Simulation Modelling Practice and Theory 89 (2018) 35–47

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Two distribution methods for multiagent traffic simulations

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

[a] *Université Paris-Est, IFSTTAR, COSYS, GRETTIA, Champs sur Marne F-77447, France*
[b] *School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom*

A R T I C L E I N F O

Keywords:
Multiagent systems
Traffic
Simulation
High performance computing

A B S T R A C T

Modeling and simulation play an important role in transportation networks analysis. With the widespread use of personalized real-time information sources, the behavior of the simulation depends heavily on individual travelers reactions to the received information. As a consequence, it is relevant for the simulation model to be individual-centered, and multiagent simulation is the most promising paradigm in this context. However, representing the movements of realistic numbers of travelers within reasonable execution times requires significant computational resources. It also requires relevant methods, architectures and algorithms that respect the characteristics of transportation networks. In this paper, we define two multiagent simulation models representing the existing sequential multiagent traffic simulations. The first model is fundamental diagram-based model, in which travelers do not interact directly and use a fundamental diagram of traffic flow to continuously compute their speeds. The second model is car-following based, in which travelers interact with their neighbors to adapt their speeds to their surrounding environment. Then we define patterns to distribute these simulations in a high-performance environment. The first is agent-based and distributes agents equally between available computation units. The second pattern is environment-based and partitions the environment over the different units. The results show that agent-based distribution is more efficient with fundamental diagram-based model simulations while environment-based distribution is more efficient with car following-based simulations.

1. Introduction

Mobility policies makers need decision support systems to decide which transportation policies they should implement. In this context, simulation is one of the important tools that allow to test strategies and multiple scenarios without impacting the real traffic [1]. However, transport systems are becoming progressively more complex since they are increasingly composed of connected entities (mobile devices, connected vehicles, etc). It becomes critical that simulation tools take into account this fact. Indeed, with the generalization of real-time traveler information, the behavior of modern transport networks becomes harder to analyze and to predict.

For these reasons, multiagent simulation, which adopts an individual-centered approach, is one of the most relevant paradigms to design and implement such applications. The design and development of multiagent traffic 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 [24], to test changes in behavior after the introduction of new mobility services, such as carpooling. A multiagent traffic simulation platform simulates the behavior of travelers interacting in a complex, dynamic and open environment, on which they have a partial

* Corresponding author.
E-mail address: hamza-mahdi.zargayouna@ifsttar.fr (M. Zargayouna).

https://doi.org/10.1016/j.simpat.2018.09.005
Received 6 January 2018; Received in revised form 4 September 2018; Accepted 7 September 2018
Available online 08 September 2018

perception [4]. Each agent tries to find the most efficient route to reach its destination in a network evolving dynamically. In some applications (e.g. [44]), an agent can potentially be informed of the status of the network and use this information to modify its route. In this kind of simulations, it is important to model and simulate a realistic number of travelers to correctly observe the effects of individual decisions. In the European project *Instant Mobility*[1] for instance, the objective was to supply a multimodal platform with individual and multimodal travel queries and dynamic positions of travelers and vehicles. To allow the platform to demonstrate its efficiency in an operational context, we implemented a simulator (called SM4T [43]), which had to be executed with an actual volume of travelers. Other examples where simulations must be scalable concern the testing of new mobility services such as car-pooling, car sharing, dial a ride, evacuation modeling, the exchange of information between connected vehicles, etc.

However, the simulation of an actual number of passengers in a big city (several millions of travelers) comes with a high cost. If the same computing infrastructure is maintained, the cost is a loss in accuracy. Otherwise, the cost is a considerable computing power and an architecture allowing the distribution of computations on many computation units. We desire to keep the same simulation quality and choose the latter approach. However, the majority of current multiagent traffic simulators do not allow such distribution. This induces limitations on the number of simulated travelers, means of transport and the size of the considered networks. Our main objective in this paper is to provide reproducible distribution patterns that could be used by existing and future implementations of multiagent traffic simulations.

In this paper, we propose to study distribution methods for multiagent traffic simulations. We define two multiagent simulation models, representing the main types of multiagent simulations of the literature. The first model is called fundamental diagram-based model, in the sense that travelers do not interact directly but use a fundamental diagram of traffic flow to continuously compute their speeds. This is the choice performed for instance by Zargayouna et al. [45], Meignan et al. [31] and Mahmassani [30] and Cajias et al. [11]. The second model is called car following-based, in which travelers interact with their neighbors to adapt their speeds to their surrounding environment. This is the most common choice performed in the literature for multiagent simulations (e.g. [27,46]). These two models correspond partially to microscopic and mesoscopic traffic simulations. Microscopic models simulate the interactions between individual vehicles, while mesoscopic models simulate small groups of traffic entities, whose interactions are described in a medium level of detail. This paper studies two distribution patterns (agent-based and environment-based) applied to these two simulation models. The objective is to discover the most suitable distribution pattern, provided a chosen multiagent traffic model. The results show that agent-based distribution is more efficient with fundamental diagram-based simulations while environment-based distribution is more efficient with car following-based simulations.

The remainder of this paper is structured as follows. In Section 2, we present the previous proposals for multiagent traffic simulation and the existing distributed multiagent platforms. Section 3 presents a simulator for the execution of both fundamental diagram-based and car following-based multiagent traffic simulations. Section 4 presents the two distribution patterns (agent-based and environment-based) and their application to the two simulation models. Section 5 explains our experimental setup and the results of our simulations. Section 6 concludes this paper.

2. Related work

2.1. The problem

The problem tackled in this paper is to distribute n agents on k connected computation units. The interactions between agents on the same units are less costly then inter-units interactions, which generates messages using the communication network. Let w_i the workload of agent a_i. Let $p(a_i, a_j)$ the interaction costs between agents a_i and a_j. If they are on the same unit, this cost is zero. The workload of a unit U_i is:

$$W(U_i) = \sum_{a_j \in U_i} w_j \tag{1}$$

The communication cost between two units U_i and U_j is:

$$P(U_i, U_j) = \sum_{a_k \in U_i} \sum_{a_l \in U_j} p(a_k, a_l) \tag{2}$$

An ideal distribution of the n agents on the k units is a multi-objective optimization which has to:

- distribute best the workload between units,
- limit to the maximum the communications between units.

Formally, find the distribution minimizing:

$$\sum_{1 \le i \le j \le k} P(U_i, U_j) \text{ with } W(U_i) = \frac{\sum_{1 \le h \le k} W(U_h)}{k} + \epsilon \tag{3}$$

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

Miyata and Ishida [33] perform an integer programming optimization of the problem. Even if this approach allows to find an optimal solution to the distribution problem, the computational complexity increases exponentially with the number of agents, which makes it unusable for large-scale traffic simulations.

