TOWARDS LARGE SCALE ROAD TRAFFIC SIMULATION EXPERIMENT

Marek Małowidzki, Tomasz Dalecki, Przemysław Bereziński, Michał Mazur

Military Communication Institute
Zegrze, Poland

{m.malowidzki,t.dalecki,p.berezinski,m.mazur}@wil.waw.pl

ABSTRACT

The Insigma project goals include traffic optimization and control. It is assumed that advanced functions including traffic monitoring and prediction, route planning, and, finally, traffic optimization and control, will be able to utilize the available road infrastructure in an optimal way in order to minimize traffic jams and related social and environmental losses. The efficiency of these mechanisms will be evaluated through large-scale simulation experiments. However, before such experiments become feasible, an appropriate simulation environment must be prepared. In the paper, we discuss the application of SUMO as a simulation environment and its integration with Insigma's traffic control layer.

Keywords: road traffic, routing, simulation, traffic simulator, SUMO, Open Street Map (OSM).

1. INTRODUCTION

The Insigma project is aiming at the development of an intelligent information system for global monitoring, detection and identification of threats. The system collects data from various kinds of sensors, cameras, and users, and processes the data to identify threats and notify appropriate public services. One of Insigma's tasks is road traffic optimization and control, which includes traffic lights, information boards, and route planning.

Insigma's goals with respect to traffic control are ambitious. On one hand, they contain a number of features related to public security (support for emergency services and special vehicles, collecting and reporting events, etc.); on the other hand, one of goals is traffic optimization and control. It is assumed that all control mechanisms will be based either on dynamic traffic data, collected and updated in real time, or on traffic forecast.

The first obvious problem is the (feasible and right) approach to evaluation of designed and implemented mechanisms. It seems that large-scale experiments are only possible in a simulated environment. Such an environment should include a road traffic simulator integrated with a control layer, responsible for traffic management. In the paper, we discuss our work on integration the SUMO simulator with the routing service we have already developed (Małowidzki et al. 2012, 2013a, 2013b).

The paper is organized as follows: First, we discuss the traffic subsystem in Insigma and the routing service's architecture and functions. Then, we overview SUMO and its key ideas related to performing simulations. Next, we comment on how we have integrated SUMO with the routing service, include an example, and describe our experience. We propose simulation scenarios. Finally, we overview related work and end the paper with summary.

2. THE INSIGMA'S TRAFFIC SUBSYSTEM

The Insigma's ultimate (and somewhat ambitious) vision is presented in Figure 1. Insigma's goals with respect to road traffic include traffic control, traffic prediction, route planning, and related functions. Despite the fact that ongoing work includes real-world equipment (advanced cameras located at crossroads, collecting detailed data about observed traffic license plate recognition software (Janowski et al. 2012), etc.), the only way to verify control algorithms is simulation. (Drivers would not be pleased finding themselves to be beta testers of our ideas.) Thus, we integrate the SUMO road traffic simulator with the control layer in order to prepare a complete simulation environment for traffic measurements and control.

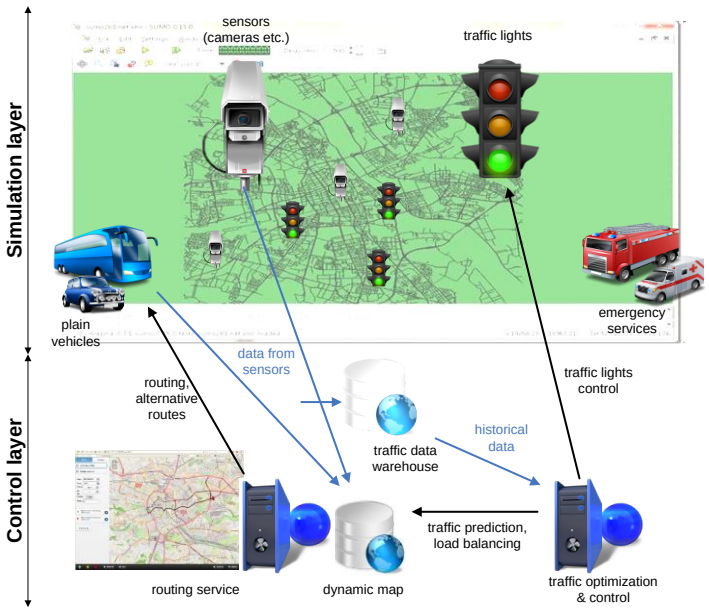

Figure 1. Insigma's traffic subsystem architecture: control signals (arrows pointing up or left) and data flows (arrows pointing down or right)

The control layer consists of a number of components. Some of them have already been implemented (the routing service, the static and dynamic maps), some are under development (traffic data warehouse), some remain to be designed and developed (load balancing, traffic optimization and control). At present, the control layer is represented by the routing

service (and the maps it utilizes), which is the main integration target; this is the reason we focus on the routing service in our paper.

