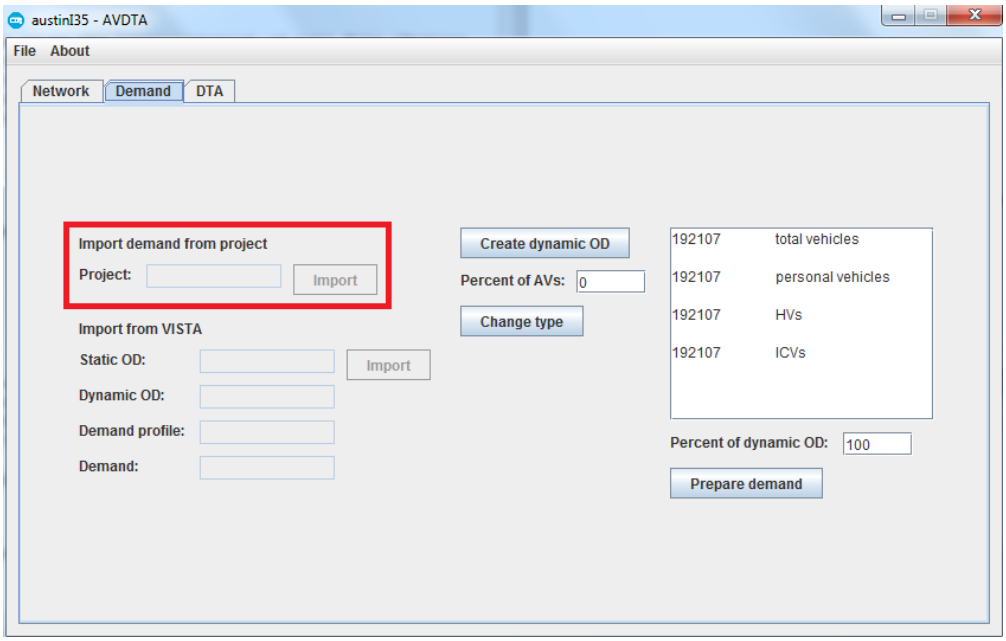
vot The value of time of the vehicle. This is used in certain control policies, such as auctions for reservations. Values of time must be non-negative.

5.3 GUI

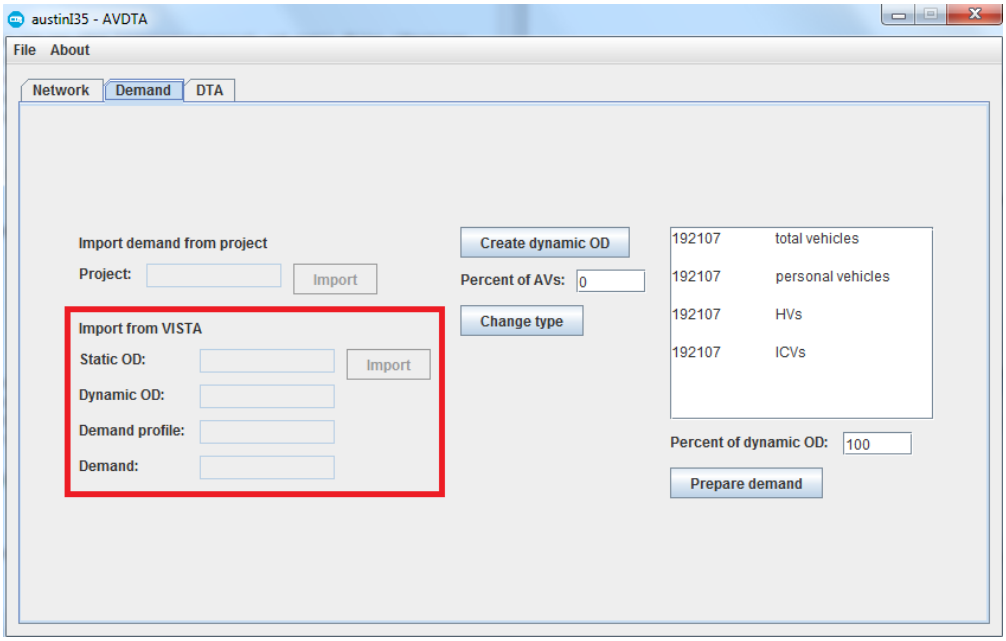
This section describes how to interact with the demand through the AVDTA GUI. The "demand" tab contains all demand interactions.

