San Francisco Dynamic Traffic Assignment Project
"DTA Anyway"

Final Methodology Report

Prepared by:

Parsons Brinckerhoff
&
San Francisco County Transportation Authority

November 28, 2012

1. Introduction

In 2010, the San Francisco County Transportation Authority (The Authority) received a grant from the Federal Highway Administration (FHWA) to implement a dynamic traffic assignment (DTA) model for San Francisco. This project, also known as "DTA Anyway", builds upon a previous DTA model developed for the Northwest quadrant of the city, and seeks to develop a "production-level" model for use by the Authority in project applications in the remainder of the city.

The project also seeks to be useful to developers of DTA models in other regions in three ways. First, it involves building a flexible toolset to facilitate the development of future DTA models. Second, it documents the process and assumptions used to develop the model, hoping to serve as an example of best practice. Finally, several particularly challenging questions are studied:

- How does DTA perform in a dense and highly congested grid network?
- How can DTA be used to study the interaction of the street network with the transit system?
- What benefits might DTA provide in evaluating congestion pricing policies?

This report describes the final methodology used by the resulting San Francisco DTA (SF-DTA) model. Several accompanying resources are available to provide a more complete picture of the project, including:

- Project Website and Code Base (see Appendix A and http://dta.googlecode.com): This site was used by the project team as a collaboration and dissemination tool. The source code for the conversion tools is available for download, as are all relevant documents.
- DTA Anyway API Documentation (http://dta.googlecode.com/git-history/dev/doc/_build/html/index.html#): This site provides an overview of the classes and scripts included in DTA Anyway.
- Final Calibration and Validation Report: While all model parameters and settings are listed here, the calibration process and validation results are documented separately in order to maintain parsimony.
- Analysis of Applications Report: Presents the results of model applications.
- Model Integration Options Report: This report presents a range of options for how SF-DTA and the San Francisco Chained Activity Modeling Process (SF-CHAMP) can be better integrated in the future.
- Future Research Topics Report: The plan for future development and integration. Note that this document is distinct from the Model Integration Options report in that it was completed after the peer review so is able to incorporate that feedback. As a result, it is able to present recommendations for future development rather than simply options.
- Peer Review Panel Report: Comments from the peer review for this project held on July 25, 2012.
- Response to Peer Review Comments (Appendix B): Documents changes made based on the recommendations of the panelists.

The remainder of this report is structured as follows. Chapter 1 provides an overview of the model system and the context. Chapter 2 describes the development of all model inputs (which involved the bulk of the effort) and the tools used to create or convert those inputs. Chapter 3 describes the final model parameters, as well as a brief overview of the calibration and validation process. Chapter 4 presents the results of a series of sensitivity tests. Chapter 5 summarizes the lessons learned.

1.1 Study Area and Policy Issues

San Francisco County has a population of approximately 800,000 within a region populated by approximately 7.2 million people[1]. Surrounded on three sides by water and on the final side by a small mountain range, San Francisco is connected to the rest of the region via two bridges: the Golden Gate to the North and the Bay Bridge to the East, two freeways to the south: US-101 and I-280, a Transbay Tube that carries BART heavy transit rail, and a network of ferries.

Peak period traffic congestion has been a problem for auto and transit users alike, with average speeds downtown averaging ten mph and eight mph respectfully[2]. Peak-period crowding on transit services keep potential users waiting while buses and trains repeatedly pass them up because they are too full. Congestion from the freeway system backs up daily onto city streets in the SoMa district, bringing cars and buses alike to a standstill.

In order to meet local goals for green house gas reduction, economic vitality, and livability[3], San Francisco is considering effective demand reduction strategies such as cordon-based congestion pricing as well as investment in cost-effective transit efficiency strategies such as bus rapid transit (BRT). Many of the questions that planners have been asking in the context of these strategies have been difficult to answer confidently within the confines of the static traffic assignment used by the Authority's regional activity-based travel demand model SF-CHAMP[4]. Among the questions that have been asked:

- What streets experience speed improvements with congestion pricing and how does transit perform compared to autos?
- Where is downtown congestion coming from and going to? Where are people on this corridor going from and going to? How many of them have origins or destinations outside of the corridor and are thus easily divertible?
- Where does traffic divert to with the implementation of this project?

Dynamic traffic assignment (DTA) promises to be able to answer these questions with more confidence. Specifically, DTA:

1. produces feasible traffic flows rather than over capacity demand,
2. allows queues to affect upstream and downstream links,
3. is sensitive to operational strategies such as signal timing, and
4. represents specific vehicles classes such as transit (and is therefore able to produce separate performance metrics for them).

Item one brings up an interesting point about whether projects should consider forecast demand regardless of hard capacity constraints, or a realistic predicted volume. Forecast demand represents demand for a facility without explicit capacity controls. Consequently, forecasted demand can sometimes greatly exceed realistic capacity. A slightly more restrictive option is to forecast capacity-informed demand, where a harsh penalty is applied to all demand that uses an over-capacity facility. While it is useful to understand demand for a facility, even under conditions of onerous travel times, this information is not useful in determining what operational impacts would actually occur. By limiting capacity and providing a more accurate estimation of travel times, realistic predicted volumes allow planners and traffic engineers to better understand and plan for likely operational conditions on the road network. Neither of these representations of demand volume tells the

[1] Census 2010
[2] sfmobility.org
[3] www.sfcta.org/sftp
[4] www.sfcta.org/modeling

entire story, but together they can provide a deeper and more nuanced picture of future travel characteristics and traffic operations than either measure alone. In short, DTA may help to provide more comprehensive information in conjunction with existing static assignment tools.

1.2 A Growing Toolbox

San Francisco's SF-CHAMP activity-based travel demand model has been in continuous use for over a decade. It has a very detailed street network and zone system within San Francisco. It includes every street, alleyway, and transit stop within the city and zone sizes are often the size of a single block in the downtown area and two to four blocks in the outskirts of the city. SF-CHAMP v4.3 *Fury* uses static user equilibrium highway assignment and an iterative, capacity-informed transit assignment[5]. Because the static user equilibrium macroscopic traffic assignment does not contain enough detail for many planning studies, many planning teams also rely upon network microsimulation models such as VISSIM to model fine-grained network operations. In order to maximize the use of our budgets, VISSIM study areas are often limited to a very small portion of the network and tools such as Synchro are used instead. Direct volumes from SF-CHAMP often need to be manipulated in order to make sense to VISSIM. DTA has the potential to both bridge the gap between SF-CHAMP and VISSIM and provide a more meaningful set of inputs, but also to replace resource intensive microsimulation for instances where the questions can be answered with DTA alone. The goal of this project is to provide a solid starting point for any project that wants to use DTA, but not to give a project-level validation across the entire city.

1.3 Previous DTA Applications

In Fall 2009, the Authority created and successfully used a subarea DTA model to examine the implications of closing some ramps on US 101 for the purposes of construction. This Northwest Subarea DTA model had 260 total zones, 3,000 nodes, 7,000 links, 240 traffic signals and 83 transit lines. Subsequently, this model has been used to study the effects of the Geary BRT project. While the subarea model has proven successful, there are several pitfalls that arose during its use that are discussed below.

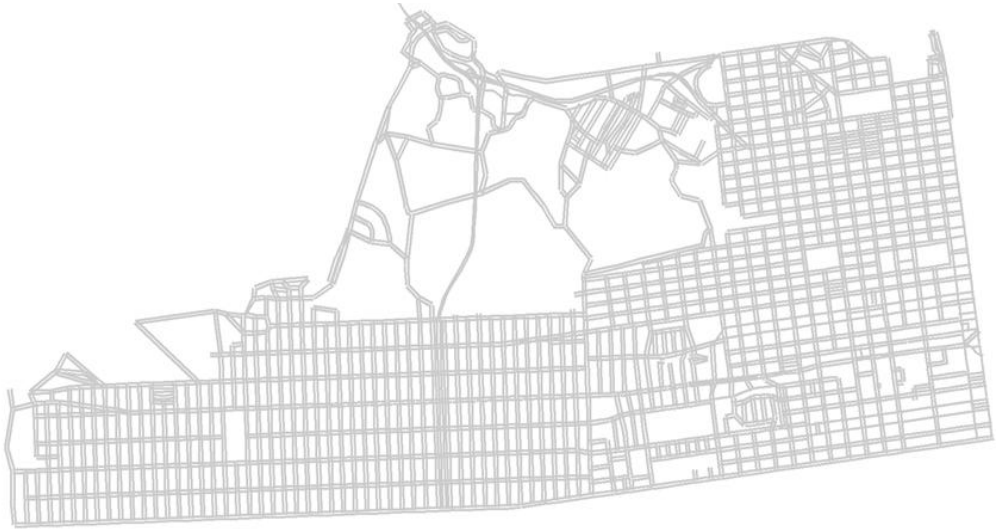

Figure 1. San Francisco Northwest Subarea DTA Network

[5] Zorn, L., E. Sall, and D. Wu. "Incorporating crowding into the San Francisco activity-based travel model." *Transportation* 39 (2012): 755-771.

- DTA represents an ideal world of network level of service knowledge. This can often mean that small shifts in travel time can create drastic shifts away from main-line roadways that may or may not be entirely true. Do drivers always know whether a slightly better alternate route exists on an obscure local roadway? We could create some more specific generalized cost functions in our citywide DTA calibration that attempt to mitigate the main-line diversion. However, the inertial effect of staying on your 'initial route' has yet to be captured.
- The subarea's eastern boundary cut across a regular street grid, forcing the subarea extraction process to assign demand to specific streets in the grid. This proved to be very restrictive and often resulted in strange results near the start of the subarea, but also farther west as well. We anticipate that the citywide DTA eliminates this pitfall by using more natural boundaries (The Pacific Ocean, San Francisco Bay, and San Bruno Mountain Range) with a limited number of external stations.
- Volumes on smaller streets are very reliant on centroid-connector placements, which made LOS analysis with the raw DTA results sometimes problematic. In the end, small 'surgical adjustments' were still required in order to achieve more realistic results at a small scale. This issue remains in our citywide DTA calibration. However, it is anticipated to be mitigated as a part of any project-level calibration process, which would involve the identification of parking facilities.

1.4 Model Development Approach

The DTA Anyway project has the primary objective of building a working DTA model with results that make sense for the PM Peak period in San Francisco. Rather than a "one-shot" model built to accomplish a specific task, it has the supporting objective of establishing a seamless process to move from SF-CHAMP to DTA for its use in a multitude of projects into the future. This process uses the SF-CHAMP network and then builds a DTA network from it, allowing for changes to one network to be reflected in both models. The San Francisco DTA also uses SF-CHAMP demand directly, resulting in a behaviorally consistent approach and avoiding the arbitrariness that can result from synthetic matrix estimation approaches. Several principles were adopted to guide the model development process:

- First, the process is automated as much as possible in order to minimize both the schedule requirements and the risk of error associated with human intervention. Therefore, code is written to prepare model inputs, convert data formats, and summarize results. Any issues that cannot be addressed in an automated fashion, such as a network coding errors, are addressed at the "source" wherever possible. This means that changes are made to the SF-CHAMP network that is converted to the DTA network, rather than made directly to the DTA network where they would be overwritten with the next import.
- Second, the model is based on real data as much as possible. If something can be readily measured, we measured it. If something could not be measured, we sought some basis or justification for our assumption. Only as a last resort, did we make a change "because it works", and in those cases the change is clearly documented. This data-driven approach is most evident in the determination of traffic flow parameters, where we conducted extensive fieldwork to measure vehicle lengths, response times, and other aspects of driver behavior. It also extends to the input development, where we imported current signal timing plans (over 1,000 of them), stop sign locations and transit routes throughout the city.
- Third, the model was developed in a highly collaborative manner. The Authority and Parsons Brinckerhoff staff worked together as part of an integrated team, sharing tasks to take advantage of

the skill sets and availability of different individuals. The project website[6] was an important tool to facilitate this collaboration. The site is integrated with the Git[7] version control system for the source code, includes an issue tracker to prevent details from slipping through the cracks, and was our repository for sharing the results of each model run and documentation on how to perform certain tasks.

- Fourth, any new code was developed in an open-source environment, but we avoided rewriting functionality that already exists elsewhere. The preference for open-source comes from a desire to provide a platform for external collaboration, and the lack of a desire to become a commercial software vendor. We also recognize that is not a prudent use of resources to replicate functionality that already exists and can be purchased for a reasonable price. The prime example of this is the DTA package itself, in this case Dynameq[8]. The Authority's experience with Dynameq has been that it produces reasonable results, has a mature user interface, and a responsive developer. Therefore the code written for this project focuses on the converting model inputs and summarizing results. It does not implement the assignment itself, nor does it provide a network editor or visualizer, which are both available in Dynameq and most other DTA software.

