2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

An Agent-Based Micro-Simulator for ITS

Behnam Torabi and Rym Z. Wenkstern
Department of Computer Science
University of Texas at Dallas
Richardson, TX, 75080
{behnam.torabi, rymw}@utdallas.edu

Mohammad Al-Zinati
Jordan University of Science and Technology
Irbid, Jordan
mhzinati@just.edu.jo

***Abstract*—In this paper we present MATISSE 3.0, a microscopic simulator for the simulation of Agent-based intelligent Transportation Systems (ATS). In MATISSE, vehicles and intersection controllers are modeled as virtual agents which perceive their surroundings through various sensors. ATS-enabled vehicles and controllers interact with one another through V2X and I2I communication mechanisms. We discuss the limitations of well-known microscopic simulators and present the features that allow MATISSE to simulate realistic ATS scenarios.**

I. Introduction

In this paper we present MATISSE 3.0 (**M**ulti-**A**gent based **T**raff**I**c **S**afety **S**imulation syst**E**m), a multi-agent microscopic traffic simulation system for the simulation of a new category of ITS that we call *Agent-based ITS* (ATS) [1]. An ATS is an autonomous traffic infrastructure which consists of:

1) *standard and ATS-enabled vehicles*. ATS-enabled vehicles are equipped with agent-based systems and sensors that allow them to monitor the driver's behavior, communicate with other vehicles, and communicate with smart traffic control devices.
2) *standard and ATS-enabled intersection controllers*. ATS-enabled controllers are equipped with agent-based systems and sensors that allow them to monitor the traffic at their intersection, communicate with other controllers, and communicate with ATS-enabled vehicles.
3) *Zone managers* which are specialized agents responsible for analyzing traffic data and communicating with ATS-enabled vehicles and intersection controllers when necessary.

In MATISSE 3.0, vehicles, intersection controllers and zone managers are modeled as virtual agents which sense their environment and if enabled, communicate and interact with one another in simulated real-time. The simulations can be visualized concurrently in 2D and 3D, and the user can modify the agents' and traffic environment's properties at run-time without interrupting the simulation. Experimental results show that 4000 virtual agents situated in a complex traffic environment can be executed concurrently on a single PC.

In the following section we discuss related works. In Section III we give an overview of MATISSE's architecture. In Section IV we discuss MATISSE's main features and in Section V we present some experimental results.

II. Related Work

A large number of microscopic traffic simulators have been proposed in the literature [8], [4], [6], [3], [5], [7], [5]. In this section, we discuss the widely used MATSim [6], MITSIMLab [3], VISSIM [5] and SUMO [7] in the context of Agent-based ITS (ATS).

A. Agent Features

1) Agents: MATSim is the only simulator classified as agent-based. Unfortunately, it does not support the features intrinsic to agents, i.e., the ability to: a) sense the environment and make decisions in simulated real-time; b) function with limited knowledge about the environment; and c) collaborate with other agents to achieve specific goals. Although MITSIMLab, VISSIM and SUMO are not agent-based, they come with extension mechanisms that allow the "agentification" (at some level) of virtual vehicles and traffic lights. Through these extensions, the user can override the standard vehicle and traffic signal behaviors. This task is strenuous and requires advanced programming knowledge. In addition, the implementation of new types of agents is not possible.

2) Communication: VISSIM is the only simulator that provides communication mechanisms (V2V and V2I). Although coordination and collaboration between simulated traffic devices is possible in MITSIMLab, this is achieved through a centralized component rather than through direct communication. None of the aforementioned simulators offers mechanisms to simulate I2I communications.

3) Knowledge Acquisition: Knowledge acquisition refers to the mechanism used by a traffic element to acquire knowledge about its environment. In realistic ITS simulations, knowledge is acquired either through sensors or communication. MATSIM provides the simulated vehicles with global knowledge about their environment. MITSIMLab provides simulated vehicles with partial knowledge (i.e., a vehicle is aware of everything within a pre-defined range). Only VISSIM offers real-time knowledge acquisition through sensors, V2V or V2I.