Two approaches exist for distributing simulations [36,40]. The first ignores communication costs and focuses on workload, distributing the agents the most equitably possible between units. The second approach is environment-based: it assumes that agents that are physically close would probably interact more. The environment is then split and each unit is responsible of a subpart of the environment and the agents in it.

2.2. State of the art

In this section, we position our work with the previous works in the literature. In the next paragraph, we present the existent multiagent traffic simulators. Then we focus on the existing proposals for distributing these platforms. We finally describe the parallel multiagent platforms.

2.2.1. Multiagent traffic simulation

There exist several multiagent traffic simulations in the literature. Most of them are car following-based models, in the sense that they rely on local interactions between traveler agents to define agents speeds. For instance, Transims [34], MATSim [27], Sumo [46] and Vissim [21] are widely used car following-based simulators of this type. AgentPolis [13], Polaris [3] and Archisim [20] are also multiagent traffic simulation platforms describing precisely the behaviors of each traveler at a microscopic scale. Some existing multiagent simulations are fundamental diagram-based model, in the sense that they compute the agents speeds based on a function mapping the number of agents traveling on an edge with their speeds. This model is generally used when an individual representation of travelers is needed but there are no reliable data about their local behavior. Zargayouna et al. [45], Meignan et al. [31], Cajias et al. [11] and Mahmassani [30] have made this choice.

2.2.2. Distributing traffic simulation

The problem of distributing multiagent traffic simulations has attracted many studies recently. Some previous works have addressed the specific problem of distributing multiagent traffic simulations. Bragard et al. [10] propose dSumo, a distribution platform applied to the Sumo microscopic simulation platform. Lee and Chandrasekar [28] propose a parallel version of Paramics [15]. However, they present small clusters and networks (grid-like).

Some proposals in the literature propose parallel but non-distributed solutions. These simulations are potentially faster but can only work with multi-threaded environment but not with distributed environments. Nagel and Rickert [34] present a parallelization of the Transims platform. They use a master-slave model for synchronizing the different units. Qu and Zhou [35] present parallel-computing framework for mesoscopic transportation simulation. Aimsun [7] also proposes such a parallel but not distributed simulation setup.

All these interesting proposals focus on distributing specific simulation platforms. In the present work, our objective is to propose distributed mechanisms that are independent from specific traffic platforms. In addition, to the best of our knowledge, our work is the only one dealing with the distribution of simulations using the two different interaction models (viz. car following-based model and fundamental diagram-based model).

Some general-purpose multiagent platforms have been specifically developed for large scale simulation in the last years. Mace3j [23], James [26] and Swages [37] are distributed multiagent platforms, but that have no available source or executable code. Jade [9] simplifies the implementation of distributed multiagent models across a FIPA compliant middleware. The platform can be distributed across multiple units and its configuration can be controlled from a remote GUI. Agents are implemented in Java while the communications relay on the RMI library. D-Mason[2] [18] is the distributed version of the Mason multiagent platform. It does not require users to rewrite their already developed simulations while overcoming the limitations on maximum number of agents. RepastHPC [17], a distributed version of Repast Simphony, uses the Repast's concepts of projections and contexts and adapts them for distributed environments. Pandora [2] is close to RepastHPC and automatically generates the code required for inter-server communications. GridABM [25] is based on Repast Simphony but takes another approach and proposes to the programmer general templates to be adapted to the communication topology of her simulation. Flame [16] allows the programmer to generate HPC simulations from finite state machines. It has also been suggested to use graphical units (GPGPU) to scale up the multiagent simulations. The TurtleKit 3 platform has been used in GPGPU [32] and Strippgen and Nagel [39] propose to use it in parallelizing multiagent traffic simulations. However, these distributed platforms do not offer fine controls on how the communications between units are performed. Indeed, the communication layer is transparent for the programmer, which makes it easier for him to implement distributed simulations, but prevents him from optimizing the distribution. The best way to manage the communications depends on the application and using such general platforms for a traffic simulator would not produce optimal results. Rihawi et al. [36] discuss the issues related to multiagent simulation in a distributed virtual environment. The authors describe methods 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 among others an adaptation of this work for a graph structure, adapted to distribute traffic-based simulations.

[2] Distributed Mason.

3. Multiagent traffic simulations

In the following, we present two multiagent traffic simulations. They are designed with the objective of representing the existing multiagent traffic simulations. They contain the main features of these types of simulation, namely the network and agents individual representation and the agents movements on this network. This section presents the common components of these simulations. The following sections present the differences between the two simulations, mainly concerning the travelers speed computation.

3.1. The multiagent system

The multiagent system, which is common to both simulations, is composed of a dynamic set of agents representing travelers, interacting with a transportation network environment. We model the transportation network in which the travelers use a graph $G(V, E)$, where $E = \{e_1, \dots, e_n\}$ is a set of edges representing the roads and $V = \{v_1, \dots, v_n\}$ is a set of vertices representing the intersections. The agents, representing the travelers, move on this network from their origins to their destinations, trying to minimize their travel costs. A traffic simulation usually adheres to the following steps. First, the simulation platform loads the parameters (simulation duration, number of generated agents, etc.) and the description of the network. Then, it creates the logical graph from the network representation, to enable shortest paths calculation and agents movements. The agents then execute one step of simulation. During a step of simulation, each agent either computes a shortest path or moves from one position to another. When an agent reaches its destination, it leaves the simulation. When the simulation duration is reached, the simulation stops and the results are collected.

When created, an agent has an origin node o and a destination node d. The first action that it executes when created and activated is to compute an $A^\star$ shortest path algorithm between o and d. The shortest path is performed on the graph G, which edges costs are dynamic, depending on the current traffic. When it has a current path, the agent moves according to it. At each activation, it moves the allowed distance following its current speed. The speed of the agent is computed following the simulation model (car following-based or fundamental diagram-based model), described in the following sections. Each time it reaches a node, the agent recomputes a shortest path, to check if the current traffic conditions have evolved and if a new shortest path has become available.[3].

These are the main components of the model that are common to both types of simulations. In the following sections, we present the specific methods for the two simulations, namely fundamental diagram-based and car following-based.

3.2. Fundamental diagram-based simulation model

In the fundamental diagram-based simulation model, the speed of an agent on an edge is computed following the number of other agents traveling on the same edge. To this end, a fundamental diagram of traffic flow is used [19] . The diagram defines a relation between the flow (vehicles/hour) and the density (vehicles/km) (cf. Fig. 1) on an edge or a part of an edge 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 move (Fig. 2).

With the distribution objective that we have, the locations of the agents and their interaction patterns are the most important. In the fundamental diagram-based model, the agents do not interact directly. The speed of the agent is computed with an interaction between the agent and the edge. The edge knows the number of agents currently using it, and based on the speed function providing the right speed to be used by the agent, based on the fundamental diagram (cf. Fig. 2).

