Patterns to distribute mobility simulations

Conference Paper · November 2016

CITATIONS
0

READS
26

4 authors:

Matthieu Mastio
Institut Français des Sciences et Technologies …
3 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Mahdi Zargayouna
Institut Français des Sciences et Technologies …
49 PUBLICATIONS **113** CITATIONS
SEE PROFILE

Gérard Scémama
Institut Français des Sciences et Technologies …
38 PUBLICATIONS **100** CITATIONS
SEE PROFILE

Omer F. Rana
Cardiff University
442 PUBLICATIONS **3,752** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Multi-Agent Systems and secured coupling of Telecom and Energy grids for next generation smart grid services (MAS2TERING) View project

Patterns to distribute mobility simulations

Matthieu Mastio[a], Mahdi Zargayouna[a], Gérard Scemama[a], Omer Rana[b]

[a] Université Paris-Est, IFSTTAR, GRETTIA Champs sur Marne, France.

[b] School of Computer Science and Informatics, Cardiff University Cardiff, United Kingdom.

{matthieu.mastio, mahdi.zargayouna, gerard.scemama}@ifsttar.fr, ranaof@cardiff.ac.uk

Abstract*—Travelers mobility simulation is a powerful tool to test strategies in a virtual environment, without impacting the quality of the real traffic network. However, existing mobility multiagent and microsimulations can only consider a sample of the real volumes of travelers, especially for big regions. With distributed simulations, it would be easier to analyze and predict the status of nowadays networks. This kind of simulations requires big computational power and methods to split the simulation between several machines. This work describes how to achieve such a distribution in a microscopic simulation context, and compare our results with a previous work on macroscopic simulation.** ***Keywords*—Distributed simulation, Multiagent systems, Microscopic simulation, High performance computing, Traffic simulation

I. Introduction

Transport systems are more and more complex and they have to evolve to integrate more connected entities (mobile devices, connected vehicles, etc). Indeed, we can now provide optimal routes for the travelers and we are also able to update these routes in real time based on new network status (congestions, accidents, bus down, canceled carpooling, etc). Giving information to the traffic network users is generally beneficial and allows to improve the global network traffic flows. However, without control, the massive spread of information via billboards, radio announcements and individual guidance may have adverse effects and create new traffic jams. With the generalization of real-time traveler information, the behavior of modern transport networks becomes harder to analyze and to predict. It is then important to model and simulate a realistic number of travelers to correctly observe these effects [23]. The ability to run a traffic simulator with real volumes of travelers at a city, a region or a country scale, would allow to observe the consequences of different information strategies on the status of multimodal traffic before implementing them in an operational context. With the results of a simulation, it is possible to compare the forecast computed with the data measured on the real network.

Furthermore, the design and development of mobility simulations are relevant in several contexts and in pursuit of various objectives. The simulation can be used to validate the impact of the use of cooperative systems [11] to test changes in behavior after the introduction of new mobility services, such as carpooling, etc. The use of a simulator in this context is interesting to define scenarios, which is difficult to achieve in a real context.

We have recently developed SM4T (*Simulator for Multiagent MultiModal Mobility of Travelers*), a simulation platform for mobility [24]. This platform simulates the behaviors of travelers interacting in a complex, dynamic and open environment. It is based on multiagent paradigm, especially suitable for this type of simulation [2], [5]. Each agent tries to find the most efficient route to reach its destination in a network evolving dynamically. An agent can potentially be informed of the status of the network and use this information to modify his original route. It is then interesting to run a simulation on the entire network because of decisions made at the microscopic level by an agent in one location can have a global influence on the network status at different location later in the simulation. In SM4T platform, as in many other multiagent mobility simulators, it may be interesting and even essential to simulate actual volumes of travelers. In the European *Instant Mobility project*[1], for instance, the objective was to supply a multimodal platform guide, with multimodal travel queries and dynamic positions of travelers and vehicles. To allow the platform to demonstrate its efficiency in an operational context, the simulator also has to execute a realistic number of travellers. Other examples where simulations has to scale up concern testing new mobility services such as carpooling, car sharing, dial a ride, the exchange of information between connected vehicles, etc.

