# Simulator Integration Platform for City Simulations

Yuu Nakajima[1] and Hiromitsu Hattori[1]

Department of Social Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
{nkjm, hatto}@i.kyoto-u.ac.jp
WWW home page:http://users/~iekeland/web/welcome.html

**Abstract.** Multiagent-based simulations are regarded as an useful technology for analyzing complex social systems and have been applied to various problems. Tackling the problems of a city involves various levels of abstraction and various target domains. Different types of human behaviors are studied separately by specialists in their respective domains. We believe that we need to integrate simulators that offer different levels of abstraction and cover various target domains. This paper introduces the architecture of a simulator integration platform and demonstrates the capability of the platform in that domains of city traffic and city electricity.

## 1 Introduction

Multiagent-based simulations (MABS) are regarded as the most powerful technology for analyzing complex social phenomena. Actors, who have various characteristics, play out their daily lives autonomously, and the summation of their actions produces social phenomena. The MABS approach, which observes interaction among diverse agents that model individual humans, is suitable for social simulations[4]. MABS are being applied various domains, e.g. traffic engineering, disaster management, sociology[1, 6, 3].

Vehicular traffic is one of the most complex systems in modern society. Comprehensive traffic simulations must involve various levels of abstraction and various target domains.

In the traffic domain, the behaviors of humans are captured at different levels of abstraction. For example, traffic simulations of a wide-area road network are based on highly abstract models such as traffic flow models[2, 10], while those of a road are based on less abstract models such as driving behavior model[5, 7]. Few simulators can deal with different levels of abstraction.

Additionally, traffic is related to various other city problems: evacuations after a disaster or enhancement of the spread of a flu pandemic. Traffic and other city problems impact each other, but there is no simulation platform that can combine multiple (different) simulators.

This paper has two goals:

1. Integration of simulators having different abstraction level Interpreting the activities of a city involves different levels of abstraction. For city traffic simulations, the movements of people are expressed by highly abstract models such as flow over a road network. For driving simulations at the level of individual roads, people are described in more specificity as actors who observe the environment and then decide their action. Integrating simulators with different levels of abstraction is the first issue. Our solution is the layered architecture, where each simulator shares the data of contact points and a high level simulator calls a low level simulator when it needs precise details of a particular phenomenon.
2. Integration of different domain simulators
   Activities in a city are interpreted as different domains. In the traffic domain, people traveling to a hospital are interpreted as vehicle movements; in the flu pandemic domain, they are interpreted as possible contacts of infected and uninfected people. Since these two domains are only weakly related, we propose an external simulator integration architecture that loosely connects different simulators by establishing mutual contact points through which messages are passed.

The remainder of this paper is as follows. Section 2 describes our approach to designing the simulator integration platform. Section 3 shows how to combine simulators that have different levels of abstraction. Section 4 describes the integration of simulators in different domains: traffic simulation and electricity simulation. Section 5 provides an analysis of the platform's performance and Section 6 places the proposed method within extant research.

## 2 Architecture

We classify the difference between simulators from two viewpoints: abstraction and target domain. First, the simulator components, the parts of the architecture, are described. Second, the layered architecture for simulators having different abstraction level is explained. Last, the external simulator integration architecture for different domain simulators.

### 2.1 Simulator Components

This architecture can handle multiple simulators, each of which captures a specialized aspect of city phenomena. Each simulator is built as an independent system module (Fig. 1, Fig. 2) and includes a simulation controller. Each simulator has its own time step because its components are executed independently.

The simulation controller requests the simulation model to calculate the state of the next step. The simulation models writes its estimate of the next time step. Simulation models consist of agent models to capture human behavior and environmental models to capture city environments.

The points connection among simulators are established by the event managers initiating and responding to event messages. When an event that involves

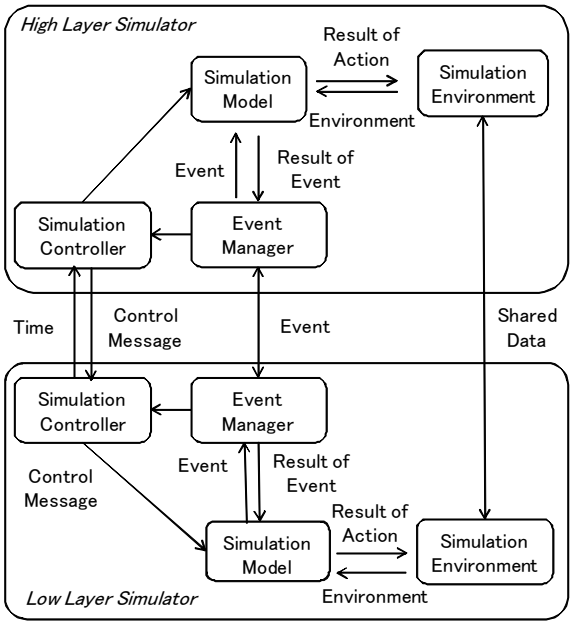


**Fig. 1.** Layer based integration architecture

another simulator occurs, the event is sent to the event manager of the appropriate simulator.

