A Demonstration of MNTG - A Web-based Road Network Traffic Generator

Mohamed F. Mokbel[1], Louai Alarabi[2], Jie Bao[3], Ahmed Eldawy[4], Amr Magdy[5], Mohamed Sarwat[6], Ethan Waytas[7], Steven Yackel[8]

[1,2,3,4,5,6,7]University of Minnesota, Minneapolis, MN 55455, USA
[8]Microsoft
{mokbel[1],louai[2],baojie[3],eldawy[4],amr[5],sarwat[6]}@cs.umn.edu wayt0012@umn.edu[7] spazard1@live.com[8]

***Abstract*—This demo presents Minnesota Traffic Generator (MNTG); an extensible web-based road network traffic generator. MNTG enables its users to generate traffic data at any arbitrary road networks with different traffic generators. Unlike existing traffic generators that require a lot of time/effort to install, configure, and run, MNTG is a web service with a user-friendly interface where users can specify an arbitrary spatial region, select a traffic generator, and submit their traffic generation request. Once the traffic data is generated by MNTG, users can then download and/or visualize the generated data. MNTG can be extended to support: (1) various traffic generators. It is already shipped with the two most common traffic generators, Brinkhoff and BerlinMOD, but other generators can be easily added. (2) various road network sources. It is shipped with U.S. Tiger files and OpenStreetMap, but other sources can be also added. A beta version of MNTG is launched at: http://mntg.cs.umn.edu.**

I. Introduction

Having access to road network traffic data is a major need to validate and evaluate indexing and query processing techniques in moving objects databases, spatio-temporal databases, and data streams. Unfortunately, such data is not easily available, and is usually of much smaller scale than needed. The process of extracting real traffic data requires installing and configuring many GPS-enabled devices and continuously monitoring the locations of such devices, which is a cumbersome task. For instance, GeoLife project [13] took more than four years to collect 17,621 trajectories dataset with the involvement of 182 volunteers in Beijing. As a result, researchers have been using existing traffic generators as a means of getting synthetic datasets that exhibit similar behavior to real data. The most two common traffic generators are Brinkhoff [1] and BerlinMOD [3], which have been widely adopted by large numbers of papers in the database literature, e.g., see [2], [5], [6], [8], [10], [12].

Even though existing traffic generators are quite useful, nonetheless, most of them suffer from the following: (1) It may take the user significant amount of effort to install and configure the traffic generation tool. For example, in order to run BerlinMOD, the user needs to first install a moving object database, i.e., SECONDO [4], and then get familiar with the script commands used to install it. After the installation, users still need to understand an extensive list of configuration parameters for each traffic generator. (2) It is not trivial to generate traffic data in arbitrary spatial regions using existing traffic generators. For example, to be able to use Brinkhoff or BerlinMOD generators for a different city than the default shipped one (Oldenburg and Berlin for Brinkhoff and BerlinMOD generators, respectively), the user needs to first obtain the road network information for the city of interest, which is a tedious task by itself. For example, to get the road network information for the city of Chicago, a user may need to understand the format of OpenStreetMap [9], and then write a program that extracts the road network of Chicago from OpenStreetMap. After obtaining the new data, the user then needs to modify the obtained format to match the required one by either Brinkhoff or BerlinMOD. Such set of tedious operations made it hard for casual users to use these traffic generators for arbitrary spatial areas. As a testimony, one can observe that almost all the literature that harnessed these generators, used their default cities.

This demo presents Minnesota Traffic Generator (MNTG) [7]; an extensible web-based road network traffic generator. MNTG is not a new traffic generator. Instead, it is a framework that encapsulates existing traffic generators and makes them easily accessible. MNTG overcomes the hurdles of existing traffic generators with three main features: (1) MNTG is a web service with a user-friendly map interface. Behind the scenes, MNTG carries the burden of configuring and running existing traffic generators. (2) MNTG can be used for any arbitrary spatial area, where users can just mark their area of interest on a map interface, and submit their traffic requests accordingly. (3) MNTG users do not need to worry about the processing time or computing resources, where MNTG has its own dedicated server that internally processes the request in a multi-threaded paradigm, and emails the user back when the data is generated. The notifying email includes a link to download and/or visualize the data.