978-1-7281-0323-5/18/$31.00 ©2018 IEEE

B. Traffic Simulation Features

1) Traffic Network Definition: SUMO, MATSIM and VISSIM provide the ability to import digital traffic maps from different sources such as OpenStreetMap. The imported maps are often incomplete (e.g., missing details about the number of lanes or lanes connections at intersections) and therefore SUMO and VISSIM use heuristics to determine the missing information. Given that the graph structure of SUMO and VISSIM defines a junction as one node in the graph, it is not possible to obtain accurate models of complex real-world intersections.

2) Run-time User Interaction: This feature refers to the user's ability to modify the simulation settings without the need to restart the simulation. Only SUMO provides limited mechanisms to modify some of the simulated environment configurations at run-time. For example, the TraCI API allows the modification of lane configurations (e.g., maximum speed, vehicles prohibited on a lane). MATSIM is the only simulator that offers the capability to trigger external events (e.g., lane closure) during the execution of the simulation. However, the user must assign the time and location of an event and predefine its effect before the simulation starts.

3) Vehicle Dynamics - Vehicle behavior: With the exception of MATSIM, all of the aforementioned traffic simulators incorporate car-following and lane-changing models. In VISSIM, car following is simulated using the model proposed in [10]. In this model, vehicles respond based on the distance and speed of the vehicles ahead. In VISSIM, it is possible to define properties such as safety distance for a class of vehicles. A rule-based model which can also be configured is used for lane changing.

4) Simulating accidents: Accident simulation is crucial for the modeling of realistic traffic dynamics [2]. Only MITSIMLab offers the capability of modeling collisions. In MITSIMLab an accident is defined before the execution of the simulation. It is assigned a start time, expected duration, position, number of lanes affected or blocked, severity, length, and the maximum speed of the vehicles passing by an accident. Predefined accidents do not allow for the simulation of realistic scenarios.

III. Overview of MATISSE's 3.0

A. High Level Architecture

MATISSE's architecture consists of three building blocks (see Figure 1) [1]. The simulator's main constituent is the *Simulation System* which includes three subsystems:

1) The *Agent System* creates and manages simulated standard and ATS-enabled vehicles and intersection controllers as well as zone managers. The various agent types communicate through the Agent-to-Agent Message Transport Service;
2) The *Environment System* creates and manages the traffic network;
3) The *Simulation Microkernel* manages the simulation workflow.

The *Control and Visualization System* renders 2D and 3D representations of the simulation and provides real-time interaction mechanisms. The *Message Transport Service* provides a configurable messaging infrastructure that allows MATISSE's building blocks to exchange information.

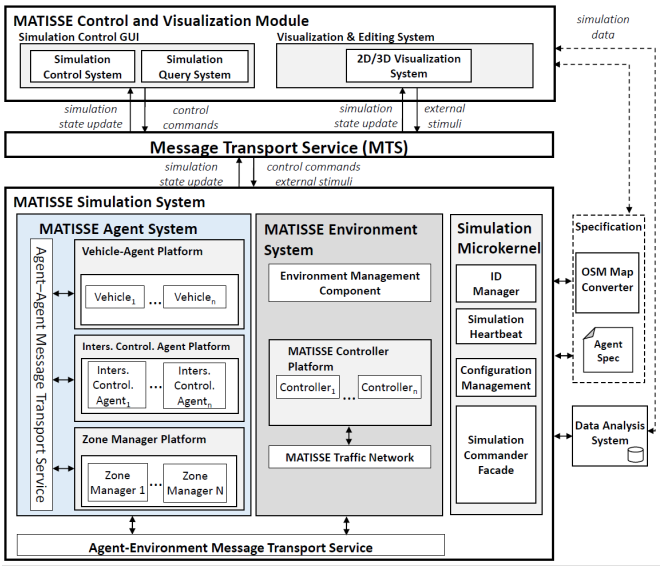

Fig. 1. MATISSE's High-Level Architecture

B. Traffic Network Structure

In MATISSE, a traffic network is specified as a directed graph where nodes represent intersections or connections between road segments, and directed edges represent road segments. The graph defines the possible traffic movements between lanes in consecutive roads. Figure 2 (a) shows the graph definition that corresponds to the traffic network illustrated in Figure 2 (b).

