December 29, 2012

RT0896
Operations Research 17 pages

Research Report

IBM Mega Traffic Simulator

T. Osogami, T. Imamichi, H. Mizuta, T. Morimura, R. Raymond, T. Suzumura, R. Takahashi, T. Idé

IBM Research - Tokyo
5-6-52 Toyosu, Koto-ku, Tokyo 135-8511, Japan

IBM Mega Traffic Simulator

Takayuki Osogami	osogami@jp.ibm.com
Takashi Imamichi	imamichi@jp.ibm.com
Hideyuki Mizuta	e28193@jp.ibm.com
Tetsuro Morimura	tetsuro@jp.ibm.com
Rudy Raymond	raymond@jp.ibm.com
Toyotaro Suzumura	toyo@jp.ibm.com
Rikiya Takahashi	rikiya@jp.ibm.com
Tsuyoshi Idé	goodidea@jp.ibm.com

IBM Research - Tokyo*

December 29, 2012

Abstract

IBM Mega Traffic Simulator (Megaffic) is an agent-based simulator of traffic flows with two unique features. First, Megaffic can build its model of simulation by directly estimating some of the parameters of the model from probe-car data. This capability is in contrast to existing agent-based simulators of traffic flows, where the values of their parameters are calibrated with iterative simulation. Second, Megaffic can run on massively parallel computers and simulate the microscopic traffic flows in the scale of an arbitrary city or even the whole Japan. This manuscript gives an overview of the design and capability of Megaffic.

*5-6-52 Toyosu, Koto-ku, Tokyo 135-8511, Japan

1 Introduction

Transportation authorities must take appropriate actions to keep their transportation system functioning. They need to decide which sections of a road should be closed to avoid unacceptable congestion around key facilities such as hospitals and fire stations when they face a natural disaster. They need to plan where to construct new roads to reduce congestion or to accommodate increased traffic flows in future. The transportation authorities can rely on traffic simulation to evaluate the effectiveness of a particular action in a particular situation.

The models of traffic simulation range from microscopic to macroscopic, depending on the level of detail [15, 8]. A microscopic model tracks the location of individual vehicles, while a macroscopic model tracks some features of flows such as speed and density. Microscopic models allow more detailed study and more faithful modeling of transportation systems than their macroscopic counterparts [8].

As a result, there has been a significant amount of effort in building traffic simulators with microscopic models [6, 1]. Nishi et al. report that a minute change in the configuration of merging lanes can significantly reduce congestion [16], but such an impact can only be captured by microscopic models. A macroscopic model does not allow us to directly study the impact of small changes that cannot be represented by that model. Also, we cannot directly study the impact of traffic control in more detail than what the macroscopic model tracks. For example, if we want to evaluate the impact on specific vehicles such as ambulances, a microscopic model would be more appropriate than a macroscopic model.

However, a microscopic model can have considerably more parameters than a corresponding macroscopic model. The values of these parameters, for example, determine how the driver of a vehicle, an agent, chooses the speed, the lane, or the route as well as the origin and the destination. We need to carefully set these values for individual agents, because they essentially determine the results of simulation. Calibrating these values is time-consuming and often relies on intuitions and knowledge of an expert of transportation systems [6, 11, 10]. Due to the difficulty of calibration, some of the details are often omitted from microscopic models. For example, the driver of a vehicle is often assumed to take the route that minimizes the travel time [20], ignoring other features such as travel distance and the number of turns.

In addition, a microscopic model can limit the scalability of traffic simulation [8]. The congestion at different sections of a city can interact with each other, so that a simulation model must include all of the roads and intersections where congestion might occur. This area might spread across the entire city, and analyzing congestion is of critical importance in major developing cities, including Beijing, Moscow, Mexico City, Sao Paolo, and Lagos. The lack of scalability can be another reason to omit some of the details from a microscopic model.

We have been developing IBM Mega Traffic Simulator (Megaffic), a traffic simulator with a microscopic model, with a goal of overcoming these two common limitations of the microscopic model of traffic simulation. We can use Megaffic to simulate an entire city without omitting necessary details from the microscopic model. In this manuscript, we give an overview of the design and capabilities of Megaffic.

Megaffic has a unique feature of being able to directly estimate some of the parameters of the model of traffic simulation from probe-car data. Probe-car data records trajectories of vehicles that are measured with the Global Positioning System (GPS). In particular, Megaffic uses the

probe-car data to set the values of the parameters that determine how the drivers of vehicles select their routes, including their origins and destinations. Existing approaches would calibrate these values by iterating simulation by varying the values of the parameters at each iteration until it finds the values where the output of the simulation matches corresponding observed values such as the number of vehicles that travel through a road segment [6, 11, 4]. Our approach is similar to [10], who advocates the use of more data to reduce the number of variables that need to be calibrated. However, [10] does not propose any specific approach.

The detailed model of route choice significantly increases the computational complexity of traffic simulation. For scalability, we have built X10-based Agents eXecutive Infrastructure for Simulation (XAXIS), a platform that allows massively parallel execution of agent-based simulation [23]. Megaffic, which is built on XAXIS, enables a nearly linear scale-up with respect to the number of cores.

The rest of the manuscript is organized as follows. We start by presenting an overview of the structure and simulation model of Megaffic in Section 2. Details about individual components of Megaffic are presented in the following sections.

2 Overview of Megaffic

Agent-based simulation of traffic flows tracks the location of each agent, representing the driver of a vehicle, who travels, interacting with other agents, according to various models of their behaviors (see Figure 1). Each agent is assigned an origin, a destination, and a departure time according to a model of origin-destination (OD) generation. The simulator creates the agent at the origin at the departure time. The agent chooses a route from the origin to the destination, according to a model of route choice, and travels along that route. The simulator lets the agent travel along that route, during which the travel speed is changed according to a model of speed selection, and the lane is changed according to a model of lane selection. In this section, we give an overview of the structure of Megaffic.

