Science of Computer Programming 168 (2018) 142–164

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Actor-based macroscopic modeling and simulation for smart urban planning

Jacopo de Berardinis [a,*], Giorgio Forcina [b], Ali Jafari [c], Marjan Sirjani [b,c]

[a] The University of Manchester, School of Computer Science, Oxford Road, Manchester, M13 9PL, UK
[b] Mälardalen University, School of Innovation, Design and Engineering, Hogskoleplan 1, Vasteras, 72123, Sweden
[c] Reykjavik University, School of Computer Science, Menntavegur 1, Reykjavik, 101, Iceland

ARTICLE INFO

Article history:
Received 8 January 2018
Received in revised form 10 August 2018
Accepted 4 September 2018
Available online 7 September 2018

Keywords:
Actor-based modeling languages

ABSTRACT

Assessing the impacts of a mobility initiative prior to deployment is a complex task for both urban planners and transport companies. Computational models like Tangramob offer an agent-based framework for simulating the evolution of urban traffic after the introduction of new mobility services. However, simulations can be computationally expensive to perform due to their iterative nature and the microscopic representation of traffic. To address this issue, we designed a simplified model architecture of Tangramob in Timed Rebeca (TRebeca) and we developed a tool-chain for the generation runnable instances of this model starting from the same input files of Tangramob. Running TRebeca models allows users to get an idea of how the mobility initiatives under study affect the traveling experience of commuters, in a short time and without the need to use the simulator during this first experimental step. Then, once a subset of these initiatives is identified according to user's criteria, it is reasonable to simulate them with Tangramob in order to get more detailed results. To validate this approach, we compared the output of both the simulator and the TRebeca model on a collection of mobility initiatives. The correlation between the results demonstrates the usefulness of using TRebeca models for unconventional contexts of application.

1. Introduction

Being part of a continuously growing population, which is expected to shift from 7.3 billion to 9.6 billion inhabitants by 2050 [1], urges us to re-think urban mobility. Such an unbridled demographic growth is worsened by an increasing urbanization trend, as people living in urban areas will rise from 54% to 66% in the next 30 years. If poorly managed, these phenomena might jeopardize the quality of life of citizens, accentuating problems like traffic congestion, higher cost of mobility, land use inefficiencies and damage to the environment.

Urged by these threats, urban planners are now embracing Smart Mobility, defined as "a complex set of projects and actions, different in goals, contents and technology intensity" [2] focused on mobility issues. Smart mobility services range from carsharing and bikesharing to more advanced ones like self-driving taxis and dynamic ridesharing systems.

* Corresponding author.
E-mail addresses: jacopo.deberardinis@postgrad.manchester.ac.uk (J. de Berardinis), giorgio.forcina@mdh.se (G. Forcina).

https://doi.org/10.1016/j.scico.2018.09.002
0167-6423/

Nevertheless, given a certain urban context, how can we evaluate the ability of a Smart Mobility Initiative (SMI), i.e. a number of smart mobility services, to meet the actual mobility needs of the population prior to its deployment? It turns out that estimating the impacts of a mobility initiative is one of the most crucial concerns in urban planning. Indeed, the current approach of using heuristics and best-practises might be risky for decision makers, since the new mobility services may be unaccepted by the population [3,4]. In this context, the only relevant work we can find in the literature is DTalite [5], i.e. an open-source mesoscopic simulator developed to provide transportation planners, engineers, and researchers with a theoretically rigorous and computationally efficient traffic network modeling tool. However, this computational model is only focused on simulating how urban traffic would change in case the road network is altered by the end user. On the other hand, we need a similar easy-to-use tool for the assessment of novel mobility services.

Driven by these motivations, *Tangramob* [6] offers a simulation environment for mobility assessment. This tool allows urban planners and transport companies to check if the effects of the simulated mobility initiatives are expected to be in line with their objectives and plans. The peculiarities of this simulator are: the adaptability to different geographical contexts; the support of multimodal trips and mobility services; the ability to reproduce real-life scenarios. Tangramob relies on an Agent-Based Model (ABM) in which every citizen is given the ability to experience with the newly introduced mobility services in order to understand which is the best way to travel daily. Following a Reinforcement Learning (RL) strategy, a simulation is organized as a series of iterations, so that a single day is simulated multiple times: this approach allows citizens to accumulate experience so as to come up with better mobility decisions. At the end of a simulation, we will be able to understand how the smart mobility initiative is accepted by the population and how it affects urban system.

However, the iterative nature of Tangramob simulations makes them computationally expensive in case of complex scenarios, i.e. when either or both the population under study is large and the number of new smart mobility services is substantial. On the other hand, an urban planner is interested in performing multiple experiments with the simulator, that is, evaluating different smart mobility initiatives so as to find out the most promising ones. For instance, a user can change the configuration of a mobility service by either increasing or decreasing the number of vehicles. Nonetheless, running as many simulations as the number of smart mobility initiatives to investigate might be time-consuming.

To address this obstacle, we reduced the complexity of the Tangramob's ABM in order to derive a Timed Rebeca (TRebeca) [7] model in which the RL-based learning process is replaced with a decision-rule process and the representation of traffic is simplified. Together with the ability of modeling persons as packets, made possible by the actor-based modeling paradigm of TRebeca, the resulting model allows to run experiments faster with the cost of loosing the microscopic detail of Tangramob simulations. In addition, to improve its usability, we developed ToolTRain, i.e. a tool chain that generates an instance of the TRebeca reference model from the same input files of a Tangramob scenario; runs the so-obtained model instance; and infers the same output variables of the simulator from the run. Despite the several simplifications, comparing the results obtained from Tangramob and ToolTRain on different mobility initiatives, for the same scenario, shows that the TRebeca model behaves similarly to that of Tangramob. This correlation lets the users exploit the TRebeca model as a tool for getting first results of a SMI without simulating it. In particular, the experimenter can use this model to understand which initiatives are more in line with his expectations, so as to simulate them later with Tangramob to get more details.

This paper is an extended version of [8]. The main extensions are presenting: the improvements we made to the original TRebeca reference model in order to obtain a more realistic representation of urban traffic and a new criterion for the selection of mobility services by citizens; a new case study where we investigate the application of the ToolTRain approach to the Electric Work Site Project of Volvo. This last contribution is an argumentation of a possible use case in which our approach could be applied, and neither the TRebeca model nor the Tangramob simulation is provided. Moreover, all sections have been extended and present new material and insights in order to allow the reader to fully understand our approach and evaluate its application in new contexts.

The structure of this paper is as follows: Section 2 gives an overview of Tangramob, together with a brief description of the Agent-Based Model (ABM) behind its design. In Section 3 we introduce Rebeca and Timed Rebeca (TRebeca) modeling languages and we provide a detailed description of the simplified TRebeca model derived from the ABM. Moreover, this section details on how we performed the derivation process, focusing on the different assumptions and heuristics adopted in order to approximate the learning process of the simulator. In Section 4 we describe how we designed the experiment to demonstrate both the correlation among the results given by the two models and the utility of the simplified TRebeca model. Section 5 show the experimental results and we discuss about their implications to confirm the former hypothesis. In Section 6 we present the Electric Work Site Project of Volvo and we investigate how the ToolTRain approach can be adopted in contexts other than mobility. Finally, we conclude with an overview (Section 7) of the related work and our research outlook (Section 8) for this project.