However, the simulation of an actual number of passengers in a big city (several million of travelers) requires both considerable computing power and an architecture allowing the distribution of computations on many hosts. The current version of SM4T [25], as the majority of current traveler mobility simulators, does not allow such distribution. This induces a limit on the number of travelers, means of transport and the size of the considered networks. Therefore, predicting the effects of regulations and information strategies on large networks with connected and informed travelers is very difficult. Our main objective is to test the scalability of this type of simulator. We aim to provide reproducible generic distribution patterns that could be used by the majority of multiagent mobility simulators.

[1] http://www.instant-mobility.com/

In a previous work [16], we have proposed distribution methods for a macroscopic simulator. Macroscopic simulations on the other hand use a function to calculate the vehicles speeds on a road, based on the number of vehicles on that road. Microscopic simulations let agents calculate their speed based on their immediate neighbour vehicles. Our goal in this article is to extend the proposed simulator for these microscopic agent-mobility simulators.

The remainder of this paper is structured as follows. In section II, we present the previous proposals for travelers mobility simulation and the existing distributed multi-agent platforms. Section III presents our previous work concerning the distribution of a macroscopic simulator. In section IV, we describe the microscopic model that we use for our experiments. Section V explains our experimental setup and how we applied our distribution methods on the microscopic simulator. We then provide our results before to conclude.

II. Related work

In this section, we position our work with the literature. In the next paragraph, we present the existent mobility simulators. Then we focus on the generic parallel multiagent platforms. Finally, we describe some theoretic work on the distribution of multiagent simulations.

There exist several multiagent simulators for travelers mobility. For instance, Transims [20] simulates multimodal movements and evaluates the impact of policy changes in traffic or demographic characteristics. AgentPolis [14] is also a multiagent platform for multi-modal transportation and MATSim [18] is a widely known platform for mobility micro-simulation. However, the mobile entities in MATSim are passive and their state is modified by central modules, which limits its flexibility and its ability to integrate new types of (proactive) agents. Sumo [4] and Vissim [9] are also widely used microscopic simulators mainly focused on traffic. There are also other simulators, such as Archisim [8], that describe very precisely the behaviors of each drivers at a microscopic scale. However, as far as we know, no distribution methods which are specific to this kind of traffic simulator, taking in account their characteristics, has been proposed.

Some general-purpose multiagent platforms have been specifically developed for large scale simulation in the last years. RepastHPC [7], a distributed version of Repast Simphony, uses the Repast's concepts of projections and contexts and adapts them for distributed environments. Pandora [1] is close to RepastHPC and automatically generates the code required for inter-server communications. GridABM [12] is based on Repast Simphony but takes another approach and proposes to the programmer general templates to be adapted to the communication topology of his simulation. Flame [6] allows the programmer to generate HPC[2] simulations from finite state machines. It has also been sugested to use graphical units (GPGPU) to scale up the multiagents simulations [19]. However, these distributed platforms do not offer fine controls on how the communications between hosts are performed. Indeed, the communication layer is transparent for the programmer, which makes it easier for him to develop distributed simulations, but prevents him from optimizing the distribution. The best way to manage the communications depends of the application and using such general platforms for a travel simulator would not produce optimal results.

More theoretical works studied general methods to address this problem. In [17] and [22] the authors propose to relax some synchronization constraints to achieve a better scalability by reducing the time the hosts wait for each other. This relaxation of constraint implies a loss a precision which is not viable in the case of a traffic simulator. We could reach a state where the vehicles overlap and occupy the same position in the network. In [21] the authors discuss the issues related to multiagent simulation in a distributed virtual environment. This paper describes methods that allow to split the virtual environment in several zones to parallelize the simulation execution. This work proposes an efficient splitting of a continuous space in two dimensions. In the present paper, we use an adaptation of this work for a graph structure, adapted to distribute traffic-based microscopic simulations.

III. Macroscopic simulation

To launch a simulation at a city scale, we need a large memory and computing power. A way to achieve it is to deploy the simulation on several processing units instead of a single one, where each unit runs the same program but owns only a part of the program data in its private memory, and all the processors are connected by a network. The advantage of this approach is its high scalability; it can be implemented on most parallel architectures and we can deploy the same simulation on larger systems if we need more power.

With the purpose of testing the efficiency of different ways to distribute the work load, we have, in a previous work [16], developed a multiagent travel simulator.