It is worth noting that there are several topics left for future development. These include:

- Rigorous validation for specific "project-level" applications;
- An enhanced approach to modeling parking, including reconfiguring centroid connectors to load at driveways;
- Assigning people to transit trips (this is being done in parallel with FAST-Trips);
- 24-hour DTA; and
- Feedback to SF-CHAMP.

These topics are discussed in more detail in the Future Research Topics Report.

1.5 Model Overview

The model covers all of San Francisco, as shown in Figure 2. This allowed for a natural breakpoint at the San Bruno Mountain State Park near the San Mateo County line where there are a limited number of external stations. In this way, SF-DTA is able to predict the north-south routing through the city, rather than relying on fixed external station volumes through a dense part of the roadway network.

The road network includes every street in the city, with 976 traffic analysis zones (TAZs) and 22 external stations. The network includes actual phasing and timing plans for nearly every traffic signal in the city, about 1,115 in total (certain mid-block or pedestrian-only signals are ignored). It also includes stop control at about 3,726 intersections based on a GIS layer of stop signs in the city.

The demand for the San Francisco DTA is taken from a subarea extraction of the trip tables produced by SF-CHAMP. SF-CHAMP covers the entire nine-county Bay Area, which is why the subarea extraction is necessary. The roadway networks are also imported directly from the network representation used in SF-CHAMP, allowing for a direct linkage between the two models. The demand is currently segmented into four user classes: autos, trucks, autos with toll, and trucks with toll. There are currently no tolls within the

[6] http://dta.googlecode.com
[7] http://git-scm.com/
[8] https://www.inrosoftware.com/en/products/dynameq

city, so the toll matrices are unused in the base year. They serve as placeholders for scenarios that have cordon-based congestion pricing. The definition of user classes is flexible such that different classes could be used for a specific application.

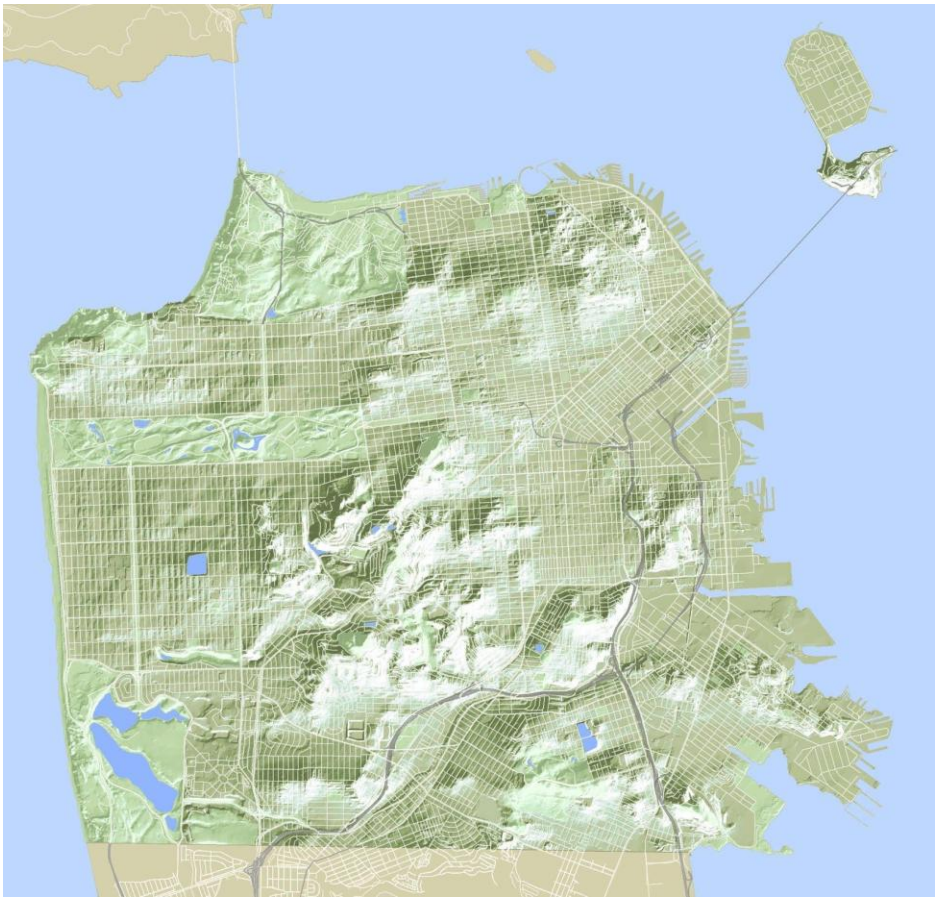

Figure 2. San Francisco County Study Area

The simulation is currently run for a PM peak period. Demand is loaded from 2:30-7:30 pm, with the simulation continuing until all traffic clears. The five-hour demand period includes a one-hour warm-up period, a three-hour P.M. peak period, and a one-hour cool-down period. During the warm-up period from 2:30 to 3:30 pm, approximately 76,300 cars and 13,000 trucks are loaded on to the network, and during the cool-down period from 6:30 to 7:30 pm, roughly 68,300 cars and 6,000 trucks are loaded onto the network. During the intermediate three-hour PM peak period in the base year, roughly 390,600 autos and 65,200 trucks are loaded onto the network, with 220,700 internal car trips and 170,000 car trips originating or ending outside the county study area. With warm-up time, the reliable period of results is approximately 3:30-6:30 pm, corresponding to the three-hour P.M. peak assignment in SF-CHAMP.

The Authority currently uses Dynameq as their DTA platform. It is complemented by tools developed for this project to facilitate the development of the inputs and the processing of the outputs. The model run process for a specific scenario is:

1. Import the network and demand data to prepare it for Dynameq.
    a. Read the Cube network and convert the links to Dynameq format

   b. Read the transit lines from Cube format and convert to Dynameq format
   c. Read the signal timing cards from an Excel file format used by the San Francisco Municipal Transportation Agency (SFMTA) and attach them to the network using street names
   d. Import the unsignalized intersections from SFMTA's GIS format
   e. Create the demand by extracting a subarea trip table from SF-CHAMP
2. Run the DTA model in Dynameq
   a. Create a new scenario
   b. Import the files created by step 1
   c. Set the necessary properties for the DTA run
   d. Execute
3. Export the results and process output for analysis
   a. Export the results from Dynameq
   b. Join traffic counts
   c. Produce automated reports

The details of these steps are described in Chapter 2.

2. Model Input Development & Code Base

This chapter describes the preparation of inputs for the DTA model using the "DTA Anyway" codebase - a set of open-source tools we developed for the purposes of this project. The objectives of this chapter are to document the network and code base development, provide a step-by-step template for someone else who is considering building a DTA model, and describe the functionality of the DTA Anyway library.

2.1 Overview of DTA Anyway Code

DTA Anyway is a Python library developed by the San Francisco DTA model development team to assist in automatically generating the DTA model network from the various input components. Key functionality includes the ability to: read and write Dynameq ASCII files, read static network files (in the form of a Cube network), write shapefiles of links and movements (essential for debugging), and other network manipulation tasks (i.e., splitting links, finding links and movements according to various attributes, setting movement priority hierarchies for an intersection, etc.).

The objective of DTA Anyway is for DTA network modifications to be made programmatically rather than via a graphical user interface (GUI) and to facilitate a quick, seamless and less error-prone process for converting a static highway network to a dynamic one. When the DTA model is fully integrated with the Travel Demand Model, then there will be no need for a static network (and if there is one, DTA Anyway could be used to generate it easily from the DTA model because the DTA network has more detail). Until that time, however, conversion from static networks to DTA networks will be a frequent occurrence, since a typical project involves multiple scenarios with different variations of a network, and each one needs to be converted to a DTA network. Thus, automating the conversion of these networks to the DTA version reduces error and streamlines the use of the DTA model.

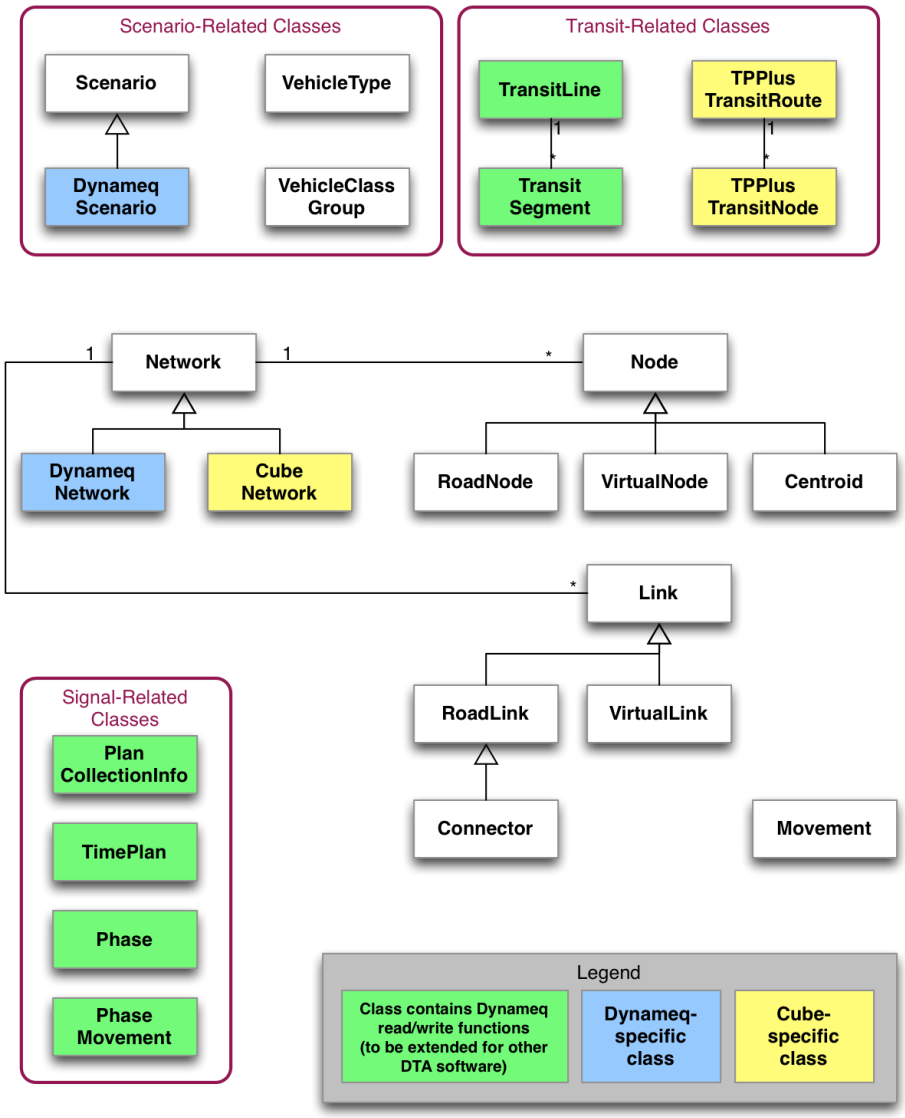

Figure 3. Class diagram of the DTA Anyway library

The DTA Anyway library has an object-oriented design, with classes representing the network itself, links, nodes, movements, traffic signal time plans, etc. A class diagram of the DTA Anyway library is shown in Figure 3, and the API is documented thoroughly on the project website[9]. The DTA Anyway library is meant to be location- and modeling-software agnostic, so that other parties interested in developing or programmatically building a DTA network (from a static network or from other sources) can use the library. Location-specific code and configuration is kept out of the DTA Anyway library, and included only in the scripts which use the DTA Anyway library.

The DTA Anyway team has developed a series of scripts to construct the San Francisco DTA model that uses the DTA Anyway library, with each script adding another level of detail to the DTA model. An overview of the script sequence is shown in Figure 4 .