There are a number of such services commercially available and successful on the market (with Google Maps as a premier example), however, specific functions that the service was to support as well as planned integration with higher-level traffic control algorithms in practice required implementing a new one from scratch.

In the following section, we discuss the routing service's architecture and key ideas.

3. THE ROUTING SERVICE

The routing service's internal architecture is shown in Figure 2. The architecture has been discussed in detail in our previous work (Małowidzki at al. 2012) but, for the completeness of the discussion, we briefly summarize it here.

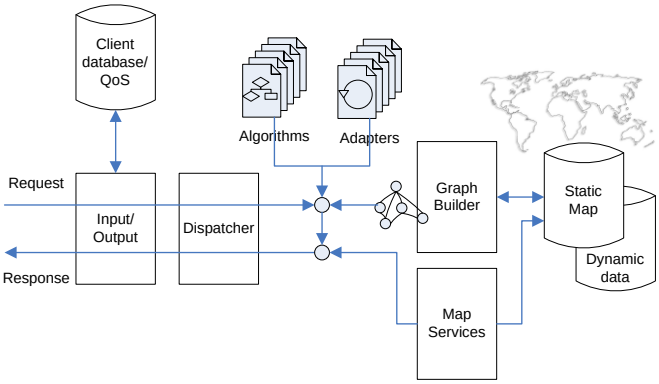

Figure 2. The route server's internal architecture

The main elements (components) are as follows:

- Input/output, a component responsible for handling messages for clients, providing optional QoS and security functions;
- Database, containing the static map and dynamic data (traffic statistics: drive and turn times, speeds, etc.). At present, the static map contains the Open Street Map (OSM) data, although the database format has been significantly modified for our purposes (refer to Małowidzki et al. (2013b) for details).
- Graph Builder, responsible for transforming map data into a graph (a set of nodes and edges) that may be used for route computations;
- Adapter(s), computing graph weights. Usually, each route type (Fast, Short, Optimal, etc.) requires a separate adapter. Their implementation may be trivial (e.g., for a Short route) or fairly complex, as it is in case of a privileged route adapter, described in Małowidzki et al. 2013a. Note that we assume that *weights are functions of time* (the start time of the drive at a given edge), which allows to take into account dynamic (current and predicted) traffic data.
- Algorithm(s), performing route optimizations. Algorithms are separated from road data by adapters; they only see the graph with edge weights computed by adapters. We have tested a number of algorithms, including Ant Colony Optimization (Bedi et al. 2007) and an adapted version of SAMCRA (Góralski et al. 2011), but found that Dijkstra-based algorithms (an optimized version using a priority queue or the A* algorithm (Hart et al. 1968)), perform best.
- Finally, the Dispatcher, managing the above-mentioned elements, and providing additional functions (e.g., alternative routes) and debugging capabilities.

The software is implemented in the Microsoft .NET 4 framework environment. The internal architecture is organized around a number of interfaces. Most crucial elements (.NET classes implementing well-known interfaces) that affect the service behavior (the graph builder, adapters, algorithms) are dynamically loaded according to the server's configuration.

We have also developed a client, implemented in JavaScript/OpenLayers environment. The client's capabilities allow to make use of most of the routing service's functions.

4. SUMO

SUMO (Simulation of Urban MObility) (Behrisch at al. 2011, Krajzewicz et al. 2012) is a microscopic traffic simulator that models the movement of vehicles in space-continuous map and uses a discrete time (with one-second resolution). Each vehicle is modeled separately and is described by a departure time (the time it starts its drive) and a route described by a set or roads.

SUMO has been implemented in the Institute of Transportation Systems at the German Aerospace Center. The version used in our simulation is 0.16.0 (released in December 2012).

The core of simulation software is written in C++. There are additional software libraries that allow to affect the simulation with Python and Java code.