2. The Tangramob simulator

Tangramob [6] (formerly SmartHub) is an agent-based simulation framework, supporting urban planners and transport companies in shaping Smart Mobility Initiatives (SMIs) within urban areas. Users can assess whether introducing a SMI can improve the traveling experience of the citizens, as well as the urban transport system. In order to consider people's acceptance, Tangramob returns an estimation of how a mobility initiative can impact on local communities, so as to figure out beforehand if a SMI can potentially succeed or not. Tangramob is built on the top of MATSim [9], a robust and well-known multi-agent traffic simulator. Both of them are based on Java and are open-source projects, although the core code

Fig. 1. Direct path.

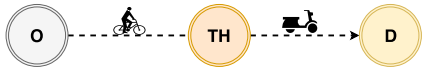

Fig. 2. 2-trip path.

Fig. 3. 3-trip path.

of Tangramob has not been released yet. Technically, a Tangramob simulation requires the following inputs specified into Extensible Markup Language (xml) format:

- **road network** of the area under study, represented as a weighted graph with nodes denoting intersections and edges standing for streets.
- **mobility agendas** of a sample population (i.e. people's mobility habits), each of which summarizes what a person does during an ordinary working day (i.e. activities) and how he moves from one place to the next one (i.e. *legs*).
- **smart mobility initiative** to be simulated, i.e. a list of geographically located containers of one or more smart mobility services, called tangrhubs. Each smart mobility service belongs to a tangrhub and it comes with a number of mobility resources (e.g. vehicles), as well as a service charge.

A Tangramob simulation returns several output files so as to provide users with a list of measures concerning how the mobility habits of citizens are expected to change after the introduction of the SMI. In particular, the following output variables are returned for each person, and then aggregated together: *traveled distance*, *traveled time*, CO_2 *emissions*, *mobility costs* and whether he has accepted the mobility initiative or not. Tangramob will also provide a measure of the resulting *urban traffic levels*, as well as a metric on the *use of mobility resources*. From the analysis of such parameters, a user can realize if the simulated initiative is in line with his expectations. If not, he can change the configuration of the SMI (e.g. relocate/add/remove tangrhubs, change a mobility service) and run new experiments.

Though we omit the description of Tangramob in this work, we need to introduce the concepts of *tangrhub*, *smart mobility service*, *smart mobility initiative* and *commuting pattern* in order to present the agent-based model and the derivation process for the TRebeca model.

A **tangrhub** is a geo-located entity providing citizens with one or more mobility services.

A **smart mobility service** is a traveling solution that citizens can choose, if resulting from an interaction with a tangrhub. Mobility services supported in the current version of Tangramob are rental solutions (carsharing, bikesharing, scootersharing) and ridesharing systems. For future developments, both Tangramob and ToolTRain will support further mobility services, e.g. public transport and taxi on-demand. In Tangramob, defining a smart mobility service requires the user to specify the service type (e.g. carsharing), the initial number of vehicles and the service charge (i.e. cost per km, per hour and fixed cost). Moreover, each mobility service m_i provided by a tangrhub th_j must belong either to one of these service types:

- *intra-hub* services, used for moving people *to* and *from* th_j thereby serving first mile trips, e.g. from a commuter's home-place to th_j and viceversa;
- *inter-hub* services, moving people from th_j to another tangrhub th_k.

A **Smart Mobility Initiative (SMI)** is the configuration of the mobility solution to simulate. The SMI specifies the number of tangrhubs, their location, their mobility services, and the characterization of each mobility service.

A **commuting pattern** is the intermodal representation of how a person moves from an origin location to his destination. Such a trip can be direct (e.g. traveling by either walking or car); or more complex (e.g. using more than one means of transport). A clear example of a commuting pattern is the route provided by the trip planner of Google Maps. In Tangramob, the complexity of commuting patterns is limited to three schemes: direct path, 2-trip path and 3-trip path (shown in Figs. 1, 2, 3, respectively). In particular, nodes O and D represent the commuter origin and destination respectively; whereas nodes TH, TH_O and TH_D depict respectively a generic tangrhub, and the nearest tangrhubs to O and D. This is possible thanks to both the concept of tangrhub and interconnection among tangrhubs via inter-hub mobility services. The resulting architecture looks similar to a computer network, in which tangrhubs play the role of routers, mobility services are cables and commuters can be seen as packets.

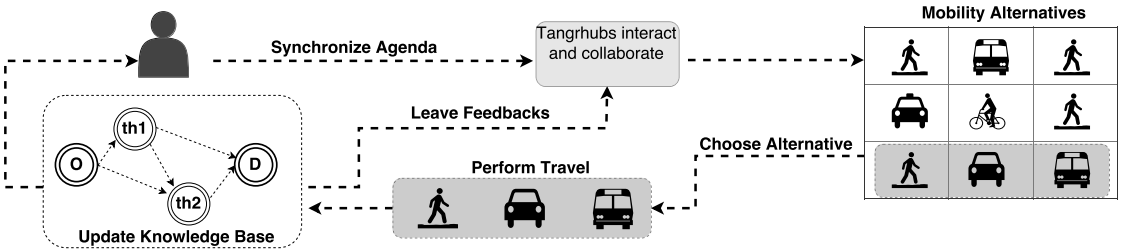

Fig. 4. Commuter–Tangrhub interaction loop.

2.1. Tangramob Agent-Based Model (ABM): an overview

This section provides an overview of the model of Tangramob, that is needed to understand the translation of the ABM into the simplified TRebeca model presented in Section 3.3.

As better detailed in [6], the ABM of Tangramob has the following two agent types: *commuter* and *tangrhub*. A commuter agent, from now on commuter, is the computational counterpart of a person of the population. Every commuter has his own mobility agenda, i.e. a sequence of daily activities (e.g. home, work) interleaved by legs, each of which tells how the commuter moves from one activity location to the next one (e.g. car, bike). Instead, a tangrhub agent acts as a local mobility service provider.

Both agents live and operate, with different perceptions, in a composite environment made of three different spaces: the temporal one, the geographical one and the space described by the current state of each smart mobility service. The geographical space is the core of the transport simulation, since the physical limitations of the road network can create bottlenecks and delays as people move with a certain pace.

As depicted in Fig. 4, every time a commuter needs to move from one place to another, an interaction with the surrounding tangrhubs takes place as follows: tangrhubs are expected to collaborate with each other in order to provide the commuter with a number of traveling alternatives for taking him to destination. Afterwards, the commuter will perform a decision-making process to select the traveling alternative that is expected to optimize his performance criteria. Once an alternative is chosen, the involved tangrhubs will reserve the required mobility services so that the commuter can start his journey. Finally, once the commuter has reached his destination, he will be asked to leave a feedback for each smart mobility service involved in the chosen alternative.