[9] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/index.html

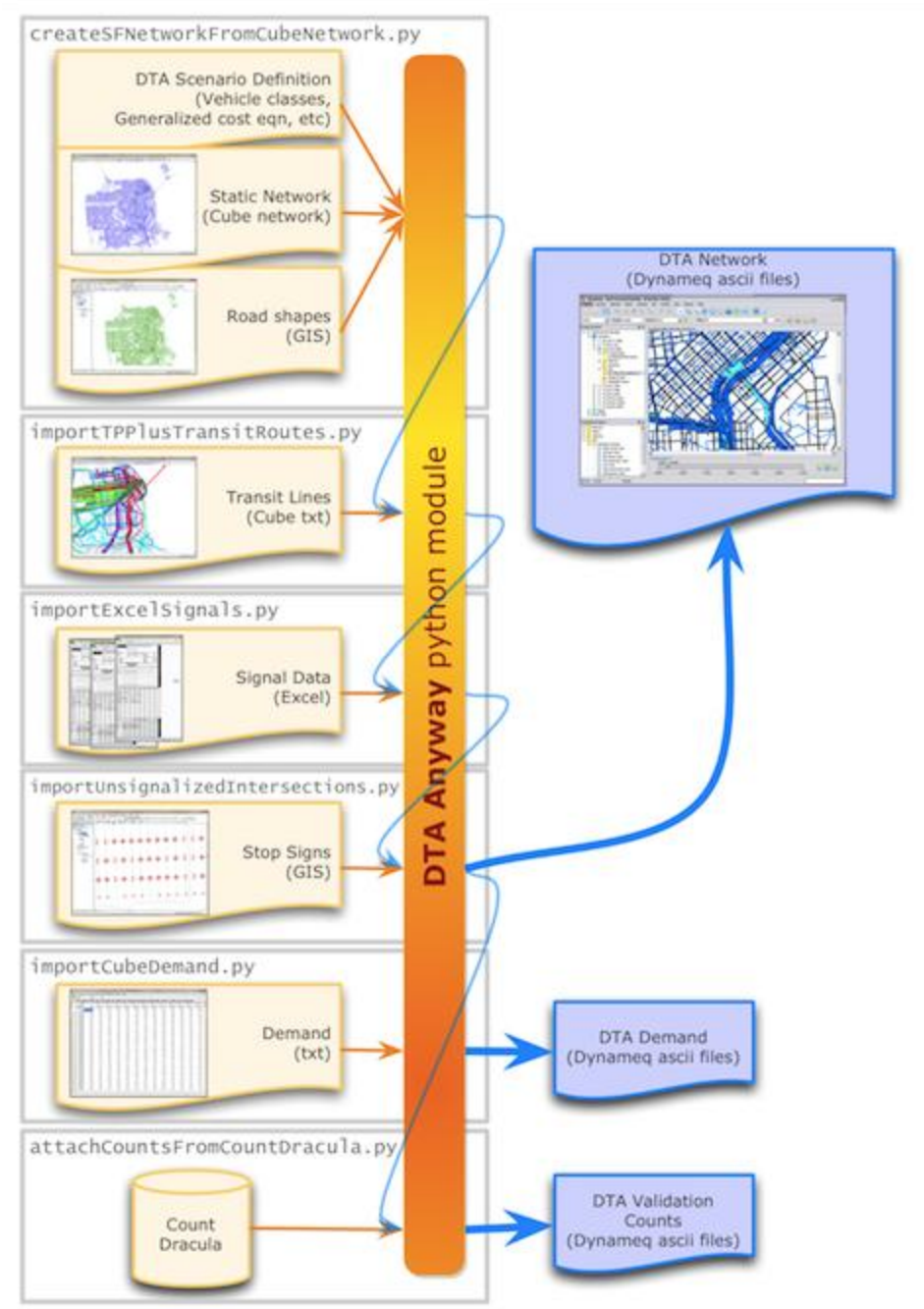

Figure 4. Overview of the script sequence used to create the San Francisco DTA network

The next two sections give an overview of installation and logging of the DTA Anyway library and are followed by descriptions of each of the San Francisco specific scripts used to create the San Francisco DTA model.

2.1 Software Requirements and Installation Instructions

The DTA Anyway Python module is not yet available as an installable module, and so installation involves downloading the source code from the online Git repository (currently hosted on the Google Code website)

into a local directory. This can be done via a Git clone command as instructed on the site[10]. As the software becomes more mature, it will be packaged into a releasable module (a pip module or an easy_install module). Initially, however, we anticipate that other agencies may need to make changes to the library itself, and more collaboration is anticipated before an installable version will be released.
In addition to downloading the source code, several additional Python modules are required or optional; these are listed in the documentation[11]. At this time, there are three required modules: NumPy, PyShp, and PyParsing. There are several other modules that are required for specific functions within the DTA Anyway codebase. All of these modules are available for installation using easy_install [12]or pip[13]. Once a local version of the DTA Anyway codebase saved, writing scripts to utilize the DTA Anyway API amounts to making sure the location of the codebase is listed in one's `PYTHONPATH` environment variable so that python knows from where to load the library on import.

2.2 Logging and Error Handling

Throughout the subsequent sections, there are various references to logging warnings, errors and having fatal errors. DTA Anyway includes a logging method, `setupLogging`() which should be called at the beginning of any script. This logging method takes three arguments:

1. The first is the `infoLogFilename`, into which informational log messages get written. These are typically summary messages that quantify specific large tasks performed in the scripts. Records of a file being read or a file being written are included here. These logs also receive error messages as well, when something slightly problematic occurred. These files are meant to be brief enough that a quick scan can reveal how the script performed. Additionally, the San Francisco scripts typically include a date/timestamp so that the log files don't overwrite each other.
2. The second is the `debugLogFilename`, into which a more verbose log gets written. These are typically used to log detailed debugging information.
3. The final argument is a boolean, `logToConsole`, indicating whether or not logging messages should also go to the console. If set to True, INFO level messages and above are logged to the console (so debugging messages are not included.)

Additionally, many methods in the DTA Anyway library give exceptions in response to unexpected events. These methods are documented as such, and the calling code should handle them appropriately. San Francisco's DTA scripts follow the convention that most errors are caught and logged but considered acceptable, while other errors are sufficient cause for the script to fail outright, requiring human intervention and a fix.

2.3 Defining the DTA Scenario and Converting the Static Network

Script name	`createSFNetworkFromCubeNetwork.py`
Inputs	• input network file (a static Cube network) • turn penalty text file (referencing input network nodes) • a shapefile from which to infer the road link shapes (referencing the input network nodes)

[10] http://code.google.com/p/dta/source/checkout
[11] http://dta.googlecode.com/git-history/dev/doc/_build/html/index.html
[12] http://packages.python.org/distribute/easy_install.html
[13] http://pypi.python.org/pypi/pip

Configuration within Script	• vehicle class names and groups • vehicle types (including vehicle names, effective lengths, response times, maximum speeds, and speed ratios) • generalized cost equations • additional network clarification such as turn pockets, lane shifts, custom response times, and a few links where the transit lanes are not the right-most lane.
Outputs	the DTA network, in Dynameq ASCII file format

The first stage in the process is to convert the input network to the DTA network. This includes several steps:

1 **The DTA Scenario is defined**. The scenario definition includes the vehicle classes, vehicle types and vehicle class groups that will travel on the network, traffic flow parameters for those vehicle types, and the generalized cost functions that determine vehicle route choice.

2 Subsequently, **the static network is read** and mappings are defined to translate the static network facilities to the DTA network facilities, including the definition of additional traffic flow attributes based on the link attributes. For example, the vehicle response time factors as well as the free flow speeds for roadway links are determined by the facility type of the link as well as the area type of the neighborhood, which is a measure of land use density and non-motorized activity. Additionally, based on data collected by the Authority, the slope of the roadway also factors into the response time factor of the link. See Section 3.2 for a more detailed description of the traffic flow parameters data collection.

3 **Transit-only lanes** are coded into the DTA network based on one of the input network link variable, `BUSLANE_PM`. A few of these are specifically configured (in the script itself) to be in the second rightmost lane when the rightmost lane is a general purpose right-turn lane, but most of these are in the rightmost lane or in the center lane, depending on the value of `BUSLANE_PM`.

4 Next, since the input static network does not include an explicit definition of them, **all intersection movements are added**. Because this step can be affected by the definition of a transit-only lane, the transit-only lanes are defined in a previous step; for example, a movement from one transit-only link to another only needs to have a transit-only movement. This method includes an argument for whether U-turn movements are to be added; for simplicity, U-turns in San Francisco are assumed to be prohibited unless they are specifically added as allowed.

5 **Turn prohibitions are applied**. Once the movements are added, the static network turn prohibition file is read and those movements are modified to be prohibited.

6 **Custom response time factors are applied.** Some sections of roadway operate in real life with more- or less-aggressive traffic flow characteristics. This most often occurs at places of large merge-, diverge-, and weaving sections where drivers must maneuver quickly and aggressively to reach their desired lanes. To accommodate these variations, the script applies a custom response time factor to one specific link which required response time adjustment. In the future, this could be operationalized by specifying a separate facility time for these sections.

7 **HOV Stubs are removed**. Placeholder links in the static network are used to connect between managed or controlled-access capacity and general flow lanes. When these HOV stub links are not used in any given scenario (usually because the HOV lane that they are connecting to doesn't yet exist in that particular modeled year), the script strips from the DTA network in order to de-clutter it and free up links and node numbers.

8 In Dynameq and other DTA software packages, centroids are nodes that represent vehicle origins and destinations, and connectors are the first and last real roadways of each vehicle trip. Since these connectors don't typically start or end at the centroid location, virtual links typically connect centroids and connectors; they are called virtual because they do not represent actual roadways. In order to be consistent with this network representation, **virtual links are inserted between centroids and road nodes, and the centroid connectors are moved to mid-block locations**. In the San Francisco static network, centroid connectors typically interface with the roadway network at intersections. Since the DTA model has a nuanced representation of turn movements, conflicts and signals or stop signs at these intersections, the unrealistic intrusion of centroid connectors at these nodes would interfere with the function of the intersection. Thus, road links adjacent to these intersections are split to create midblock nodes, and the connectors are moved to join the roadway network at the new nodes.

9 **Turns are allowed from transit-only lanes**. In San Francisco, private automobiles are legally allowed to turn right from a transit-only lane when that transit lane is the right-most lane. Similarly, left turns for private autos are allowed from a center transit lane (for a two-way street) or a left-side transit-only lane (for a one-way street), where left turns are allowed. In this step, these movements are accommodated in the DTA network by splitting the link into two portions: the portion closest to the downstream intersection where turn movements are made and the upstream portion farther from the intersection. The upstream portion of the link remains a true transit only lane that does not allow any auto or truck infringement. Meanwhile, the downstream segment is coded as a mixed traffic right turn lane. This enhances model realism by allowing right turning vehicles to use transit lanes to prepare for turning movements, but has the weakness of allowing autos to enter the bus lane even if they are not making a turn. These through-moving vehicles are still required to merge back into a mixed flow lane prior to the next segment of true transit-only lane.

10 A few known **turn pockets are added** by reading in an override file**.**

11 **Overlapping links are fixed**. Centroid connectors that overlap a road link too closely cause Dynameq to error, so virtual nodes for overlapping links are moved slightly so that they no longer conflict.

12 **Short links are adjusted**. Due to a very intense urban environment, some links in our network are very short. However, Dynameq requires all road links to be longer than the longest vehicle type. Thus, each road link is examined and for those that are too short, their lengths are asserted to be this minimum length, which is 0.029 miles, or 153 feet. -- the length of an articulated Muni bus.

13 **Unused nodes are deleted**. The final step in this process is to delete unused nodes. The San Francisco static network includes a number of nodes that are not used in DTA, such as those involved in bicycle- or pedestrian-only links, transit access links, etc. These nodes are unconnected to the road network and are therefore easily removable.

Further documentation and source code for this step are available on the project site[14] ().

2.4 Importing Transit Routes

Script name	`importTPPlusTransitRoutes.py`
Inputs	• the DTA network, in Dynameq ASCII file format

[14] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_createSFNetworkFromCubeNetwork.html

	• the transit line files, in TP+ TRNBUILD format
Configuration within Script	• a mapping from the transit line mode number to the DTA transit line type (bus or tram) • a mapping from the transit line mode number to the DTA transit vehicle type • index into the TP+ TRNBUILD headway array to use
Outputs	the transit line file, in Dynameq ASCII file format

After the static network is converted to a DTA network, the accompanying static network's transit configuration is imported. First, the Cube TP+ transit network file is read and parsed into a set of `TPPlusTransitRoute` instances. These instances include additional information from the `TPPlusTransitRoute`, including a mode number and an array of headways. Then the `TPPlusTransitRoute` instances are converted into DTA `TransitLine` instances. This process is done as follows:

1 **Create a DTA `TransitLine` for each transit route specified.** For each `TPPlusTransitRoute` instance, a new `TransitLine` instance is created. A mapping from the Cube TP+ transit line mode number to the DTA transit type (bus or tram), and a similar mapping to the vehicle type (regular bus, articulated bus, cable car, 2-car light rail, etc.) is used to define the attributes of the `TransitLine` instance. This mapping is provided as configuration within the script itself. The script additionally specifies which headway to use by specifying an index into the array of headways in the `TPPlusTransitRoute.` In order to prevent the situation where all buses depart at the same time (the start time of the simulation), headways are translated into schedules by choosing a random start time within the headway for each line.