### 2.2 Layered Integration Architecture

Activities of citizens are interpreted on different levels of abstraction when they are modeled in simulations. For example, a high level of abstraction is used to examine gross movements, i.e. the morning movement of people from the suburbs into the city center. A low level of abstraction would be used to examine crowd movement in a shopping area.

Simulators that have different abstraction levels deal with the same phenomenon from different aspects. Because of this, these simulators are strongly linked to each other. We propose here a layered architecture where the higher level layer controls the lower layer and the simulators are connected by data exchanges (Fig. 1).

The simulation model in the higher level layer opens an extensible point as events type for replacing its simulation model to less abstracted simulation model. The simulator of high level abstraction sends the event to the event manager when the simulation model requires more precise calculation. The simulation controller in the higher level simulation sends a request to the appropriate lower level simulator. The data of each simulation environment is accumulated in

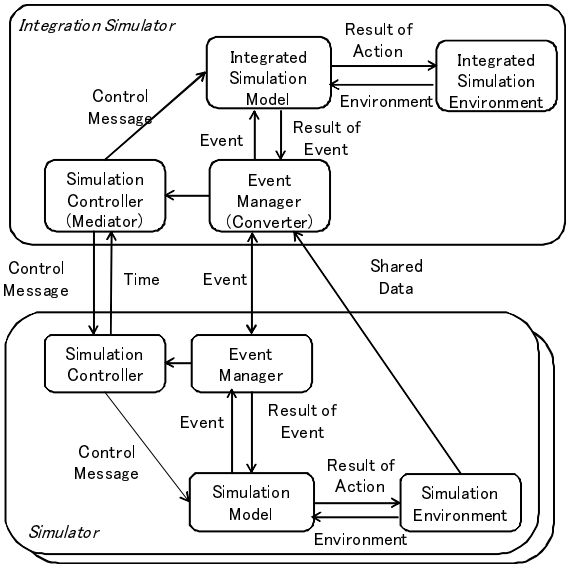


**Fig. 2.** External simulator integration architecture

each simulator but the simulation models exchange the data associated through shared environments.

### 2.3 External Simulator Integration Architecture

The activities of citizens must be interpreted from different point of views. In the traffic domain, people rushing to a hospital are interpreted as the movement of vehicles. In the flu pandemic domain, they are interpreted as contacts between infected and uninfected people. We describe the integration of different domain simulators in this section.

Different domain simulations have a weak relationship and share a little data. Loose coupling is desirable for connecting the various kinds of simulators. We proposed the external simulator integration approach that uses an external integration simulator (Fig. 2). The integration simulator manages the simulation processes and events representing the interaction between target simulators.

The integration simulation controller controls the simulation process of each simulator. Each simulator receives requests to calculate the next state from the integration simulation controller and returns the simulation time of the next state. The integration simulation controller decides the next controller to activate considering the simulation times of each simulation.

Objective simulators for integration are registered with the event manager in the integration simulator. The event manager gathers the events that occur in each simulator and sends them to the appropriate simulators after converting them into events that can be handled by the simulators.