5.3.1 Import demand

The left hand side contains options to import demand from other sources. The first option is to import the demand from another DTA project. This will copy all demand files from the specified project, overwriting any demand files in the current project. Click on the text field to select the project.

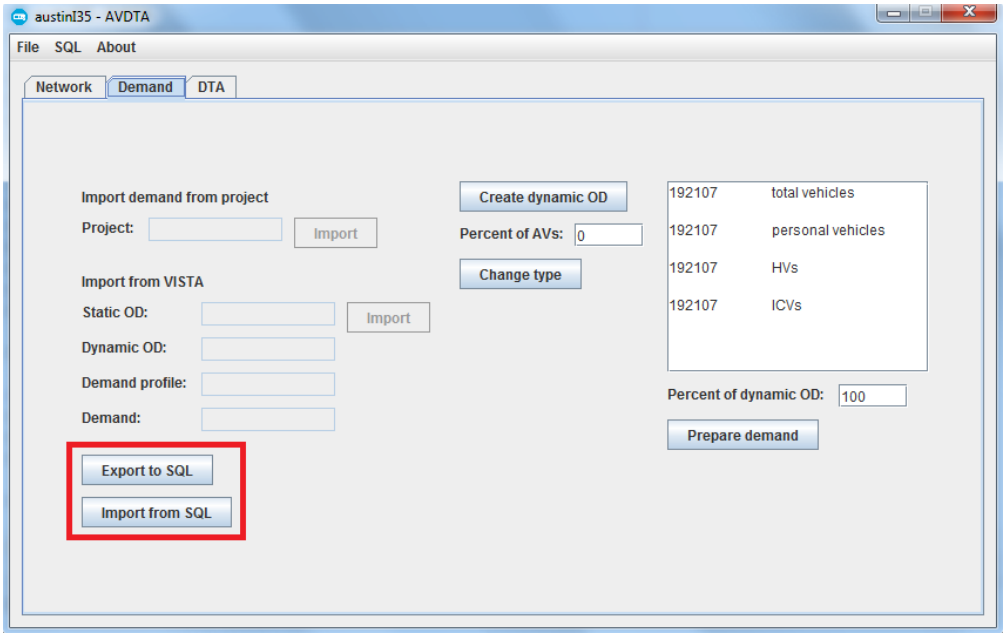


The second option is to import demand from VISTA. This requires the `static_od`, `dynamic_od`, `demand_profile`, and `demand` tables from the VISTA database. Copy them into text files, and select the text files by clicking on the text fields. Note that the file format of AVDTA is not the same as the file format of VISTA, so using the GUI to import demand is recommended.



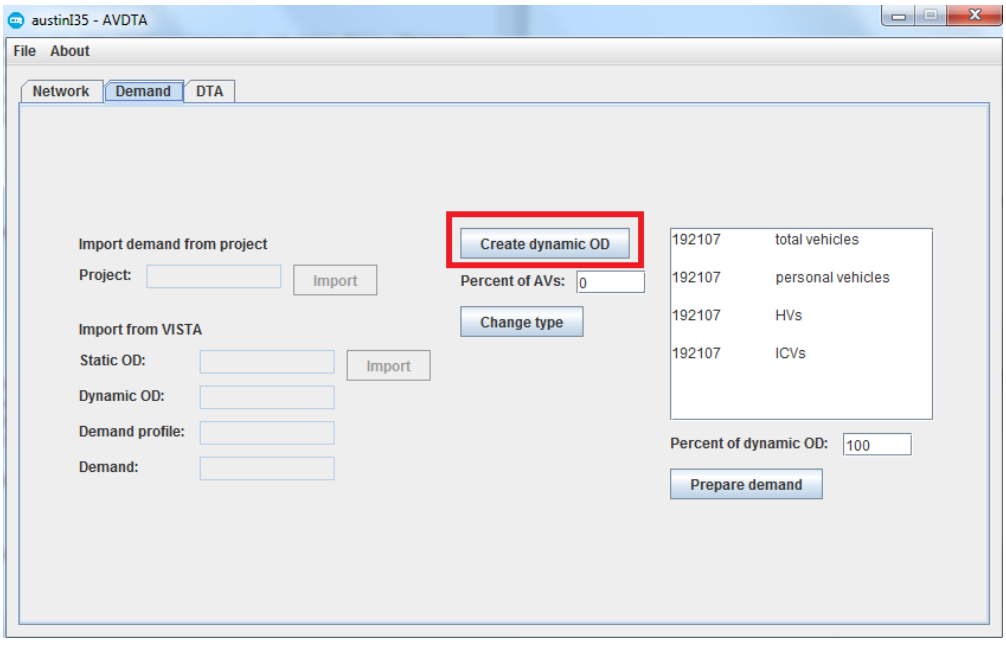
The third option is to export or import data to/from SQL. This allows data to be manipulated within SQL, then returned to AVDTA in the appropriate format. To use SQL, the connection must first be set up (Section 2.3.1). Exporting will create and populate