2 **Traverse the transit route and create `TransitSegments` for the `TransitLine`.** For each node pair along the `TPPlusTransitRoute`, the corresponding link is searched for within the DTA network and added to the DTA `TransitLine` as a link in the `TransitSegment`, which is essentially a list of links. However, many of these links will have been split in the preceding step (Converting the Static Network). Therefore, when a node pair is not found in the network, a shortest path is sought between the two and that series of links is used. This shortest path is then added to the `TransitSegment`. If the shortest path includes more than a configurable number of links (4 in our case), then it is assumed that the issue is more significant than a couple of link splits, such as the situation where a light rail vehicle transitions to a grade-separated section of off-street links. In this situation, the shortest path is not added, and the current `TransitSegment` is deemed complete. The iteration over the node pairs in the `TPPlusTransitRoute` continues.

3 **When a light rail line goes off the roadway network (i.e. underground or on grade-separated tracks), skip that section.** Most nodes that are specified in the TP+ TRNBUILD file are roadway nodes and therefore in the DTA network. However, light rail lines have some sections that are on grade-separated right-of-way or underground. In these situations, the nodes will not be found, and these sections will be skipped because they do not affect traffic. To skip these sections, the current `TransitSegment` will be completed and a new `TransitSegment` will be initiated when the transit line nodes are found within the roadway network again.

At the end of the process, the entire transit network (comprised of `TransitLine` instances, themselves comprised of `TransitSegment` instances) are visualized in Dynameq, to ensure that only the expected transit lines (e.g. light rail lines with off-street portions) are broken into disconnected segments.

Further documentation and source code for this step are available on the project site[15].

2.4.1. Importing Transit Routes from GTFS

Since San Francisco public transit service is published in GTFS format, GTFS-to-DTA transit conversion was also scripted during the research and development of a person-based transit assignment model, FAST-TrIPs[16]. This alternative transit specification has the benefit of additional accuracy by virtue of it being schedule-based rather than frequency-based. However, the GTFS specification does not include node numbers consistent with the static network, and so transit stops must be mapped to roadway links via their published coordinates. The process is therefore more computationally-intensive than the TP+ conversion, and it requires more manual route validation and tweaking to account for discrepancies between GTFS coordinates and those in the roadway network. This script utilizes the googletransitfeed open source library for GTFS parsing[17]. Further documentation and source code for GTFS-to-DTA is available on the website[18].

2.5 Importing Signals

Script Name	`importExcelSignals.py`
Inputs	• the DTA network, in Dynameq ASCII file format • signal card data, in Excel workbook format • movement overrides (a comma-separated text file) • U-turn prohibitions (a comma-separated text file)
Configuration within Script	• this script is highly customized to the San Francisco application and the SFMTA's specific signal data format
Outputs	• the DTA network, in Dynameq ASCII file format, including signal phasing for signalized nodes

The signal plan importing process reads in and processes all the signal plans (in the base 2012 scenario, there are 1,102 in San Francisco) in SFMTA's signal spreadsheet format, and saves signal phasing data on the corresponding network node. The code accomplishes the translation with the following steps:

1. reads in and parses the signal spreadsheet (AKA the *signal card)*,
2. finds the correct node in the DTA network,
3. assigns the offsets, signal phases and durations to the appropriate movements, and
4. validates the signal plans.

The signal plan validation step confirms:

- all possible movements through that intersection have some green time (or at least permitted right turn on red time),and
- conflicting movements do not have simultaneous green time.

Signal Cards

[15] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_importTPPlusTransitRoutes.html
[16] Khani, A. E. Sall, L. Zorn and M. Hickman. "Integration of the FAST-TrIPs Person-Based Dynamic Transit Assignment Model, the SF-CHAMP Regional, Activity-Based Travel Demand Model, and San Francisco's Citywide Dynamic Traffic Assignment Model" To be presented at the 92nd TRB Annual Meeting in January 2013.
[17] http://code.google.com/p/googletransitdatafeed/
[18] http://code.google.com/p/dta/source/browse/scripts/importGTFS.py?name=release-1.0

The SFMTA maintains an Excel format signal timing card for every signalized intersection in San Francisco. Each signal card provides complete cycle, phase, and coordination offset information for a single intersection. If timing plans change throughout the day the signal card shows all timing plans and the period during which they are in effect. The vast majority of San Francisco's traffic signals operate on fixed timing plans, but actuated signalization is employed at some intersections. Signal cards for actuated signals explain the rules under which the actuation operates, but also provide a default fixed timing plan. The signal cards are quasi-standardized. While many of the signal cards, especially more recent timing plans, follow a standardized structure, there are numerous examples of formatting or content arrangement differences among older timing plans. This inconsistency presented challenges for parsing the signal cards in a systematic manner.
Signal cards are version controlled with ordinal change numbers. Whenever an SFMTA engineer updates the signalization for an intersection, the engineer will issue a revised signal card with a new change number, the name of the traffic engineer, and the date of the change. The signal card for an intersection with the highest change number is the most recent (and currently effective) signal timing plan.

Identify Node in Network

The DTA Anyway code identifies which node the signal plan should be assigned to by matching the movements in the signal plan (which are named according to street names) to the names of the incoming streets in the DTA Network with the same number of street names as are in the signal card. The street name matching is done approximately, but the code also standardizes street naming by deleting leading zeros from numbered streets (i.e. 07th becomes 7th) and removing roadway type from the name (Ave, St, etc.). This ensures that the names can be matched even where the signal card and the network may not have the same street name format. In the case of streets with the same name post-processing (i.e. 3rd Ave and 3rd St both become 3rd) the differentiation is done based on the cross-streets. This only works if an Avenue and Street do not have the same cross-streets, but that has not been a problem with this San Francisco network.

Signal Phase Identification and Processing

Once the node corresponding to the signal plan is identified, the signal phases are read in and processed. Each signal card contains the phasing for every possible day of week / time of day combination, so the signal import script specifies which time of day should be read in. The process identifies the green, yellow, and red time for each signal phase and all of the movements associated with that phase.

Phases are listed in the signal cards by their streetnames (i.e Folsom) and optionally followed by a direction (i.e. SBLT, which means Southbound Left Turn) identifying a specific movement. Phases without a specified direction are assumed to include all movements associated with this incoming link. The signal import script matches the streetnames of the phase with the movements with an incoming streetname of the selected node. If there is a movement listed with any phase that does not have a corresponding link in the DTA network, the program will give an error that the movement listed is not an available movement in the DTA network, and that signal will not be added to the network. Outgoing links of the movement are identified by either set angles specified in the `getTurnType` function in `Movement.py`, or by an override file which was necessary due to several intersections with odd geometries.

The override file is a comma-separated file which has fields for Incoming Direction (NB, SB, EB,WB), Incoming Street Name, Cross Street Name, Outgoing Direction, Outgoing Street Name, turn type, vehicle permissions (if any), and number of lanes (if not default). This file is read in and will override the automatic

turn type, permissions, and number of lanes if they differ from the default. If the angle of two outgoing links indicates the same type of movement, they are both assigned to that movement type. There are two possible labels for each turning movement (i.e. LT and LT2 for left turns and RT and RT2 for right turns). Both sets of movements are treated the same, and no preference is given to the first or secondary movement with any label. Set angles are currently set to between -45 and 45 degrees for through movements, between 45 and 135 degrees for right turns, between 135 and 180 degrees for secondary right turns (RT2), between -135 and -45 degrees for left turns, or between -135 and -180 degrees for secondary left turns (LT2).

Once the signal phases in the signal card are associated with movements in the DTA network, the signal import script identifies the duration of each phase. Phase lengths may vary based on the time of day, so the signal import script selects the phasing plan that encompasses the time specified. The code does not use the scenario start time because it must allow for the fact that although our scenario begins at 14:30 to allow for warm-up time, we are interested in using the signal plans from the PM peak period. By specifying a signal time of 15:30 or 16:30, the code will be sure to select phase times that are associated with the PM peak signals. The current version of the DTA Anyway API can only assign one time plan per signal, but this could be adapted in the future to allow multiple timing plans to be used based on time of day.

The final stage in the signal importing code is the signal plan validation, which flags conflicting movements with green time at the same time or signals that can't find all the movements in the signal card in that node in the DTA network. Conflicting movements are detected based on the turn type and whether the movements are both protected. For two perpendicular streets, conflicting movements are: both through, one through and the other left turn, and both left turns. For two parallel streets, conflicting movements are: one through movement and the other left turn, and both left turns. If a two conflicting protected movements are identified, the code gives an error and does not add the signal plan to that node. Additionally, if there are any node movements which do not have any permitted or protected movement time, the code will give an error and not assign the signal plan to that node. These are validation steps that are also done in Dynameq, so the code simply ensures that none of the signal plans added to the network will generate an error in Dynameq that will prevent the simulation from running.

Several features in the DTA Anyway signal import script are specific to Dynameq's signal timing interpretation. Specifically, if there are two movements in a signal phase in Dynameq and only one of them goes to yellow and then becomes red in the next phase, the yellow time can be added to the first signal phase. Dynameq recognizes that any movement that also has green time in the following phase will not remain green during the yellow time in the first phase. This is an important assumption which allows us to create signal phasing plans that include that yellow time only for the movements where they exist, keeping accurate signal coordination along facilities.

The primary shortcoming of this method is that it generates a fixed signal plan for the entire simulation period. With longer simulation times, different signal plans may be used throughout the simulation period for some locations. At the time of this writing, however, only one signal plan can be generated for the whole simulation period. This is a modification that may be made as part of future extensions and updates to the code.

Further documentation and source code for this step are available on the project site[19].

2.6 Importing Stop Signs

Script Name	`importUnsignalizedIntersections.py`
Inputs	• the DTA network, in Dynameq ASCII file format • stop sign shapefile (each row represents a stop sign)
Configuration within Script	• mapping from stop sign street names to DTA network street names (most are the same but some are different; "Ninth Street" vs "9th Street", etc.) • a few corrections for anomalies found in the shapefile • critical gap parameters for custom priorities
Outputs	• the DTA network, in Dynameq ASCII file format, including stop sign information and custom priorities related to stop signs

Stop sign intersection control is added after the signalized intersection control is added. If an intersection in the DTA network already has a signal associated with it, the stop sign import script will ignore it. This is because it is more likely for a stop sign to be upgraded to a signal rather than a signal to be downgraded to a stop sign.

SFMTA has provided a stop sign shapefile which is read in this step. Each feature in the shapefile is an individual stop sign and includes the street the stop sign is facing, the cross street for the stop sign, and the direction that the stop sign faces. The unsignalized intersection import script uses the following algorithm to translate the shapefile into unsignalized control for the DTA network:

1. Assign each stop sign to an intersection in the DTA network using both the coordinates as well as the street names in the stop sign shapefile.
2. Determine the type of unsignalized control by comparing the number of stop signs at each intersection with the number of incoming road links.

In Dynameq, standard two-way stops give the higher facility type the right of way. However, there are several cases in the San Francisco network that there are two way stop controls that have equal facility types or where the higher facility type is required to stop. In these cases, Dynameq requires the definition of a "custom priority", which are written out in their own Dynameq ASCII file. The custom priority gives all the incoming movements without stop signs priority over the incoming movements that have stop signs even if they have the same facility type.

In the base 2012 network, around 1,800 intersections are all-way stops, 900 intersections are two-way stops, and 1,000 intersections have custom priorities defined. At the end of this script, one last validation step is performed: for each road node without any signals nor stop control, an entry is made into the log noting the street names for intersections with no traffic control. This log should be reviewed prior to using the Dynameq network

Further documentation and source code for this step are available on the project site[20].