MNTG is extensible to support: (1) *various traffic generators*. Currently, MNTG is shipped with the two most common traffic generators, Brinkhoff and BerlinMOD, yet, it also has the interface that can be used to add new traffic generators. (2) *various road network sources*. It is currently shipped with the support for U.S. Tiger files [11] and OpenStreetMap [9], yet, it also has the interface that can be used to add other sources for road network data.

A beta version of MNTG is launched as a web service for public use; a prototype can be accessed via http://mntg.cs.umn.edu. The beta version supports both Brinkhoff and BerlinMOD traffic generators on U.S Tiger files and OpenStreetMap data. The extensibility interface for adding

*This work is supported in part by the National Science Foundation, USA, under Grants IIS-0952977 and IIS-1218168.

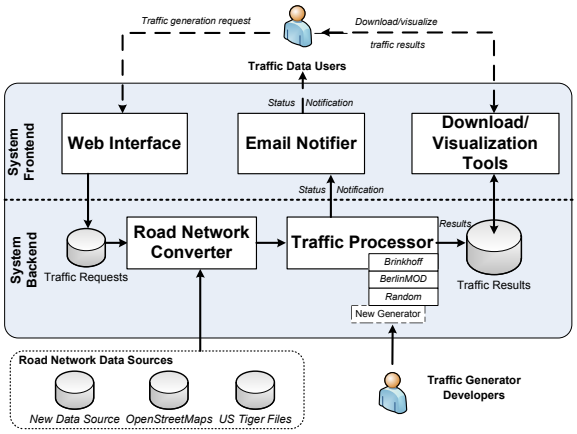

Fig. 1. MNTG System Overview.

more generators or other road network sources is currently working internally under our support, and will be shown in the demo. Since its launch in February 2013, MNTG has received more than 500 traffic generation requests from researchers world wide. We envision that MNTG will be the de facto standard for generating road network data for researchers in spatial and spatio-temporal databases worldwide.

II. System Overview

Figure 1 gives an overview of MNTG architecture. A user interacts with MNTG through its *system front-end* that includes a *web interface* that allows users to submit traffic generation requests, an *email notifier* that retrieves the status updates from the back-end and keeps users posted, and *download and visualization tools* that allow users to download and/or visualize the generated data. Internally, MNTG system back-end processes the traffic generation requests, where it includes two main components: (1) *Road Network Converter*, which is responsible for extracting the road network data from different data sources. It currently uses US tiger files or OpenStreetMaps. Yet, it is extensible to support other road network sources. (2) *Traffic Processor*, which takes the road network data and feeds it to the underlying traffic generator. It currently support both Brinkhoff and BerlinMOD data generators. Yet, it is extensible to include other data generators.

III. Road Network Converter

MNTG employs a *road network converter* that (a) receives the user area of interest, (b) extracts the road network data for the area of interest, and (c) sends the extracted data to the *traffic processor*. We will first explain (and demonstrate) how to include a road network data source in MNTG, then, we will discuss two case studies that are already realized in MNTG; US Tiger files, and OpenStreetMap.

A. Adding Road Network Data to MNTG

To add a new road network data source, MNTG provides a template that consists of the following two function headers:

1) `ExtractRoadNetwork`. This function takes a rectangular area, defined by two <*latitude*, *longitude*> coordinates, as its input, and produces the road network information of the selected area as its output. This includes pruning all information outside the selected area as well as pruning the non road network information of the selected area.
2) `PrepareStandardOutput`. This function takes the road network information from `ExtractRoadNetwork` as its input, and produces two standard text files `node.txt` and `edge.txt`, that contain the final set of nodes and edges in the selected area, respectively. The text files are in a standard format to make it portable to various traffic generators employed by MNTG *traffic processor*.

B. Case Study 1: U.S. Tiger Files

US Topologically Integrated Geographic Encoding and Referencing (Tiger) Files [11] are published by US census bureau on a yearly basis to provide the most recent information about US geographical data, which include city boundaries, road networks, address information, water features, and much more. A very unique feature of US Tiger Files is that the files are partitioned and organized based on US counties. In other words, all roads in a county are packed as one compressed file with a unique file identifier, e.g., `tl_2010_01001_roads.zip`, where `tl` means tiger line, `2010` indicates the publishing year of the data, `01001` is a unique identifier for the county (in this case is *Autauga, Alabama*), and `roads` represents the type of data.