2.1 Input data

Megaffic can build its simulation model from map data, census data, and probe-car data (see Figure 2). The map data have the information about road segments and intersections. The census data give a table, which we refer to as an OD table, whose entry represents the number of trips from a subarea of the map to another during each hour of a day. The probe-car data record the trip histories of multiple drivers of vehicles, where a trip history is a sequence of longitude, latitude, and time observed with the GPS. It is highly recommended that all of these data are prepared for accuracy of simulation. When any of these data are missing, Megaffic can still run by feeding randomly generated data or by setting default values for some of the parameters of the simulation model.

When probe-car data is unavailable from an area that needs to be simulated, we can use the probe-car data from another area. For example, one can use the probe-car data from the greater Tokyo area to determine the parameters in the model of route selection and those of OD generation. These parameters can then be used in simulation of Hiroshima city, which is over

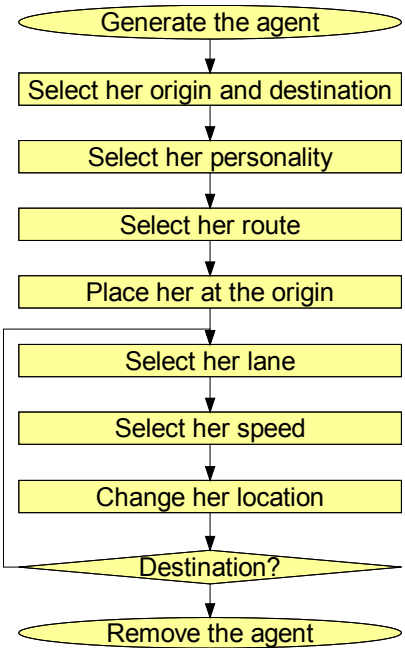

Figure 1: Simplified life of an agent

400 miles away from Tokyo (i.e., outside the greater Tokyo area). An advantage of our approach is that we can transfer the models estimated in one area to simulation in other areas, because these models consist only of abstract elements such as the personality of the drivers of vehicles and the tendency of a type of a landmark to become an origin or a destination.

2.2 Basic modules of Megaffic

The first step of building the simulation model is map-matching. Because the GPS data are prone to error, we use a reliable technique of map-matching with a hidden Markov model [21] to recover the routes that are taken by those drivers of vehicles from the probe-car data. See Section 3 for more details about the map-matching module of Megaffic. In the following, we refer to those data after map-matching as probe-car data.

Megaffic uses the technique of L_1-regularized Poisson regression with adaptive baseline [12] in its model of OD generation for effective utilization of two data sources that are complementary to each other: (i) the census data that have the exact number of trips but at the coarse granularity of subareas and (ii) the probe-car data that have a small number of sampled trips but with the exact information about the locations of their origins and destinations. The two sources of data are integrated based on the information about landmarks, including hotels and railway stations, that are available in the map data.

The technique of [12] gives the probability that a pair of intersections becomes an origin and a destination, respectively, for each hour of a day. The model of OD generation creates agents together with their origins, destinations, and departure hours according to this probability. Exact departure time can be determined uniformly at random within the departure hour. See Section 4 for more details about what the OD-estimation module does.

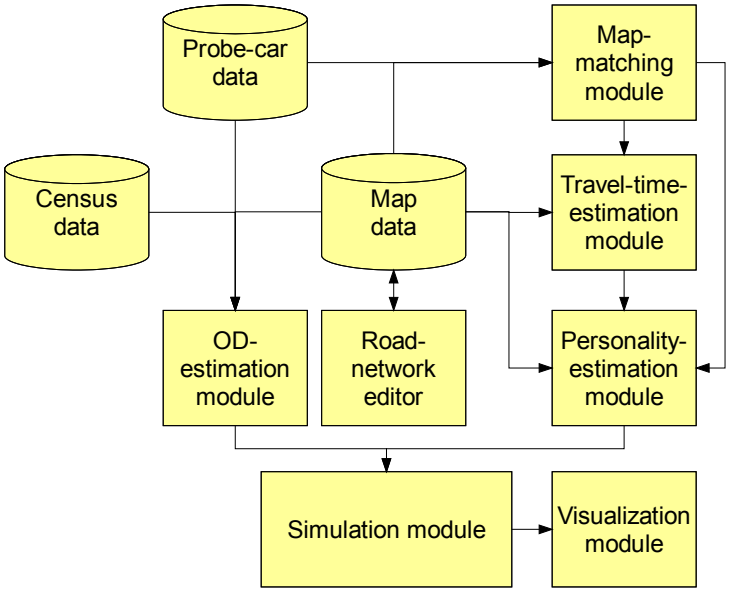

Figure 2: Data flow.

Megaffic's model of route choice determines the route of an agent from his origin to his destination, taking into account three quantities: travel time, travel distance, and the number of turns. Specifically, the route that minimizes the weighted sum of the three quantities is selected, where the weight depends on individual agents. We refer to the weight used for an agent as the agent's personality.

Megaffic can estimate the personality of a driver whose trip is recorded in the probe-car data. For each driver whose trip history is recorded in the probe-car data, the personality-estimation module of Megaffic estimates the personality of that driver (specifically, the driver's perceived importance of the three metrics: travel time, distance, the number of turns). Our algorithm efficiently compares a selected route against combinatorially many routes that have not been selected to give the best estimate on the personality. See Section 5 for more details about the personality-estimation module.

To generalize the limited set of observed personality to the personality of all of the agents in the simulator, we fit the set of observed personality to a mixture of Dirichlet distributions. Namely, the personality of an individual agent is generated probabilistically according to the fitted mixture of Dirichlet distributions.