Each feedback quantifies the traveling experience of a commuter using a specific mobility service. This value is computed by means of a scoring function which takes into account some performance criteria of the commuter, such as travel time and traveled distance. The use of a feedback allows a person to progressively make more informed decisions.

To enable this behavior, Tangramob simulations are iterative: each iteration corresponds to a typical day in which commuters try new mobility services and record their experience. This online optimization approach, driven by feedback, enables commuters to improve their traveling experience iteration by iteration. Thus, the more the users can experience the new mobility services the higher is the detail of the output of the simulation.

At the end of a simulation, commuters may change their original mobility habits in favor of those mobility services that can better accommodate their needs (we call them subscribers). In case the simulated SMI does not suit some commuters, those agents will be restored with the traveling habits owned before the SMI introduction.

3. From Tangramob ABM to TRebeca

In this section, we introduce Rebeca and TimedRebeca modeling languages, showing their features and the motivations behind their involvement in our work. Afterwards, we present the TRebeca model, and we argue about its derivation from the Tangramob ABM.

3.1. Reactive Object Language: Rebeca

Reactive Object Language (Rebeca) [10] is an actor-based language designed to connect practical software engineering domains and formal verification methods. In short, Rebeca is a language for modeling event-based distributed systems. Moreover, it represents an interpretation of the actor model adopting a Java-like syntax which is supported by verification tools. The semantics of Rebeca is presented by Labeled Transition System (LTS). Systems are modeled by concurrently executing reactive objects called *rebecs* which can interact with one another by asynchronous message passing. In particular, a Rebeca model consists of the definition of reactive classes (i.e. rebecs), each of which corresponds to a specific actor type of the system. Technically, a reactive class comprises three parts: known rebecs (i.e. the other rebecs with which it can communicate), state variables (like attributes in object-oriented languages) and message server definitions, defining the behavior of the actor itself (like methods in object-oriented languages). Each message server has a name, an optional list of parameters and its body, which can be described as the actual behavior of the rebec once such kind of message is received;

it includes a number of statements, i.e. assignments, sending of messages, and selections. The computation of a Rebeca model is event-driven [10], since messages can be seen as events. Each rebec takes a message from its message queue and executes the corresponding message server atomically. Communication among rebecs takes place by asynchronous message passing as follows: the sender rebec sends a message to the receiver rebec and continues its work; the message is put in the message queue of the receiver and it stays there until the receiver serves it. The behavior of a Rebeca model is hence defined as the parallel execution of the released messages of the rebecs.

In addition to the modeling features, Rebeca offers a formal verification approach in order to check the model correctness. More in detail, the language is supported by the Rebeca Verifier tool, which uses model checking algorithms for verifying Rebeca models. For the sake of clarity, despite the potentialities that Rebeca offers, we aim at using it just for performance analysis purposes rather than properties verification tool.

3.2. Timed Rebeca: TRebeca

Timed Rebeca (TRebeca) [7,11,12] is a timed extension of Rebeca language with timing primitives. TRebeca supports the modeling and verification of distributed systems with timing features, and its semantic is presented in Timed Transition System. Time is represented in terms of discrete time steps. The timing primitives provided to the modeler by the TRebeca language aim at representing the following functionalities:

- **computation time:** the amount of time units required for a computation;
- **message delivery time:** the amount of time units needed for delivering a message;
- **message expiration:** the amount of time within which a message is still valid;
- **periods of occurrences of events:** the frequency with which events are repeated.

In a Timed Rebeca model, each rebec has its own time clock which can be seen as a synchronized and distributed clock. Methods (i.e. message servers) are still executed non-preemptively as in Rebeca models. The specification of time features are made possible by the extension of the Rebeca syntax with the statements: *delay, deadline and after*.

- **Delay:** *delay(t)* statement increases the clock of the rebec of t time units (t is a natural number). Such a delay aims at representing the computational time needed for a rebec for executing and finishing a process.
- **Deadline:** *r.m() deadline(t)*, in which r is a rebec, m is a method and t is a natural number, aims at representing the message expiration feature. In this specific case the message m should be processed by rebec r earlier than t time units.
- **After:** the statement *r.m() after(t)*, in which r is a rebec, m is a method and t is a natural number, represents the message delivery time: m is added to the message queue of rebec r, but it cannot be processed before t time units after message sending.

As it emerges from its features, the Timed Rebeca modeling language perfectly satisfies our purposes, since it allows to capture timing features, as well as to represent the interactions between the ABM's agents. In fact, timing is needed for organizing the actions performed by commuters during the course of a 24-hour day.

3.3. The simplified Tangramob model in TRebeca

Tangramob is a complex and fine-grained framework capable of simulating millions of events and interactions between agents and the environment. For this reason, it is prohibitive to reproduce the ABM as it is into a TRebeca model, since its executions would result into an unmanageable state space explosion. Thus, the designed TRebeca model is widely simplified and abstracted, but still keeping the core features. In detail, the TRebeca model is composed of the two following rebec types (i.e. actors): *CommuterGenerator* and *Tangrhub*. The following sections describe the actors involved in the model and how we abstracted the behavior of the commuter agent.

3.3.1. The commuter as a message

From the ABM of Tangramob outlined so far, the reader can notice that for the *Tangrhub* agent we defined its rebec counterpart, whereas the *Commuter* actor does not exist. Indeed, in the simplified model we do not consider the commuter as an entity capable of reasoning and acting when some event occurs, but it is simply treated as a message to be sent among tangrhubs. With this assumption, each Tangrhub plays the role of a delivering system in which packets (i.e. commuters) are dispatched. According to the mobility agenda of each commuter, it sends and receives messages to and from each other. The specifications regarding the sender (i.e. origin tangrhub), the receiver (i.e. the destination tangrhub) and the time in which the message has to be sent, are encapsulated within the message itself, and they reflect the mobility agenda of the corresponding commuter (i.e. their daily plan).

This information are represented in TRebeca by means of a two-dimensional array (from now on commuter matrix), in which rows represent commuters, whereas columns contain their mobility needs. In detail the columns of the commuter matrix represent:

- id: commuter identifier
- th_h: home closest tangrhub
- $time_{home}$: time before leaving home
- $time_{fm}$: walk time from home to th_h
- th_w: workplace closest tangrhub
- $time_{work}$: time spent working
- $time_{lm}$: walk time from th_w to work

The commuter matrix is pre-computed and the data are derived from the file which contains the commuters' mobility agenda mentioned in Section 2 (i.e. plans.xml). The derivation process will be detailed in Section 3.4.