In this case, the function `ExtractRoadNetwork` identifies all counties that overlap with the rectangle area selected by the user. Then, it loads the road network files for all overlapped counties and filters out the road segments outside the selected region. Then, the `PrepareStandardOutput` function converts Tiger file format to the standard output format for `node.txt` and `edge.txt`.

C. Case Study 2: OpenStreetMap

OpenStreetMap [9] (OSM) is a free GIS service, contributed mainly by volunteers to record all spatial landmarks (including road networks) worldwide. All data in OSM is encapsulated into a large XML file `Planet.osm` that includes four primitive data types: (1) *Node*, that represents a spatial point by its latitude and longitude coordinates, (2) *Way*, that consists of a sequence of *nodes* to construct a line or a polygon, (3) *Relation*, that specifies the relation between *ways* and *nodes*, e.g., two *ways* are connected together, and (4) *Tags*, that provide description for other data types, i.e., *node*, *Way* and *Relation*, using a key-value pair.

In this case, the function `ExtractRoadNetwork` first retrieves all geographical data with an "osm" file, based on the spatial area specified by the user, from OpenStreetMap interface. Then, it parses all XML key-value pairs in the downloaded "osm" file to extract the road network information. Then, `PrepareStandardOutput` converts the road network information (extracted from the "osm" file) to the standard output format for the files `node.txt` and `edge.txt`.

IV. Traffic Processor

MNTG provides a wrapper around existing traffic generators to ensure their ease of use. The *traffic processor* component of MNTG is responsible on: (1) receiving the road

network data from the *road network converter* and feeds it to the underlying generator, (2) running the underlying selected traffic generator, and (3) producing the generating traffic to be downloaded and/or visualized on a map interface. We first explain how to include a traffic generator in MNTG. Then, we discuss two case studies that are already realized in MNTG; Brinkhoff and BerlinMOD traffic generators.

A. Adding a Traffic Generator to MNTG

To add a new traffic generator, MNTG provides a template that consists of the following three function headers:

1) `RoadNetworkConvert`. This function takes a road network in the standard format of two text files `node.txt` and `edge.txt` as its input. The output would be the same road network, but in a different format that is required by the traffic generator.
2) `TrafficGeneration`. This function first pops up a dialogue interface to prompt the user to enter various parameters for the generation request, e.g., number of objects and time interval. Then, it takes these parameters along with the specified road network extracted from `RoadNetworkConvert`, and makes an external execution call to the traffic generator.
3) `TrafficResultConvert`. This function takes its input as the output of the `TrafficGeneration` function. Then, it converts the input into a standard text format as < *Object ID*, *Timestamp*, *type*, *latitude*, *longitude*> that can be easily downloaded as a text file and/or visualized a map.

B. Case Study 1: Brinkhoff Generator

Brinkhoff traffic generator is one of most widely used traffic generators [1] (cited 650+ per Google Scholar), where its general idea is to simulate the object movements from two random locations using the shortest path. To support Brinkhoff generator inside MNTG, we implement its three abstract functions as follows: The `RoadNetworkConvert` function converts the output of the *Road Network Converter* into two binary files based on the need of Brinkhoff, i.e., `request_ID.node` and `request_ID.edge`. The `TrafficGeneration` function prepares Brinkhoff configuration file, i.e., `property.txt`, and assembles the calling command based on the user's request. The `TrafficResultConvert` function converts the traffic data produced by Brinkhoff generator into our standard output format.

C. Case Study 2: BerlinMOD Generator

BerlinMOD is another very popular traffic generator [3], where it simulates human movements during the weekdays and weekends. Users can specify their work and home areas in the road networks, then the generator simulates the users movements based on two rules: (1) during the weekdays, a user leaves Home in the morning (at 8 a.m.+ *T1*), drives to Work, stays there until 4 p.m.+ *T2* in the afternoon, and then returns back Home, (2) during the weekends, a user has an 0.4 probability to do an additional trip which may have 1-3 intermediate stops and ends at home.