A. The simulation environment

We model the transportation network in which the traveler evolve with a graph $G(V,E)$ where $E = \{e_1, ..., e_n\}$ is a set of edges representing the roads and $V = \{v_1, ..., v_n\}$ is a set of vertices representing the intersections. The agents, representing the travelers move in this network from their origins to their destinations, trying to minimize their travel time. The travel time on an edge depends on the number of agents on it.

In this simulator, the agents do not interact directly. Instead, the simulation uses a fundamental diagram of traffic flow, that gives a relation between the flow (vehicles/hour) and the density (vehicles/km) (cf. fig 1) to calculate the speed of the agents at each time. The fundamental diagram suggests that if we exceed a critical density of vehicles k_c, the more vehicles are on a road, the slower they will move (fig 2).

B. Distribution

The designed simulator is representative and generic. It is as simple as possible and as complex as needed, including

[2]High Performance Computing

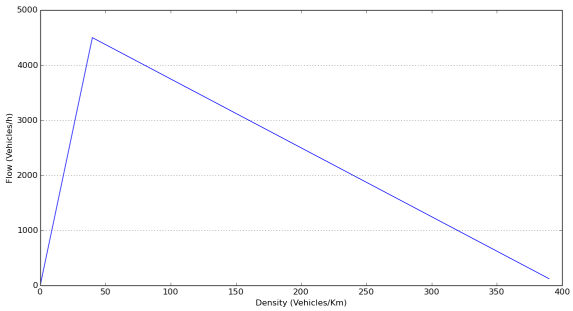


Fig. 1. Flow in function of density

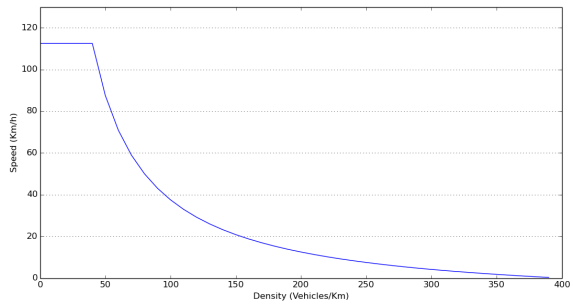
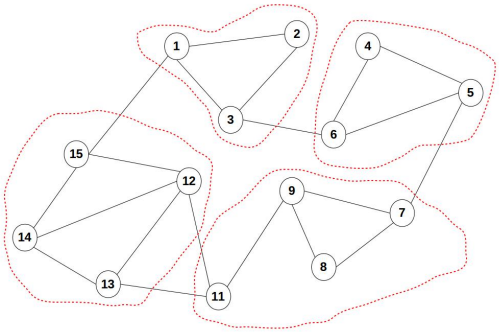


Fig. 2. Speed in function of density

travel on a network in a consistent manner with the current load of the roads. We have kept in this simulator only the aspects that can have an impact on the distribution problem. Based on this simulator, we started working on distribution patterns to run it on several processing units. To do so, we split the workload between the available cores the most efficiently possible.

In this simulation model, the main workload is generated by the calculation of the shortest paths. Indeed, as the travel times evolve dynamically, agents paths have to be recalculated several times during the simulation. That is why we chose to consider an agent as a unit of workload. Therefore, in order to distribute this model, we need to split the agents between the servers. We have tested our simulator with two different distribution methods.

1) Environment distribution: The first approach tries to keep agents that are close in the transport network on the same server. We cut the network in as many parts as we have cores at disposal, and split it between the different cores. Each server is only aware of what is happening on the part of the graph that it is managing, and the agents that are in the same location are now likely to be on the same server. If an agent reaches a part of the network that is not managed by his current core, he has to be transferred to the proper core. In order for the environment distribution method to be effective, each core has to manage approximately the same number of agents and the number of edges cut by the partition has to be minimized (to reduce the number of agents being transferred between cores).