[19] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_importExcelSignals.html
[20] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_importUnsignalizedIntersections.html

2.7 Importing Demand

Script Name	`importCubeDemand.py`

Inputs	• a set of demand files (comma-delimited ASCII files), and for each one:
	  ○ a start time and end time to be used from that file
	  ○ the time step to be used for that portion of time
	  ○ the proportion of the input demand to be used for the DTA demand
	• the DTA network, in Dynameq ASCII file format
	• the vehicle class name (this should be one of the set specified in the DTA Scenario)
	• a start and end time for the demand
	• an optional peaking profile (a comma-delimited ASCII file)

Configuration within Script	• None
Outputs	• an O-D matrix, in Dynameq ASCII format

The final step that is completed as part of a standard scenario import using DTA Anyway is the import and validation of demand to the DTA model. This task is accomplished with the following steps:

1. **Run subarea extraction for San Francisco Static model -** Cube TP+ scripts extract subarea demand for the study area boundary for each applicable time period and write it out to a comma-separated files that contain six columns: an origin TAZ, a destination TAZ, and then one column indicating the demand for each DTA vehicle class (`Car_NoToll`, `Car_Toll`, `Truck_NoToll` and `Truck_Toll`). This occurs in Cube before `importCubeDemand.py` is run.
2. For each vehicle class, `importCubeDemand.py` is run. This script:
   a. Reads in the comma-separated demand files for all time periods for this vehicle class
   b. Applies the optional peaking profile to the demand (if specified). If a row in the peaking profile input file matches the start and end time of one of the input demand files, then the demand will be distributed according to the factors. The sum of the factors must add to one for the specified time period. When this is included, then the timeStep specified with the input demand file will be ignored, and the timeStep for this demand period will instead be the time period for the demand period divided by the number of time factors.
   c. Writes the relevant O-D demand matrices in Dynameq ASCII format
   d.

Currently, three time periods are used from the SF-CHAMP model: midday (9:00 AM to 3:30 PM), PM peak (3:30 PM to 6:30 PM), and evening (6:30 PM to 3:00 AM). Only the last hour of the midday demand is used, and it serves as a warm-up period so that the network is realistically congested when the PM peak trips begin, resulting in more realistic travel times for those trips. Similarly, only the first hour of the evening period is used for network cool down, so that the trips starting at the very end of the PM peak are faced with more realistic levels of network congestion.

The peaking profile used in the San Francisco demand import process is derived from PeMS traffic count data at network entry locations; the contents of this file follows. The peaking profile is therefore only applied to the PM peak period data from the Cube demand input file. This peaking profile is used for all San Francisco vehicle classes.

Start Time	End Time	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6

15:30	18:30	0.15	0.16	0.16	0.18	0.18	0.17

Further documentation and source code for this step are available on the project site[21] .

2.8 Importing Counts

Script Name	`attachCountsFromCountDracula.py`
Inputs	• the DTA network, in Dynameq ASCII file format
Configuration within Script	Filter definitions for the types of counts to query: • all counts • all midweek counts (Tuesdays, Wednesdays, Thursdays) • recent (2009-2011) counts • recent midweek counts For each of these filters, we query for 5 different count types: • 5-minute movement counts (from 4p-6p) • 15-minute movement counts (from 4p-6:30p) • 15-minute mainline counts (from 4p-6:30p) • 60-minute mainline counts (from 4p-6p) • 3-hour mainline counts (from 3:30-6:30p)
Outputs	• For each filter, for each count type, a count data file in Dynameq's User-defined Attribute Values File Format that corresponds either a link or a movement with a sequence of average counts for that movement (according to the count type) • An Excel workbook containing the given counts in their raw form and aggregrate (via averaging) across count days

The final set of data processed using DTA Anway are the traffic counts used for deriving the demand peaking profile mentioned previously as well as for the DTA model validation. To facilitate this process, the Authority has also embarked on the creation of a counts database system, called Count Dracula[22]. Built using Geodjango, Count Dracula stores movement counts and mainline counts along with their locations, collection dates and times, source files, and other attributes, and the Django framework enables easy querying and visualization of the counts using a web interface.

The code to attach these counts to the DTA network utilizes both the DTA Anyway and Count Dracula libraries, matching up Count Dracula's intersections and mainline links with those of the DTA network. Count Dracula's understanding of the network is more limited than DTA Anyway, with locations based on street names and cardinal directions. Thus, the process for **attaching a mainline count location to the DTA network** is as follows:

1 Given the street name on which the count was taken (the `on_street_label`) and the direction of that link (NB, SB, EB or WB), in addition to the names of the cross streets at both ends of the link (the `from_street_label` and the `to_street_label`), `Network.findLinksForRoadLabels()` iterates through its road nodes to find candidate endpoints for the link based on the names of the streets that have an endpoint at the node.

[21] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_importCubeDemand.html
[22] https://github.com/sfcta/CountDracula

2 If a set of potential start nodes and a set of potential end nodes are found, try to walk from each potential start node to a potential end node as long as the link direction matches the direction from Count Dracula and the DTA link name matches the Count Dracula link name. This is done because the original link may have been split, so this method will return a sequence of links from the start node to the end node.
3 The first such sequence that is found is returned. If no list is found, then no match is made.
4 `attachCountsFromCountDracula.py` attaches the count to the first link in the returned list of links.

The primary shortcoming of this process is that Count Dracula has a more limited understanding of link directionality than DTA Anyway. That is, Count Dracula does not read a shapefile nor does it have a more nuanced grasp of geometry of a link, so links with significant curvature will have a single direction in Count Dracula, but they might translate to (after splitting) links with different directions in DTA Anyway, and therefore fail to map. Around 10-20% of links from Count Dracula fail to map to the DTA Anyway network.

Similarly, the process for **attaching a movement count location to the DTA Anyway network** is as follows:

1 Given the street names/directions involved in the movement (the `incoming_street_label`, `incoming_direction`, `ougoing_street_label`, `outgoing_direction`, and `intersection_street_label`) as well as the intersection road node number itself (which can be used here since Count Dracula is based on the same static network as the DTA network, so the node numbers are shared), `Network.findMovementForRoadLabels()` looks up the given road node.
2 **Street name-based matching:** Next, the method verifies the `incoming_street_label` matches an incoming link going in the `incoming_direction`, and that the `outgoing_street_label` matches an outgoing link going in the `outgoing_direction`. However, the `incoming_direction` does not have to be the primary direction; for example, if `incoming_direction` is southbound but the DTA network has the candidate link going southeast (primarily east), then this the incoming link is considered valid. This is to account for DTA Anyway's more nuanced (and therefore mismatched) notion of link geometry.
3 If Street name-based matching fails, then **direction-based matching** is attempted. This is similar but instead of requiring that the street names match, the directions are required to match and a unique match is required. That is, if an incoming southbound link is sought and only one incoming southbound link matches the road node, it is assumed to be a match (note this southbound DTA link must have its primary direction be southbound). This is done because Count Dracula has an imperfect understanding of street names at intersections where the street names change, often attributing links on both sides of the street to the same name.

The Count Dracula database includes a mix of 5-minute, 15-minute, 60-minute and 3-hour counts acquired from multiple sources such as project study counts, SFMTA, and the Caltrans Performance Measurement System (PeMS). Since the DTA model validation is for 2010, counts from 2009 to 2011 were used for validation, and these were further filtered to midweek days (Tuesday, Wednesday and Thursday). Figure 5

maps the locations of the count locations available for validation.

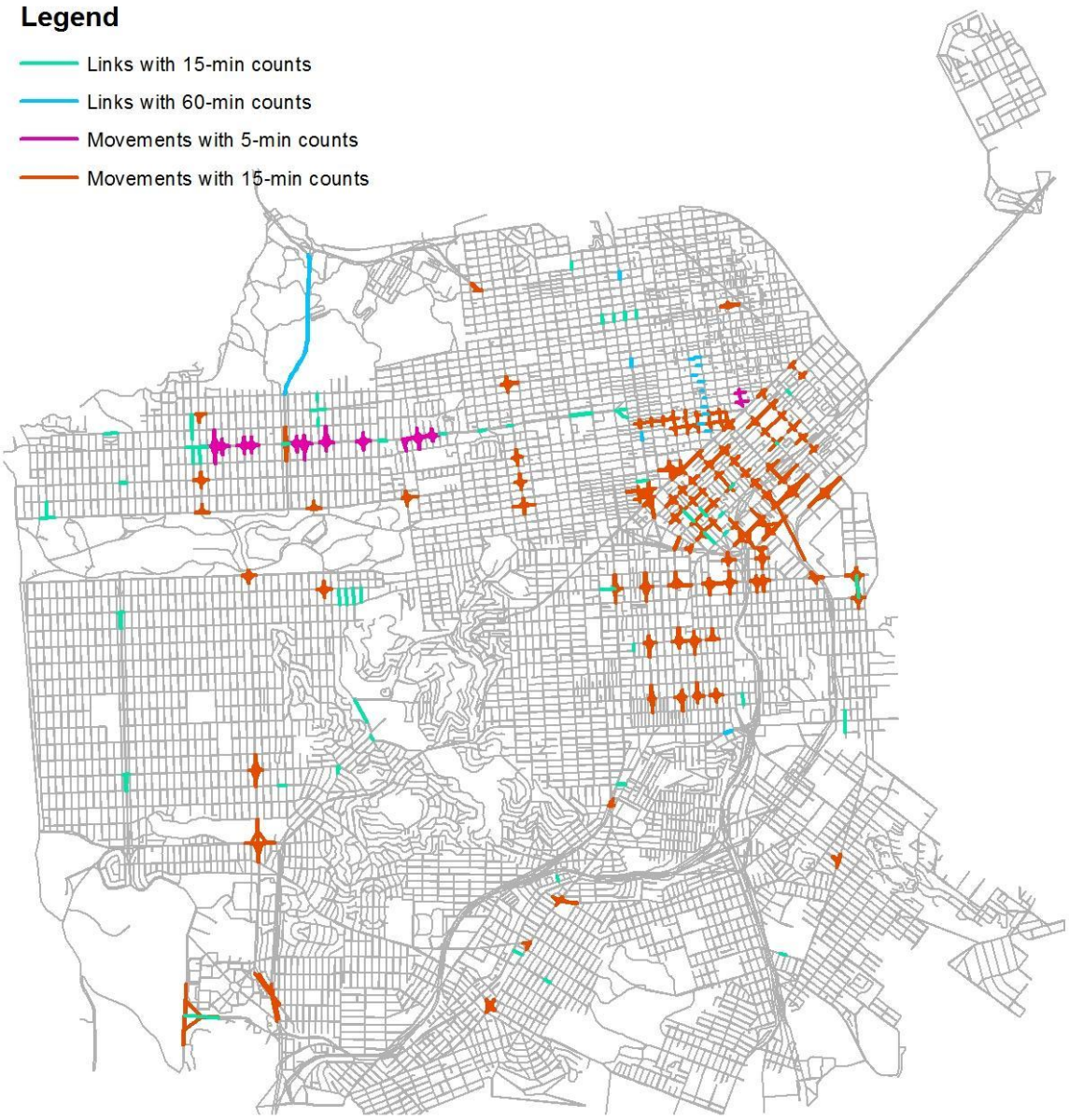

Figure 5. Recent, midweek counts used for DTA validation

Further documentation and source code for this step are available on the project site[23].

3. Final Model Parameters

As part of the calibration process, numerous parameters were adjusted in order to create a well-converged, gridlock-free, dynamic assignment where traffic volumes and travel times fit as closely as possible to observed values. The adjusted parameters include speed-flow parameters such as link speeds, vehicle lengths, vehicle

[23] http://dta.googlecode.com/git-history/release-1.0/doc/_build/html/script_attachCountsFromCountDracula.html

response times, and link response time factors. Additionally, refinements were made to the generalized cost expression and modeling parameters (such as critical gaps) to best approximate the actual traffic flow patterns. Additional information about the calibration and validation process and the current validation of the model is included in the Final Calibration and Validation Report

.

The final model parameters were chosen based on how well model runs using a set of parameters match observed travel and traffic conditions in San Francisco. When evaluating a model run the primary requirements are that a reasonable level of convergence has been reached and that the assignment is free of infinite delay gridlock. Once these conditions are met an assignment is validated against observed link volumes, observed movement volumes, and observed travel times for select corridors. After each test is completed, the results are exported and compared to observed counts and speeds from the Authority's database, taking care to limit the counts used to only those taken in the middle of the week and only from more recent years. Our model flows are based on 2010 demand, so it was important to use only more recent counts to maintain as much accuracy as possible in the comparison. Each assignment is also scanned for questionable traffic patterns, queue lengths, levels of congestions and path choices that appear to differ from real-world conditions.

3.1 Functional Classifications

Many of the network and speed-flow parameters in the DTA model are assigned according to the functional classification of links in the DTA network. Both SF-CHAMP and the San Francisco DTA model networks classify links by area type and facility type.

The area type classification provides an indication of an area's density. Dense, busy, downtown areas have a higher order (lower number) area type designation while suburban areas have lower order area type designations. There are six area type classes in the SF-CHAMP nine-county San Francisco Bay Area regional network, but only the densest four area types are present in San Francisco. Facility type designations describe the functional role of road facility. The lowest order facilities are alleys, local streets, and collectors. The next level includes minor, major, and super arterials. Freeways and expressways are the highest order facility types.