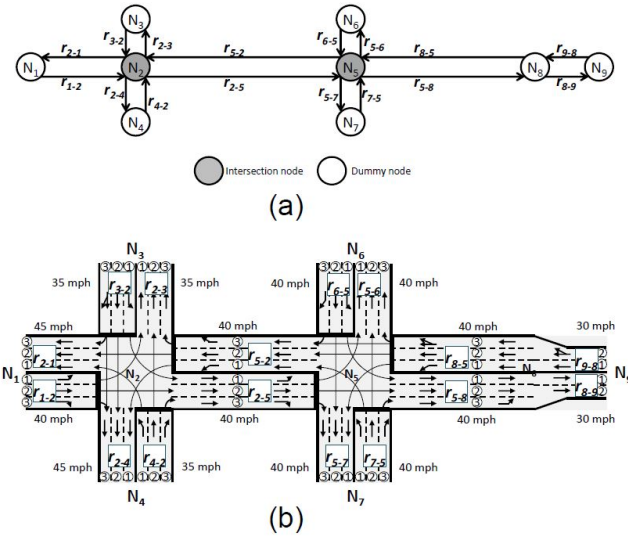

Fig. 2. Traffic Network Definition in MATISSE

IV. MATISSE's 3.0 Features

A. Creating Virtual Agents

MATISSE was built as a multi-agent simulation system from the ground up. It provides several predefined concrete classes for vehicle agents. These classes can be instantiated to create various types of virtual traffic agents equipped with diverse sensing and communication mechanisms. The modeler can also easily create new types of agents by implementing concrete classes extending from abstract classes provided in MATISSE's *vehicle agent package*. The vehicle agent package consist of four main modules:

1) *Interaction module*: This module handles the vehicle agents interaction with external entities. It consists of three sub-modules: a) The *perception module* implements mechanisms necessary for a vehicle agent to perceive the traffic environment using sensors; b) The *communication module* handles all communications between the vehicle agent and other simulated agents; and c) The *route guidance module* implements mechanism that allow a vehicle agent to access the traffic network information (e.g., route guidance, congestion levels on roads).
2) *Knowledge module*: This module represents the vehicle agents memory. It includes the vehicle agents knowledge about itself (e.g., acceleration, deceleration capabilities, maximum speed) and knowledge acquired through sensing and communication (e.g., approaching vehicle).
3) *Task module*: This module defines the tasks that a vehicle agent can perform. *MoveTask* and *TurnTask* are used by the vehicle to travel in the environment.
4) *Planning module*: This module implements different vehicle agent planning strategies. The *TravelRoutePlanningModule* implements plans used by the vehicle to find a travel route, while the *MovementDynamicsPlanningStrategy* computes a set of possible actions, their reward and risk values and selects an action based on the driving behavior.

Figure 3 shows a section of the code for a vehicle agent. A demo illustrating MATISSE's features is available at: *mavs.utdallas.edu/its*