More in detail, the closest tangrhubs to home (i.e. th_h) and workplace (i.e. th_w) are computed by means of the Euclidean distance function. For what concern the elapsing time before leaving home (i.e. $time_{home}$) and the working time (i.e. $time_{work}$), they are directly extrapolated from the agenda without any particular treatment. First mile (i.e. $time_{fm}$) and last mile (i.e. $time_{lm}$) walking times, differently from the model presented in [8], are pre-computed taking into account also those personal features which could affect commuters walking speeds, i.e. age and gender. Eq. (1) shows the computation of their values.

$$time_{fm,lm} = distance_{fm,lm} \times \beta / (speed_{walk} \times factor_{age} \times factor_{gender}) \tag{1}$$

In which, $distance_{fm}$ and $distance_{lm}$ represent respectively the Euclidean distance between commuter's home and his closest tangrhub and between th_w and his workplace. β is the beeline factor; $speed_{walk}$ represents the average walking speed; $factor_{age}$ and $factor_{gender}$ are values that characterize the commuter's speed, either increasing or decreasing it according to his age and gender. These values are the same used in Tangramob and are based on results of a comprehensive literature review of [13].

Analyzing the structure of the commuter matrix, some abstractions we applied in order to reduce the complexity of the ABM of Tangramob, emerge. Indeed, considering the characterization of the first and last mile trip, we assume that each commuter will perform intra-hub trips only by walking.

Another strict assumption regards the commuting pattern selection process. Indeed, in the commuter matrix, each commuter is provided with two tangrhubs. This implies that all of them are expected to travel by means of inter-hub mobility services only. Thus, by adopting always the 3-path commuting pattern (Fig. 3). Indeed, in this model we assume that commuters have only two commuting patterns:

$$home \rightarrow th_h \rightarrow th_w \rightarrow work \quad \text{and} \quad work \rightarrow th_w \rightarrow th_h \rightarrow home$$

Such assumptions, as well as treating the commuter as a message and not as an actor, lead the system to be more lightweight and computationally faster than Tangramob. Although, this gain implies a loss of detail on that information related to the commuters movements.

3.3.2. The Tanrghub rebec

Differently from the commuter, the tangrhub agent has been translated into a rebec named *Tangrhub*. Its behavior is similar to the one designed for the ABM, i.e. managing its mobility services and providing commuters with vehicles. Additionally, since commuters are modeled as messages, each *Tangrhub* has the responsibility of delivering commuters to the next *Tangrhub*. Every time this occurs, an available mobility service resource (i.e. a vehicle) is released to a commuter, which will use it for reaching the next *Tangrhub*. Instead of letting commuters select a mobility service, in the TRebeca model this decision-process is emulated by *Tangrhubs*. Indeed, every time a commuter is scheduled, a *Tangrhub* releases a vehicle, if any, of the selected service (s^p) with the best trade-off between its current fleet ($fleet_{s_i}$) and a priority value ($priority_{s_i}$), as shown in Eq. (2).

$$s^p = \max_{s_i}(fleet_{s_i} \times priority_{s_i}) \qquad i \in [0..numOfServices] \tag{2}$$

In the model presented in [8], $priority_{s_i}$ is associated to each mobility service. This means that the higher is the value, the more the service is preferred. However, this approach is not very commuter-centric and in line with Tangramob since the priority value related to each mobility service is the same for all the commuters. In order to let commuters decide which mobility service is more in line with their travel needs, instead of assigning a priority value to each mobility service, we pre-calculate for each person a list of values which aim at representing the personal expectations related to those mobility services introduced with the SMI. Thus, commuters have their own priority values associated to each mobility service (s_i). As shown in Eq. (3), these values take into account several personal aspects of the commuter itself, like the age ($factor_{age}$) and the gender ($factor_{gender}$), as well as some specific mobility service features, like travel times ($time_{s_i}$) and costs ($cost^{hr}_{s_i}$, $cost^{km}_{s_i}$, $cost^{fixed}_{s_i}$). Thus, the higher is the value, the more the commuter prefers the corresponding service.

$$
\begin{aligned}
priority_{s_i} &= 1/(time_{s_i} \times cost_{s_i}); i \in [0..numOfServices] \\
cost_{s_i} &= (time_{s_i} \times cost^{hr}_{s_i}) + (distance(th_h, th_w) \times cost^{km}_{s_i} + cost^{fixed}_{s_i}) \\
time_{s_i} &= distance(th_h, th_w)/(avgSpeed_{s_i} \times factor_{age} \times factor_{gender})
\end{aligned} \tag{3}
$$

The information related to the priority values, similarly to the commuter matrix, are saved in a two dimensional array data structure, in which each row represents a commuter and each column corresponds to a mobility service.

Once a mobility service is selected according to the mentioned approach, during the run of the model each Tangrhub must memorize how such service is used so as to produce the outputs for evaluating the goodness of the SMI. For this reason each Tangrhub comes with two more data structures: the distances matrix between tangrhubs (D) and their velocity matrix (V). The first one is a standard distance matrix, in which at each position $D_{(th_i,th_j)}$ the distance in meters from tangrhub i to tangrhub j is pre-computed. Such distance is calculated by means of the Dijkstra Shortest Path Algorithm since, as described in Section 2, the topology of the urban network is represented by means of a graph.

The velocity matrix V aims at representing the average speeds between all the tangrhubs by using a specific mobility service. In particular, $V^{s_k}_{(th_i,th_j)}$ represents the weighted average of the speeds, traveling from tangrhub th_i to tangrhub th_j by means of the service s_k.

Considering that travel times and travel costs are strictly dependent on the selected mobility service, it is not possible to pre-compute their values in the pre-processing phase. Thus, since the commuters are no longer actors and the environment is just abstracted by means of a distances matrix, we provide the Tangrhub rebec with a data structure named delays matrix so as to emulate travel delays. Such bi-dimensional array is randomly pre-computed and each row represents a commuter. Columns contain the delay percentage with which the travel for the specific sub-trip is penalized in both ways. In detail:

- $delay_{(home,th_h)}$: trip delay $home \rightarrow th_h$
- $delay_{(th_h,th_w)}$: trip delay $th_h \rightarrow th_w$
- $delay_{(th_w,work)}$: trip delay $th_w \rightarrow work$
- $delay_{(work,th_w)}$: trip delay $work \rightarrow th_w$
- $delay_{(th_w,th_h)}$: trip delay $th_w \rightarrow th_h$
- $delay_{(th_h,home)}$: trip delay $th_h \rightarrow home$

The just presented approach represents a strong assumption of the ABM. However, we believe that such a simplification allows us to easily abstract from the original design of Tangramob and, as a consequence, reduce the computational burden.

The state variables of the Tangrhub rebec are:

- *mobServiceFleet*: the amount of available vehicles for each mobility service,
- *usedMobServiceFleet*: the number of used vehicles for each mobility service,
- *unusedMobServiceFleet*: the unused vehicles for each mobility service,
- *timeUsedMobService*: the usage times of a mobility service,
- *costUsedMobService*: the usage costs of a mobility service.

3.3.3. The CommuterGenerator rebec

The *CommuterGenerator* rebec is in charge of monitoring the progress of commuters travels and their scheduling. In particular, it checks whether commuters are leaving home or they have just performed the last sub-trip, i.e. they are coming back from workplace. Moreover, this rebec is notified every time a commuter experienced a service disruption, i.e. it did not find any available mobility service. Its state variables are:

- arrivedCommuters: the number of commuters that have finished their day,
- commuterAborts: commuters that experienced a mobility service disruption.