Once the personality of an agent is determined, Dijkstra's algorithm [3] can be used to find the route, from his origin to his destination, that minimizes the convex combination of the three quantities under consideration, where his personality is used as the weight in the convex combination. To take into account the number of turns, Dijkstra's algorithm is run on a network whose vertex represents a segment of road and whose edge represents a connection from a segment of road, which we refer to as the first road-segment, to a neighboring segment of road, which we refer to as the second road-segment. The edge cost then represents the convex combination of the travel time along the second road-segment, the travel distance along the second road-segment, and the indicator (zero or one) of whether there is a turn from the first road-segment to the

second. The travel time used to determine the route is updated periodically (e.g., every ten simulated minutes), depending on the simulated travel time.

While an agent travels along his route, he adjusts his speed, depending for example on the distance to the preceding vehicle. We use Gipps' car-following model as our model of speed-selection [5]. An agent also changes his lanes, depending for example on the turn that he is going to make next and on the space available in the neighboring lane. We use the model of lane-selection proposed by Toledo et al. [25]. The default values of the parameters in the model of speed-selection and those of lane-selection are set as recommended in [5, 25]. See Section 6 for details about how Megaffic manages the movement of vehicles.

Megaffic is built on top of XAXIS for enabling massively parallel execution of simulation. Megaffic and XAXIS are written with Java and X10 languages and runs on any platform that supports these two languages. The computational resources that are required by Megaffic depend on the scale of simulation. Megaffic has been tested on a wide range of machines from a standard workstation to a supercomputer with 128 computing nodes (TSUBAME 2.0 at the GSIC Center of the Tokyo Institute of Technology). See Section 7 for more details about XAXIS.

2.3 Other capabilities of Megaffic

The model of Megaffic can be further extended to incorporate more detailed behavior of the drivers of vehicles. For example, the drivers of vehicles have different sensitivity to the risk of delay [26]. Our model of route selection can be extended in such a way that the drivers of vehicles select different routes depending on the sensitivity to the risk of delay, where the sensitivity is estimated from probe-car data in the personality estimation module.

Travel time distribution, which will be needed for such estimation, can be fitted by the use of the travel-time-estimation module of Megaffic, which gives the probability density function of the travel time along each road segment based on the realized travel times that can be inferred from the probe-car data. Our sophisticated interpolation engine allows us to reliably estimate the probability density function even for the road segments that have only a few samples of observed travel times. See Section 8 for more information about what the travel-time estimation module does.

Megaffic also provides tools for editing its input and output to help what-if analysis. A decision maker would want to run simulation with varying map data to see how the results change. The road-network editor of Megaffic is a graphical interface that allows us to easily add/delete a road or modify the properties of a road. The visualization module of Megaffic can convert the log file of Megaffic into the KML (Keyhole Markup Language) format, so that external GIS tools such as Google Earth and IBM Intelligent Operations Center can be used to visualize the results of the simulation.

3 Map-matching module

It is getting increasingly popular for cars to provide their trajectory data to receive services of high quality, including car navigation based on the latest traffic conditions. Because the trajectory data is usually obtained with GPS, it is sampled at discrete epochs and prone to measurement

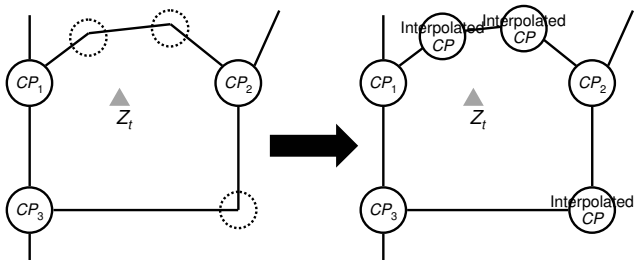

Figure 3: Representing road network data by CPs and interpolated CPs

errors. The first step of pattern recognition from such trajectory data is map-matching that finds the most likely route given a sequence of GPS data points.

This map-matching has turned out to be surprisingly difficult even with moderate sampling rate and moderate error [14]. A difficulty of map matching stems from the tradeoff between the route that traverses the places that are close to the GPS points and the route that is likely to be traveled by drivers. A naive approach of connecting the points on roads that are respectively closest to given GPS data points often results in a winding route that is very unlikely to be traveled. The map-matching module of Megaffic is built with a new map-matching algorithm that is based on a hidden Markov model (HMM).

3.1 Map-matching problem

Formally, the input and output of map-matching is as follow:

Input: A sequence of T GPS observation points, $Z = (Z_t \mid t = 1, 2, \ldots, T)$, where each Z_t is composed of latitude and longitude of the moving object at time t, and a directed road network $G = (V, A)$, such that $V = (\mathrm{cp}_i \mid i = 1, 2, \ldots, N)$ is a collection of nodes that correspond to cross-points (or, CPs) on the road network, and $A = (\mathrm{r}_j \mid j = 1, 2, \ldots, M)$ is a collection of arcs (directed edges) that correspond to road segments on the road network.

Output: A connected sequence of arcs $R = (r_i \mid i = 1, 2, \ldots, T')$, where $r_i \in A$, that denotes the roads that are matched to the observed points.

The task of map-matching is to find the most likely sequence of CPs on the roads that were traversed by the moving objects among possible sequences. The accuracy of map-matching depends on the granularity of road segments in the road network. The road segments are represented by interpolated CPs as illustrated in Fig. 3, where, for example, the road between CP_1 and CP_2 is represented by interpolated CPs that are useful for representing road curves. For simplicity, we call all points as cross-points (or, CPs) although *the interpolated ones are not intersections of multiple roads*. Many map data, such as the OpenStreetMap[1], represent roads as sequences of CPs.

[1] www.openstreetmap.org

3.2 Hidden Markov model for map-matching

In an HMM for map-matching, the moving object is modeled to move according to a Markov process between CPs on the road segments. These road segments are not directly observed, and are considered as hidden states, but the output of the hidden state, namely the GPS coordinates, are. The distribution of the observed GPS coordinates depends only on the hidden state. The the basic components of the HMM include the emission probability, the state transition probability, and the initial state probability.