B. Importing Traffic Networks

In MATISSE, the modeler can either create a virtual traffic network through a graphical interface by "snapping" road segments or by importing entire networks through Open Street Map (OSM). The modeler can also import a section of an OSM network from a file system or an online viewer. Several advanced algorithms have been developed to reliably convert OSM graphs to MATISSE graphs, and automatically generate missing information, e.g., number of road lanes, traffic light locations, and allowable traffic movements. In this section, we give an overview of OSM network structures and some of MATISSE's conversion algorithms.

```
public class VehicleAgent

extends AbstractAgent<VehicleAgentState, VehicleKnowledgeModule, VehicleInteractionModule,
    VehiclePlanningModule, VehicleTaskModule>
implements Serializable
{
    public VehicleAgent(VehicleAgentState state, CellBounds cellBounds) {
        super(state);
    }
    @Override
    protected VehicleInteractionModule createInteractionModule(VehicleKnowledgeModule knowledgeModule) {
        return new VehicleInteractionModule(new VehiclePerceptionModule(knowledgeModule),
                new SimpleAgentCommunicationModule(knowledgeModule.getId()));
    }
    @Override
    protected VehicleKnowledgeModule createKnowledgeModule(VehicleAgentState state) {
        return new VehicleKnowledgeModule(state);
    }
    @Override
    protected VehiclePlanningModule createPlanningModule(VehicleKnowledgeModule knowledgeModule,
            VehicleTaskModule taskModule, VehicleInteractionModule interactionModule) {
        VehiclePlanGenerator planGenerator =
                new VehiclePlanGenerator(knowledgeModule, taskModule, interactionModule);
        VehiclePlanExecutor planExecutor =
                new VehiclePlanExecutor(knowledgeModule);
        return new VehiclePlanningModule(planGenerator, knowledgeModule, planExecutor);
    }

    @Override protected VehicleTaskModule createTaskModule(VehicleKnowledgeModule arg0) {
        return new VehicleTaskModule(arg0);
    }
```

Fig. 3. Code for vehicle Agent

1) OSM Network Structure: OSM networks are in the form of XML formatted files. They contain three types of elements: a) *node*, b) *ways* and c) *relations*. Elements can have *tags* and *keys* to describe their features. A *node* holds coordinates of a location. A *way* is an ordered list of *nodes*. A *way* can either be *open* or *closed*. A *way* is closed if it has the same first and last *nodes*. It is open otherwise. *Relations* are included in an OSM file to describe the relations between *ways* and *nodes*. In OSM file format, *ways* that are marked as "highway" represent roads.

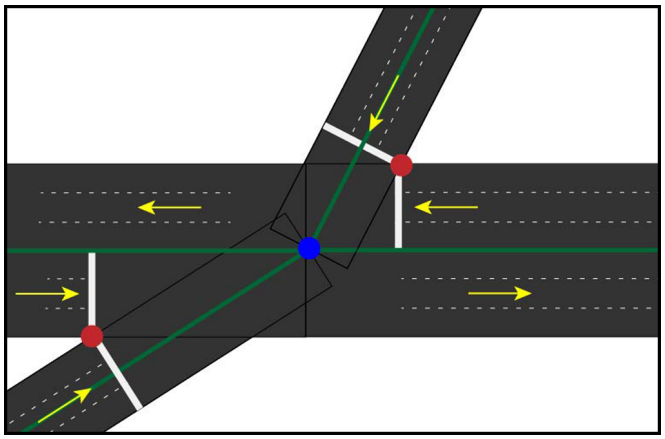

Fig. 4. Generate an intersection from an OSM file

2) Overview of MATISSE's Conversion Algorithm: To convert an OSM network, MATISSE first reads the OSM file and extracts *ways* that represent roads. Then, it adds nodes associated with roads into its network graph. Next, it connects the nodes in the order that is specified in the OSM file. Then, it widens the roads depending on their number of lanes. The number of lanes is usually specified in OSM files. In case it is undefined, MATISSE estimates it based on the road type.

In certain cases, widening roads creates overlaps between road surfaces at intersections (see Figure 4). For these cases, stop bar positions have to be computed. MATISSE places a stop bar at the position where a road surface crosses the surface of its adjacent roads. In figure 4, red circles show crossing points.