Zargayouna et al. [45], Meignan et al. [31], Cajias et al. [11] and Mahmassani [30] for instance adhere to this fundamental diagram-based model.

3.3. Car following-based simulation model

In the car following-based simulation model, the speed of an agent on an edge is computed following the position and speed of the vehicles surrounding him (cf. Fig. 3). In this model, the information available to each agent is only local. The agents perceive a part of their environment and calculate their next move given the perceived information. This implies many local communications between the agents, because their actions are conditioned by the actions of the other agents present in their surroundings. This model is generally based on a car-following model described in the following.

At each activation, each agent i computes its speed based on the speed and position of the agent $i - 1$ before him:

$speed_i(t + T) = \alpha speed_i(t) + \beta s_{i,i-1}(t)$

with T the reaction time and $s_{i,i-1}(t)$ the relative speed of i.

If there is no vehicle preceding the agent, it accelerates until it reaches the speed limit of its edge.

In this model, we cannot rely on the fundamental diagram of traffic to continuously have the current travel times on the edges. Instead, each agent registers its experienced travel time when it reaches the end of the edge (as proposed by Wahle et al. [41]). The shortest path calculation is based on the graph where the travel times costs are fed by the agents following this procedure.

In contrast with the fundamental diagram-based model, the agents in the car-following model do interact directly. The speed of the agent is computed with a direct interaction between the considered agent and the agents before him. This difference between the

[3] The graph being directed, turnarounds are only possible at nodes and there is no need for the agent to execute a shortest path while traveling on an edge.

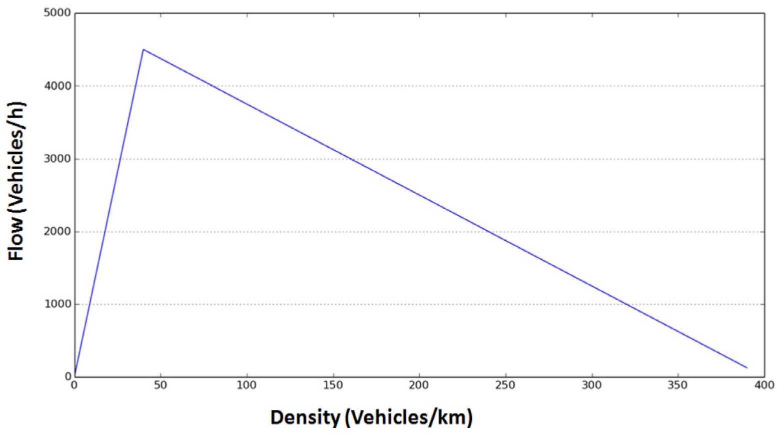

Fig. 1. Fundamental diagram.

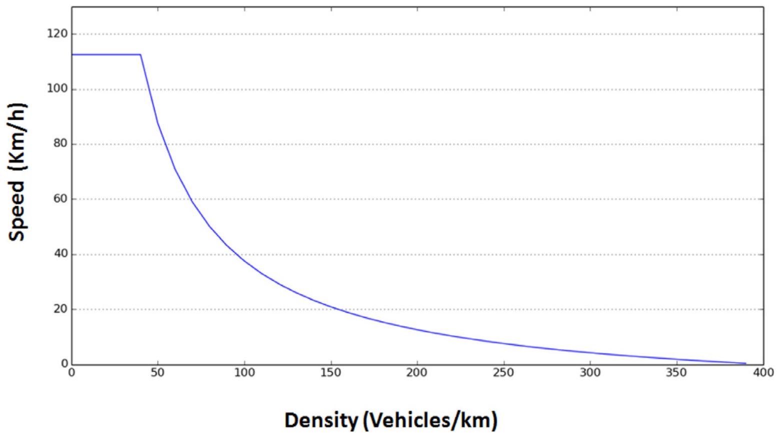

Fig. 2. Speed in function of density.

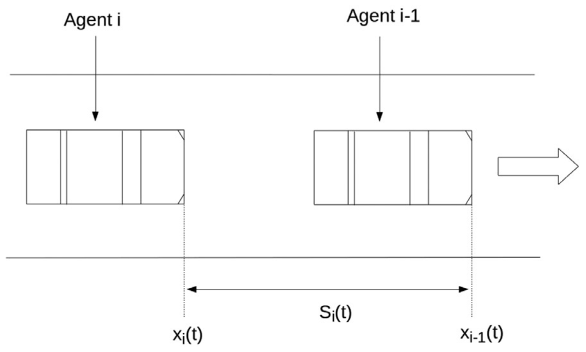

Fig. 3. The car-following model.

two models conditions the choice of the relevant distribution pattern for the considered simulation type. The distribution patterns are described in the following section.

Behrisch et al. [8], Maciejewski and Nagel [29], Doniec et al. [20] and Cameron and Duncan [12] for instance adhere to this car following-based model.

4. The distribution methods

We define two patterns to distribute traffic simulations. The patterns are the same than those identified by Rihawi et al. [36] for general-purpose situated multiagent simulations, and we believe that they present two representative distribution patterns for this

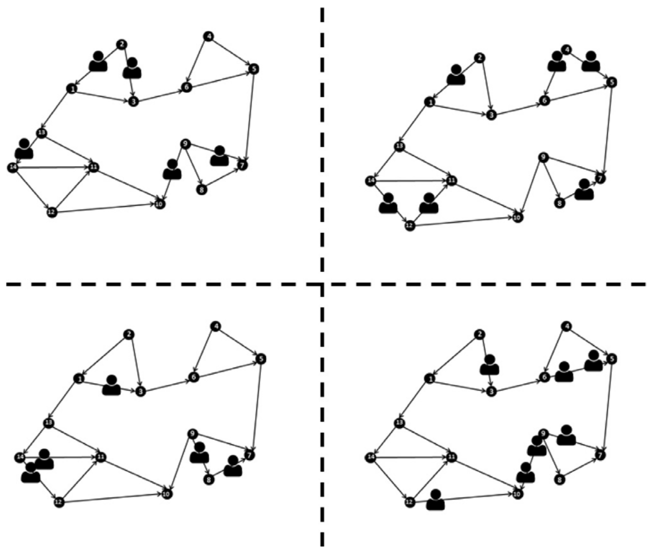
Fig. 4. Agents distribution.

kind of simulations. The first pattern (called agent-based distribution) is the distribution model used by Barceló et al. [6]. It consists in the duplication of the transport environment on all processing units, and the equal dispatching of the agents on each one. As a consequence, agents stay on the same unit during all the simulation. The second pattern is the mostly used pattern in the literature. It consists in partitioning the transportation environment and the dispatching of each subpart of the environment - and all the agents in it - on each processing unit. In this pattern, agents might have to move from one unit to another if their itinerary crosses several subparts of the transport environment.