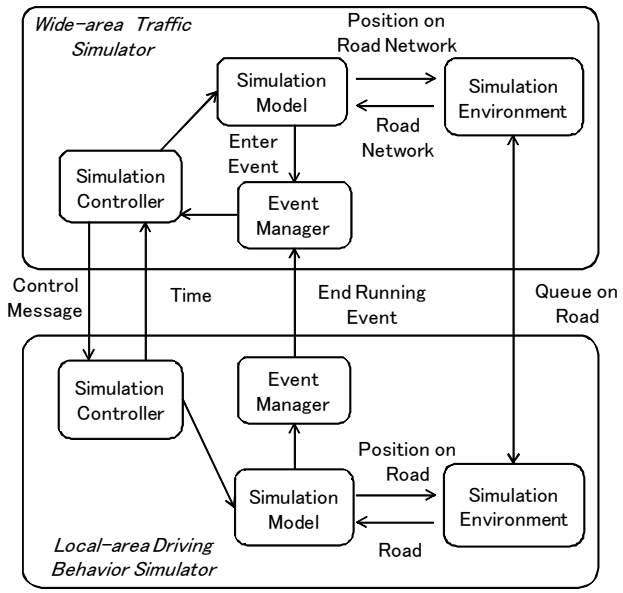


**Fig. 3.** Integration wide-area traffic simulator with local-area driving simulator

Each simulator stores a different simulation environment. A simulation environment can be altered only by its own simulator, while some part can be read by the other simulators. The connection between simulators is realized by message passing to decrease the dependency between simulators and protect the simulation environments from external simulators.

## 3 Integration of Different Abstraction Level Simulators

To explain our approach, we describe the example of connecting a traffic flow simulator for wide-area road networks, with a driving behavior simulator for individual roads. Fig. 3 provides a system diagram of the integration.

Traffic flow simulations using simple agents are popular in the traffic domain. A simulator with a high level of abstraction uses simple nodes and links to model road networks. This is suitable when estimating the relation between traffic demand and traffic flow across wide areas[2]. However, this approach fails to provide realistic driving behavior simulations on particular roads. This is because details of the road structure (*e.g.*, the width of lanes) or surrounding environment including neighboring vehicles cannot be represented, the simulator fails to consider such local factors.

### 3.1 Simulator for Global Traffic

The agents in a global traffic simulation select appropriate routes and then pass along the routes.

The route selection module reads road network data and OD (Origin-Destination) data of agents. Road network data mainly describes the structure of the road network while the OD data consists of tuples of the starting point and the destination point of each agent. The agent selects the route that has minimum cost as identified from map information and the average trip time of each road. A route plan consists of paths, mode choice, daily activity, and so on.

The route execution module deals with abstracted road networks, not two-dimensional spaces. The route execution module realizes a queue-based simulator; that is, the road network is represented as a network of FIFO (First-In, First-Out) queues. Each agent moves over this queue-network between queues according to its scheduled routing plan given vacancies in the next queue. Traffic flows in this simulator are composed of agent transfers between queues.

### 3.2 Simulator for Local Traffic

In the simulator for local traffic, the agent is regarded as a virtual driver and vehicle. They move in a two-dimensional space rather than the abstract road network. The module reads agent ID and road ID from the simulation environment and gets details of the road's structure and surrounding environment including neighboring vehicles.

### 3.3 Integrating Simulators of Different Abstraction Levels

The integration of a local road simulator and a wide-area traffic simulator proceeds as follows (Fig.4).

1. At the initial step, a set of initial plans (routes) is generated based on free speed travel times generated by the wide-area traffic simulation.
2. The traffic flow simulation is run using the generated plans. On the abstract road network in the wide-area traffic simulation, agents move from entering queue to running queue or leaving queue to entering queue.
3. When an agent enters a running queue in the wide-area simulation, an entering event is generated. The wide-area simulator calls the appropriate local area simulator and specifies the event.
4. The local driving simulation calculates the driving behavior of the agents in the specified running queue for the next step. The wide-area traffic simulation and local driving simulation are jointly aware of all running queues.
5. If an agent in a running queue arrives at the end of the road (queue), the agent is moved from the running queue to leaving queue in the wide-area traffic simulation.
6. If the simulation time of the local driving simulation is earlier than that of the wide-area traffic simulation, the local driving simulator updates the time and goes to step 4.
7. Step 2 to step 6 must be iterated before the simulation time is at the end.

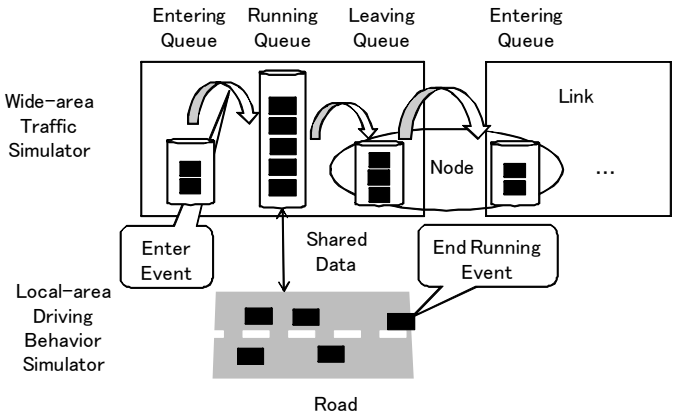


**Fig. 4.** Behavior of cars on road network and road

## 4 Integration of Different Domain Simulators

We connect traffic and electricity-consumption simulators to elucidate our integration technology. This section explains how these two simulators can be integrated. Fig. 5 provides a system diagram of the integration.