3.3.4. The TRebeca model event graph

In order to describe how actors interact in the TRebeca model, we provide the reader with its event graph (Fig. 5) and some relevant parts of the pseudo-code. These give an intuitive and highly abstracted view of events and causality relations. The graph has labeled nodes which represent events and their owner rebec. Edges show the causality relations among nodes, and can be either conditional (thick edges) or mandatory (thin edges).

As shown in Fig. 5 and Listing 1, the *CommuterGenerator*, once inizialized its state variables, starts the run by sending a message to itself that triggers the *fireCommuters* event (message server). At this point, the *CommuterGenerator* evaluates the mobility agenda of each commuter ($comm_k$) by exploring the commuter matrix. Afterwards, commuter by commuter it sends a *serveCommuter* message to each origin tangrhub, which is the closest tangrhub to home in the first trip (i.e. th_h), after a specific time unit (Eq. (4)). The Boolean parameter within the message aims at representing which travel it is serving (i.e. true: from home to work, false: from work to home).

$$
thArrTime^{comm_k} = time^{comm_k}_{home} + time^{comm_k}_{fm} + (time^{comm_k}_{fm} \times delay^{comm_k}_{(home,th_h)}/100) \tag{4}
$$

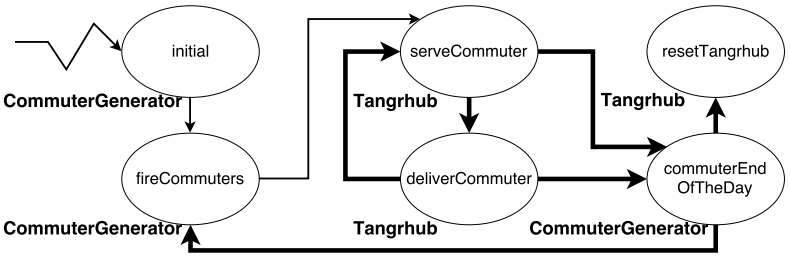

Fig. 5. Event Graph of the TRebeca Model.

```
reactiveclass CommuterGenerator(numOfCommuter) {
  knownrebecs {
    /* All the configured Tangrhub (n + 1) rebecs */
    Tangrhub th_0;
    ...
    Tangrhub th_n; }

  CommuterGenerator(...) {
  1) init statevars;
  2) self.fireCommuters(); }

  /* Each commuter is delivered to the closest tangrhub */
  msgsrv fireCommuters() {
    foreach commuter in commuterMatrix:
      1) thArrTime: the arrival time to the closest tangrhub (i.e. th_o);
      2) serve commuters: th_o.serveCommuter(commuter,true) after(thArrTime); }
```

Listing 1: CommuterGenerator rebec pseudo-code.

When a *Tangrhub* receives this message, one of these three actions is possible (Listing 2).

1. in case a mobility service is available and the commuter is reaching his workplace, it sends a message to the next *Tangrhub* (i.e. th_w), after *travTimes* time unit,
2. in case the commuter is coming back home and there is an available mobility service, it sends a message to the *CommuterGenerator* for informing it,
3. if no service is available, it sends a message to the *CommuterGenerator*, which informs it that the commuter did not find any vehicle for traveling (i.e. service disruption).

For the sake of clarity, *travTimes* is the time in which the commuter k is expected to arrive to the destination Tangrhub (th_d), starting from the origin Tangrhub th_o, by means of the vehicle of the mobility service s_p, as shown in Eq. (5):

$$travTimes^{comm_k}_{(th_o,th_d)} = (D_{(th_o,th_d)}/V^{s_p}_{(th_o,th_d)} + \\ + D_{(th_o,th_d)}/V^{s_p}_{(th_o,th_d)} \times delay^{comm_k}_{(th_o,th_d)}/100) \tag{5}$$

The first action triggers a *deliverCommuter* event, which represents the travel of a commuter by means of a mobility service. After that, a message is delivered to the next Tangrhub for informing it about the following commuter trip, triggering again a *serveCommuter* event.

On the other hand, actions 2 and 3 will trigger a commuterEndOfTheDay event (Listing 3). Every time this occurs, the *CommuterGenerator* updates the number of arrived commuters (i.e. those who finished their activities), and a service disruption is registered if the commuter did not find a service. In case the number of arrived commuters is equal to the total number of commuters, the *CommuterGenerator* restores the initial state of the system by sending a *resetTangrhub* message to each *Tangrhub*. The aim of this message is to represent the end of the day, thus to mimic the Tangramob end of an iteration. Restoring the initial configuration of the model means for the model checker to reach a state that is equivalent to the initial state of the model. Indeed, in this state, the variables of each rebec have the same value, whereas the timestamp is obviously different. However, the equivalence criterion between states does not consider the time, thus for the model checker, the just restored state and the initial one are equivalents. As a consequence, the run of the model is interrupted since all the following states will be equivalent to those that have been already generated in the state space. It is worth saying that the resetTangrhub message is not a specific stop message that roughly ends the run of the model, leaving unexplored the following states. But, it is just a message that restores the initial condition of the system.

```
reactiveclass Tangrhub(numOfCommuter) {
  knownrebecs {
    Tangrhub th_0;
    ...
    Tangrhub th_N;
    CommuterGenerator controller; }

  Tangrhub(...) {
    1) init statevars; }

  /* Each commuter asks for a mobility service for a trip */
  msgsrv serveCommuter( commuterId, isFirstTrip){
    if ( no mobility service available ):
      controller.commuterEndOfTheDay(commuterId, true);
    else:
      1) select a service according to the current fleet available and the commuter service priority value;
      2) update the fleet of the selected mobility service;
      3) compute the travel statistics for the selected service (times, distances, costs and emissions);
      4) perform the travel by the selected mobility service:
        th_d.deliverCommuter(commuterId, selectedService, isFirstTrip) after (travTimes); }

  /* The commuter performs the travel from a Tangrhub to the next one */
  msgsrv deliverCommuter(String commuterId, int selectedService, boolean isFirstTrip){
    /* The fleet of the destination Tangrhub is updated */
    serviceFleet[selectedService]++;
    /* Check whether the current trip is the last one */
    if (isFirstTrip ):
      self.serveCommuter(commuterId, false) after(timeToReturnBack);
    else:
      controller.commuterEndOfTheDay(commuterId, false) after(homeArrivalTime); }

  msgsrv resetTangrhub() { /* restore the initial configuration of this Tangrhub */ } }
```

Listing 2: Tangrhub rebec pseudo-code.

```
/* Keep track of all those commuters who terminated their daily travels and encountered a service disruption */
msgsrv commuterEndOfTheDay(commuterID, isAborted) {
  /* In case all commuters terminated, each Tangrhub is resetted */
  if (all commuters came back home):
    foreach tangrhub_i in tangrhubs:
      tangrhub_i.resetTangrhub(); }
```

Listing 3: CommuterGenerator *commuterEndOfTheDay* msgsrv.