static_od, dynamic_od, demand_profile, and demand, tables in the SQL database. Importing will copy the contents of these tables into AVDTA.

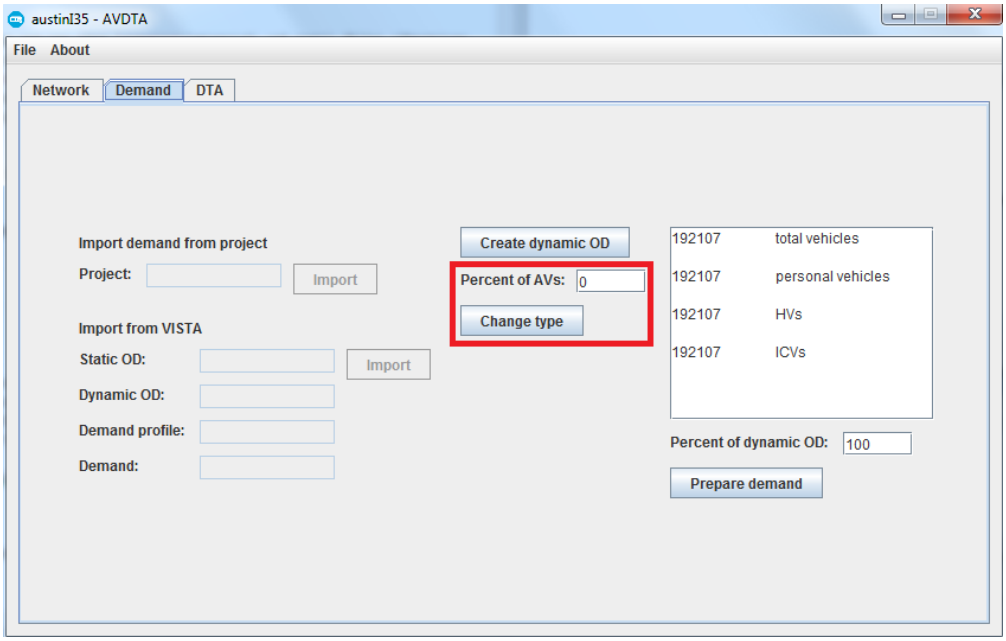


5.3.2 Creating demand

The middle section includes options to change the dynamic_od.txt file. The first button will generate the dynamic_od.txt file based on the static_od.txt and the demand_profile.txt files, as discussed in Section 5.2.2.