Previous papers did not try to combine with city traffic simulators and electricity consumption simulators which were developed independently. However, the rapid penetration of electric vehicles is forcing the emergence of a loose relationship. The charging of an electric vehicle is interpreted as vehicle movement (to the charging station) i.e. traffic domain, and as an electricity consumer i.e. electricity domain.

### 4.1 Traffic Simulation

The traffic simulation used in this section is the one described in Section 3. Traffic agents in the simulation leave their home, go to shopping areas or/and working places and return home. When the agents arrive/leave a facility, arriving/leaving events are generated and sent to the event manager. The traffic simulator transmits the arriving/leaving events and destination change events to the electricity-consumption simulator.

### 4.2 Electricity-Consumption Simulation

The electricity-consumption simulator has several models: electric consumption, electric generation, and electric charging of electric appliances connected to facilities. The simulator calculates electric consumed in each facility step by step. The electricity simulator transmits connecting/disconnecting events to the traffic simulator. The electricity-consumption simulator has a time-step of 10 seconds, which is different from that of the traffic simulator.

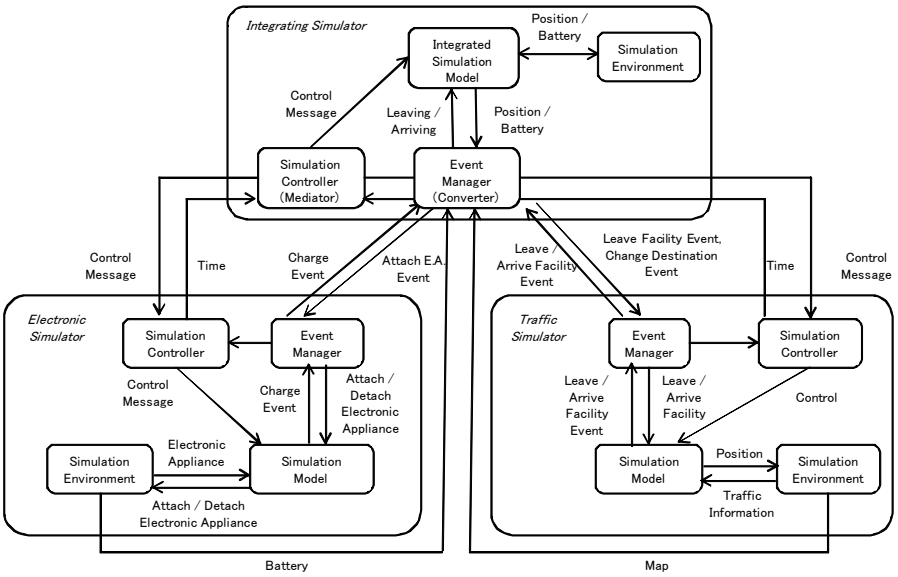


**Fig. 5.** Integration traffic simulator with electronic simulator

### 4.3 Integration of Traffic Simulation and Electricity Simulation

The integration controller (IC) calls the simulator that has the earlier simulation time. The traffic and electricity-consumption simulations are linked by the arriving/charging/leaving behavior at charging facilities. IC calculates electric consumption based on the movement distance of electric vehicle and let the agent decide whether to go to the charging station or not. IC can match objects (agents and facilities) in the traffic simulation with those in the electricity-consumption simulation because the simulator has object ID tables.

IC proceeds as follows.

- When IC receives a "leaving facility" message from the traffic simulator, it records vehicle ID and facility ID. If the battery reserve of the electric vehicle is under the threshold that triggers a search for a charging station, IC locates the charging station nearest the vehicle and sends a destination change message to the traffic simulator. IC reads and records the battery reserve of the electric vehicle.
- When IC receives an "arriving at facility" message from the traffic simulator, it calculates the electricity consumption based on the distance moved, gained from map information, and updates the battery status of the vehicle. If the battery reserve is under the threshold that triggers charging, IC sends a charging event to the electricity-consumption simulator.
- When the vehicle is fully recharged in the electricity-consumption simulator, the vehicle disconnects and the appropriate event is sent to IC. Additionally