4.1. Agent-based distribution

The first distribution pattern is agent-based, since it clusters the set of agents in k equal parts (with k the number of available processing units), and distributes each subset on a unit and executes the simulation (cf. Fig. 4). The transportation network is duplicated on each unit. This method ensures that each unit has the same amount of work at any time of the simulation. In the following, we describe the use of this pattern for both simulation models that we have defined.

4.1.1. Fundamental diagram-based simulation with agent-based distribution

In a fundamental diagram-based simulation, when it is distributed following the agent-based distribution pattern, every unit continuously informs the other units of its network state. Indeed, every unit does not have a complete view of the network state, since only a part of the agents evolve in the unit. Thus, every unit sends the list of edges together with the number of agents currently on them. Each unit is then capable of computing the relevant speed for each agent (using the fundamental diagram of traffic) and to compute the shortest paths for the agents.

4.1.2. Car following-based simulation with agent-based distribution

When distributed following the agent-based distribution pattern, the agents in a car following-based simulation do not use a fundamental diagram of traffic to compute their speeds. Instead, they need to know the state of the agents preceding them. To do so, they interrogate the edges in the other units to know if there are agents preceding them, and if it is the case, to know their states. Moreover, the units exchange the current travel times (provided by the agents as explained in the car following-based model), in order to compute the shortest paths for the agents.

4.2. Environment-based distribution

The second approach to distribute traffic simulations is environment-based. Its main idea is that it tries to keep on the same unit the agents who are geographically close in the transport network (cf. Fig. 5). To this end, the network is partitioned in k parts (with k the number of available processing units), and distribute it between the different units. Each unit 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 area are now likely to be on the same unit. If an agent reaches a part of the network that is not managed by its current unit, it moves to the proper unit. In order for the environment distribution method to be effective, each unit has to manage approximately the same number of agents and the number of edges

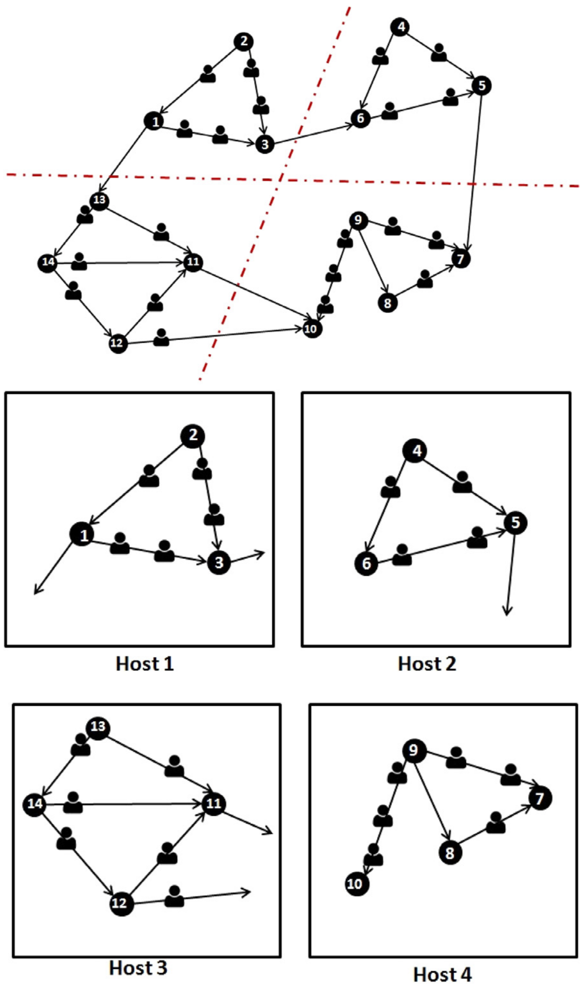

Fig. 5. Environment distribution.

```
Require: Graph $G = (V, E)$, number $k$ of partition
Ensure: Partition P
(1) $P \leftarrow P_0, \ldots, P_{k-1}$
(2) $V' \leftarrow V$
for $p \in [0, k-1]$ do
    (3) $v \leftarrow$ random vertex of $V'$
    (4) $P_p \leftarrow \{v\}$
    (5) $V' \leftarrow V' \setminus \{v\}$
end for
while $|V'| > 0$ do
    (6) $p \leftarrow$ index of the lightest partition
    (7) $m = min_{v \in V'}(1+\epsilon)$(number of $v$'s neighbors $\in P_p$) − (number of $v$'s neighbors $\notin P_p$)
    (8) $mv$ = random vertex of $v \in V' | (1+\epsilon)$(number $v$'s neighbors $\in P_p$) − (number of $v$'s neighbors$\notin P_p$) $= m$
    (9) $P_p \leftarrow P_p \cup \{mv\}$
    (10) $V' \leftarrow V' \setminus \{mv\}$
end while
(11) Return $P$
```

Algorithm 1. *Differential Greedy* algorithm.

connecting the partitions has also to be minimized (to reduce the number of agents being transferred between units).

The problem of graph partitioning has been widely studied in the scientific literature. We propose a method derived from the Differential Greedy algorithm by Fiduccia and Mattheyses [22] that allows us to use the algorithm with weighted vertices while producing more connected partitions (Algorithm 1). For edges partitioning, we make the same choice as Cetin et al. [14] by not cutting edges in the middle. We associate each edge with the partition of its origin node.

The algorithm starts by creating a minimal partition with only one node each (instructions (1)–(5)). Then, while there are nodes to be associated to partitions, the algorithm:

- chooses the lightest partition P_p, in terms of agents present in it (instruction (6))
- finds the nodes that are the most connected with the nodes already in P_p and that are the least connected with the nodes that are not in P_p. The parameter ϵ gives more or less importance to the nodes that are close to the partition (instruction 7).
- chooses one of these nodes, adds it to the partition and removes it from the nodes to process (instructions 8–10).

Our modification of the original differential greedy algorithm concerns the choice of the current partition to treat. The “lightest” partition in the original algorithm concerns the number of nodes in the partition, while in our algorithm, it concerns the number of agents in the partition. The second difference concerns the use of ϵ, which encourages the connectivity of the sub-partitions. Indeed, having sub-partitions that are not connected could lead to agents going back and forth to the same unit while staying in the same neighborhood.

4.2.1. *Fundamental diagram-based simulation with environment-based distribution*

When used with an environment-based distribution, the computation units in the fundamental diagram-based simulation exchange the current travel times on the transport edges, to be able to compute the shortest paths for the agents. However, since all the agents on an edge are present on the same unit, they do not need to exchange the number of agents per edge. The fundamental diagram of traffic and the speeds of the agents can indeed be defined locally.

4.2.2. *Car following-based simulation with environment-based distribution*

When distributed following the environment-based distribution pattern, the agents in a car following-based simulation need to know the state of the agents preceding them. In contrast with the agent-based distribution model, the agents preceding them are by definition present on the same computation unit. The interrogation of the edges is then local to the concerned computation unit. The units keep on exchanging the current travel times (provided by the agents) to compute the shortest paths for the agents.