The speed and flow properties of individual links are assigned in the SF-CHAMP and DTA networks according to the combination of area type and facility type classifications of individual links. Actual link-level observed free-flow speeds and speed limits are not considered, except to ensure that the speed and flow properties are reasonable for each facility and area type combination. Figure 6 and Figure 7, below, show a map of the geographic distribution of road link area type and facility type classifications throughout the San Francisco DTA network.

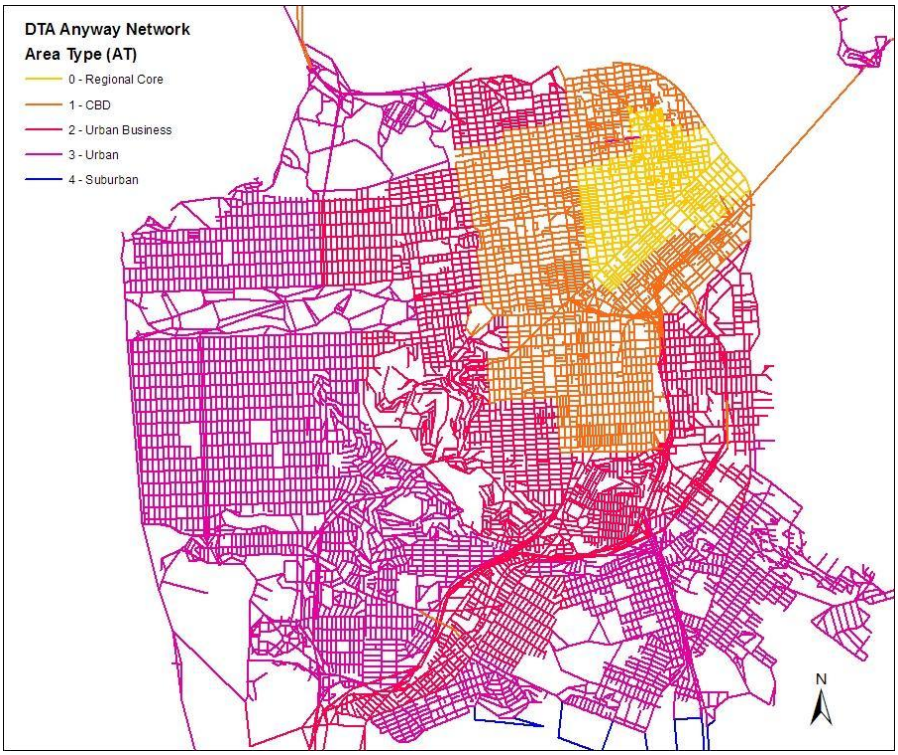

Figure 6. Map of Area Type Classifications

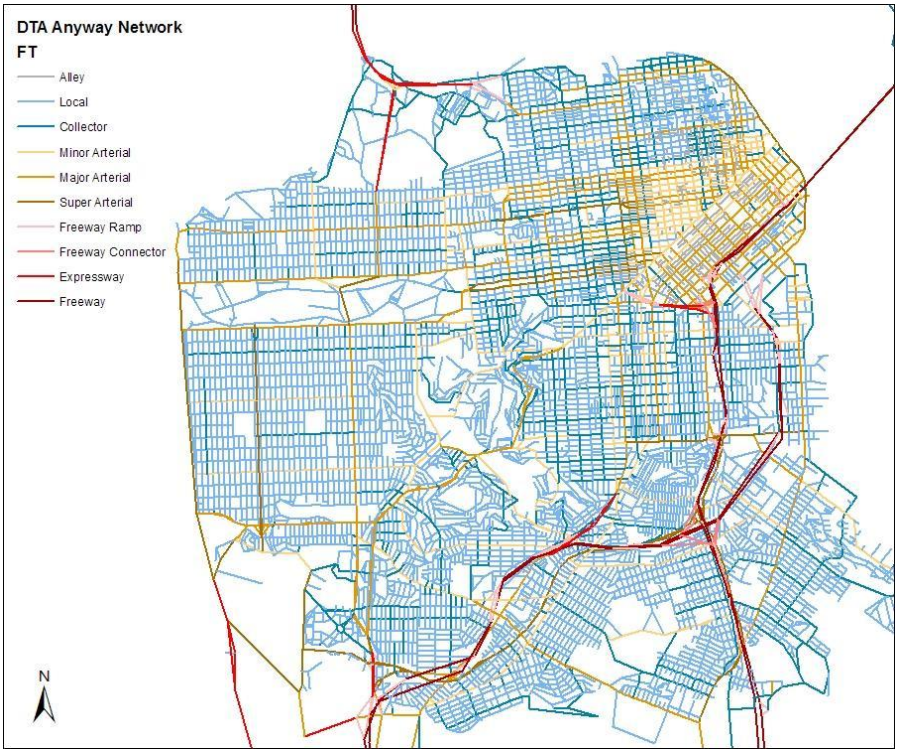

Figure 7. Map of Facility Type Classifications

3.2 Network and Demand Inputs

At this time, the San Francisco DTA model includes all network and control clean-up through the Nov. 13 calibration run. The current network representation includes several specific assumptions related to bus lanes, travel on lower order road facilities, and interactions between pedestrians and vehicles that warrant highlight and explanation.

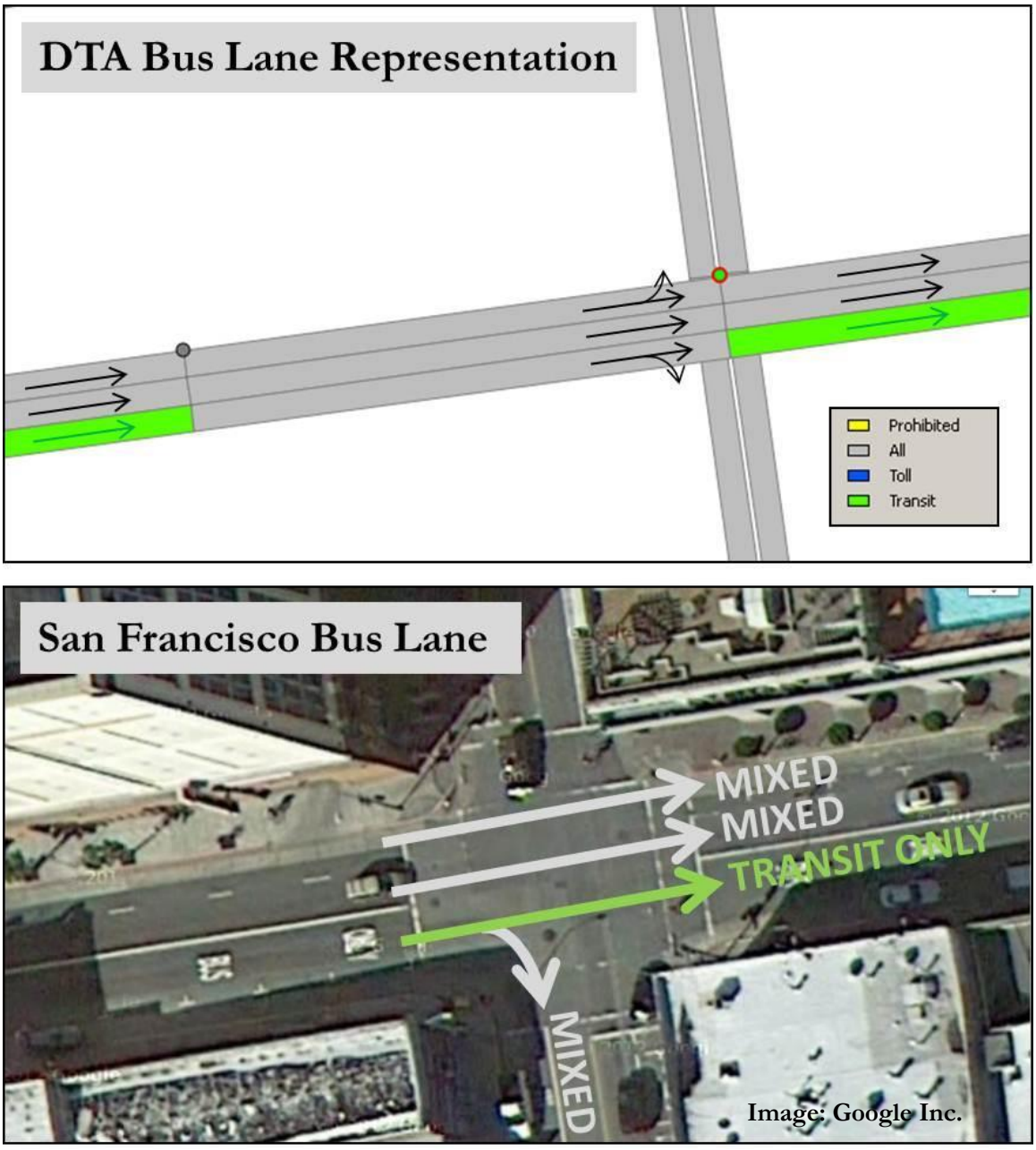

Figure 8. Comparison of Actual Bus Lane Permissions and DTA Anyway Representation of Bus Lanes

The current representation of bus lanes (shown in Figure 8) allows all traffic to use a portion of bus lanes upstream from an intersection in order to execute turning movements across bus lanes. This partially reflects real world conditions, where auto traffic is allowed to enter curbside bus lanes to make right turns, adding to

the effective capacity of the roadway, but it also allows mixed traffic that is not preparing for a turning movement to use the lane as well. The version of Dynameq that is being used for this project allows movement restrictions that are specific to user class, but not lane restrictions, so it is not possible to permit right turning autos and through moving transit vehicles to share a lane without also inaccurately allowing auto traffic to continue through the intersection. The current bus lane representation is a necessary interim step because coding the bus-only lanes to exclude the right turns results in severe gridlock within the network. Further discussion of DTA Anyway bus lane representation is given in Section 2.3.9 of this report.

Another important assumption made in this calibration run is that the perceived travel times on local and collector links is higher than the actual travel time. This is based on an assumed perceived disutility of these smaller roads that is not necessarily captured by travel time. In order to incorporate this measure of travel time perception, the generalized cost includes an additional penalty of ½*Free-flow time for alley, local, and collector links.

In an attempt to partially capture the effects of interactions between vehicles and pedestrians, signalized turning movement saturation flow rate has been reduced in central areas of San Francisco where pedestrian activity is highest. The current parameters indicate a signalized turning movement saturation flow rate of up to 1,800 vehicles per hour (but less if the approach link has a lower saturation flow rate) for most turning movements in San Francisco. This maximum turning movement saturation flow rate is reduced by 10% in San Francisco's downtown core (for turning movements from links classified as area type zero) and 5% in neighborhoods adjacent to the core (links classified as area type one). This representation of vehicle-pedestrian friction reflects how high pedestrian activity in downtown areas reduces vehicle turning movement capacity, but it does not reflect the fine-grained differences in pedestrian activity - and subsequent impacts on traffic capacity - caused by block to block variations in pedestrian activity in the city's core and in peripheral commercial districts.

The demand used in this final set of calibration, sensitivity, and applications tests remains unchanged from its extraction out of SF-CHAMP. In order to allow for adequate "warm up" and "cool down" periods before and after the simulation period, the demand loading period begins at 14:30 and ends at 19:30. In order to estimate the demand for the 14:30 to 15:30 and 18:30 to 19:30 periods, the SF-CHAMP demands for the midday and evening periods were used with factors based on truck and car counts at external stations which were then adjusted to better match the PM-peak observed volumes and travel times. The demand factors (portion of demand from the full period trip tables used) for the midday and evening tables used in this model are shown in Table 1.

Table 1. Off-Peak Demand Factors

	MD Demand Factor	NT Demand Factor
Car_NoToll	0.13364	0.22594
Truck_NoToll	0.05128	0.03921

3.3 Speed Flow Parameters

In Dynameq there are default values for passenger car effective length (20.40 ft) and response time (1.25 sec). Alternately, users can define vehicle classes with custom effective lengths and response times. The default and user specified effective length and response time values can also be altered for entire scenarios or for

individual links through the use of multiplier factors (Effective Length Factor (ELF) for EL and Response Time Factor (RTF) for right turns).