The emission probability is denoted as $\Pr(Z_t|\mathrm{CP}_t = \mathrm{cp}_i)$, which is the probability of observing Z_t given that the (hidden) position of moving object on the road network at time t is cp_i. Following [14], the emission probability is modeled to follow a Gaussian noise centered at cp_i, and thus

$$\Pr(Z_t \mid \mathrm{CP}_t = \mathrm{cp}_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{\|Z_t - \mathrm{cp}_i\|_{\mathrm{gc}}}{\sqrt{2}\sigma}\right)^2,$$

where $\|Z_t - \mathrm{cp}_i\|_{\mathrm{gc}}$ denotes the great circle distance on the surface of the earth between the true location of the moving object and the observed point. The parameter σ is the standard deviation of the GPS measurement.

The state transition probability gives a conditional probability of the object to move from a cross-point to another cross-point on the road network. We propose to set the probabilities to be proportional to the distance between the CPs on the road network, as follow.

$$\Pr\left(\mathrm{CP}_{t+1} = \mathrm{cp}_j \mid \mathrm{CP}_t = \mathrm{cp}_i\right) \propto \exp(-\beta d_{ij}),$$

where β is a parameter to control the effect of the shortest

The initial state probability gives the likelihood of the initial CP of the moving object. The probabilities are approximated from the emission probabilities with regard to the first GPS observation, i.e., $\Pr(Z_1 \mid \mathrm{CP}_1 = \mathrm{cp}_i)$.

Given these probabilities, the most likely sequence of CPs can be computed from the HMM using the Viterbi algorithm, which is a popular method in speech and text recognition tasks. For each GPS point, we first find all nearby CPs (including the interpolated ones) nearby by range query, and compute emission probabilities for each pair of the GPS point and the nearby CPs. This can be done efficiently by storing all CPs on the road network with the *k-d tree* or nearest neighbor data structures. We then apply the Viterbi algorithm on the GPS points and CPs. See [21] for more details about our map-matching technique.

4 OD-estimation module

Conventional approaches for generating an origin and a destination (OD) of traffic rely on OD tables that are constructed based on surveys. An entry of an OD table gives the number of trips from a zone to another, but the exact locations of the origins and the destinations of those trips are usually unknown. There also exist approaches for microscopic OD estimation from probe-car data, but such approaches can be applied only to the area where probe-car data is available.

The OD-estimation module of Megaffic is built with a new microscopic ODs estimation method that takes advantage of multiple sources, including probe-car data, census data, and

landmark data from a map. Here the landmark data includes the locations of various types of objects such as hotels and railway stations. The new OD estimation method constructs a prior distribution based on the census data, uses L_1-regularized Poisson regression for propagating the sparse probe-car data based on the landmark data, and finally outputs a set of the ODs. The use of the landmark data is the key to overcome the difficulty of OD-estimation that stems from the sparsity of probe-car data.

4.1 Traffic OD estimation problem

The traffic OD estimation problem is to inference a joint probability function over the locations of the origin and the destination of a trip. We seek to generate a set of ODs according to this joint probability function. Here we generate an origin or a destination on a cross-point, *i.e.*, a node of a road-network graph[2].

Each cross-point is associated with the information about the frequency of being the origin or the destination of a trip and the information about nearby landmarks. Let $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ be the set of cross-points, where $|\mathcal{C}|$ denotes the total number of cross-points in a graph. Each cross-point, c_n, is associated with $o_n \in \mathbb{N}$, the number of origins, and $d_m^n \in \mathbb{N}$, that of destinations. Specifically, o_n is the number of trips in the probe-car data that depart from c_n; d_m^n is the corresponding number of trips that depart from c_m and terminate at c_n. Each cross-point, c_n, also has information about landmarks, $\boldsymbol{l}_n \in \mathbb{R}^{|\mathcal{L}|}$, where $|\mathcal{L}|$ is the number of the types of landmarks. The element i of $\boldsymbol{l}_n$ denotes the number of type-i landmarks that are close to c_n.

We assume that an OD table at the level of zones is available. Such an OD table can be constructed from person-trip counts of census data. The (i, j)-th entry of the OD table denotes the number of trips from the i-th zone, z_i, to the j-th zone, z_j. The zone is defined in the census data and typically contains about a hundred cross-points.

4.2 L_1-regularized Poisson regression

The L_1-regularized Poisson regression that we use for OD estimation is an extension of Poisson regression. Poisson regression models the number of origins (or destinations) from a cross-point with a Poisson distribution, which has a parameter, μ, that represents the mean and the standard deviation. In Poisson regression, μ is represented with a log-linear model as $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}) = \exp(\boldsymbol{\theta}^\top \boldsymbol{x})$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of d parameters, $x \in \mathbb{R}^d$ is a vector of d features, and $\top$ denotes the transposition.

The objective of L_1-regularized Poisson regression is to find the parameter that minimizes the following objective function [7]:

$$J(\boldsymbol{\theta}) = \sum_k - \log p(n_k \,|\, \boldsymbol{x}_k, \boldsymbol{\theta}) + \lambda \, |\boldsymbol{\theta}|_1 \tag{1}$$

where

$$p(n \,|\, \boldsymbol{x}, \boldsymbol{\theta}) \equiv \frac{\left(\exp(\boldsymbol{\theta}^\top \boldsymbol{x})\right)^n \, \exp\left(-(\exp(\boldsymbol{\theta}^\top \boldsymbol{x})\right)}{n!},$$

[2]Our approach can be applied to the case where we generate ODs is on road segments, *i.e.*, links of a road-network graph.