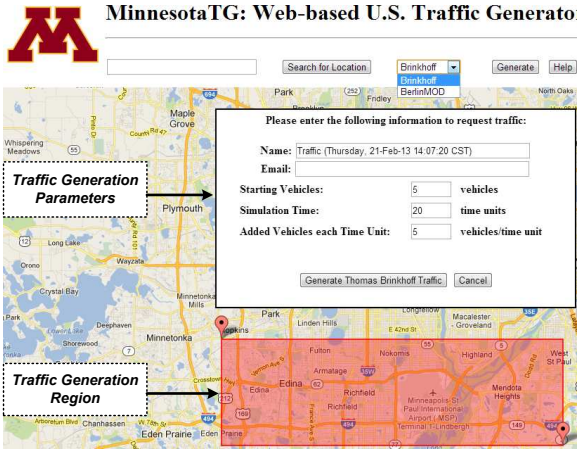

Fig. 2. MNTG Web GUI: Traffic Generation (Google Maps Interface)

To run the BerlinMOD traffic generator, a user would need to set up a SECONDO database [4], and uses a set of scripts to query it. To support BerlinMOD generator inside MNTG, we install and configure a SECONDO database in our server and implement the three abstract functions as follows: The `RoadNetworkConvert` function reads the standard road network files and transforms it to the format used in BerlinMOD. Ultimately, it produces a data file named `street.data`, where the road segments are represented by a pair of locations bounded by a set of brackets. The `TrafficGeneration` function prepares a customized query script for the SECONDO database, i.e., `BerlinMOD_DataGenerator_RequestID.SEC`. It does so by preparing a generic script in advance, and updates the parameters based on the user request. Then, the function runs the modified script to generate the traffic data. The `TrafficResultConvert` function converts the traffic data produced by BerlinMOD into the standard output format.

V. Demonstration Scenario

The audience interact with MNTG through the following steps: (1) *traffic request submission*, where audience can submit their requests online by selecting the traffic generation area from either Google Maps or OpenStreetMap interface and the traffic generator method, (2) *traffic data download and visualization*, where audience can either download or visualize their requested data, (3) *adding a new road network data source*, where audience can add a new road network data source to MNTG, and (4) *adding a new traffic generator*, where audience can register a new traffic generator, and then uses it in their new traffic generation request.

A. Traffic Request Submission

Figure 2 gives the *interface* for MNTG. To generate data, a user performs the following four steps: (1) either drag/zoom the map or write an address in the search field to get the geographical area of interest, (2) draw a rectangle on the map for the traffic generation area, (3) select the traffic generator from the drop down menu, e.g., *Brinkhoff* or *BerlinMOD*, and (4) Click on the *Generate* button, and enter the parameters for the traffic simulation, including the user email address.

Once the request is submitted, MNTG sends an email to the user acknowledging the receipt of that request. Upon

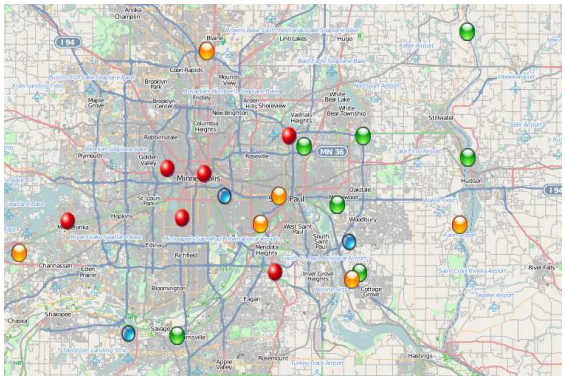

Fig. 3. MNTG Traffic Visualization (OpenStreetMap Interface)

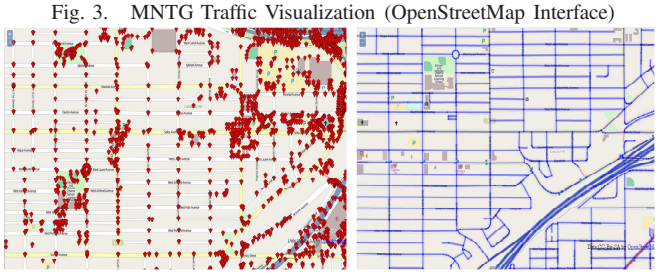

Fig. 4. Adding New Road Network Data Source

completion of the traffic request, another email is sent to the user with two hyper links. One for downloading the generated data, and one for visualizing the data on MNTG web interface. The time MNTG takes to complete the request mainly depends on the request size and the underlying traffic generator, e.g., number of objects and simulation time.