A list of parameters that are currently used in the San Francisco DTA model is provided below. Table 2 shows the free-flow speed parameters according to area type and facility type classification. These values are derived from locally collected observed data and are representative of local conditions. Table 3 shows response time factors according to facility type classification and the slope of the link. The relative differences in response time between flat, uphill, and downhill streets reflect observed conditions. Table 4 shows saturation flow rates. The saturation flow rates are somewhat higher than values observed in the field. There are two reasons why this deviation is reasonable. First, the streets where saturation flow rates were observed feature especially narrow lanes. Saturation flow rates on typical San Francisco streets may be somewhat higher. Second, INRO, the company that produces Dynameq, advised that the DTA model would perform best if traffic flow parameters create an outside envelope around observed conditions rather than tightly matching observed figures. Table 5 shows effective lengths, maximum vehicle speeds, and response times for cars and trucks. Effective vehicle lengths are shorter than those observed in the field, however since observations were conducted on streets with narrow lanes, typical effective vehicle lengths throughout San Francisco may be closer to the final parameter values. Also, in areas with heavy congestion, vehicles may reduce spacing to avoid blocking intersections. A lower effective vehicle length provides slack to account for possible driver responses to long queue lengths. The final response times are slightly lower than those observed in local surveys. The reduction was necessary in order to prevent the assignment from experiencing gridlock. These adjustments are deemed acceptable because they helped to improve the overall performance of the model and the fit of the model results. Table 6 shows the critical gap and default follow-up time values used.

Table 2. Free Flow Speeds

Free Flow Speed (mph)	Regional Core	CBD	Urban Biz	Urban
Alley	10	10	10	10
Local	18	18	18	15
Collector	23	23	20	20
Minor Arterial	26	26	28	30
Major Arterial	28	28	30	32
Super Arterial	30	30	33	36
Freeway Ramp	30	30	35	35
Fwy-Fwy Connector	35	40	45	45
Expressway	60	65	65	65
Freeway	60	65	65	65

Table 3. Response Time Factors

Response Time Factor*	Flat	Upward Slope	Downward Slope
Alley	1.0	1.1	0.9
Local	1.0	1.1	0.9
Collector	1.0	1.1	0.9
Minor Arterial	1.0	1.1	0.9
Major Arterial	1.0	1.1	0.9
Super Arterial	1.0	1.1	0.9
Freeway Ramp	1.0	1.1	0.9
Fwy-Fwy Connector	1.0	1.1	0.9

Expressway	1.0	1.1	0.9
Freeway	1.0	1.1	0.9

* Response time factors are multiplied by the response time of the vehicle class (1.0s for cars, 1.25 for trucks) to get that vehicle's response time on each link.

Table 4. Saturation Flow Rates

Saturation Flow (vphpl)*	Regional Core	CBD	Urban Biz	Urban
Alley	1,342	1,342	1,342	1,342
Local	1,760	1,760	1,760	1,633
Collector	1,923	1,923	1,831	1,831
Minor Arterial	1,999	1,999	2,044	2,084
Major Arterial	2,044	2,044	2,084	2,121
Super Arterial	2,084	2,084	2,138	2,185
Freeway Ramp	2,084	2,084	2,170	2,170
Fwy-Fwy Connector	2,170	2,239	2,296	2,296
Expressway	2,418	2,449	2,449	2,449
Freeway	2,418	2,449	2,449	2,449

* Saturation flow rates displayed are for flat terrain. Due to differences in the response time factor for grade, uphill streets have slightly lower saturation flow rates and downhill streets have slightly higher saturation flow rates.

Table 5. Length, Speed, and Response Time

Factor \ Vehicle Class	Car	Truck
Effective Length (ft)	21	31.5
Effective Length Factor	1.0*	1.0*
Maximum Speed (mph)	100	70
Vehicle Type Response Time (s)	1.0	1.25

* Effective length factor is 1.0 throughout most of San Francisco, but it is currently set to 0.95 in and around the San Francisco CBD (defined as area types zero and one (AT0 & AT1).

Table 6. Critical Gap and Follow-up Time

		Critical Gap (s)	Critical Wait (s)	Follow-Up Time (s)
Control	Crossing	3.20	60	--
	Merging	4.80		--
AWSC	Thru	--	--	3.20
	Right Turn	--	--	3.20
	Left Turn	--	--	3.20
TWSC	LT from Major	3.28	60	2.20
	FT from Minor	4.96		3.30
	TH on Minor	5.20		4.00
	LT from Minor	5.68		3.50
Merging	Merge	3.28	60	2.60
Signalized	U-Turn	5.20	60	4.00
	LT conflict Thru	3.60		2.50*
	LT conflict RT	3.60		2.50*
	Right Turn	--		2.50*
	Thru	--		1.80*
	Turn on Red	4.96		4.00

* Signalized left turn and right turn follow-up times are designated for individual links according to the area type classification of the incoming link. Through movement follow-up times are also designated. When follow-up times are designated for specific links, those values override the settings in this table.

3.4 DTA Specifications

The following settings correspond to Dynameq settings required to run the assignment.

Assignment

These values define the period of the simulation.

Start of demand	14:30
End of demand	19:30
End of simulation period	00:30 (+ 1 day)
Transit lines simulation	Yes
Re-optimization	No
Re-optimization iteration(s)	0

Demand

These values specify how the demand is input, and the generalized cost equations.

Cars	Class	Car_NoToll
	Matrix	Car_notoll
	Paths	20
	Intervals	20
	Types(%)	Car = 100
	Movement expression	ptime+(left_turn_pc*left_turn)+ (right_turn_pc*right_turn)
	Link expression	fac_type_pen*(1800*length/fspeed)+toll_link*toll_cost
Trucks	Class	Truck_NoToll
	Matrix	truck_notoll
	Paths	20
	Intervals	20
	Types(%)	Truck=100,
	Movement expression	ptime+(left_turn_pc*left_turn)+(right_turn_pc*right_turn)

	Link expression	fac_type_pen*(1800*length/fspeed)+toll_link*toll_cost

Interpreting the above, trips are being assigned to the network using the following generalized cost equation:

```
GeneralizedCost = Time + LeftTurnPenalty + RightTurnPenalty
                  FacilityTypePenalty + TollPenalty
```

where:

`LeftTurnPenalty` = 30 seconds if a movement is a left turn
`RightTurnPenalty` = 10 seconds if a movement is a right turn
`FacilityTypePenalty` = 50% of free flow travel speed if link is alley, local or collector
`Toll Penalty` = Toll as specified in the scenario

The turning movement penalties prevent the abundance of paths zigzagging across the grid network. The facility type penalty is set to be equal to half of the link's free flow time for locals and collectors. This was implemented in order to get reasonable paths while still respecting the observed free-flow times on locals and collectors. An operating cost was included in some versions of the generalized cost expression, but it is not used in the final calibration model. When applied, the operating cost term is calculated based on an assumed fuel cost/driving cost multiplied by the value of time in this network to arrive at an operating cost in units of seconds. The toll penalty is only used in scenarios that include congestion pricing. The toll penalty is calculated by dividing the toll cost assigned to a link by the assumed value of time to achieve a time cost, in seconds, of paying a toll. To be consistent with the static model, the current model assumption is that the value of time for all drivers is $15 per hour in 1989 dollars.

Control Plans

These settings relate to the import of signal control plans to Dynameq, which are read from the SFMTA signal timing format in Excel, and written to a format that Dynameq can read.

excelSignalsToDynameq	14:30 - 21:30

Results

These settings specify the time steps used by Dynameq.

Simulation results	14:30:00 - 00:30:00 (+ 1 day) -- 00:05:00
Lane queue animation	14:30:00 - 00:30:00 (+ 1 day) -- 00:05:00
Transit results	14:30:00 - 00:30:00 (+ 1 day) -- 00:05:00

Advanced

These values are settings for the DTA method used by Dynameq.

Traffic generator	Conditional
Random seed	1

Travel times averaged over	450 s
Path pruning	0.001
MSA reset	3
Dynamic path search	No
MSA method	Flow Balancing
Effective length factor	1.00
Response time factor	1.00

4. Final Validation Assessment

As discussed above, the SF-DTA parameters were calibrated to produce reasonable model results. The accompanying *Final Model Calibration and Validation Report* describes that calibration process in detail with a focus on the changes that were necessary to get to that final state. Several important issues addressed during calibration were:

- Updating speed flow parameters to generate an appropriate amount of congestion without resulting in complete gridlock.
- Setting the perceived time in the generalized cost function to encourage travelers to utilize the arterial system over local roads.
- Accounting for the conflicts between turning vehicles and pedestrians in crosswalks, particularly in the downtown area.
- Coding the bus lanes to allow right turning vehicles to use those lanes without allowing through auto traffic to use the lanes.
- Handling special cases of complex signal phasing and timing plans.
- Extending the demand loading period to include sufficient warm up and cool down time before and after the 3-hour PM peak period.

The *Final Model Calibration and Validation Report* also presents the model validation results. The model is able to achieve a stable convergence with a mean relative gap of 2.7%. Overall, the model predicts volumes that are 13% lower than the counts, with the difference uniformly distributed throughout the PM peak period. The RMSE is high for low volume roads, but drops to around 40% for links carrying at least 500 vehicles per hour. 65% of all links, and 76% of arterial links fall within the maximum desirable deviation recommended by the Caltrans Travel Forecasting Guidelines[24]. This meets the target of 75% of arterial links within the maximum desirable deviation. The model predicts travel times reasonably well, with the exception of a handful of outliers, and the overall pattern of traffic is both logical and similar to what the static model predicts.

Overall, this is deemed to be a reasonably good "system-level" validation, making the model suitable for application. For major projects, a more detailed "project-level" validation may be warranted to ensure that

[24] State of California Department of Transportation Travel Forecasting Guidelines, November 1992, accessed at: http://ntl.bts.gov/DOCS/TF.html.

the study area does not fall within a specific area where there are a number of outliers or where the model under-performs.

The *Final Model Calibration and Validation Report* also presents the results of several sensitivity tests. These tests show that changing the random number seed or making a trivial network change can result in modest differences in traffic flows and travel times throughout the network. This stability is a topic recommended for further investigation to ensure that the noise level remains small enough so as not to obscure the differences being measured by project tests.

A test of loading 2040 demand on the network reveals that it is possible for the network to clear and that the relative gap is high but moving in the right direction. It also reveals that doing so would result in a large traffic jam throughout much of the city, as would be expected. It is unknown how much feedback to SF-CHAMP might mitigate the traffic levels observed here.

Finally, the model was applied to two hypothetical projects: a BRT scenario and a congestion pricing scenario. In both cases, the model produced a reasonable change, and one that was somewhat different from what the static model predicted.

Through this effort, we have identified a number of areas for further improvement, which are outlined in the Future Research Topics Report. Those research directions will be pursued, but in the meantime, we feel comfortable with the current state of the model for use in the initial round of project applications.

5. Lessons Learned

Several lessons can be learned from our experience with this project. Those lessons are summarized here, and broken into categories.

Software and Application Process

Integrated process makes application easier. Implementing an integrated application process saves time in model application and ensures that each scenario can be run with the correct demand. By an integrated application process we mean the automated the process of creating inputs from SF-CHAMP to feed SF-DTA such they run on a consistent set of networks and with a consistent trip table. While this involved some up-front investment, the automated tools ensured that the effort involved in running test applications for BRT and congestion pricing was minor, and also allowed the demand changes from those scenarios to be captured by DTA.

Subarea extraction causes loss of temporal dimension of demand. The current process of extracting subarea demand from a regional model results in a loss of the temporal dimension of that demand and should be improved. Subarea extraction is necessary to generate trip tables for DTA because SF-CHAMP is applied for the full 9-county Bay Area, whereas SF-DTA is applied only for San Francisco. The peer review panel recommended further refinement to this process, and an approach (termed External Geographic Representation) to getting around this limitation is proposed in the accompanying *Future Research Topics Report*. The proposed approach would maintain the external TAZ for trip ends outside of San Francisco, allowing the time-of-day demand to be tabulated directly from the trip lists. This could be important if, for example, the work trips departing

from the CBD have tend to all leave at the same time causing a specific surge of traffic on part of the network.

Calibration and Validation

Matrix estimation is not necessary. This project demonstrates that a DTA model of this scale can be reasonably well calibrated and validated without relying on origin-destination matrix estimation. The model produces reasonable travel times, has logical overall flow patterns, and meets the Caltrans guideline for maximum desirable deviation for 75% of arterial count locations. There remains room for improvement, and further refinement may be warranted for specific project applications, but the system-wide results are reasonable. Avoiding matrix estimation allows demand for future-year and alternative scenarios to be fed directly to DTA without concern for whether the matrix adjustments remain valid.

DTA models are subject to cliff effects. At several points the calibration process, we would make a seemingly small change to that would result in the assignment breaking down into complete gridlock. This result was most apparent when calibrating the traffic flow parameters where an effective vehicle length of 21 ft produced reasonable results, whereas an effective vehicle length 22 ft resulted in gridlock throughout the CBD and an effective vehicle length of 20 ft resulted in very light congestion in the CBD. This behavior is in contrast to a static assignment model, which usually predicts a smooth response. The DTA model's response is realistic, because it reflects the nature of traffic where adding a small number of vehicles beyond a critical threshold can reduce the flow dramatically. While realistic, this behavior makes calibration more challenging because comparisons to traffic counts are not meaningful if modeled traffic cannot flow through the network. It also has implications for model application if a relatively small change can have a big impact.