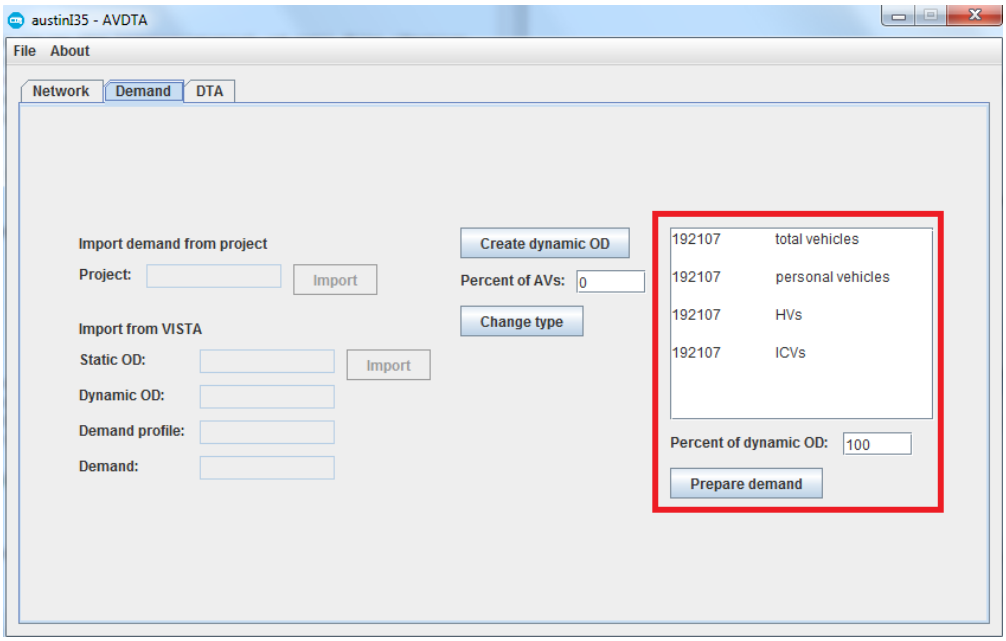


The second option will change the type of vehicles to 111 for HVs or 121 for AVs (see Section 5.2.1 for types) in the `dynamic_od.txt` file based on the specified percent of AVs.



The right hand side describes the individual vehicle trips. The text area lists the total numbers of vehicles, then breaks down the numbers of vehicles by type (driver, engine, and behavior). Click the "prepare demand" button to generate the `demand.txt` file from the `dynamic_od.txt` and `demand_profile.txt` files. You can also scale the percent of demand.

Preparing demand uses a random number generator when the number of trips is not an integer. If d is the number of vehicle trips in `dynamic_od.txt`, prepare demand will create either $\lfloor d \rfloor$ or $\lceil d \rceil$ trips depending on the outcome of the random number generator.


austinI35 - AVDTA
File
About
Network
Demand
DTA
Import demand from project
Project:
Import
Import from VISTA
Static OD:
Import
Dynamic OD:
Demand profile:
Demand:
Create dynamic OD
Percent of AVs:
0
Change type
192107
total vehicles
192107
personal vehicles
192107
HVs
192107
ICVs
Percent of dynamic OD:
100
Prepare demand