λ (≥ 0) is a regularization parameter, and $|\cdot|_1$ denotes the L_1-norm. This minimization problem can be solved iteratively with the quadratic Taylor series expansion around the current estimate of $|\boldsymbol{\theta}|$ and the Lasso regression [9, 7].

For OD estimation, we use landmark information, $\boldsymbol{l}_i$, as the feature vector of an origin, c_i. The feature vector for a destination is $\boldsymbol{l}_i^j \equiv [\sqrt{\mathrm{vec}(\boldsymbol{l}_i \otimes \boldsymbol{l}_j)}^\top, \boldsymbol{l}_j^\top]^\top$, where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix, and $\otimes$ denotes the outer product operator.

Here we also include additional features, which we refer to as adaptive baselines, in the Poisson regression to reduce the effect of the bias that the distribution of the ODs from the probe-car data might have. We define the adaptive baselines on the basis of the assumption that the samples in the probe-car data is uniform distributed in a zone. Specifically, for the origin, the adaptive baseline of a cross-point, c, in a zone, z_i, is given as the logarithm of the conditional expected number, n_o, of origins from c:

$$b(c) = \log(\mathbb{E}[n_\mathrm{o} \mid z_\mathrm{origin} = z_i]),$$

where $\mathbb{E}$ denotes the expectation operator, and z_origin is the random variable representing the zone of a generic origin. For the destination, the adaptive baseline of a pair of cross-points, c in a zone z_i and c' in a zone z_j, is analogously given by

$$b(c, c') = \log(\mathbb{E}[n_\mathrm{d} \mid z_\mathrm{origin} = z_i, z_\mathrm{destination} = z_j]),$$

where n_d denotes the number of destinations to c'.

These adaptive baselines have corresponding parameters, θ_o^b and θ_d^b. The resulting vector of features for the origin is $\tilde{\boldsymbol{l}}_i = [b(c_i), \boldsymbol{l}_i^\top]^\top$, and the corresponding vector of parameters is $\tilde{\boldsymbol{\theta}}_\mathrm{o} = [\theta_\mathrm{o}^b, \boldsymbol{\theta}_\mathrm{o}^\top]^\top$. The corresponding vectors for the destination are defined analogously. This adaptive baseline for the origin can be understood by the observation that, if we set $\theta_\mathrm{o}^b = 1$ and $\boldsymbol{\theta}_\mathrm{o} = \mathbf{0}$, the mean (and standard deviation) of the estimated Poisson distribution, $\mu_{\tilde{\boldsymbol{\theta}}_\mathrm{o}}(c_i) = \exp(\tilde{\boldsymbol{\theta}}_\mathrm{o}{}^\top \tilde{\boldsymbol{l}}_i)$, is equal to $\mathbb{E}[n_\mathrm{o} | z_\mathrm{origin} = z_i]$. That is, θ_b is expected to center around 1, and $\boldsymbol{\theta}_\mathrm{o}$ around $\mathbf{0}$. Our objective function can hence be derived from Equation (1) as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{|\mathcal{C}|} -\log p(o_i \mid \tilde{l}_i, \tilde{\boldsymbol{\theta}}_\mathrm{o}) + \lambda(|\theta_b - 1| + |\boldsymbol{\theta}_\mathrm{o}|_1).$$

The optimization problem of minimizing our objective function remains convex and thus can be solved by the use of an approach for solving Equation (1).

To generate samples of ODs, we use an OD table at the zone level as prior information. First, a pair of zones for an origin and a destination is chosen according to the OD table. Then an origin is selected from the chosen origin zone according to the learned Poisson distribution, $p(n_\mathrm{o} | \tilde{\boldsymbol{l}}_i, \tilde{\boldsymbol{\theta}}_\mathrm{o})$. Namely, a cross-point c_i is selected as the origin with probability $\propto \exp(\tilde{\boldsymbol{\theta}}_\mathrm{o}^\top \tilde{\boldsymbol{l}}_i)$. A destination is then selected analogously, according to $p(n_\mathrm{d} | \tilde{\boldsymbol{l}}_i^j, \tilde{\boldsymbol{\theta}}_\mathrm{d})$. Noted that our approach uses probe-car data only in the phase of training our model. Once the model is trained, we can generate ODs without probe-car data. See [12] for more information about our technique of OD estimation.

5 Personality-estimation module

The trajectories of humans are often approximated with simple stochastic processes, including random walks, Levy flights and diffusion processes. Although such stochastic processes might well approximate macroscopic mobility patterns of humans or other animals, they do not account for the particular choice of trajectories or routes that depends on individual human's preference or tendency in decision making.

Understanding the driver's selection of routes is crucial for planning, forecasting, and controlling traffic systems. There has been a significant amount of research toward uncovering the drivers' preferences for various characteristics of routes, including travel time, risk of delay, and cost. An important limitation of such prior work is in the considered set of possible alternatives. In the prior work based on discrete choice models of route selection, an arbitrary set of representative routes is first identified from combinatorially many and potentially infinite alternatives. It is then examined which routes are chosen over the others within the identified set of the representative routes. The number of the representative routes must be small so that the preferences of drivers can be estimated.

The personality-estimation module of Megaffic studies what distinguishes the trajectory selected by an individual driver from all of the other possible alternatives. Each trajectory, p, is examined in terms of the following four quantities:

$C_{\mathrm{N}}(p)$**:** magnitude of turns,

$C_{\mathrm{T}}(p)$**:** the nominal travel time based on speed limit,

$C_{\mathrm{D}}(p)$**:** the travel distance, and

$C_{\gamma}(p)$**:** the value of the entropic risk measure of the travel time, where gamma is the parameter of the entropic risk measure and represents the sensitivity to the risk.

The latest implementation of Megaffic handles these four quantities, but one might want to include other quantities such as cost.