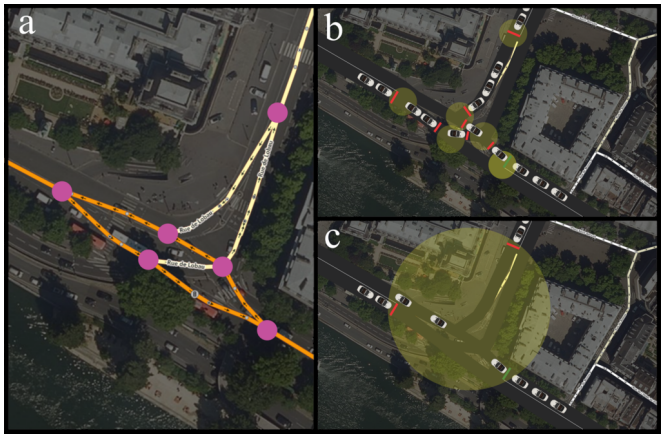

Fig. 5. An Intersection in Paris and Its Representation in Open Street Map

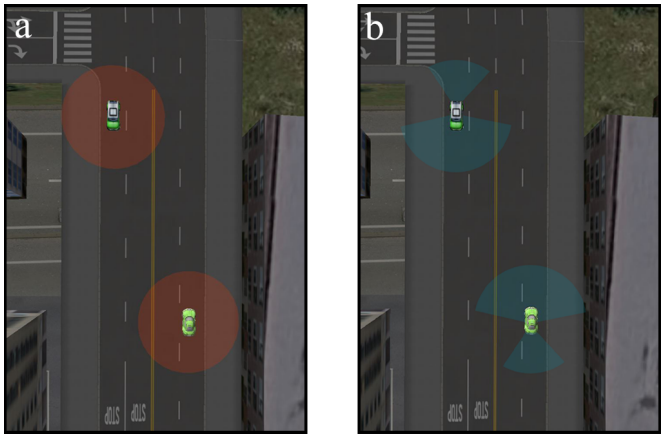

Fig. 6. Vehicle agents perception. a) Circle of Influence b) Vision cones

After forming intersections, signalized intersections need to be determined. As discussed in Section II, most micro-simulators represent a signalized intersection with one node in their network structure. However, in Open Street Map a signalized intersection is not necessarily represented by one node, but often with an arbitrary number of nodes. Figure 5(a) shows an OSM graph for a complex signalized intersection in Paris, France. The single signalized intersection is represented using 6 nodes. Micro-simulators such as SUMO or VISSIM convert each of the OSM nodes into one signalized intersection which results in an incorrect representation of the real network topology (see Figure 5(b)). As shown in Figure 5(c), MATISSE's network structure and conversion algorithms allow an accurate conversion of the information.

C. Initial Vehicle Distribution

At the start of the simulation, the user defines the total number of vehicles to be run as well as entry and exit points. The initial vehicle distribution can be automatically generated by MATISSE or specified by the user through a graphical interface.

D. Vehicle and Intersection Controller Agents

Virtual ATS-enabled vehicles and intersection controllers are equipped with sensors and perceive the environment within their sensor range, called *circle-of-influence* (COI). They are able to communicate with other enabled vehicles and intersection controllers located within their COI (see Figure 6(a)).

For standard virtual vehicles, a *vision cone* is used to simulate a human driver's vision range and perception of the environment. Other virtual human sensors such as auditory and olfactory sensors are available. The virtual driver's *level of distraction* is directly related to its level of perception of its direct environment (See Figure 6(b)).

The ranges of the various sensors can be altered during the execution of simulation.

E. Vehicle Behavior

Unlike most simulators which use predefined car-following and lane-changing algorithms, in MATISSE 3.0, vehicle agents compute the set of possible actions based on their current perception of the environment, the maximum acceleration and deceleration rates and their flexibility in steering. They assign a reward and a risk value to each action. The reward indicates the impact that an action has in helping the vehicle achieve its objectives (e.g., reach its destination as fast as possible, follow traffic rules, drive smoothly). For each action, the agent also assesses the risk that an action may result in a collision. For normal driving behavior, the agent ignores the actions that are considered dangerous and executes the action with the highest reward.