During the selection of the simulator, we also considered MATSim but we found SUMO to be more user-friendly and contain most functions we would require.

SUMO's key features are as follows:

- Open source code;
- Maturity of the project; new versions appearing regularly;
- Sufficient documentation and examples;
- Usage of XML, which provides configuration flexibility. We successfully developed a map converter for SUMO (see section 6).
- An included converter for importing OSM maps;
- A convenient API (called TraCI; available in C++, Python and Java) that allows to control the simulation;
- Last but not least, a good GUI for simulation visualization.

Regarding SUMO drawbacks, it does not support privileged vehicles (that would not have to obey the rules of the road), which is important in Insigma (support for emergency services is one of explicit goals of the project, see Małowidzki et al. (2013a)). This poses a problem with simulating this type of vehicles.

5. PERFORMING SIMULATIONS IN SUMO

This section describes key simulation issues in SUMO. In the next section, we comment on how our "control loop" affects the way simulations are performed.

Map preparation. SUMO's approach to modeling map data is quite convenient. Three separate XML files describe, respectively, nodes, edges and connections. There are additional tools for map data processing. The main tool, *netconvert*, enables to coalesce these files into a SUMO map. OSM data import is easy.

Routing vehicles. SUMO provides a dedicated tool, *activitygen*, for modeling traffic demands, but we decided to implement our own tool. (The main reason was that we wanted to have a better control of the demands. Additionally, we found activitygen's configuration to be complex and insufficiently documented.) The tool supports three traffic classes: driving to work, business traffic, and transit traffic. We assume our map covers a city with residential districts and some center/production area. Our tool, given a population of the city, uses heuristics to generate flows for each traffic class. The flows are defined by a start and an end point for each vehicle. Then, we are either able to use SUMO's routing functions or compute the routes ourselves (section 6).

Moving vehicles. During each simulation step, which equals 1 second, and for every simulated vehicle, SUMO's engine checks whether the vehicle can move ahead and then selects an appropriate speed value. The value may be set to:

- The maximum allowed for a road;
- The previous value increased by an acceleration factor;
- The previous value decreased by a braking factor, if a vehicle is approaching a crossroad or an obstacle (a slower vehicle ahead).

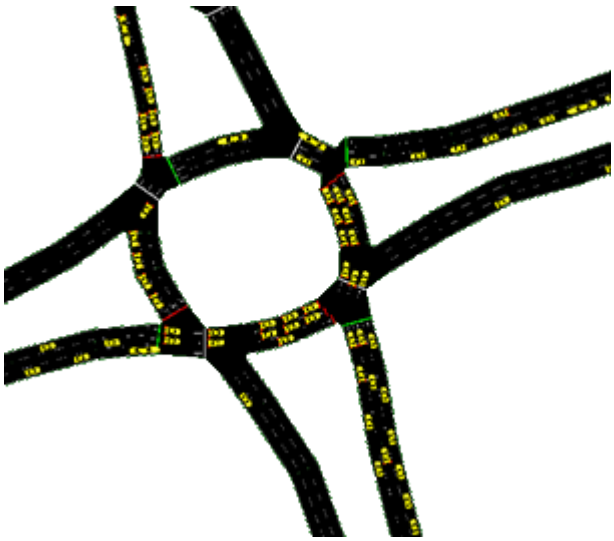

Figure 3. A screenshot that presents a crossroad simulated in SUMO (Dmowski Roundabout in Warsaw)

Then, a vehicle is moved according to the speed value.

Road sensors. SUMO provides abstract detectors that can be used to collect data in any place we would like to observe:

- Aerial induction loops provide average speed of vehicles passing them;
- Crossroads may be monitored as black boxes providing average times needed to pass a crossroad in a particular relation (direction).

Traffic control. Traffic control may be performed mainly through traffic lights. Two main control types are considered: Configuring time slices for a green light (in a given direction) and synchronizing subsequent crossroads along selected major roads to assure a "green wave," that is, uninterrupted traffic through a number of crossroads.

6. SUMO AND ROUTING SERVICE INTEGRATION

This section describes the integration of SUMO and our routing service.

Map preparation. Our database model is based on OSM but contains important extensions. Thus, we needed a dedicated tool able to convert our graphs into SUMO maps. Through an internal Graph Handler interface, it is possible to intercept the graph after it has been constructed by the Graph Builder. The intercepted graph is an input to a converter component, which generates the three required XML files (section 5), which are finally converted to a SUMO map by netconvert.