We assume that each trajectory, p, is selected in such a way that the convex combination of the four quantities relative to their minimum possible values over all of the possible trajectories,

$$\alpha_{\mathrm{N}}\frac{C_{\mathrm{N}}(p)}{\underline{C}_{\mathrm{N}}}+\alpha_{\mathrm{T}}\frac{C_{\mathrm{T}}(p)}{\underline{C}_{\mathrm{T}}}+\alpha_{\mathrm{D}}\frac{C_{\mathrm{D}}(p)}{\underline{C}_{\mathrm{D}}}+\alpha_{\mathrm{R}}\frac{C_{\gamma}(p)}{\underline{C}_{\gamma}}, \tag{2}$$

is minimized, where $\underline{C}_{\mathrm{N}}$ denotes the minimum value of $C_{\mathrm{N}}(q)$ for all possible trajectories, q, from the first link of p to the last link of p; $\underline{C}_{\mathrm{T}}$, $\underline{C}_{\mathrm{D}}$ and $\underline{C}_{\gamma}$ are defined analogously. We then seek to find the values of $(\alpha_{\mathrm{N}},\alpha_{\mathrm{T}},\alpha_{\mathrm{D}},\alpha_{\mathrm{R}},\gamma)$ that best explains a selected trajectory, p.

The values of $(\alpha_{\mathrm{N}},\alpha_{\mathrm{T}},\alpha_{\mathrm{D}},\alpha_{\mathrm{R}},\gamma)$ are determined for each of the trajectories recorded in the probe-car data. The simulation module of Megaffic uses this empirical distribution of $(\alpha_{\mathrm{N}},\alpha_{\mathrm{T}},\alpha_{\mathrm{D}},\alpha_{\mathrm{R}},\gamma)$ to generate a sample value of $(\alpha_{\mathrm{N}},\alpha_{\mathrm{T}},\alpha_{\mathrm{D}},\alpha_{\mathrm{R}},\gamma)$, which is assigned to an agent. The route of the agent is then determined in such a way that Equation (2) is minimized.

6 Simulation

In this section, we explain the details about how agents flow in Megaffic. Megaffic primarily manages three types of objects: *vehicle*, *road*, and *cross point*. As we described in Section 2, a *vehicle* object is an agent that is given an origin and a destination and travels along a route connecting the origin and destination. A *road* object manages *vehicles* on a road segment by a sorted queue based on the position of the vehicles. When a road segment has multiple lanes, those lanes are handled as different *road* objects. These *road* objects have pointers to neighboring lanes so that vehicles can change the lanes following the pointers. We define a *cross point* as an end point of a road segment, i.e., an intersection. A *cross point* has queues of vehicles in order to transfer vehicles from incoming road segments to outgoing road segments.

The *cross point* keeps track of the positions of vehicles every simulated second. This process of changing positions alternates between two phases. The first phase changes the positions of the vehicles along the incoming roads for each *cross point*, and the second phase places the vehicles that reached the end points of the incoming road segments to outgoing road segments. Megaffic executes these processes of different *cross points* in parallel with synchronization such that a new phase starts after the previous phase completes for all *cross points*.

More specifically, in the first phase, Megaffic updates the speeds, the lanes, and the positions of vehicles along the incoming roads of each *cross point* based on drivers' models. In addition, the first phase handles the new vehicles at their departure time. These new vehicles, however, only choose their routes in the first phase and do not enter the road segments to keep the consistency of parallelization. In the second phase, Megaffic places the new vehicles, as well as the ones that reached the end points of road segments in the first phase, along its outgoing road segments if these outgoing road segments have remaining room.

7 XAXIS

X10-based Agent eXecutive Infrastructure for Simulation (XAXIS) is a platform for a massive agent-based simulation and is written with a parallel programming language, X10 [22]. In XAXIS, an agent manager manages agents that are asynchronously executed by the activity of X10 in each place. An agent in a place can communicate with an agent in another place by sending a communication message. This communication message is hidden from users. Developers of simulators can thus concentrate on mounting the logic of the agent. A message that is developed using X10 can be transmitted and received in XAXIS by calling the method of the agent manager in the place of a receiver.

In our latest implementation, the activity of X10 is mapped to a *cross point* of Megaffic. An alternative design would be to map the activity of X10 to a *vehicle*. See Section 6 for the functions of a *cross point* and a *vehicle*. See [23] for more information about XAXIS.

8 Travel-time estimation module

Drivers are heterogeneous in that they tend to take different routes from a common origin to a common destination. The effectiveness of traffic simulation relies on how well we can model such