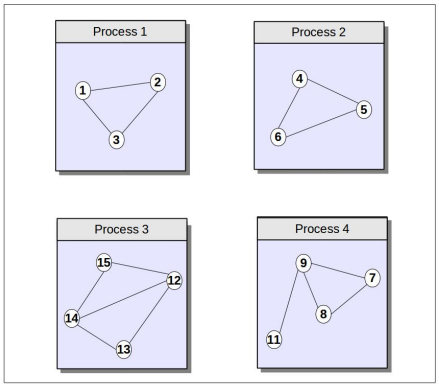


Fig. 3. Environment distribution

The problem of partitioning a network has been widely studied in the scientific literature. We proposed a method derived from the Differential Greedy algorithm [10] that allowed us to use the algorithm with weighted vertices while producing more connected partitions (Algorithm 1).

Algorithm 1 Modified *Differential Greedy* algorithm

Require: Graph $G = (V, E)$, number k of partition
Ensure: Partition P
$P \leftarrow P_0, ..., P_{k-1}$
$V' \leftarrow V$
for $p \in [0, k-1]$ **do**
  $v \leftarrow$ random vertex of V'
  $P_p \leftarrow \{v\}$
  $V' \leftarrow V' \setminus \{v\}$
end for
while $|V'| > 0$ **do**
  $p \leftarrow$ index of the lightest partition
  $m = min_{v \in V'}(1+\varepsilon)$(number of v's neighbors $\in P_p$) − (number of v's neighbors $\notin P_p$)
  $mv =$ random vertex of $v \in V' | (1+\varepsilon)$(number v's neighbors $\in P_p$) − (number of v's neighbors$\notin P_p$) $= m$
  $P_p \leftarrow P_p \cup \{mv\}$
  $V' \leftarrow V' \setminus \{mv\}$
end while
Return P

2) Agents distribution: The second distribution pattern cuts the set of agents in k equal parts (with k the number of

available servers), and distribute each subset on a server and runs the simulation. This method ensures that each core has the same amount of work at any time of the simulation. Due to the absence of inter-agents communication, this method is particularly well adapted with the macroscopic model.

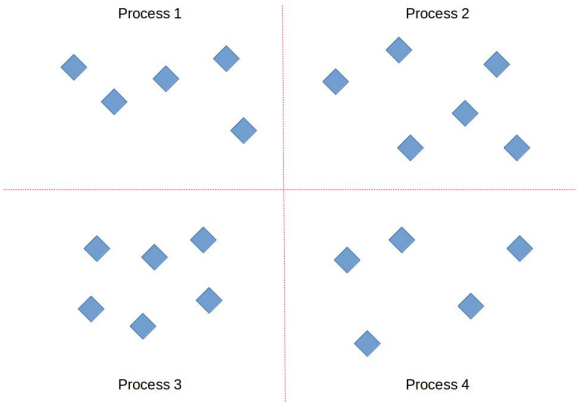


Fig. 4. Agents distribution

IV. Microscopic simulation

The agents distribution showed much better results with a macroscopic simulator, based on the fundamental diagram of traffic. Indeed, with this method, there is a perfect balance of the workload, without being penalized by inter-cores communications, since the agents do not interact locally in macroscopic models.

However, many multiagent traffic simulators of the literature implement microscopic behaviors to model the agents movements. In such a paradigm, the information available to each agent is only local. The agents perceive a part of their environment, delimited by their aoi[3] and then calculate their next move given the perceived information. This implies many local communications between the agents, because their actions will be conditioned by the actions of the other agents present in their aoi.

To represent simulators with microscopic behavior, we implement a generic microscopic multiagent traffic simulator. Our goal here is to represent the interactions in this kind of simulators, not to get an operational microscopic traffic simulation. In our model, the behavior of an agent is influenced by the agent before him, using a car-following model. This generates a lot of local interactions and represent micro simulations behavior, whatever the car-following model they use.

The simulator uses the same graph structure. Each agent represents a traveler evolving in a traffic network. Agents appear nondeterministically with an origin and a destination vertices. They compute the shortest path based on the current status of the network before to start traveling. They ask for a new shortest path each time they reach a vertex in their path, to check whether a new shortest path becomes possible, following the dynamics of the network. The simulation ends when all the travelers have reached their destinations or when a time step threshold is reached.