The conversion retains crucial road parameters such as speed limits, lane counts, etc. Database identifiers are preserved as well, as they are later needed to identify roads in computed routes. Traffic control at crossroads is based on either traffic lights, priorities (specified on the basis of road signs) or the right hand rule.

Routing vehicles using the routing service. The traffic demands are prepared as usually, using our tool described above. Having for each vehicle the start and the destination points, we do not rely on SUMO but instead ask the service to compute routes. Note that the routes are based on dynamic data, which are continuously updated (read from SUMO sensors and delivered to the dynamic map, see below). The interface is implemented in Python.

Requests to the routing service may be synchronous or asynchronous. The synchronous mode may cause some time synchronization issues – see the discussion below – and is supported for testing purposes. In a more realistic approach, with asynchronous requests, a vehicle issues a request and continues its drive along the previous route; as soon as new route is available, it is passed to the vehicle. In case the new route cannot be applied (e.g., a vehicle has just passed a place where, according to the new route, it should have taken a turn), it is discarded and the vehicle continues its drive while a new routing request is scheduled.

The data flow is presented in Figure 4. Data from SUMO sensors feed the dynamic map, and are consumed by the routing service when calculating routes. The simulation/control loop is thus closed.

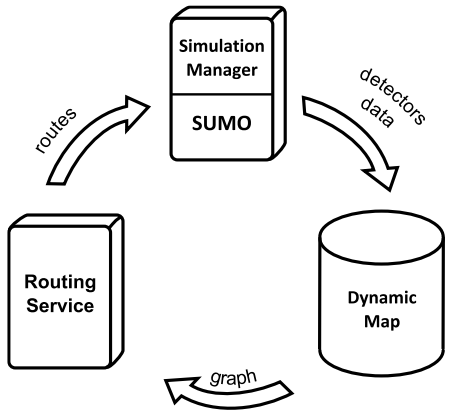

Figure 4. Information flow between SUMO and the routing service

Road sensors. We collect traffic data in the following way:

- Aerial induction loop provide us with average speed values. The loops are placed on every road.
- Multi entry/exit detectors are put on every crossroad; they are a source of travel times through a crossroad in every relation.
- SUMO can dump data periodically to XML files and we do it every 15 minutes.
- Data collection may be performed in one of the two modes:
  - Offline mode that only records data from simulation to files and allows a later "replay," i.e., setting the values in the dynamic map;
  - Online mode that requires both SUMO and control layer services be running simultaneously.

Vehicle sensors. Additionally, we support a simulated GPS Tracker sensor. Such a sensor collects raw GPS vehicle positions and instant speed values, and delivers them to a tracker service. Data are collected from a set of selected cars (possibly, from all cars), stored locally, and delivered in larger packs (for improved performance). A number of parameters can be configured. GPS data are supplemented with some debugging information, which allows to perform on-line analysis of the tracker service's correctness.

Traffic control. Traffic control will be performed using two cooperating mechanisms:

- The routing service, guiding individual vehicles and applying alternative paths to distribute load. We assume that traffic prediction (Małowidzki et al. 2013b) will enable such distribution and possible jamming of "best" routes will be eliminated.
- Traffic lights control, performed by high-level traffic optimization and control algorithms; this remains to be implemented.

Clock synchronization and performance. SUMO itself is faster than real time. It may be slowed down by our integration layer (and additional processing involved). Clock synchronization is important for most control mechanisms to work properly. It is easy to artificially slow the simulation (make 1 second of simulated time equal to 1 second of wall time by introducing artificial delays). However, a coordination in case simulation is slower than real time may pose some problems, although our current experience suggests SUMO will be fast enough.

We are going to simulate tens of thousands of vehicles and the routing service may become a bottleneck (at present, a typical request takes some 3-4 seconds to execute, with most of the time spent on graph construction). Fortunately, as the simulation will be limited to a single city, we plan (if necessary) to build and store the graph in memory, which should definitely improve performance.

7. EXAMPLE

The following example demonstrates the effectiveness of car routing using our service. Two selected cars use different routes (Figure 5.): v1 drives along the shortest route (the red one, shown at the top) computed by SUMO while v2 employs the routing service, which is supplied with current traffic data, to get the fastest route (the green, bottom one). As a result, v1 enters a severe traffic jam and is considerably delayed; v2 takes a lightly loaded detour and arrives to a destination much earlier.