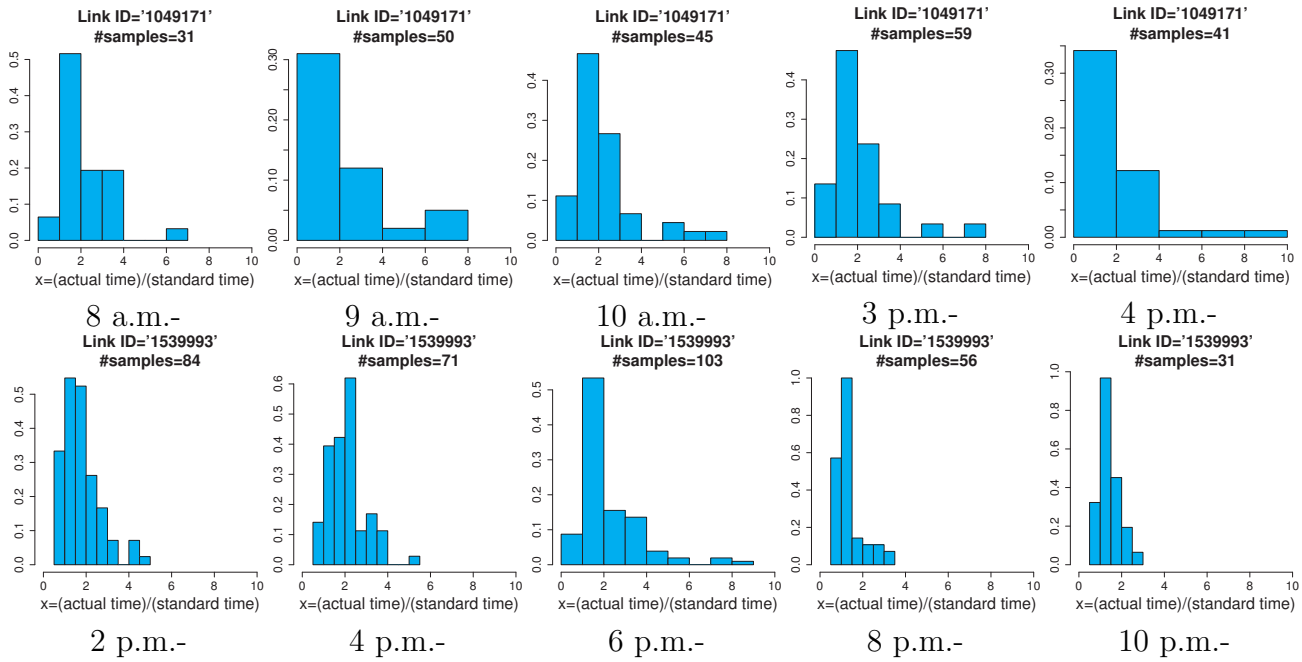

Figure 4: Histograms of travel time relative to the standard time based on speed limit (Figure 2 from [24]).

drivers' heterogeneity. The prior work suggests that the routes selected by drivers are sensitive to the distribution of travel time, and the sensitivity depends on particular drivers.

For realistic simulation, it is thus essential to estimate the distribution of the travel time for every link in E, the set of all of the links in a given road network, with high accuracy. Stochastic optimization can then find the optimal route that minimizes the value of a risk measure that corresponds to a particular driver's sensitivity to risk. Minimizing risk measures results in diversified choices of routes, because the risk measures and their parameters vary among drivers.

A limitation in existing approaches for fitting travel-time distributions is a parametric assumption. Many regression methods use the Ordinary Least Squares principle to estimate the mean of travel time or its logarithm and fit the travel-time into a Gaussian or log-normal distribution [2]. We find that the actual travel-time often has a small probability of being very long, resulting in heavy-tailed or multimodal distributions (See Figure 4).

8.1 Travel-time estimation problem

Our nonparametric estimator gives a travel-time distribution for each link, including those where travel-time samples are missing, based on the assumption that similar links have similar travel-time distributions. Formally, let Y_e be the random variable that represents the travel time along a link, $e \in E$, relative to $\tau_e^{(0)}$, the corresponding nominal travel time defined with the speed limit. Our goal is to estimate the probability density function, $f_e(\cdot)$, of Y_e for every $e \in E$, given the set of training samples. After we fit $f_e(\cdot)$, the probability density function of the (absolute) travel time, $X_e \triangleq \tau_e^{(0)} Y_e$, can be computed by rescaling $f_e(\cdot)$. Let us index all of the edges by $e_1, e_2, \ldots, e_{|E|}$. Let $\pi[1], \ldots, \pi[m]$ be m indices of edges where the travel times have been observed

several times or more.

We model the distribution of the relative travel-time with a nonparametric form

$$f_e(y) = \frac{\lambda_0\,\varphi_0(y) + \sum_{i=1}^{m} \lambda_i\, K(e, e_{\pi[i]})\,\varphi_i(y)}{\lambda_0 + \sum_{i=1}^{m} \lambda_i\, K(e, e_{\pi[i]})}, \tag{3}$$

where $\Phi \triangleq \{\varphi_0, \varphi_1, \ldots, \varphi_m\}$ is a set of basic density functions, $K(e, e_{\pi[i]})$ is a similarity function between e and $e_{\pi[i]}$, and $\boldsymbol{\lambda} \triangleq (\lambda_0, \lambda_1, \ldots, \lambda_m)^\top$ is a vector of link importance. A basis density function $\varphi_i(\cdot)$ needs to satisfy $\int_0^\infty \varphi_i(y)dy \equiv 1$, and the link similarity function $K(\cdot,\cdot)$ must be non-negative. Equation (3) resembles the nonparametric Nadaraya-Watson kernel regression [13, 27] except that we add link-independent weight and basic density function, λ_0 and $\varphi_0(\cdot)$. The terms λ_0 and $\varphi_0(\cdot)$ are introduced for defining $f_e(y)$ even when $\forall i \in \{1, \ldots, m\}, K(e, e_{\pi[i]}) \equiv 0$.

8.2 Nonparametric Conditional Density Estimator

First, we fit each basic density function with a mixture of gamma or log-normal distributions based on the observed travel time along the corresponding link. In fitting the basic density functions, we use convex clustering that is guaranteed to converge to the global optima and is accelerated with the Sequential Minimal Optimization (SMO).

The $f_e(\cdot)$ for the remaining links are then given by Equation (3). To optimize the importance weights, $\boldsymbol{\lambda}$ in Equation (3), we exploit a convex optimization algorithm called the Kullback-Leibler Importance Estimation Procedure (KLIEP) that can also be accelerated with SMO. The similarity metric, $K(\cdot,\cdot)$, is computed with a diffusion kernel on a graph, which can incorporate similarity for pairs of links that are not directly connected. Since the number of links is typically huge in a road network, the diffusion kernel matrix needs to be sparse. To guarantee computational feasibility and using the fact that the traffic on a link only affects nearby links, we approximate the matrix exponential in the diffusion kernel with a power of the matrix.

Our algorithm converges quickly to a globally optimal estimator even with large probe-car datasets. More information about the travel-time estimation module of Megaffic can be found in [24].