[3]area of interest

A. The car-following model

At each time step of the simulation, the agents determinate their speed based on the speed and position of the agent before him. The variables needed to describe our model are the following:

1)	$x_n(t)$	position of n at t time
2)	$x'_n(t)$	n speed at t time
3)	$x''_n(t)$	n speedup at t time
4)	$s_n(t) = x_{n-1}(t) - x_n(t)$	inter-agent distance
5)	$s'_n(t) = x'_{n-1}(t) - x'_n(t)$	relative speed
6)	T	reaction time

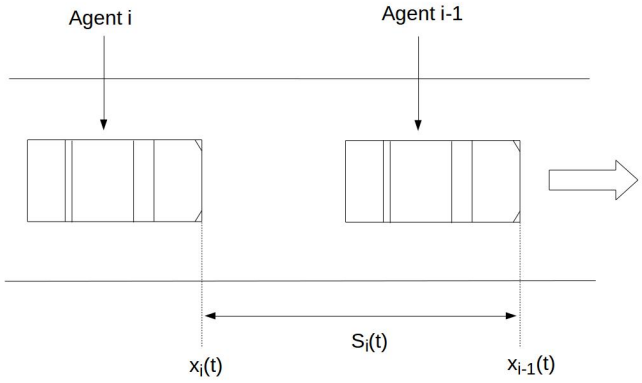


Fig. 5. The pursuit model.

Thus, at any time step, the speed of an agent is given by the relation:

$$x''_n(t+T) = \alpha s'_n(t) + \beta s_n(t) + \gamma x'_n(t)$$

If there is no vehicle preceding the agent, he will accelerate until it reaches the speed limit of his edge.

V. Experiments and Results

A. Implementation

We have modified our previous simulator to test the two distribution methods with the microscopic model. We use Python to develop our simulator for its efficiency in quick prototyping. Python is a mature portable language with a lot of well tested scientific libraries and is along with C and Fortran one of the most used languages for high performance computing [15]. Here, we do not seek absolute performance, but we aim to study the relative efficiency of different distribution methods. Thus we believe that Python is a relevant choice.

The inter-process communications are managed by MPI, which is the standard language for parallel computing with a huge community of users. MPI offers a simple communication model between the different processes in a program and has many efficient implementations that run on a variety of machines [4].

We have launched the distributed simulations on an experimental cluster we have set up. For our tests, we used two hosts under Linux Mint 17.2 Rafaela (kernel version 3.16.0-38-generic) each with a processor Intel Xeon CPU E7-4820 (32 cores at 2Ghz) with 250GB of memory. We ran the simulation on two configurations: the first is a sequential version of the program on a single core, the second is a distributed version on the whole 64 cores.

The simulation is performed for 100 time steps on a 200 nodes power-law graph generated with the Barabasi-Albert model [3], which is often used to represent transportation networks [13].

B. Results

We compare the two methods of distribution (agent-based and environment-based distributions) with the different paradigms (micro and macro) increasing the number of agents (from 10,000 to 500,000).

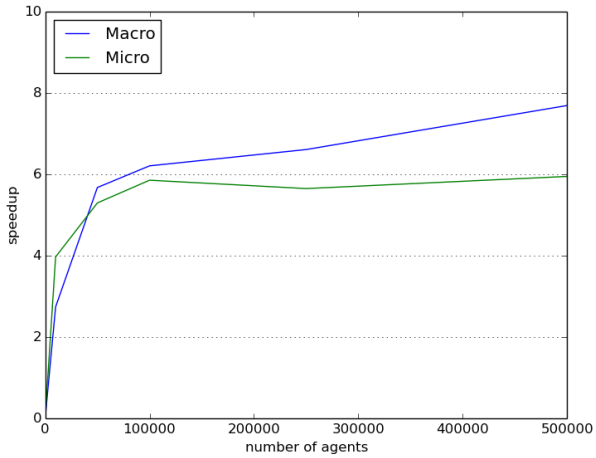


Fig. 6. Speedup for the environment distribution.

The results are shown in table. I, and the speedups for the two distributions methods applied on the different paradigms are plotted in Fig. 6 and Fig. 7. The speedup measures how many times the distributed simulation is faster compared to the corresponding sequential execution.

As we can see, the agent distribution is really efficient for a macroscopic model (14 times faster with 500,000 agents). There is no local interactions in this paradigm. This method allows us to get a perfectly balanced load all along the simulation, while keeping the amount of inter-servers communications at the minimum.

[4]MPI4PY is an efficient interface that allows to use MPI with Python.