5. Experiments and results

5.1. *Implementation*

A way to execute a distributed simulation is to define a distributed program where each computation unit, while executing the same program, 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. Indeed, it can be implemented on most parallel architectures and we can deploy the same simulation on larger systems if we need more computing power and memory. We use Python to develop our simulator for its efficiency in quick prototyping. Python is a mature portable language with many well tested scientific libraries and is along with C and Fortran one of the most used languages for high performance computing [38]. 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 use the igraph tools for the modeling of the traffic network in the form of a directed graph. Three python classes are defined. The Simulation class loads the parameters, the graphs and creates the agents. The Context class manage the agents, while the Agent class manages the agents behaviors, including their movements and shortest paths calculations.

We have executed the distributed simulations on an experimental cluster that we have set up. For our tests, we used two units under Linux Mint 17.2 Rafaela (kernel version 3.16.0-38-generic) each with an Intel Xeon processor CPU E7-4820 (32 cores at 2 Ghz) with 250GB of memory. A ssh server is installed on each machine. The ssh protocol is used by MPICH 3.0.4, to encapsulate the MPI messages. We ran the simulations on six configurations: the first is a sequential version of the program on a single core and the five others are distributed versions with 4, 8, 16, 32 and 64 cores. The behaviors in terms of traffic of the sequential and all the distributed versions are rigorously the same.

We have considered two networks. The first is a real network concerning the Paris–Saclay region, France, with 1895 nodes and 3831 edges. The number of daily travelers using this network is around 110,000. The origins-destinations of the travelers are based on real data travel patterns. We consider from 10,000 travelers to 500,000 travelers in our simulations. That means that we represent from around 10% to around 500% of the real volumes of travelers in our simulations. The second is a virtual network of 200 nodes

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

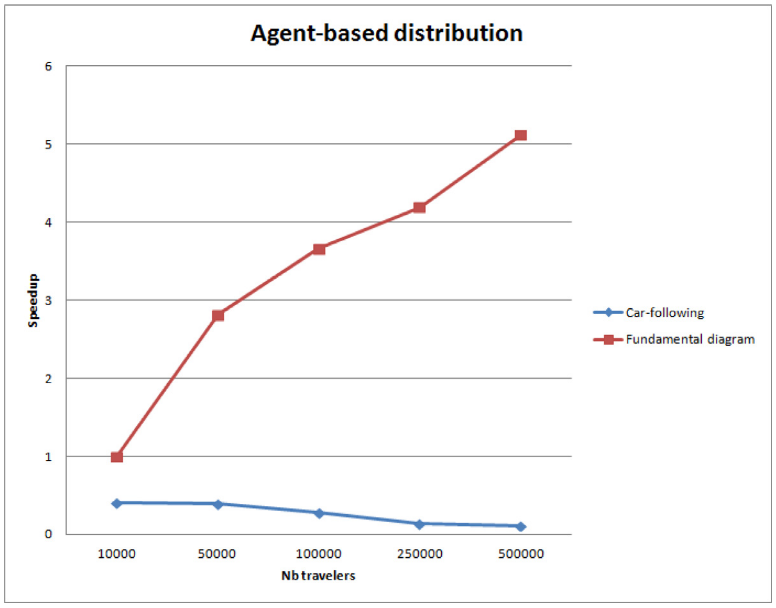

Fig. 6. Speedup for the agent-based distribution (Paris–Saclay Network).

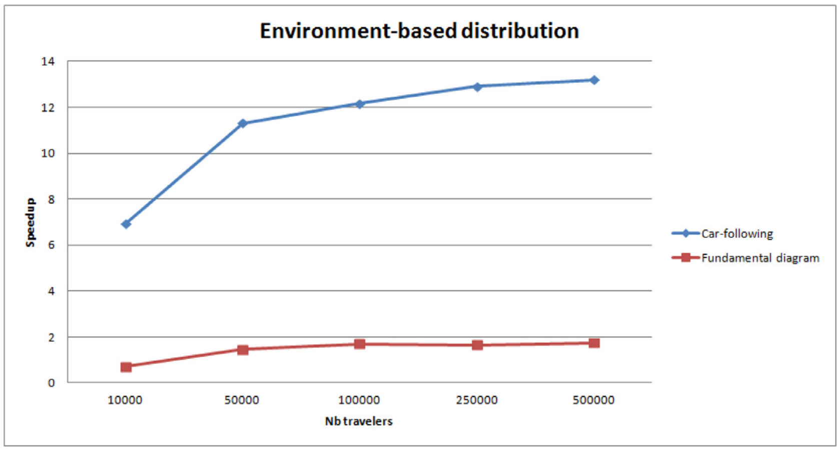

Fig. 7. Speedup for the environment-based distribution (Paris–Saclay Network).

power-law graph generated with the Barabasi and Albert [5] model with random origins and destinations. We have executed each simulation 10 times and provide the average results[5] We consider no lane-changing behaviors in the implemented car-following model.

5.2. Results

5.2.1. Distribution method × simulation model

In this section, we compare the two methods of distribution (agent-based and environment-based distributions) with the different simulation models (car following-based model and fundamental diagram-based model) with increasing the number of agents (from 10,000 to 500,000).

The speedup measures how many times the distributed simulation is faster compared to the corresponding sequential execution. The speedups for the two distributions methods applied on the different paradigms are plotted in Figs. 6 and 7. As we can see, the agent distribution is efficient for a fundamental diagram-based model (more than 5 times faster with 500,000 agents). There is no local interactions in this type of simulations and therefore this method allows to get a perfectly balanced load all along the simulation, while keeping the amount of inter-servers communications at the minimum.

[5] The standard deviation is less than 5% in all the simulations.

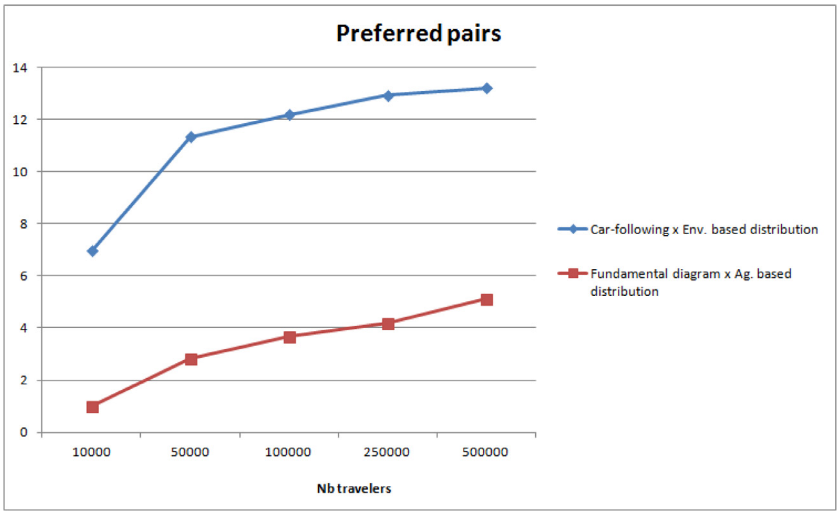

Fig. 8. Preferred pairs (Paris–Saclay Network).