B. Traffic Data Download & Visualization

Audience can use MNTG to download or visualize their requested data. Downloaded data will have a standard text format as follows:

```
OID TS Type       Lat             Lng
0   0  newpoint   44.98636   -93.29820
1   0  newpoint   44.99894   -93.18128
```

MNTG also stores the generated traffic data inside a MySQL database, which can be used for visualization. Traffic visualization is performed using Google Maps v3 API for displaying overlays in HTML. The data is loaded via ajax into the web page which then creates an overlay for each time stamp of the moving object. Figure 3 gives an example for traffic visualization, where the colored circles on the map represent the moving objects. The user can either view the animated object movements or move the handler on the bottom scroll to view the snapshot at a specific time.

C. Adding a Road Network Data Source

The demo attendees can add a new road network data source to MNTG. We have a prepared data set, termed *HighwayRoadNetwork*, which includes only the highways in US. We will show the audience how to register this new road network with MNTG using two abstract functions. Once registered, audience will see that they can submit a new request to MNTG using the new road network. Figure 4 depicts the nodes and edges of the *HighwayRoadNetwork* on a Google Map, which is given to the user as a verification for the new data to import.

D. Traffic Processor Extension

The audience may register a new traffic generator to MNTG. For demonstration purpose, we prepared a new traffic generator, termed *RandomTraffic*, which requires binary format of road networks, runs with a java file, and outputs the traffic result in a binary format. The *RandomTraffic* generator takes its input as the number of moving objects and average speed. Then, it selects random points as one per each moving object. For each random point, we select another random destination, and calculate the shortest path. Given the average speed, objects move along the shortest path till their destinations.

We will show the audience the java template class file for the *RandomTraffic* generator, which includes the implementation of its three abstract functions, `RoadNetworkConvert`, *RandomTraffic*, and `TrafficGeneration`. Then, we will upload this file to MNTG, which indicates that a new traffic generator is registered. Audience will then restart MNTG to see that they can submit a new traffic generator request using the *RandomTraffic* generator.

REFERENCES

[1] T. Brinkhoff. A Framework for Generating Network-based Moving Objects. *GeoInformatica*, 6(2), 2002.

[2] S. Chen, C. S. Jensen, and D. Lin. A Benchmark for Evaluating Moving Object Indexes. *VLDB Journal*, 1(2), 2008.

[3] C. Düntgen, T. Behr, and R. H. Güting. BerlinMOD: a Benchmark for Moving Object Databases. *VLDB Journal*, 18(6), 2009.

[4] R. H. Güting, T. Behr, and C. Düntgen. Secondo: A platform for moving objects database research and for publishing and integrating research implementations. *IEEE Data Engineering Bulletin*, 33(2), 2010.

[5] H. Hu, J. Xu, and D. L. Lee. PAM: An Efficient and Privacy-Aware Monitoring Framework for Continuously Moving Objects. *IEEE TKDE*, 22(3), 2010.

[6] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A Hybrid Prediction Model for Moving Objects. In *ICDE*, 2008.

[7] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, and S. Yackel. MNTG: An Extensible Web-based Traffic Generator. In *SSTD*, 2013.

[8] M. F. Mokbel, X. Xiong, and W. G. Aref. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases. In *SIGMOD*, 2004.

[9] OpenStreetMaps. http://www.openstreetmap.org/.

[10] H.-P. Tsai, D.-N. Yang, and M.-S. Chen. Mining Group Movement Patterns for Tracking Moving Objects Efficiently. *IEEE TKDE*, 23(2), 2011.

[11] US TIGER LINES. http://www.census.gov/geo/maps-data/data/tiger-line.html.

[12] W. Wu, W. Guo, and K.-L. Tan. Distributed Processing of Moving K-Nearest-Neighbor Query on Moving Objects. In *ICDE*, 2007.

[13] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. GeoLife2.0: A Location-Based Social Networking Service. In *MDM*, 2009.