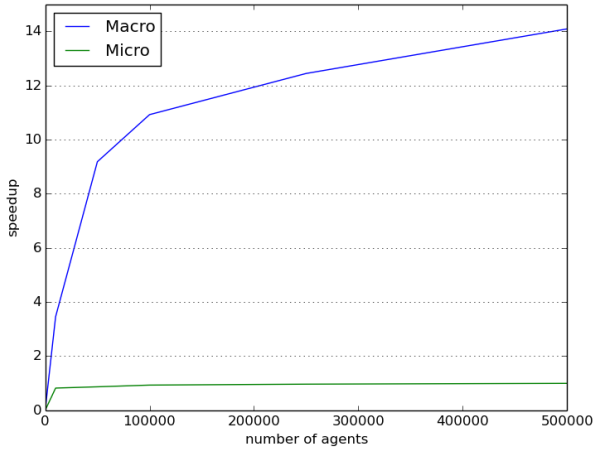


Fig. 7. Speedup for the agent distribution.

However, this method is particularly ineffective in the case of a microscopic simulation. Indeed, the agents will now interact a lot with other agents that are not situated in the same server. This will generate a lot of communications between the servers, and the gain of the parallelization will be annihilated by the time required by these communications. This method is even less efficient than the sequential execution for the microscopic model (speedup < 0).

For a macroscopic simulation, the environment distribution is less efficient than agent distribution. It is well adapted for a microscopic simulation though, showing similar results than for a macroscopic simulation. This method is six times faster than a sequential execution applied in a microscopic simulation, while there is at the moment no dynamic load balancing mechanism implemented for this method. Indeed, for now, the traffic network is only splitted once at the beginning of the simulation. If an important number of agents are concentrated in the same part of the network, they will be nevertheless in the same server. It will hence take more time for this server to calculate all the shortest paths and, all the other servers will have to wait for it. If a server is overloaded it can slow down all the simulation.

VI. Conclusions and perspectives

In this paper, we applied two distribution methods on a microscopic travel simulator, and compared the results with our previous work on a macroscopic simulator. We have seen that, while well suited for macroscopic simulators, the agents distributed method can not be applied in a context where many local interactions occur.

However, we have good results with the environment distribution method. As most of the simulators of the literature are based on a microscopic approach, we think that this method could be applied with great benefit on them.

The environment distribution is currently done statically, at the beginning of the simulation. We believe that

number of agents	10,000	50,000	100,000	250,000	500,000
Sequential Macro (1 cores)	30.9	142.8	288,3	714,3	1540.9
Agents distribution Macro (64 cores)	8.9	15.5	26.4	57.4	109.4
Environment distribution Macro (64 cores)	11.3	25.2	46.5	108.2	200.5
Sequential Micro (1 cores)	62.6	302.4	642.3	1686.2	3413.4
Agents distribution Micro (64 cores)	76.3	348.8	690.7	1747.9	3434.7
Environment distribution Micro (64 cores)	15.8	57.1	109.8	298.7	574.3

TABLE I. Computational times (in seconds) for a 100 time steps simulation on a 200 nodes scale free graph.

the speedup could be largely improved by adding dynamic load balancing mechanisms. We will delve into this aspect in a future work.

References

[1] E. Angelotti, E. Scalabrin, and B. Avila. PANDORA: a multi-agent system using paraconsistent logic. In *Fourth International Conference on Computational Intelligence and Multimedia Applications, 2001. ICCIMA 2001. Proceedings*, pages 352–356, 2001.

[2] F. Badeig, F. Balbo, G. Scemama, and M. Zargayouna. Agent-based coordination model for designing transportation applications. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 402–407. IEEE, 2008.

[3] A.-L. BarabÃąsi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct. 1999.

[4] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - simulation of urban MObility - an overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 55–60, 2011.

[5] N. Bessghaier, M. Zargayouna, and F. Balbo. An agent-based community to manage urban parking. *Advances in Intelligent and Soft Computing*, 155:17–22, 2012.

[6] S. Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth, and C. Greenough. Exploitation of high performance computing in the FLAME agent-based simulation framework. In *2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, pages 538–545, 2012.