9 Concluding remarks

Megaffic is still being extended to meet the needs of its users. The contents of this report are thus subject to change in future versions of Megaffic. For example, the model of route selection can be made more sophisticated by incorporating a rich model of sequential decision making under risk or uncertainty, which is discussed in [18, 19, 17].

What is essential to Megaffic is a couple of unique design principles, which will stay unchanged. First, Megaffic can use probe-car data to directly estimate some of its parameters without regard to the standard approach of calibration, which is known to be quite difficult for microscopic model of traffic simulation. Second, Megaffic allows massively parallel execution. Megaffic is built on top of XAXIS, an infrastructure for agent-based simulation built with a parallel programming language, X10.

There are cases where it is difficult to obtain a sufficient volume of probe-car data. The probe-car data might be available from nearby cities, where the behavior of the drivers of vehicles is close to that in the city whose traffic follows need to be simulated. Then Megaffic can use that probe-car data to build some of the models of simulation. Otherwise, we can use classical techniques of building the models of simulation. For example, the parameters of those models can be calibrated [6, 11, 4]. An advantage of Megaffic in the latter cases would be its scalability.

Megaffic is a research asset of IBM Research - Tokyo and can be licensed, subject to a license agreement. Contact the asset owner, Tsuyoshi Ide, for licensing.

Acknowledgments

The authors thank S. Kato and S. Suzuki for their help in Megaffic development. In particular, S. Kato designed and implemented much of the simulation module of a previous version of Megaffic. S. Suzuki helped implement the personality-estimation module.

This work was supported by "Promotion program for Reducing global Environmental loaD through ICT innovation (PREDICT)" of the Ministry of Internal Affairs and Communications, Japan and by Japan Science and Technology Agency, CREST.

References

[1] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - simulation of urban mobility: An overview. In *Proceedings of the Third International Conference on Advances in System Simulation*, pages 55–60, 2011.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] G. Flötteröd, Y. Chen, and K. Nagel. Behavioral calibration and analysis of a large-scale travel microsimulation. *Networks and Spatial Economics*, to appear.

[5] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[6] G. Gomes, A. May, and R. Horowitz. Congested freeway microsimulation model using VISSIM. *Transportation Research Record: Journal of the Transportation Research Board*, 1876:71–81, 2004.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[8] K. Jeannotte, A. Chandra, V. Alexiadis, and A. Skabardonis. Traffic analysis toolbox - volume ii: Decision support methodology for selecting traffic analysis tools. Technical Report FHWA-HRT-04-039, Federal Highways Administration, 2004.

[9] R. C. Kelly, M. A. Smith, and R. E. Kass. Accounting for network effects in neuronal responses using l_1 regularized point process models. In *Advances in Neural Information Processing Systems*, volume 22, pages 1099–1107, 2010.

[10] H. Liu, Q. Yu, W. Ding, D. Ni, H. Wang, and S. Shannon. Feasibility study for automatic calibration of transportation simulation models. In *Proceedings of the 44th Annual Simulation Symposium*, pages 87–94, 2011.

[11] N. E. Lownes and R. B. Machemehl. VISSIM: A multi-parameter sensitivity analysis. In *Proceedings of the 2006 Winter Simulation Conference*, pages 1406–1413, 2006.

[12] T. Morimura and S. Kato. Statistical origin-destination generation with multiple sources. In *Proceedings of the 21st International Conference on Pattern Recognition*, 2012.

[13] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9(1):141–142, 1964.

[14] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *ACM GIS*, pages 336–343, 2009.

[15] D. Ni. A spectrum of traffic flow modeling at multiple scales. In *Proceedings of 2010 Winter Simulation Conference*, pages 554–556, 2010.

[16] R. Nishi, H. Miki, A. Tomoeda, and K. Nishinari. Achievement of alternative configurations of vehicles on multiple lanes. *Physical Review E*, 79(6):066119, 2009.

[17] T. Osogami. Iterated risk measures for risk-sensitive Markov decision processes with discounted cost. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 567–574, July 2011.

[18] T. Osogami. Robustness and risk-sensitivity in markov decision processes. In *Advances in Neural Information Processing Systems*, December 2012.

[19] T. Osogami and T. Morimura. Time-consistency of optimization problems. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12)*, pages 1945–1951, July 2012.

[20] B. Raney and K. Nagel. An improved framework for large-scale multi-agent simulations of travel behavior. In P. Rietveld, B. Jourquin, and K. Westin, editors, *Towards Better Performing European Transportation Systems*, pages 305–347. Routledge, London, 2006.

[21] R. Raymond, T. Morimura, T. Osogami, and N. Hirosue. Map matching with hidden markov model on sampled road network. In *Proceedings of the 21st International Conference on Pattern Recognition*, 2012.

[22] V. A. Saraswat, V. Sarkar, and C. von Praun. X10: Concurrent programming for modern architectures. In *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '07)*, pages 271–271, 2007.

[23] T. Suzumura, S. Kato, T. Imamichi, M. Takeuchi, H. Kanezashi, T. Idé, and T. Onodera. X10-based massive parallel large-scale traffic flow simulation. In *Proceedings of the 2012 ACM SIGPLAN X10 Workshop*, page Article No. 3, 2012.

[24] R. Takahashi, T. Osogami, and T. Morimura. Large-scale nonparametric estimation of vehicle travel time distributions. In *Proceedings of the 12th SIAM International Conference on Data Mining*, pages 12–23, 2012.

[25] T. Toledo, M. Ben-Akiva, and H. N. Koutsopoulos. Modeling integrated lane-changing behavior. *Transportation Research Record*, 1857:30–38, 2003.

[26] P. Ulleberg and T. Rundmo. Personality, attitudes and risk perception as predictors of risky driving behavior among young drivers. *Safety Science*, 41(5):427–443, 2003.

[27] G. S. Watson. Smooth regression analysis. *Sankhya Ser. A*, 26:359–372, 1964.