3.4. ToolTRain: infer, generate, run, infer and collect

Since we aim at using the TRebeca model as a lightweight simulation tool, we need to generate new instances of the model from given scenarios, so as to collect similar output data of Tangramob after the model run. However, this is still not enough to get significant results due to the iterative nature of Tangramob and its queue-based traffic simulation.

For this purpose, we implemented ToolTRain: a tool-chain specifically designed for generating a TRebeca model from the simulator's input files according to some abstraction rules; running the resulting model; and inferring the output from the model run. Fig. 6 shows the 3 building blocks of ToolTRain which are outlined below.

3.4.1. Model inference and generation

Starting from the input files of Tangramob (Section 2), a TRebeca model is generated according to the following points:

- A subset of the input population is selected as potential users of the new mobility services. The remaining commuters, i.e. those who live or work too far or too close from/to tangrhubs, are assumed to travel by car or walk respectively. This step is called the commuter filtering process.
- For each potential user, from now on *potential subscriber*, the tangrhubs closest to his home (th_h) and to his workplace (th_w) are chosen, and the corresponding 3-path commuting pattern (Fig. 3) is fixed for him.
- First mile trips (traveling towards a tangrhub) and last mile trips (traveling from a tangrhub to the destination) are performed by walk, thus the travel distance of such trips are computed as Euclidean distance from point to point.

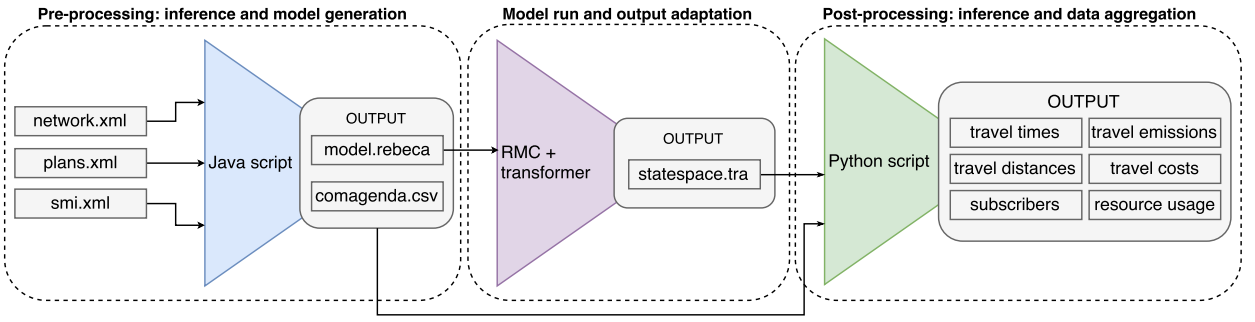

Fig. 6. The architecture of ToolTRain.

- For each potential subscriber, a priority value for those mobility services introduced with the SMI are calculated by means of Eq. (3). This latter takes into consideration both features of the commuter as well as characteristics of the service itself.
- Recalling the graph-like nature of the road network input, distances among tangrhubs are computed with the Dijkstra's shortest path algorithm. These values are expected to determine the travel time of inter-hub trips, depending on the characteristics of the vehicles provided by the new mobility services.
- Random delays are generated for all trips in order to emulate urban traffic.
- For each inter-hub trip a weighted velocity average is computed. These velocities are specific for each mobility services introduced with the SMI.
- A data structure similar to the commuter matrix presented in Section 3.3.1 is generated for post-processing purposes.

3.4.2. Model run

The so-generated TRebeca model is run with Rebeca Model Checker (RMC), a tool for direct model checking of TRebeca models. In particular, running a TRebeca model in RMC results in the generation of the whole state space of the model. Next, in ToolTRain, the so-generated state space is converted into a tiny representation in which only a list of pre-defined state variables are reported for each state for further analysis. This is done by means of the state space transformer tool of the Rebeca suite. Therefore, it is worth remarking that we actually use RMC for performance evaluation rather than correctness check.

3.4.3. Data inference and aggregation

Since we are interested in the results of a smart mobility initiative, the converted state space is fed into a post-processing script to collect the observed variables at the very last reached state, which corresponds to the end of a day. Next, in order to emulate people's acceptance of a mobility initiative, all those commuters who encountered a service disruption (i.e. no vehicles available at a tangrhub) during the model run are selected and treated differently. In particular, as similarly done in the pre-processing step, those commuters are assumed to travel by car or by walk, and their travel metrics are computed accordingly. Finally, once every potential subscriber has been processed, all the progressively-collected travel metrics are aggregated in order to generate the same output files of Tangramob.

3.5. Comparing Tangramob and TRebeca models

All the previously outlined abstraction rules are meant to simplify the original ABM, thereby removing its computationally-expensive features at the cost of loosing the microscopic detail of Tangramob simulations. Thus, traffic is emulated with random delays, whereas the online (and iterative) optimization process, used for modeling people's acceptance, is replaced by decision-rules, as well as all the abstractions and simplifications made to the original ABM. These rules are both encoded in the TRebeca model and achieved by the commuter filtering process of ToolTRain. An example of decision rule is the one used for emulating the selection process of mobility services: a commuter does not use past experience anymore, but he/she chooses a service according to Eq. (2). Nevertheless, even though we pay in model expressiveness, a complete pass of ToolTRain is much faster than a Tangramob simulation. Indeed, as shown in Section 5.3, it turns out that running the TRebeca model can drastically reduce the computational burden of Tangramob, and this is a considerable gain if one needs to try many SMIs. Moreover, it is worth remarking that once a TRebeca model is generated, there is no need to repeat this step in case other mobility initiatives differ just in the number of vehicles per service.

Concerning the modeling techniques, Tangramob lies on an agent-based model which is conceptually similar to the actor-based one of the TRebeca counterpart. Indeed, agent's perceptions can be thought of as triggering message forwarding in actor-based models: an actor receiving a message can be seen as an agent perceiving a change in the environment. This similarity makes it possible to translate the agent-based model into an actor-based one, keeping its conceptual integrity with no particular compromises.

For what concerns the analysis capabilities, both models allow to observe the same information, with an exception made for traffic levels. In particular, as previously mentioned, we did not model traffic dynamics in the TRebeca model, thus it is not possible to have a measure of the road occupancy during the day.

Finally, if we look at the usability of the models, we can notice that both require the same input files to perform a model run/simulation. Moreover, thanks to ToolTRain, even the output data is presented in the same way (graphs and tables) of the simulator, thus we can conclude that using the TRebeca model is as intuitive as using Tangramob.