However, this method is particularly ineffective in the case of a car following-based simulation. Indeed, the agents interact continuously with the other agents that are not situated in the same unit. This generates many communications between the servers, and the gain of the parallelization is annihilated by the time required by these communications. This method is even less efficient than the sequential execution for the car following-based model (speedup < 1).

For a fundamental diagram-based simulation, the environment-based distribution is less efficient than agent-based distribution. It is well adapted for a car following-based model simulation though. This method is up to 14 times faster than a sequential execution applied in a car following-based simulation. The explanation of the poor results of the fundamental diagram-based simulations with environment-based distribution is that the communication of edges costs between units, necessary for the computation of vehicles speeds, takes too much time and penalizes these simulations.

Fig. 8 shows the preferred pairs (fundamental diagram with agent-based distribution and car-following with environment-based distribution). The curves show that the pair environment-based distribution with car-following model is more effective than the pair agent-based distribution with fundamental diagram based.

5.2.2. Impact of network type

To assess the impact of network type and size on the distribution, we have executed the same methods on a virtual network of 200 nodes power-law graph generated with the Barabasi–Albert model [5]. Origins and destinations are this time generated randomly. The Fig. 9 provides the results for the agent-based distribution and Fig. 10 provides the results for the environment-based distribution. For agent-based distribution, the findings are the same: the fundamental diagram-based simulations behave way better

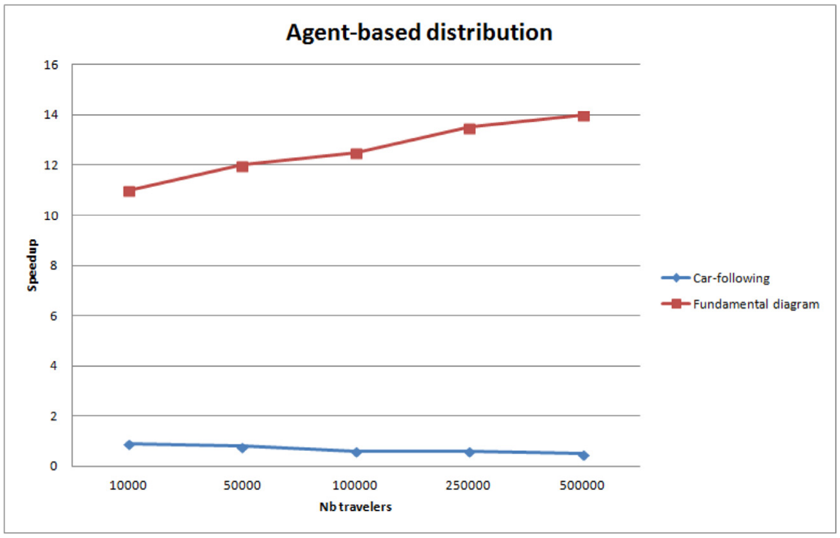

Fig. 9. Speedup for the agent-based distribution (Barabasi network).

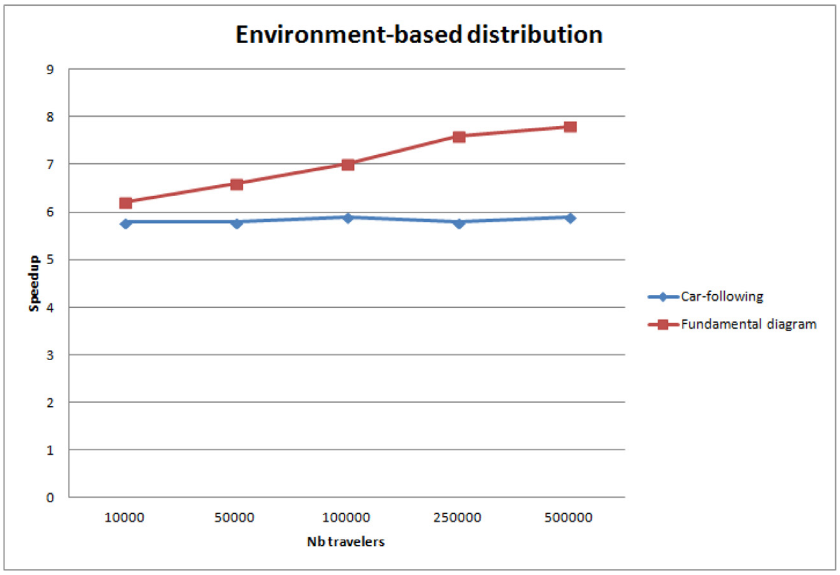

Fig. 10. Speedup for the environment-based distribution (Barabasi network).

than car following-based simulations. However, for environment-based distribution, the findings are different: the two simulations types profit of the distribution at the same level (fundamental diagram-based simulations perform even slightly better). We explain this difference by the network size. Indeed, Paris–Saclay network is ten times bigger than the considered virtual network. The communication of edges costs between units, necessary for the computation of vehicles speeds, is a lot less costly with the virtual network and does not penalize the fundamental diagram-based simulations anymore.

5.2.3. *Increasing units numbers*

To better assess the scalability of the different distribution methods, we execute the different simulations on the Paris-Saclay network with increasing number of available units: 4, 8, 16, 32 and 64. We consider 100,000 travelers in these simulations, which correspond to the real number of daily travelers. The results are reported in Fig. 11 for agent-based distribution and Fig. 12 for environment-based distribution. The results show that the speedup is increasing fast for environment-based distribution with car following-based simulation and for agent-based distribution with fundamental diagram-based simulation. The speedup is stable or increasing slowly for agent-based distribution with car following-based simulation and for environment-based distribution with

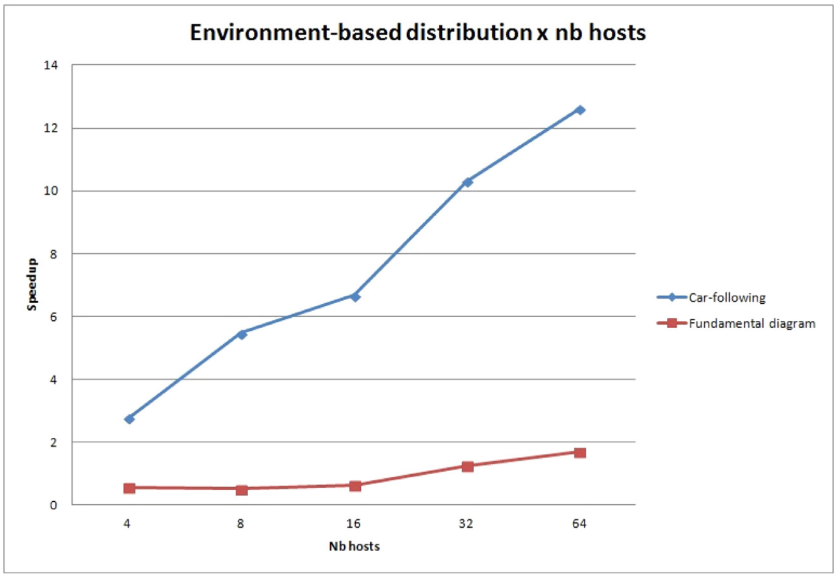

Fig. 11. Impact of the number of units (agent-based distribution).