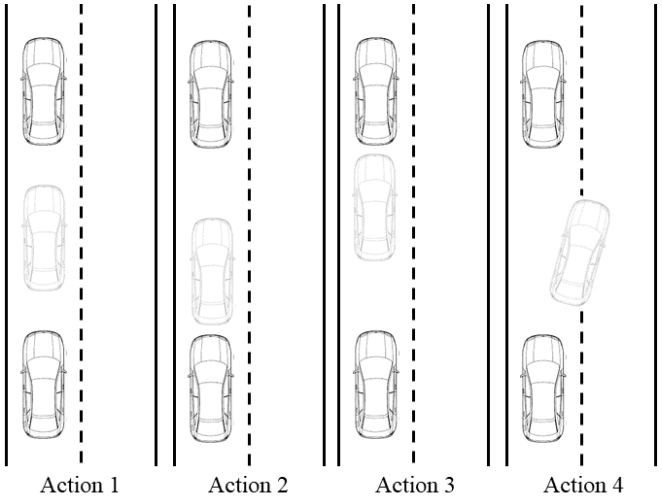

Fig. 7. Four possible actions of a vehicle

Various driving behaviors can be simulated by assigning different values to the risk-aversion factor and the importance of an objective in the reward values. Figure 7 shows a scenario where the vehicle agent in the back can take different actions. In *Action 1* a vehicle agent maintains its current speed and steering. In *Action 2* it maintains current steering and decreases its speed. In *Action 3* it maintains current steering and increases its speed. Finally, in *Action 4* it maintains its current speed and steers the vehicle 20 degrees to the right. If we assume that with respect to the vehicle

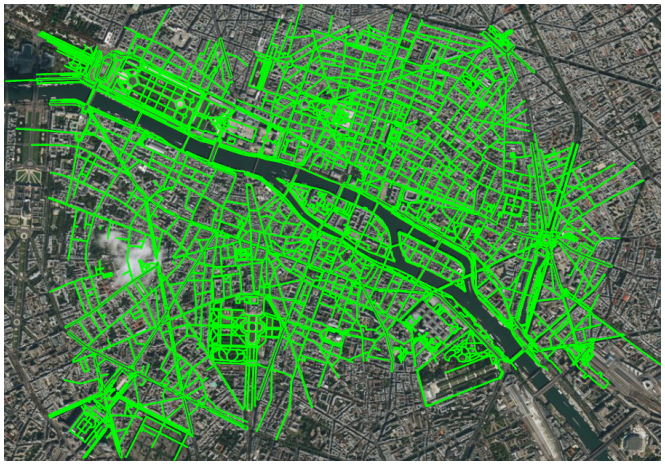

Fig. 8. 2D visualization of Paris's Traffic Network.

agent's objectives, *Actions 3* has the highest reward and *Action 4* has the lowest reward, and with respect to collision, *Action 3* has the highest risk and *Action 2* has the lowest risk then, among the four actions, an aggressive agent will choose *Action 3*, a defensive agent will choose *Action 2*, a regular agent will select *Action 1*, and a careless agent will choose *Action 4*.

F. Agent Property Modification At Run Time

With MATISSE, the user can modify agent properties such as circle-of-influence, vision range and speed, and display these properties at run-time. Also, the user can track agents by their IDs and change properties of a group of agents.

During the simulation, the user can modify the driver's level of distraction. This may dynamically introduce unexpected accidents and unpredicted traffic behavior.

V. Experimental Results

The experiments discussed in this section were run on a multicore PC (Intel Core i7 X980 CPU (3.33GHz), 6.00 GB, 64-bit Windows 7, 12GB Memory DDR3 2673.2MHz, NVIDIA GeForce GTX 480 GPU). A simulated model of Paris's road network was produced in MATISSE 3.0. The model includes 2981 road segments and 231 signalized intersections in addition to the 665 non-signalized intersections. Figure 8 shows a 2-D representation of the traffic network. Tables I shows the number of road segments, classified based on the number of lanes. Tables II and III summarize the types of signalized and non signalized intersections, classified based on the number of incoming and outgoing lanes.