Figure 5. The jammed (upper) and the lightly loaded (bottom) routes (fragments shown)

8. CURRENT EXPERIENCE

Running simulations using an artificial graph is simple. Unfortunately, a real-world map conversion is a source of numerous problems, especially in case of a map that is not precise enough or is missing important road data. Conversion tools are either imperfect or not documented sufficiently (for example, the conversion between geographical coordinates and SUMO positions is quite tricky). Serious problems are caused by "micro-crossroads," that is, very short road segments between crossroads: They often cause a car to be unable to leave such a segment, clogging the crossroad forever. A common case is a deadlock of two opposite car directions, both trying to take a left turn and blocking each other. (These problems may be caused by map inaccuracies or

simulator bugs.) In order to successfully run our simulation on the OSM map for Warsaw, we had to perform a simulation, analyze deadlock points, correct the map, and repeat these steps multiple times. That proved to be a time consuming and tedious work.

Apart from the problems mentioned above, we find running simulations in SUMO as relatively straightforward and enjoying. The strong feature of SUMO is its GUI front end, which allows to visualize moving cars in a nice manner and enables a full control over the simulation. It is uncomplicated to analyze the simulation and identify problems (such as, e.g., deadlocked crossroads). The main API, TraCI, is also functional enough; it allows, among other things, to route cars and control the traffic lights. Thanks to the fact that TraCI clients may be written in Python, we could easily write simulation scripts with desired logic and interfaces to our services (the routing service, the dynamic map, the GPS tracker, and, in future, the traffic control service).

9. SIMULATION SCENARIOS

Simulation scenarios we plan to perform include the following ideas:

Evaluation of control mechanisms. First of all, we are going to evaluate the influence (and efficiency) of additional, more complex and more intelligent control features to the overall traffic system performance. Thus, we are going to enable subsequent control mechanisms and compare results for the following cases (from simplest to most complex):

- routing service based on a static map only;
- routing service provided with current dynamic traffic data;
- routing service provided with both current dynamic traffic data and traffic prediction;
- as above, with load balancing enabled;
- as above, with traffic lights control enabled (a full scenario).

Reliable vs. irresponsible drivers. Simulated "drivers" will be offered a number (probably, two or three) alternative routes, with the preferred one advised by a load balancing function. We could compare two cases, when all drivers select the recommended route or when some of them "know better" and do not obey.

Full data about current traffic situation vs. "unsurveyed areas." We could be able to compare the case when precise data are available for all roads and the case when only main roads are monitored (or, even worse, some sensors fail and report erroneous values). Additionally, the performance of GPS Tracker component could be compared with accurate data from higher-level sensors (i.e., directly from SUMO).

Traffic prediction accuracy. We could check what happens when traffic has been predicted perfectly and what happens if, for an unknown reason, actual traffic differs significantly from the forecast.

Modeling unexpected events. We plan to model accidents, intended traffic jams or sudden road closures and observe how the control layer copes with such a situation.

Modeling privileged vehicles. SUMO does not support privileged vehicles directly but some limited experiments are still possible. For example, we could try to control the traffic lights along a privileged vehicle's route in order to assure a green light at every crossroad.

Note that most of the above scenarios compare the case of an "ideal" world (cooperating drivers, full information available, no unexpected events) with scenarios when something goes wrong, which often happens in the real world.

10. RELATED WORK

During related work review, we were mostly interested in (possibly) large-scale urban traffic simulations. Uppoor et al. (2013) generate a synthetic (although realistic) dataset of 24-hour car traffic for a 400-km^2 area around the city of Koln; the dataset could be then employed in other studies (e.g., research on wireless networks with on-board terminals in moving vehicles). They use OSM as map data and SUMO as the simulation tool. Another example includes simulating Tel Aviv Metropolitan Area in MATSim (Bekhor, Dobler, and Axhausen 2010) in order to compare and match traffic flows (of the original traffic model for Tel Aviv and the flow computed in MATSim). Balmer, Nagel, and Raney (2004) perform a large-scale, 24-hour microscopic traffic simulation for Switzerland (for the whole country). Garcia-Nieto, Alba and Olivera (2011) report on the usage of SUMO for finding successful cycle programs of traffic lights for large urban areas. Ben-Akiva and Davol (2002) present a case study of simulations employed for traffic model calibration for an area near Stockholm.