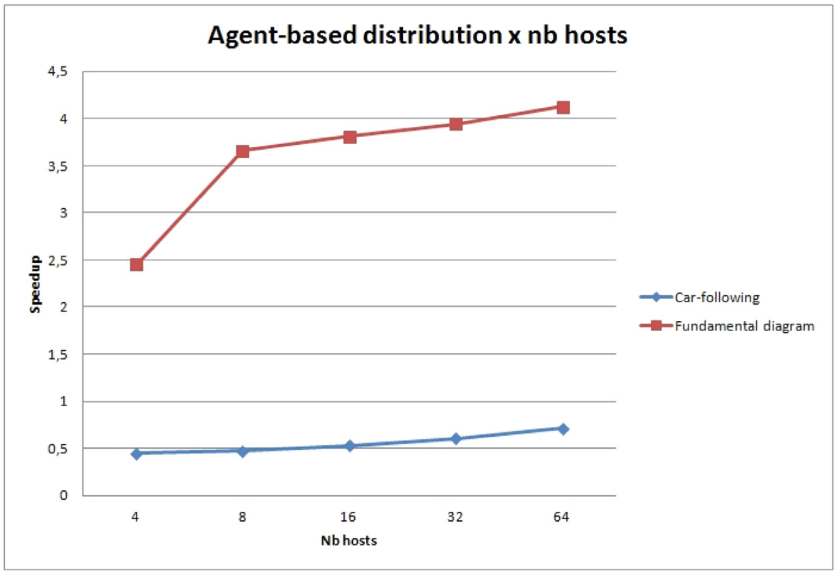

Fig. 12. Impact of the number of units (environment-based distribution).

fundamental diagram-based simulation. These results confirm the findings of Section 5.2.1: environment-based distribution is efficient with car following-based simulation and agent-based distribution is efficient with fundamental diagram-based simulation.

6. Conclusions and perspectives

The work presented in this paper is motivated by the scalability limitations of existing simulation platforms, and specifically of our simulation platform SM4T [45]. We applied two distribution methods on two types of multiagent traffic simulators. The results show that agent-based distribution is well suited for fundamental diagram-based simulators while environment-based distribution is well suited for car following-based simulations. To use the distribution methods presented in this paper, we have to first classify the simulation platform at hand. For SM4T, the simulation is fundamental diagram-based because agent's speeds are computed with a function associated with the edges that they are using. As a consequence, distributing SM4T should be performed following an agent-based distribution model. Of course, the actual implementation of the distributed version of the simulation platform has to be done. But now we know which distribution pattern would me more relevant. This process could be applied with other simulation platforms to be distributed.

In the near future, we will propose a load balancing algorithm that is able to dynamically balance the loads of a traffic simulation. We will also test this method on a cloud-like environment (single core units, linked by a network). We also plan to consider multimodal multiagent traffic simulators. The presence of different transport modes and networks could encourage to mix the patterns presented in this paper with a distribution per transport mode. We are also working on the integration of information networks (such as social networks or intervehicular interaction [42]) and their impact on the distribution performance. Indeed, if travelers interact often, they should be preferably executed on the same units, or else they would generate too many communication and deteriorate the performance of the system. Finally, other aspects than simulation models might impact the distribution, such as the presence of a multi-threading or message-passing environments, and we will consider investigating them. The experiments in this paper were performed on a research simulator, and we look forward to confirming them with other models from the literature.

References

[1] A. Abadi, T. Rajabioun, P.A. Ioannou, Traffic flow prediction for road transportation networks with limited traffic data, IEEE Trans. Intell. Transp. Syst. 16 (2015) 653–662.
[2] E. Angelotti, E. Scalabrin, B. Avila, PANDORA: a multi-agent system using paraconsistent logic, Fourth International Conference on Computational Intelligence and Multimedia Applications, 2001. ICCIMA 2001. Proceedings, (2001), pp. 352–356, https://doi.org/10.1109/ICCIMA.2001.970493.
[3] J. Auld, M. Hope, H. Ley, V. Sokolov, B. Xu, K. Zhang, Polaris: agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations, Transp. Res. Part C 64 (2016) 101–116, https://doi.org/10.1016/j.trc.2015.07.017.
[4] F. Badeig, F. Balbo, G. Scemama, M. Zargayouna, Agent-based coordination model for designing transportation applications, IEEE, 2008. Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on, 402–407.
[5] A.L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512, https://doi.org/10.1126/science.286.5439.509.
[6] J. Barceló, J. Ferrer, D. García, R. Grau, M. Forian, I. Chabini, E. Le Saux, Microscopic traffic simulation for ATT systems analysis. a parallel computing version, Contrib. 25th Ann. CRT (1998).
[7] J. Barcelo, J.L. Ferrer, D. García, M. Florian, E. Le Saux, Parallelization of microscopic traffic simulation for att systems analysis, Equilibrium and advanced transportation modelling, Springer, 1998, pp. 1–26.
[8] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO - simulation of urban MObility - an overview, SIMUL 2011, The Third International Conference on Advances in System Simulation, (2011), pp. 55–60.