4. Experimental design and setup

So far, we outlined the ABM of Tangramob and the derivation process of the corresponding TRebeca reference model. We also argued how this last model can be useful for users to get an idea of a smart mobility initiative without the need to simulate it. However, using ToolTRain as lightweight preprocessing for Tangramob requires us to prove this hypothesis:

H *(Relational equivalence). Given a network, a population and a SMI, ToolTRain can approximate Tangramob, i.e. there is a positive correlation between their outputs.*

In other words, we seek to evaluate the *alignment* of the two computational models under the same inputs. Alignment was introduced by Axtell et al. [14] in order to establish a framework of concepts and methods to determine whether two models can produce the same results, which in turn is the basis for critical experiments and for testing whether one model can subsume another. The proposed framework allows to determine whether two models claiming to deal with the same phenomena can, or cannot, produce the same results. More precisely, Axtell et al. [14] identify two categories of equivalence, i.e.

- *distributional equivalence* if the two models produce distributions of results that cannot be distinguished statistically;
- *relational equivalence*, if the two models can be shown to produce the same internal relationship among their results.

Therefore, considering the introduction of assumptions and simplifications in the TRebeca reference model, we are interested in demonstrating the relational equivalence between Tangramob and ToolTRain. More in detail, since Tangramob returns several output files, testing H1 is equivalent to testing different sub-hypothesis, one per output variable of the simulator. With the exception of urban traffic, which is not represented in the TRebeca model, we need to demonstrate that ToolTRain return, similarly to Tangramob, the following outputs:

- travel times;
- travel distances;
- CO_2 emissions;
- mobility costs;
- number of subscribers;
- mobility fleet usage.

To test the positive correlation between the output variables of both these approaches, we propose a comparative experiment which also allows us to appreciate the usefulness of ToolTRain. First, we choose 9 smart mobility initiatives to evaluate and we partition them into 3 groups, according to the number of Vehicles Per-Capita (VPC) of each one. In particular, we define the following partitions: *light-SMIs* ($\text{VPC} \leq 0.05$); *medium-SMIs* ($0.05 \leq \text{VPC} \leq 0.10$); *massive-SMIs* ($\text{VPC} \geq 0.10$). Each group thus represents a kind of intervention that the urban planner can evaluate on the basis of his/her goals.

Then, we feed each SMI into ToolTRain, together with a fixed set of input variables described later. Once the computation is over, we can observe how the output variables mentioned above differ for each SMI. Such analysis allows us to get a coarse-grained idea of the impacts of a mobility initiative on the urban system. Therefore, for each group we select the most promising SMI, i.e. the one that minimizes travel times, traveled distances, mobility costs, CO_2 emissions and the number of unused vehicles while maximizing the number of subscribers. The selected SMIs are then simulated with Tangramob and their results are compared with the ones returned by ToolTRain. This allows to test H1.

For the sake of a fair comparison, the 9 Smart Mobility Initiatives (SMIs) proposed in this experiment all share the same scenario, i.e. the urban road network of the chosen geographical area together with its population. Since the purpose of this experiment is to test the positive correlation between the output variables of Tangramob and ToolTRain, the number and location of tangrhubs is fixed for all the SMIs, as well as the types and charge of the mobility services provided by each tangrhub. This allows for greater control of the variables involved in the experiment, thus giving more emphasis to the contribution of the actual distribution of mobility resources among tangrhubs.

The following subsections are meant to provide more insights on both the chosen urban area; its population; and the configuration of the smart mobility initiatives investigated in this paper in terms of mobility resource distribution and service charges.

Fig. 7. Urban road network of Ascoli Piceno with tangrhubs.

Table 1
Cost and priority values per mobility service.

	Cost per hour	Cost per km	Fixed cost
Bikesharing	0.5 €	0 €	0.01 €
Carsharing	13 €	0.1 €	0.01 €
Scootersharing	2.5 €	0.1 €	0.01 €

4.1. The scenario of Ascoli Piceno: urban road network, tangrhubs and sample population

As a common testbed for all the SMIs we choose the city of Ascoli Piceno (Italy), a mid-sized town of 50K inhabitants. There are no particular reasons behind this choice apart from the fact that the authors are more familiar with this urban environment.

Fig. 7 shows the portion of urban road network used for this experiment: it covers an area of about 15 km^2 and includes both the main residential and commercial areas of the city. Moreover, the city center is located westward and it is worth mentioning that the consistent and numerous traffic limitations imposed within this area have a tremendous effect on those citizens who cross it. Besides the urban road network, Fig. 7 also reports the location of the tangrhubs as triangles. Both the number and the actual location of the tangrhubs considered for this experiment were obtained from a cluster analysis tool provided by Tangramob and better detailed in [6]. In short, this tool tries to push a variable number of tangrhubs towards the activities of commuters (e.g. home, work, shopping), thereby minimizing the mobility to and from tangrhubs (first and last mile trips).

Considering the scarce availability of fine-grained data on the mobility behavior of Italian commuters, the population used for this experiment was synthesized from a set of local statistical indicators. In particular, the generation process of the synthetic population is detailed in [6] and it is based on statistical data as follows: 57% are male and 43% female; the minimum and maximum daily working hours are 5 and 9 respectively. Finally, 15% of commuters leave home at 7 AM, 65% at 8 AM, 15% at 9 AM and 5% at 10 AM. Only commuters who can generate traffic are considered for this experiment.

As similarly done in traffic simulations, the sample population is just a small yet representative portion of the whole population. This is done for speeding up the computation of the model. In our case, we found that a sample of 2068 commuters is still enough to capture the ordinary mobility patterns of the inhabitants of this part of Ascoli Piceno.

4.2. Configuration of the investigated SMIs

As can be seen in Table 1, each type of mobility service comes with a personal charge that is computed as the sum of 2 travel-dependent factors, i.e. travel time and travel distance, plus a fixed contribution (e.g. cost of subscription or membership). These values are set according to the actual average service charges in Europe and are considered fixed for this experiment. The user is however free to investigate how different policies or pricing schemes can impact the adoption of services by commuters.

Table 2
The investigated Smart Mobility Initiatives (SMIs).

Tangrhub	Service type	Light-SMIs			Medium-SMIs			Massive-SMIs		
		SMI-1	SMI-2	SMI-3	SMI-4	SMI-5	SMI-6	SMI-7	SMI-8	SMI-9
TH 0	bikesharing	0	2	2	2	2	4	5	10	25
	carsharing	2	2	6	2	4	4	5	5	25
	scootersharing	0	0	1	2	2	2	5	5	25
TH 1	bikesharing	0	2	2	2	4	4	5	5	25
	carsharing	2	2	5	2	2	2	5	10	25
	scootersharing	0	0	1	2	2	4	5	5	25
TH 2	bikesharing	0	3	3	10	10	10	35	35	25
	carsharing	3	3	4	5	5	7	30	20	25
	scootersharing	3	3	3	5	7	7	30	20	25
TH 3	bikesharing	0	3	3	5	5	5	10	10	25
	carsharing	2	2	3	3	3	5	10	15	25
	scootersharing	0	2	2	3	5	5	8	8	25
TH 4	bikesharing	0	0	2	5	5	5	10	20	25
	carsharing	0	2	3	3	3	5	15	15	25
	scootersharing	2	2	2	3	5	5	15	15	25
TH 5	bikesharing	0	2	2	3	5	5	5	5	25
	carsharing	2	2	3	2	2	2	5	10	25
	scootersharing	0	0	1	2	2	7	5	5	25
TH 6	bikesharing	0	0	1	7	7	7	20	10	25
	carsharing	2	2	3	4	4	6	15	15	25
	scootersharing	0	2	2	4	6	6	20	15	25
TH 7	bikesharing	0	2	2	5	5	5	15	15	25
	carsharing	2	2	3	4	6	6	15	10	25
	scootersharing	0	0	2	5	5	7	15	15	25
TH 8	bikesharing	0	0	2	5	5	5	10	10	25
	carsharing	2	2	3	3	3	5	10	15	25
	scootersharing	0	2	2	3	5	5	10	10	25
total fleet		22	44	68	101	119	140	338	333	675