TABLE I
Number of Road Segments with Various Number of Lanes

Type	1	2	3	4	5	6
Count	1820	540	348	254	13	6

TABLE II
Number of Signalized Intersection with various incoming and outgoing lanes

Type	1×1	1×2	1×3	2×2	2×3	3×3	*Other*
Count	33	63	63	3	6	3	60

TABLE III
Number of Non-Signalized Intersection with various incoming and outgoing lanes

Type	1×1	1×2	1×3	2×2	2×3	3×3	*Other*
Count	372	110	78	4	3	0	98

Experiment 1: Simulating ATS Traffic Scenarios
The purpose of this experiment is to get an initial evaluation of *road usage* and *communication costs* for traffic scenarios involving a combination of ATS-enabled and standard vehicles.

For the assessment of road usage, vehicles were added to the simulation until the road network reached its capacity. Figure (9) shows the average road occupancy for different percentages of autonomous vehicles. Road usage reaches its higher level when all vehicles are ATS-enabled.
For the assessment of communication costs, the simulations were run with 3000 vehicles. Figure (10) shows the number of exchanged messages between vehicles for various distributions of ATS-enabled vehicles. As expected, the number of exchanges messages increases exponentially when the number of ATS-enabled vehicles increases.

Advanced experimental results on the assessment of agent-based traffic congestion reduction using MATISSE 3.0 can be found in [9].

Experiment 2: Simulation Scalability
The purpose of this experiment is to evaluate MATISSE's performance in simulating large scale agent-based traffic scenarios. The experiment starts with an empty network of the city of Paris. Ten ATS-enabled virtual vehicles are