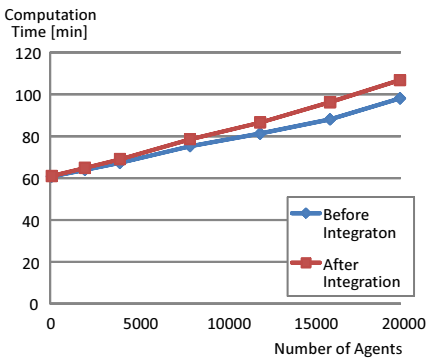


**Fig. 6.** Performance of layer based integration

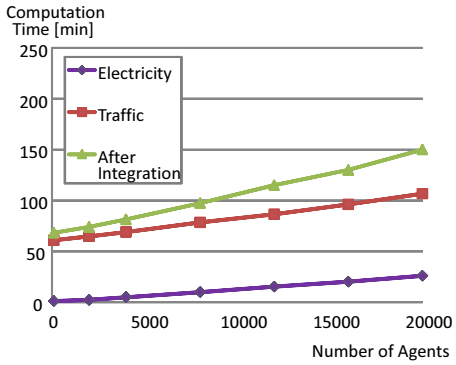


**Fig. 7.** Performance of external simulator integration architecture

when the electric vehicle does not have another activity in the current facility, it leaves the facility and a "leaving facility" event is sent to IC.

## 5 Evaluation

### 5.1 Layerd Integration

We investigate how our layered integration approach affects the computational overhead time. The computational time of integration simulation (after integration) and only wide-area simulation (before integration) are compared. We changed the number of agents from 0 to 20,000.

This evaluation was conducted to estimate the relationship between driving behavior on local roads and route selection in a wide area network. Each agent was given two simple driving rules "If current speed is slower than own desired speed, accelerate", "If there are slow cars in front, pass them if possible".

In this experiment, we used the road network of Kyoto city; it consisted of about fifty thousand links and one hundred thousand nodes in a square 20km x 20km area. We generated ODs (origin-destination) pairs randomly and assigned each to a different traffic agent. Simulation time step for wide-area traffic was one second and for local traffic it was 0.5 seconds. The simulation was executed over a 24 hour period.

The computer used in the following experiment had a Core i7 (8 core) 3.02GHz CPU and 12GB of memory. Simulators are executed on JVM (Sun JRE1.6.0_23(64bit)) on Windows 7 (64bit). The version of MATSim was 0.1.1. These simulators use only one thread except initialize stage.

Fig. 6 plots the computation time versus the number of agents. As you can see, the computation time is directly proportional to the number of agents and no penalty was created by the integration.

### 5.2 External Simulator Integration

We investigate how the external simulator integration architecture affects the computational time for evaluating the proposed platform. The computational time of the integrated simulation and those of each simulation (traffic and electricity-consumption) are compared. We changed the number of agents; from 0 to 20,000.

This simulation scenario assumes that it is conducted to estimate the peak change in electric consumption caused by the use of electric vehicles. All traffic agents in the simulation use electric vehicles. Both simulation environments have the same homes and work locations facilties.

The electricity-consumption simulation had 10 second time steps. The simulation machine and the traffic simulation used the same specifications as the simulation described in Section 5.1.

Fig. 7 plots the computation time versus the number of agents. As you can see, the computation time is directly proportional to the number of agents and the integration incurred no penalty.

## 6 Discussion

### 6.1 Simulation Module Coupling

The most open approach to integrating multiple simulators is to release all interfaces to control the simulators and share all simulation environments. The integrated simulators would have a high degree of coupling in this case but a small degree of coupling is desirable for maintenance and reuse of each simulator.

We proposed the layered integration architecture for connecting simulators having different levels of abstraction. There is a strong relationship between the simulators because each simulator deals with the same phenomena from a different point of view. Therefore, our architecture takes the approach that a higher layer simulator calls the lower layer simulator and both exchange data via connection point. Our approach to integration has lower cost than the full integration approach because interactions between simulators are very simple. However, simulator coupling can be very high.

We proposed integration method by external integration simulator for combining different domain simulators. Each phenomenon in different domain has weaker relationship than that of different abstraction level simulators. External simulator integration is suitable because the simulation processes and simulation environments can keep independent each other. The implementation cost for integration with the approach is larger than layered approach because interactions between each simulators becomes complex. However, the couplings of simulators become low.