[9] F.L. Bellifemine, G. Caire, D. Greenwood, Developing multi-agent systems with JADE, Wiley, 2007.
[10] Q. Bragard, A. Ventresque, L. Murphy, Self-balancing decentralized distributed platform for urban traffic simulation, IEEE Trans. Intell. Transp. Syst. (2016) 1–8.
[11] R. Cajias, A. Gonzalez-Pardo, D. Camacho, A multi-agent traffic simulation framework for evaluating the impact of traffic lights. ICAART (2), (2011), pp. 443–446.
[12] G.D. Cameron, G.I. Duncan, Paramics - parallel microscopic simulation of road traffic, J. Supercomput. 10 (1996) 25–53.
[13] M. Čertickỳ, M. Jakob, R. Píbil, Z. Moler, Agent-based simulation testbed for on-demand mobility services, Procedia Comput. Sci. 32 (2014) 808–815.
[14] N. Cetin, A. Burri, K. Nagel, A large-scale agent-based traffic microsimulation based on queue model, Proceedings of the Swiss Transport Research Conference (STRC), (2003), pp. 1–22.
[15] Y. Chen, G. Chen, K. Wu, Evaluation of performance of bus lanes on urban expressway using paramics micro-simulation model, Procedia Eng. 137 (2016) 523–530.
[16] S. Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth, C. Greenough, Exploitation of high performance computing in the FLAME agent-based simulation framework, 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 and IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), (2012), pp. 538–545, https://doi.org/10.1109/HPCC.2012.79.
[17] N. Collier, M. North, Repast HPC: a platform for large-scale agent-based modeling, in: W. Dubitzky, K. Kurowski, B. Schott (Eds.), Large-Scale Computing, John Wiley & Sons, Inc., 2011, pp. 81–109.
[18] G. Cordasco, R.D. Chiara, A. Mancuso, D. Mazzeo, V. Scarano, C. Spagnuolo, Bringing together efficiency and effectiveness in distributed simulations: The experience with d-mason, 89, 1236–1253, http://sim.sagepub.com/content/89/10/1236, doi:10.1177/0037549713489594.
[19] C.F. Daganzo, The cell transmission model: a dynamic representation of highway traffic consistent with the hydrodynamic theory, Transp. Res. Part B 28 (1994) 269–287.
[20] A. Doniec, R. Mandiau, S. Piechowiak, S. Espié, A behavioral multi-agent model for road traffic simulation, Eng. Appl. AI 21 (2008) 1443–1454.
[21] A. Ehlert, A. Schneck, N. Chanchareon, Junction parameter calibration for mesoscopic simulation in vissim, Transp. Res. Procedia 21 (2017) 216–226.
[22] C. Fiduccia, R. Mattheyses, A linear-time heuristic for improving network partitions, 19th Conference on Design Automation, 1982, (1982), pp. 175–181, https://doi.org/10.1109/DAC.1982.1585498.
[23] L. Gasser, K. Kakugawa, Mace3j: fast flexible distributed simulation of large, large-grain multi-agent systems, ACM, 2002. Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2, 745–752.
[24] M. Guériau, B. Dafflon, F. Gechter, Vips: a simulator for platoon system evaluation, Simul. Modell. Pract. Theory 77 (2017) 157–176.
[25] L. Gulyas, G. Szemes, G. Kampis, W. de Back, A modeler-friendly API for ABM partitioning, ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, (2009), pp. 219–226, https://doi.org/10.1115/DETC2009-86650.
[26] J. Himmelspach, A.M. Uhrmacher, Plug'n simulate, IEEE, 2007. Simulation Symposium, 2007. ANSS'07. 40th Annual, 137–143.
[27] N. Kühnela, T. Thunigb, K. Nagelb, Implementing an adaptive traffic signal control algorithm in an agent-based transport simulation, Procedia Comput. Sci. 130 (2018) 894–899.
[28] D.H. Lee, P. Chandrasekar, A framework for parallel traffic simulation using multiple instancing of a simulation program, ITS J. 7 (2002) 279–294.
[29] M. Maciejewski, K. Nagel, Towards multi-agent simulation of the dynamic vehicle routing problem in matsim, Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics - Volume Part II, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 551–560.
[30] H.S. Mahmassani, Dynamic network traffic assignment and simulation methodology for advanced system management applications, Networks Spatial Econ. 1 (2001) 267–292.
[31] D. Meignan, O. Simonin, A. Koukam, Simulation and evaluation of urban bus-networks using a multiagent approach, Simul. Modell. Pract. Theory 15 (2007) 659–671, https://doi.org/10.1016/j.simpat.2007.02.005.
[32] 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 (2014) 485–510, https://doi.org/10.3166/ria.28.485-510.
[33] N. Miyata, T. Ishida, Community-based load balancing for massively multi-agent systems, Springer, 2007. International Conference on Autonomous Agents and Multiagent Systems, 28–42.
[34] K. Nagel, M. Rickert, Parallel implementation of the transims micro-simulation, Parallel Comput. 27 (2001) 1611–1639.
[35] Y. Qu, X. Zhou, Large-scale dynamic transportation network simulation: a space-time-event parallel computing approach, Transp. Res. Part C. 75 (2017) 1–16, https://doi.org/10.1016/j.trc.2016.12.003.
[36] O. Rihawi, Y. Secq, P. Mathieu, Effective distribution of large scale situated agent-based simulations, ICAART 2014 - Proceedings of the 6th International Conference on Agents and Artificial Intelligence, Volume 1, ESEO, Angers, Loire Valley, France, 6–8 March, 2014, (2014), pp. 312–319, https://doi.org/10.5220/0004756903120319.
[37] M. Scheutz, P. Schermerhorn, R. Connaughton, A. Dingler, Swages-an extendable distributed experimentation system for large-scale agent-based alife simulations, Proc. Artif. Life X (2006) 412–419.
[38] N. Shivashankar, V. Natarajan, Efficient software for programmable visual analysis using morse-smale complexes, Springer, 2015. Topological Methods in Data Analysis and Visualization, 317–331.
[39] D. Strippgen, K. Nagel, Multi-agent traffic simulation with cuda, IEEE, 2009. High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, 106–114.
[40] G. Vigueras, M. Lozano, J.M. Orduña, F. Grimaldo, A comparative study of partitioning methods for crowd simulations, Appl. Soft Comput. 10 (2010) 225–235.
[41] J. Wahle, A.L.C. Bazzan, F. Klügl, M. Schreckenberg, The impact of real-time information in a two-route scenario using agent-based simulation, Transp. Res. Part C 10 (2002) 399–417.
[42] M. Zargayouna, F. Balbo, K. Ndiaye, Generic model for resource allocation in transportation. application to urban parking management, Transp. Res. Part C 71 (2016) 538–554.
[43] M. Zargayouna, A. Othman, G. Scemama, B. Zeddini, Multiagent simulation of real-time passenger information on transit networks, IEEE Intell. Transp. Syst. Mag. V (2018) 20–pages. To appear.
[44] M. Zargayouna, B. Zeddini, G. Scemama, A. Othman, Agent-based simulator for travelers multimodal mobility, Front. Artif. Intell. Appl. 252 (2013) pp81–90.
[45] M. Zargayouna, B. Zeddini, G. Scemama, A. Othman, Simulating the impact of future internet on multimodal mobility, IEEE Computer Society, 2014. The 11th ACS/IEEE International Conference on Computer Systems and Applications AICCSA'2014, 230–237.
[46] J. Zhang, A. El Kamel, Virtual traffic simulation with neural network learned mobility model, Adv. Eng. Software 115 (2018) 103–111.