A single bottleneck can cause the entire network to become gridlocked. A closely related lesson learned is that a single bottleneck can cause gridlock throughout the network. Conversely, if important bottlenecks are missing, the model will show much less congestion than reality. Much of the effort in calibrating the model involved identifying and resolving specific locations that were the source of backups, or that should be the source of further backup. An example discussed in the *Final Calibration and Validation Report* occurred on northbound US-101 where drivers must choose between going west towards the Central Freeway and east towards the Bay Bridge. Vehicles changing lanes at this split causes a backup that did not clear for hours after the end of the demand loading period. Understanding the location of bottlenecks is important because it determines whether the response should be targeted (update network coding, check signalization) or global (traffic flow parameter or generalized cost function changes).

A data driven approach provides a valuable starting point. The DTA process involves a wide enough range of parameters that could be calibrated that the dimensionality of the problem means that an optimal solution cannot easily be identified. To navigate through this process, we found a data driven approach to be valuable because it provided a foundation to know where certain values should be set. By this, we mean that field observations were conducted to measure traffic flow parameters under local operating conditions. While these observations provided a starting point, flexibility is still needed in calibration. For this model, sensitivity testing (as described in section 4.1 of the *Final Calibration and Validation Report*) demonstrated that we could achieve better results by deviating from the observed values.

Sensitivity Testing and Applications

Sensitivity testing is part of calibration. The sensitivity testing and model application process were found to be highly informative, and should be included as part of model calibration.

Model stochasticity can affect comparisons between scenarios. The random seed and the minor network change tests, and to a lesser degree the BRT scenario, demonstrated that there is some amount of stochasticity in all of the models which can affect the comparison of base to applications results. Each of these tests showed some modest changes due to noise likely caused by the stochastic nature of how DTA models are applied. Further tests to determine the full impact of different values of the random seed may indicate a need to run any future tests with multiple random seed values and average the results to determine the most accurate levels of flow and travel times for each scenario. While this is a limitation, it goes hand-in-hand with DTA's greater realism and ability to better capture diversion. There may be ways to mitigate the impacts of this issue, but further research is required to determine the best method to use.

DTA predicts more diversion. The BRT scenario demonstrated that under the congested conditions tested, that the DTA model predicts more diversion than the static model run for the same scenario. This occurs because the DTA model does not allow volume to exceed capacity, whereas the static model can simply force more traffic through the same corridor. The level of diversion predicted by the DTA model appears intuitively more reasonable, but the result should ideally be evaluated against empirical data, such as before-and-after counts from a similar project.

Appendices and Related Documents

Appendices:

A – Project Website and Codebase

B – Responses to Peer Review Panel Comments

Related Documents[25]

- **Final Calibration and Validation Report**
- **Analysis of Applications Report**
- **Future Research Topics Report**
- **Peer Review Panel Report**
- **Integration Options Report**
- **Initial Calibration and Validation Report**

[25] Related documents can be found on the website http://dta.googlecode.com

Appendix A: Project Website and Code Base

The project source code and the documentation for the API can be found at http://dta.googlecode.com, which will load the Project Home page for DTA Anyway.

To see the code API documentation, click on the link labeled "Code Documentation" on the Project Home page.

To download the source, go to tab labeled "source" and follow the instructions there.

To view the source without downloading, go to the tab labeled "source" and then click the "Browse" link. The code is developed in the "dev" branch, but release code will be pushed to the "master" branch; choose "dev" in the Branch drop-down to see the most recent version of the code base.

To see the batch file that drives the entire process for creating the San Francisco DTA network, follow this link: http://dta.googlecode.com/git-history/dev/doc/_build/html/script_importFullSanFranciscoNetworkDataset.html

Appendix B: Response to Peer Review Comments

The following section responds to short-term recommendations by the SF DTA peer review panel. Longer term recommendations are addressed in the *Future Research Topics Report*.

Traffic Flow Model

The DTA settings currently used appeared to make sense to the panel. It was suggested that further changes to these settings may be guided by the calibration and sensitivity tests conducted.

Dozens of sensitivity tests on traffic flow parameters were conducted in order to arrive at the final model specification. They are documented in section 4 of the *Final Calibration and Validation Report*.

The free-flow speed for links with stop-controlled intersections could be too high. For overestimation issue on local streets, imposing twice the free flow time may not be ideal. There might also be an aversion component in addition to the time component. It might be better to try and tweak the traffic flow parameters such as reaction time. This factor could act as a "perception penalty" and could be a function of number of stop signs. Alternatively, a separate facility type for residential low-volume streets could be created. In addition to this, a reaction time factor that includes friction due to pedestrian traffic on such streets could be tested.

We created an "alleyway" facility type with more restrictive traffic flow parameters. The free-flow speeds for links with stop-controlled intersections were adjusted downward in order to account for the deceleration/acceleration of stopping. Finally, we added a perception penalty to the generalized cost for alleyways, local, and collector facility types.

Make sure every intersection has control

This is now an error check that the codebase performs. While not every intersection currently has control, those that have no control are identified by the code so that additional data gathering can be done to identify whether those nodes should be signalized or stop-controlled.

Regarding pedestrian impedance: there might be value in developing link/node specific reaction time factors targeted at specific areas in the network which this could be important. They could also be derived in a systematic way by using information on aggregate pedestrian demand within a radius of a node (buffering). In addition to area type and facility type currently being used to classify the various parameters, there is potential to use a third dimension – "intersection type" that would allow to make targeted improvements.

A simplified approach for modeling pedestrian impedance was undertaken that approximates pedestrian turning movement friction by increasing headways for turning movements in areas of San Francisco with high levels of pedestrian activity. The current implementation assigns higher follow-up times to right and left turning movements that lack a dedicated turning movement phase for intersections designated with area type

classification zero or one. The details of this approach are described in section 2.4 of the *Final Calibration and Validation Report*.

In addition to the current area-type based accounting of pedestrian friction, the Future Research Topics Report includes a topic on Improved Non-Motorized Mode Representation. That future direction would involve developing a methodology to account for pedestrian and bicycle friction in a "demand-specific" manner, meaning that it would rely in some form on pedestrian and bicycle volumes predicted by SF-CHAMP.

Validation

Check path reasonableness

Paths and routing issues were evaluated during specific investigations during the calibration process.

Look at VMT/VHT for whole network to see if demand is 'missing'

The network-wide VMT and VHT are reported in the *Analysis of Applications Report* for the two test scenarios. The network-wide totals are similar between the static model and DTA.

Generalized Cost

The panel felt that including distance in the generalized cost function might help towards the traffic overestimation issue on local streets. It is probably more applicable for truck traffic than autos.

This was tested for both autos and trucks using the assumed fuel costs ($0.16/mile) multiplied by the link length and the value of time. This measure created high levels of congestion in the CBD, so we also tested a value with half of the original coefficient on distance. While that created less congestion, neither measure greatly improved the count-matching or speed-matching. The impacts of adding the distance term to the generalized cost were generally negative, so it was not included in the final calibrated version of the model.

Demand

The panel thought that the demand carving for a subarea was a non-trivial process and there should be some more focus on that. External geographical information could be preserved and the demand from external stations could be shifted based on the travel time to the model subarea. In addition, a temporal profile at external gateways and in internal zones could be created to avoid a flat or uniform loading profile to the DTA model. Since traffic counts are available, they may be used to create the temporal profile. Alternatively, departure times from the household survey could also be processed to obtain a temporal profile.

The temporal profile has been adjusted based on counts obtained from the PEMS database (pems.dot.ca.gov) for four network entry locations. Additional detail about how this profile was derived is available in section

2.4 of the *Final Calibration and Validation Report*, with the final coefficients shown in Section 2.7 of the *Final Methodology* report.

The panel felt that a longer warm-up period may be needed to improve the accuracy of the simulated traffic in the current modeling period (4:30 – 6:30 PM). This could also incorporate a portion of mid-day demand to create background traffic which would already be present before the model period. Further, there might be some value in simulating the entire 3-hour period (3:30 – 6:30 PM) in the DTA model to facilitate a straightforward comparison to the static model.

The simulation period has been expanded to incorporate demand from the midday and evening periods. Signal timings are continued throughout the extended simulation.

Validation

Standards: In the collective experience of the panel members, it was felt that there may not be any validation standards that are broadly accepted and also there may be no national benchmark for root mean squared error (RMSE) of flows and speeds. The panel thought the reason for this may be the limited number of studies currently existing in the DTA arena and different limitations being associated with various DTA packages used in these studies. The panel members recommended that regular Caltrans static validation standards could be used as a starting point and then extended for more refined time periods. It was noted here again that a 3-hour validation period would facilitate a more direct comparison with the static model. The panel members also stressed on the consistency of reporting structure that needs to be maintained for such a comparison.

For the purpose of this project, two validation standards were evaluated. The first is the Ohio RMSE target, which provides a target percent root mean square error by volume group. The second was the Caltrans guideline for maximum desirable deviation of link volumes. These are described in section 3.1 of the *Final Calibration and Validation Report*.

Stability: The panel noted that specific value of relative gap may not be as important as the stability of the gap. Maximums and minimums of traffic characteristics such as speeds may be checked to see if those have stabilized over iterations. It might also help to look at variation in VMT and VHT as additional measures of stability.

Relative gap stability was specifically used as a criteria in our model calibration.

Conflicting Data: The general response of the panel to this was to obtain more data so that more cross-checking can be done. The panel felt more observed traffic data on local streets would be useful for validation given that there appears to be considerable overestimation of traffic. Expanding the number of observed traffic count locations was also offered as a long term consideration. The panel felt that the current number of 200 locations may be insufficient for a city the size of San Francisco. The panel noted that there are 400 count locations in the SACOG

area for a population of about a million people and that counts should be geographically distributed and not correlated.

During the development of DTA Anyway, the Count Dracula system was also revised and updated extensively. Counts were added for additional locations in the network, and truck counts were also added where available. A map of the traffic count locations is available in section 2.8 of the *Final Methodology Report*. As described in section 3.3 of the *Final Calibration and Validation Report*, the final validation is based on traffic counts on 454 links.

Sensitivity Tests

To aid calibration and validation, more sensitivity tests need to be conducted from the perspective of future and alternative policy scenarios. It might help to devise some future scenarios tests and targeted policy tests. Since the Authority already has an idea about the initial policies that it would like to use this model for, it might be useful to build some tests around them and evaluate the results qualitatively first. Sensitivity tests on both demand and traffic flow parameters would help understand which parameters affect to what extent which in turn could help identify the parameters that need to be focused on.

In addition, the panel recommended conducting sensitivity tests around traffic flow parameters such as jam density and also flow averaging parameters. Since sensitivity is contextual, the panel suggested analyzing if the ranking of investments might change due to certain changes in these parameters. Based on which parameters affect the ranking of investments to what extent, the modeling team may be able to focus more on those parameters during calibration and validation. The panel thought that there are no standards as to how sensitive a model is to the DTA settings since the sensitivity might be dependent on the DTA package being used and modeling assumptions made in them.

The panel offered various methods to check the model's sensitivity to network changes. At first, it was suggested that progression of traffic on major arterials be confirmed. Another basic test would be to visually inspect the relevant paths for reasonableness. Finally, the panel recommended examining areas in the network that are specifically affected by bottlenecks and queues. These are the areas where static model would be significantly inaccurate in predicting the traffic flow patterns.

Sensitivity tests were conducted around the following dimensions: traffic flow parameters, random number seeds, demand, and network changes. These are discussed in Section 4.

Network Coding

The modeling team described a method of splitting a bus-lane link and making one-half of the link right-turn only to deal with bus lanes in the model. The panel suggests that targeted reaction

time factors adjustments could potentially be used to accommodate more throughput going through the general purpose lanes and better represent the traffic flow on these lanes.

The team kept the approach of link splitting as described in section 3.2 of the *Final Methodology Report*.

Demand

Could overlay demand to represent circling vehicles downtown looking for parking.

This was considered but determined to be outside the scope of this initial calibration exercise.

Include a more geographically consistent representation of external stations.

This was considered but determined to be outside the scope of this initial calibration exercise.

The panel mentioned that rounding of fractional trips may also be a source of issues in traffic prediction. Even if there are no trips lost in total, there may be significant loss of trips in specific zones. The panel recommended that bucket rounding be used over arithmetic rounding.

This feature was tested, but not implemented in the final validation. Dynameq's internal bucket rounding procedures are still used.