In addition to the above-mentioned SUMO and MATSim, there are a number of other urban traffic simulation tools. For example, MAINSIM (Dallmeyer and Timm 2012), which as able to simulate not only vehicles but bikes and pedestrians as well (although work on similar features in SUMO is ongoing (Krajzewicz et al. 2012)). DIESIS (2008) contains a categorized list of transportation systems simulation tools.

11. SUMMARY

The Insigma's goals related to traffic optimization and control are ambitious and require that appropriate simulation environment be prepared first. We have developed an advanced routing service and have successfully integrated it with SUMO. Future work will include planned control layer components and large-scale simulations. We hope we will be able to report valuable results.

ACKNOWLEDGMENTS

The work has been co-financed by the European Regional Development Fund under the Innovative Economy Operational Program, INSIGMA project no. POIG.01.01.02-00-062/09.

REFERENCES

Balmer, M., Nagel, K., Raney, B. 2004. Large–scale multi-agent simulations for transportation applications. *Journal of Intelligent Transportation*

Systems: Technology, Planning, and Operations 8, 4 (2004), 205–221.

Bedi, P., Mediratta, N., Dhand, S., Sharma, R., Singhal, A., 2007. Avoiding Traffic Jam Using Ant Colony Optimization - A Novel Approach. *Conference on Computational Intelligence and Multimedia Applications*.

Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D. 2011. SUMO - Simulation of Urban MObility: An Overview. *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011.

Bekhor, S., Dobler, C., Axhausen, K. W. 2010. Integration of activity-based with agent-based models. An example from the Tel Aviv model and MATSim. ETH Zürich, Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau (IVT) (2010).

Ben-Akiva, M. E., Davol, A. 2002. *Calibration and Evaluation of MITSIMLab in Stockholm*. Transportation Research Board Meeting, January 2002.

Dallmeyer, J., Timm, I. J. 2012. MAINSIM - MultimodAl INnercity SIMulation. *35th German Conference on Artificial Intelligence* (KI-2012).

DIESIS. 2008. D2.3 Report on available infrastructure simulators. Design of an Interoperable European federated Simulation network for critical InfraStructures (DIESIS) project report, 2008.

Garcia-Nieto, J., Alba, E., Olivera, A. C. 2011. Enhancing the Urban Road Traffic with Swarm Intelligence: A Case Study of Córdoba City Downtown. *Intelligent Systems Design and Applications (ISDA)*.

Google Maps Developer Documentation: https://developers.google.com/maps/documentation/

Góralski, W., Pyda, P., Dalecki, T., Batalla, J. M., Śliwiński, J., Latoszek, W., Gut, H. 2011. *On Dimensioning and Routing in the IP QoS System*. Journal of Telecommunications and Information Technology, nr 3, p. 21-28.

Hart, P. E., Nilsson, N. J., Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. SSC4 4 (2): 100–107.

Janowski, L., Kozłowski, P., Baran, R., Romaniak, P., Glowacz, A., Rusc, T. 2012. *Quality assessment for a visual and automatic license plate recognition*. Multimedia Tools and Applications.

Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L. 2012. Recent Development and Applications of SUMO – Simulation of Urban Mobility. *International Journal on Advances in Systems and Measurements*, vol 5 no 3 & 4, 2012.

Małowidzki, M., Bereziński, P., Dalecki, T., Mazur, M. 2012. Advanced Road Traffic Service Demonstrator. MCC'2012, Gdańsk, Poland.

Małowidzki, M., Dalecki, T., Bereziński, P., Mazur, M. 2013a. *Traffic Routes for Emergency Services*. Accepted for EUROSIM'2013.

Małowidzki, M., Mazur, M., Dalecki, T., Bereziński, P. 2013b. *Route Planning with Dynamic Data*. Accepted for MCC'2013.

MATSim: Agent-Based Transport Simulations: http://www.matsim.org/

OpenGTS™ - Open GPS Tracking System: http://opengts.sourceforge.net/

OpenLayers: http://openlayers.org

OpenStreetMap: http://www.openstreetmap.org/

SUMO: Simulation of Urban Mobility: http://sumo.sourceforge.net/

Uppoor, S., Trullols-Cruces, O., Fiore, M., Barcelo-Ordinas, J. M.. 2013. Generation and Analysis of a Large-scale Urban Vehicular Mobility Dataset. *IEEE Transactions on Mobile Computing*, 2013.