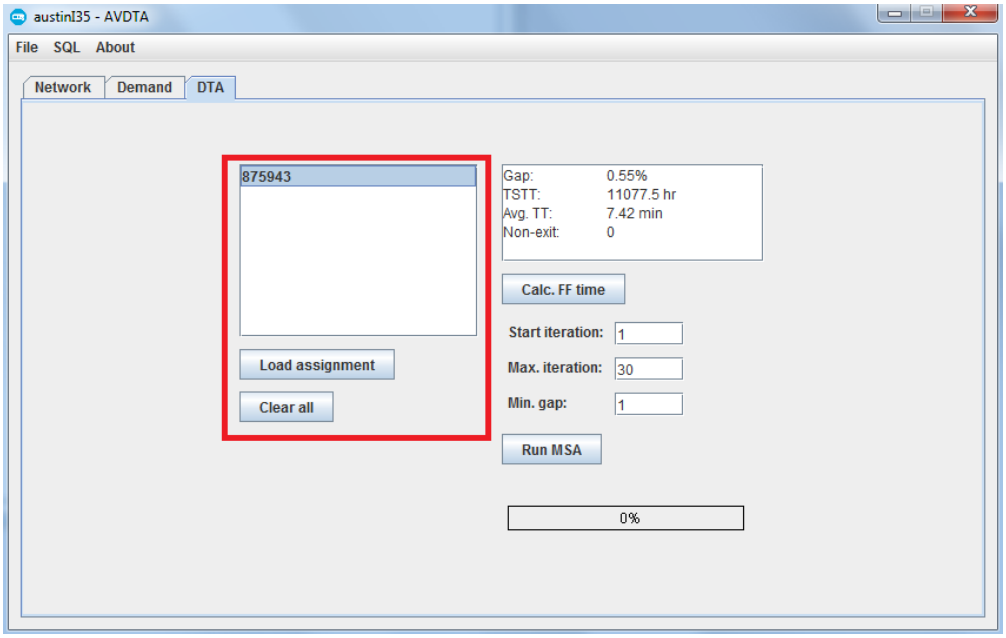
Chapter 6

Dynamic traffic assignment

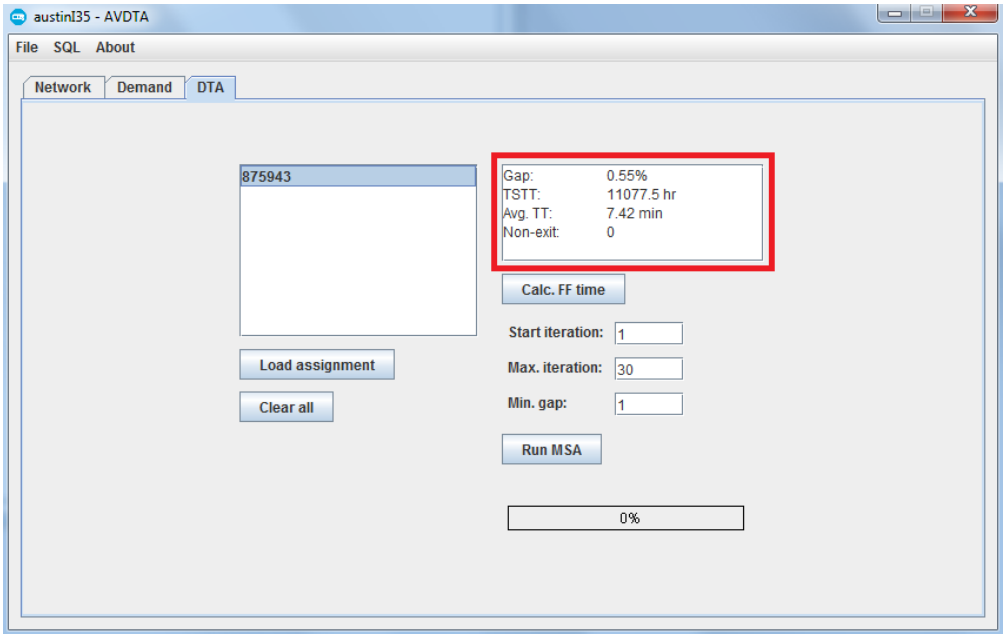
The purpose of AVDTA is to solve DTA with AVs, and the "DTA" tab contains options for loading and creating assignments. (An *assignment* determine the route each vehicle takes.) Currently, AVDTA uses the method of successive averages to find an UE assignment.

When a project is opened, no assignment is loaded by default. The `assignments` folder contains all previously stored assignments. Every time MSA is run, AVDTA will store the last iteration as a new assignment. Assignments are created with a random number as the name. However, assignments can be renamed as follows: Each assignment is a subfolder in the `assignments` folder. Renaming the subfolder will rename the assignment.

The AVDTA GUI provides a list of all existing assignments:

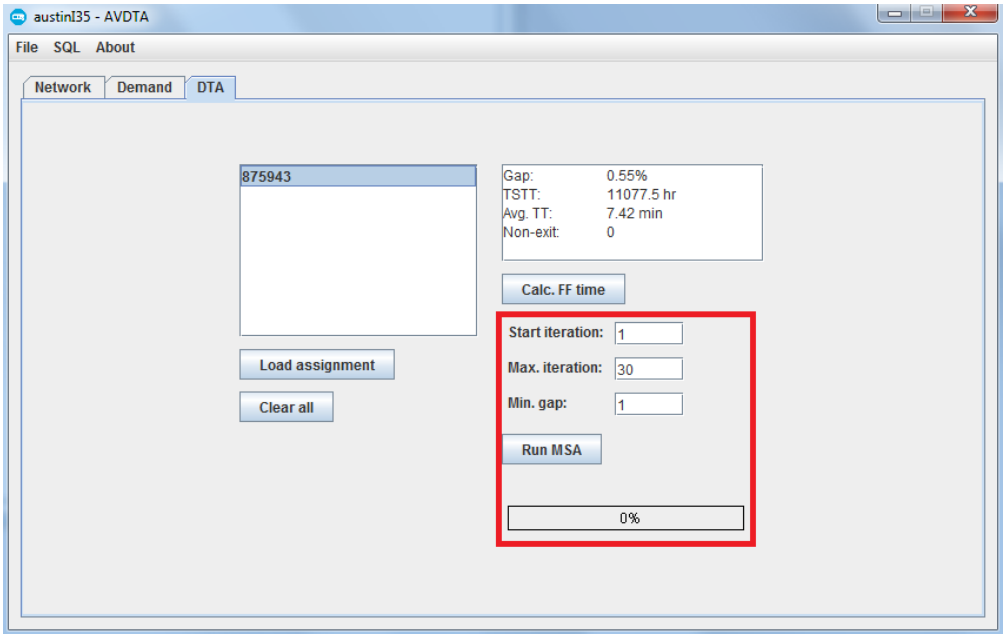


Selecting an assignment will fill the text area on the right with some summary statistics.