[7] N. Collier and M. North. Repast HPC: A platform for large-scale agent-based modeling. In W. Dubitzky, K. Kurowski, and B. Schott, editors, *Large-Scale Computing*, pages 81–109. John Wiley & Sons, Inc., 2011.

[8] A. Doniec, R. Mandiau, S. Piechowiak, and S. Espié. A behavioral multi-agent model for road traffic simulation. *Eng. Appl. of AI*, 21(8):1443–1454, 2008.

[9] M. Fellendorf and P. Vortisch. Microscopic traffic flow simulator vissim. In *Fundamentals of Traffic Simulation*, pages 63–93. Springer, 2010.

[10] C. Fiduccia and R. Mattheyses. A linear-time heuristic for improving network partitions. In *19th Conference on Design Automation, 1982*, pages 175–181, June 1982.

[11] M. Gueriau, R. Billot, N.-E. El Faouzi, S. Hassas, and F. Armetta. Multi-Agent Dynamic Coupling for Cooperative Vehicles Modeling. In 30/01/2015, editor, *The Twenty-Ninth Conference on Artificial Intelligence AAAI'2015 - (DEMO Track)*, Jan. 2015.

[12] L. Gulyas, G. Szemes, G. Kampis, and W. de Back. A modeler-friendly API for ABM partitioning. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 219–226, 2009.

[13] M.-B. Hu, R. Jiang, Y.-H. Wu, W.-X. Wang, and Q.-S. Wu. Urban traffic from the perspective of dual graph. *The European Physical Journal B*, 63(1):127–133, 2008.

[14] M. Jakob, Z. Moler, A. Komenda, Z. Yin, A. X. Jiang, M. P. Johnson, M. Pechoucek, and M. Tambe. Agentpolis: towards a platform for fully agent-based modeling of multi-modal transportation (demonstration). In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 1501–1502, 2012.

[15] H. P. Langtangen and X. Cai. On the efficiency of python for high-performance computing. In *Modeling, Simulation and Optimization of Complex Processes*, pages 337–357. Springer Berlin Heidelberg, Jan. 2008.

[16] M. Mastio, M. Zargayouna, and O. Rana. Towards a distributed multiagent travel simulation. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 15–25. Springer, 2015.

[17] D. Mengistu and M. v. Lowis. An algorithm for optimistic distributed simulations. In *Modelling, Simulation, and Identification / 658: Power and Energy Systems / 660, 661, 662*. ACTA Press, 2011.

[18] M. Michal and N. Kai. Towards multi-agent simulation of the dynamic vehicle routing problem in matsim. In *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics - Volume Part II*, PPAM'11, pages 551–560, Berlin, Heidelberg, 2012. Springer-Verlag.

[19] F. Michel. Délégation GPU des perceptions agents : intégration itérative et modulaire du GPGPU dans les simulations multi-agents. application sur la plate-forme turtlekit 3. *Revue d'Intelligence Artificielle*, 28(4):485–510, 2014.

[20] K. Nagel and M. Rickert. Parallel implementation of the transims micro-simulation. *Parallel Computing*, 27(12):1611–1639, 2001.

[21] O. Rihawi, Y. Secq, and P. Mathieu. Effective distribution of large scale situated agent-based simulations. In *ICAART 2014 6th International Conference on Agents and Artificial Intelligence*, volume 1, pages 312–319. SCITEPRESS Digital Library, 2014.

[22] M. Scheutz and P. Schermerhorn. Adaptive algorithms for the dynamic distribution and parallel execution of agent-based models. *Journal of Parallel and Distributed Computing*, 66(8):1037–1051, 2006.

[23] M. Zargayouna, A. Othman, G. Scemama, and B. Zeddini. Impact of travelers information level on disturbed transit networks: a multiagent simulation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2889–2894. IEEE, 2015.

[24] M. Zargayouna, B. Zeddini, G. Scemama, and A. Othman. Agent-based simulator for travelers multimodal mobility. *Frontiers in Artificial Intelligence and Applications*, 252:pp 81–90, Jan. 2013.

[25] M. Zargayouna, B. Zeddini, G. Scemama, and A. Othman. Simulating the impact of future internet on multimodal mobility. In *The 11th ACS/IEEE International Conference on Computer Systems and Applications AICCSA'2014*. IEEE Computer Society, 2014.