The proposed architectures require the creation of simulator components for controlling the simulation process, interfaces for observing some part of the simulation environments, and event types. IC can be reused because it also can be regards as simulator component.

### 6.2 Related Work

Some platforms that execute in distributed environment have been proposed. ASON[1], Repast[2], ZASE[9] are the large-scale multiagent simulation environment. These platform provide supports for multi-threading and agent data managing and. The platforms help developers to implement simulator which can be used in parallel and distributed environment. Platforms that reuse existing simulators have been proposed [8]. The platform integrates Repast and Ascape[3] modules as reusing existing simulators.

These simulators are focused on extending scalability of simulators. While on the proposed architecture is focused on integration of simulators which capture various phenomena in a city with different abstraction level.

One future direction of this study is to support parallel and distributed environment for increasing computational resources because precise simulation of various phenomena in a city required taking much of calculation. We try to implement a simulation integration platform on the large-scale multiagent simulation platforms listed in this section.

## 7 Conclusion

Multiagent simulations are increasingly seen as the most attractive approach to analyze complex phenomena in a city and are being applied to various domains. The phenomena captured by the different simulators can be related. There is no platform that can integrate different simulators so that they can support and complement each other.

Many simulators have already been developed and they contain knowledge and technology created from previous work in their respective application areas. We proposed a simulation platform that can integrate multiple simulators. We classified the difference in simulators from two viewpoints: abstraction and target domain. Our contributions are as follows.

1. Integration of simulators having different abstraction level
   We proposed a layered architecture wherein each simulator shares data across contact points and the higher level simulator calls the lower level simulator when the former needs precise data about a target phenomenon.
2. Integration of different domain simulators
   We proposed an architecture based on an external integration simulator controller for loosely connecting different simulators via points for message passing.

One future direction of this study is to apply the proposed platform to tackle realistic problems. In particular, as shown in Section 4, we will try to capture the effect of electric vehicles which behave as actors who in the traffic domain and as electricity consumers across an entire city.

[1] `http://cs.gmu.edu/~eclab/projects/mason/`
[2] `http://repast.sourceforge.net/`
[3] `http://ascape.sourceforge.net/`

## Acknowledgment

This work was supported by Panasonic Corp. - Kyoto University Joint Research: Crowd Navigation for Region EMS Considering Individual Behaviors and Preferences and Kyoto University Global COE Program: Informatics Education and Research Center for Knowledge-Circulating Society.

## References

1. Balmer, M., Cetin, N., Nagel, K., Raney, B.: Towards truly agent-based traffic and mobility simulations. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004). pp. 60–67 (2004)
2. Balmer, M., Meister, K., Rieser, M., Nagel, K., Axhausen, K.W.: Agent-based simulation of travel demand: Structure and computational performance of matsim-t. In: Proceedings of the 2nd TRB Conference on Innovations in Travel Modeling. pp. 1–30 (2008)
3. Deguchi, H., Kanatani, Y., Kaneda, T., Koyama, Y., Ichikawa, M., Tanuma, H.: Social simulation design for pandemic protection. In: Proceedings of The 1st World Congress on Social Simulation (WCSS-2006). vol. 1, pp. 21–28 (2006)
4. Epsterin, J., Axtell, R.: Growing Artificial Societies: Social Science from the Bottom Up. MIT Press (1996)
5. Halle, S., Chaib-draa, B.: A collaborative driving system based on multiagent modelling and simulations. Journal of Transportation Research Part C 13, 320–345 (2005)
6. Kitano, H., Tadokor, S., Noda, H., Matsubara, I., Takahasi, T., Shinjou, A., Shimada, S.: Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In: Proceedings of the IEEE Conference on Systems, Men, and Cybernetics, 1999. vol. VI, pp. 739–743 (1999)
7. Paruchuri, P., Pullalarevu, A.R., Karlapalem, K.: Multi agent simulation of unorganized traffic. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002). pp. 176–183 (2002)
8. Scerri, D., Hickmott, S., Padgham, L., Drogoul, A.: An Architecture for Modular Distributed Simulation with Agent-Based Models. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2010). pp. 541–548 (2010)
9. Yamamoto, G., Tai, H., Mizuta, H.: Consideration on realizing a hundred millions agent-based simulation. The IEICE transactions on information and systems (Japanese edetion) 90(9), 2423–2431 (2007-09-01)
10. Yamashita, T., Izumi, K., Kurumatani, K., Nakashima, H.: Smooth traffic flow with a cooperative car navigation system. In: Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005). pp. 478–485 (2005)