Clicking "Load assignment" will assign vehicles to the selected assignment.

Options to run MSA are on the right hand side. The starting iteration controls the initial step size ($\lambda = \frac{1}{i}$). When working with a previous assignment, it may be helpful to start at a lower step size. The stopping criteria are the maximum number of iterations and minimum gap. If either is reached, MSA will stop.



The status bar will estimate the completion and remaining time, based on computation time for previous iterations and the stopping criteria of the maximum number of iterations.

Appendix A

Abbreviations

Abbreviation	Definition
AST	assignment interval
AV	autonomous vehicle
BEV	battery-electric vehicle
CV	connected vehicle
CTM	cell transmission model [3, 4]
DLR	dynamic lane reversal [5, 8]
DNL	dynamic network loading [1]
DTA	dynamic traffic assignment [1]
DUE	dynamic user equilibrium [1, 13]
FCFS	first-come-first-served [6]
GUI	graphical user interface
HV	conventional (human-driven) vehicle
ICV	internal combustion vehicle [7]
LTM	link transmission model [14, 15]
MSA	method of successive averages [12]
UE	user equilibrium [13]

References

[1] Chiu, Y.-C., Bottom, J., Mahut, M., Paz, A., Balakrishna, R., Waller, T., and Hicks, J. Dynamic traffic assignment: A primer. *Transportation Research E-Circular*, E-C153 (2011).

[2] Conde Bento, L., Parafita, R., Santos, S., and Nunes, U. Intelligent traffic management at intersections: Legacy mode for vehicles not equipped with v2v and v2i communications. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on* (2013), IEEE, pp. 726–731.

[3] Daganzo, C. F. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological 28*, 4 (1994), 269–287.

[4] Daganzo, C. F. The cell transmission model, part ii: network traffic. *Transportation Research Part B: Methodological 29*, 2 (1995), 79–93.

[5] Duell, M., Levin, M. W., Boyles, S. D., and Waller, S. T. On system optimal dynamic lane reversal to improve traffic efficiency for autonomous vehicles. In *Transportation Research Board 95th Annual Meeting* (2016), no. 16-4922.

[6] Fajardo, D., Au, T.-C., Waller, S., Stone, P., and Yang, D. Automated intersection control: Performance of future innovation versus current traffic signal control. *Transportation Research Record: Journal of the Transportation Research Board*, 2259 (2011), 223–232.

[7] Levin, M., Duell, M., and Waller, S. Effect of road grade on networkwide vehicle energy consumption and ecorouting. *Transportation Research Record: Journal of the Transportation Research Board*, 2427 (2014), 26–33.

[8] Levin, M. W., and Boyles, S. D. A cell transmission model for dynamic lane reversal with autonomous vehicles. *Transportation Research Part C: Emerging Technologies 68* (2016), 126–143.

[9] Levin, M. W., and Boyles, S. D. A multiclass cell transmission model for shared human and autonomous vehicle roads. *Transportation Research Part C: Emerging Technologies 62* (2016), 103–116.

[10] Levin, M. W., Boyles, S. D., and Patel, R. Paradoxes of reservation-based intersection controls in traffic networks. *Transportation Research Part A: Policy and Practice 90* (2016), 14–25.

[11] Levin, M. W., Fritz, H., and Boyles, S. D. Optimizing reservation-based intersection controls. *In review at IEEE: Transactions on Intelligent Transportation Systems* (2015).

[12] Levin, M. W., Pool, M., Owens, T., Juri, N. R., and Waller, S. T. Improving the convergence of simulation-based dynamic traffic assignment methodologies. *Networks and Spatial Economics* (2014), 1–22.

[13] Wardrop, J. G. Road paper. some theoretical aspects of road traffic research. In *ICE Proceedings: Engineering Divisions* (1952), vol. 1, Thomas Telford, pp. 325–362.

[14] Yperman, I. The link transmission model for dynamic network loading. *status: published* (2007).

[15] Yperman, I., Logghe, S., and Immers, B. The link transmission model: An efficient implementation of the kinematic wave theory in traffic networks. In *Proceedings of the 10th EWGT Meeting* (2005).