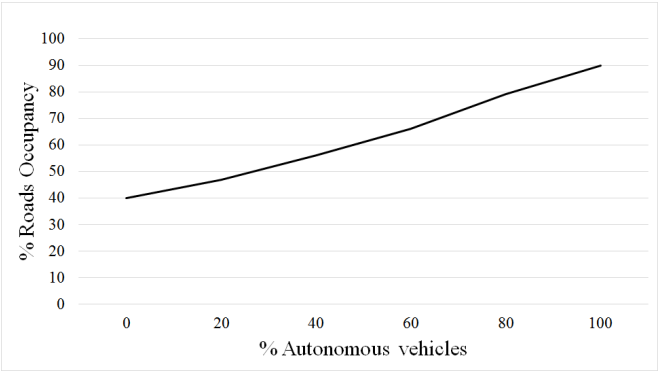

Fig. 9. Road Occupancy for Different Number of Autonomous Vehicles

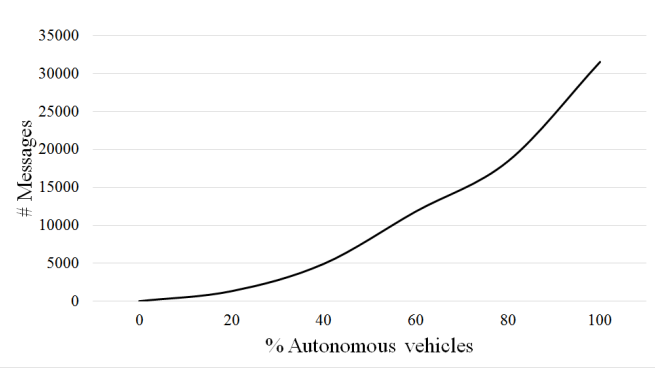

Fig. 10. Number of Messages for Different Number of Autonomous Vehicles

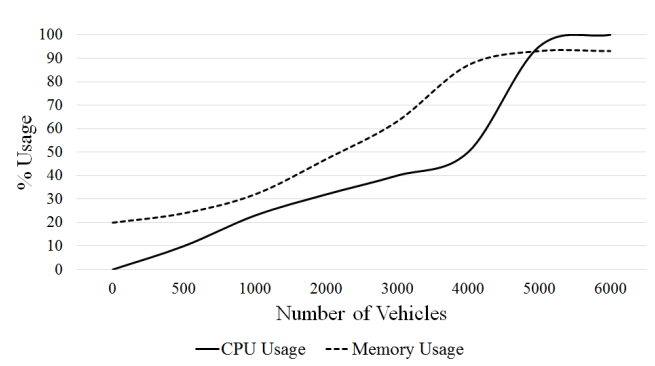

Fig. 11. Memory and CPU Usasge for Different Number of Vehicles in the Simulation

added to the simulation every second until the number of vehicles reaches 6000. Figure (11) shows the performance of MATISSE in terms of CPU and memory usage. Initially, 20% of the memory is used by the operating system and MATISSE's 3.0 core module. As expected, as the number of vehicles increases (between 1000 and 4000 vehicles), the memory usage increases until it reaches 90%. With respect to CPU usage, the increase is not as important. Nevertheless, we notice that when the memory usage reaches 90%, the CPU usage increases drastically. This is due to thrashing. When the number of vehicles is higher than 4000, both CPU and memory usage reach their maximum value and the simulation performance is drastically impacted. Based on these results, we conclude that the memory size is the limitation for MATISSE.

VI. Conclusion

In this paper we presented MATISSE 3.0, a microscopic simulator for the simulation of Agent-based Intelligent Transportation Systems (ATS). We discussed the core features that allow MATISSE to simulate realistic ATS scenarios. Experimental results show that MATISSE can execute simulations with up to 4000 ATS-enabled vehicles on a single PC.

Various advanced agent-based algorithms for traffic congestion reduction and urban evacuation have been developed and validated in MATISSE. Our goal is to continue experimenting with the simulator and investigate the development of a capability that will allow the integration of real-time data into the simulation.

References

[1] Mohammad Al-Zinati and Rym Z Wenkster. Simulation of traffic network re-organization operations. In *Distributed Simulation and Real Time Applications (DS-RT), 2016 IEEE/ACM 20th International Symposium on*, pages 178–186. IEEE, 2016.

[2] Rahaf Alsnih and Peter Stopher. Review of procedures associated with devising emergency evacuation plans. *Transportation Research Record: Journal of the Transportation Research Board*, (1865):89–97, 2004.

[3] Moshe Ben-Akiva, Haris N Koutsopoulos, Tomer Toledo, Qi Yang, Charisma F Choudhury, Constantinos Antoniou, and Ramachandran Balakrishna. Traffic simulation with mitsimlab. In *Fundamentals of Traffic Simulation*, pages 233–268. Springer, 2010.

[4] Bruno Castro da Silva, Ana LC Bazzan, Gustavo K Andriotti, Filipe Lopes, and Denise de Oliveira. Itsumo: an intelligent transportation system for urban mobility. In *International Workshop on Innovative Internet Community Systems*, pages 224–235. Springer, 2004.

[5] Martin Fellendorf. Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. In *64th Institute of Transportation Engineers Annual Meeting*, pages 1–9. Springer, 1994.

[6] Andreas Horni, Kai Nagel, and Kay W Axhausen. *The multi-agent transport simulation MATSim*. Ubiquity Press London, 2016.

[7] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3&4), 2012.

[8] Jeffrey Miller and Ellis Horowitz. Freesim-a free real-time freeway traffic simulator. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 18–23. IEEE, 2007.

[9] Behnam Torabi, Rym Z Wenkstern, and Robert Saylor. Agent-based decentralized traffic signal timing. In *Proceedings of the 21st International Symposium on Distributed Simulation and Real Time Applications*, pages 123–126. IEEE Press, 2017.

[10] R Wiedemann. Simulation des straßenverkehrsflusses. schriftenreihe heft 8. *Institute for Transportation Science, University of Karlsruhe, Germany*, 1994.