Finally, Table 2 reports the full configuration of each smart mobility initiative in terms of vehicle distribution per service. In particular, SMIs 1, 2, 3 belong to the *light* group, SMIs 4, 5, 6 are the *medium* ones and SMIs 7, 8, 9 fall within the *massive* cluster.

5. Experimental results: towards a validation

In this section, we show the results of the 9 SMIs runs with ToolTRain, then we select 3 of them according to the performance criteria discussed in the previous section. Afterwards, to test H1, we compare the output variables of the selected SMIs with the ones returned by Tangramob on the same setup. A comparison of the computational times of the 3 SMIs is also provided to further support the use of ToolTRain as a lightweight pre-processing tool. Finally, we conclude this section with a discussion on the improvements made on the travel times performance measure with respect to the version of ToolTRain presented in [8].

5.1. Experimental results of the 9 SMIs

The output variables that we are going to discuss, outlined in Section 4, can be gathered into the following three categories: (i) number of subscriptions, (ii) commuters' travel performance measures and (iii) mobility resources usage.

5.1.1. Number of subscriptions

A subscriber is a commuter who, at the end of the simulation, is expected to change his traveling habits in favor of the new mobility services. Thus, the number of subscribers is a measure of people acceptance of an SMI. Fig. 8 shows that light-SMIs can attract between 20% and 42% of the whole population; the medium-SMIs could involve from 60% up to 72%; whereas the massive ones could interest around 85% of the population. These results show that the number of subscribers grows with the number of vehicles provided, both in the light and medium SMIs. However, when the number of subscriber is close to the total number of citizens, such as in the massive-SMIs, increasing the number of vehicles is not sufficient anymore. For instance, SMI-7 is more successful than SMI-9, even though the total amount of vehicles is less than half the other.

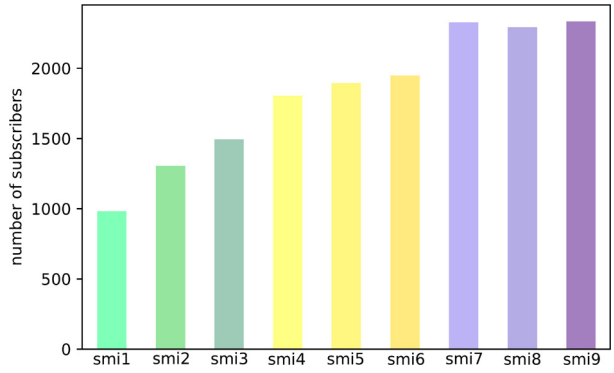

Fig. 8. N. of subscribers. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

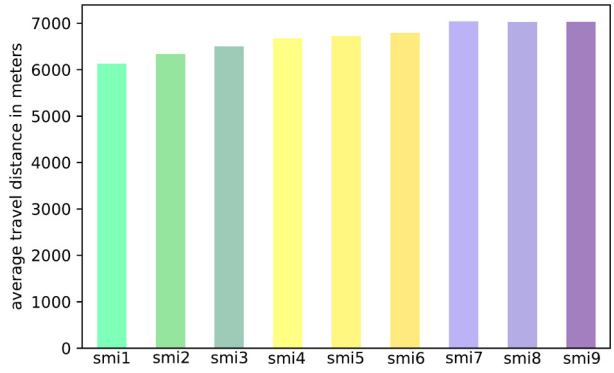

Fig. 9. Travel distances.

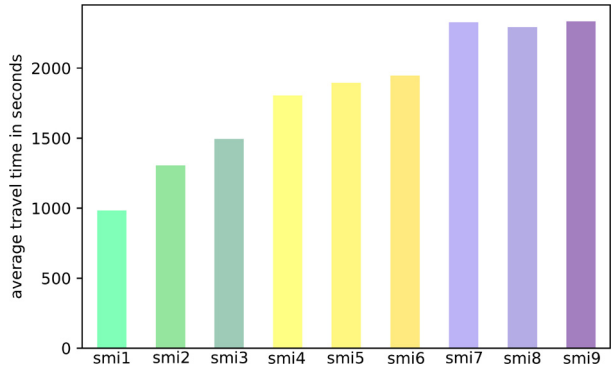

Fig. 10. Travel times.

5.1.2. Commuters' performance measures

The performance measures of commuters depend on the number of subscribers. Indeed, concerning travel times and travel distances (Figs. 9 and 10), their averages grow as the number of subscribers of the SMI increases. These trends are due to the fact that subscribers will extend their usual trips because they now pass through two tangrhubs instead of making a direct trip from origin to destination. Moreover, subscribers perform their first-mile and last-mile trips by walk, which is considerably time-consuming.

Even mobility costs and CO_2 emissions (Figs. 11 and 12) follow the trend of subscribers, but in an inverse relationship: the higher the number of subscribers, the lower the average CO_2 emissions and mobility costs. Specifically, the CO_2 decrease is due to the fact that all the tangrhubs are provided with green vehicles. Thus, when the amount of subscribers is around 85% (smi-7, smi-8 and smi-9), the carbon footprint of a commuter is almost zero. Concerning the mobility costs decrease with the subscriptions growth, this is due to the fact that commuters are just paying for the time spent traveling. The fixed costs of owning a vehicle are thus shared with the community. Indeed, in smi-1 the average daily cost of traveling is just less than €8 per commuter; in smi-7 it is 4 times less.

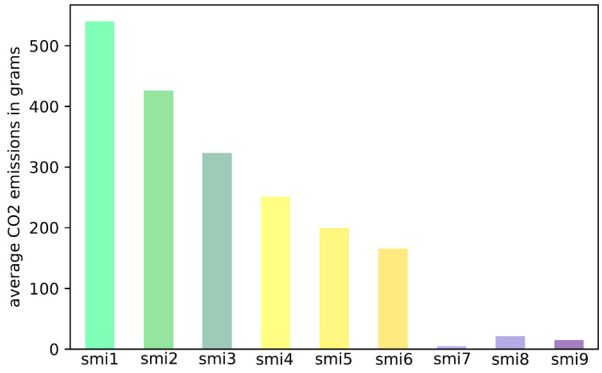

Fig. 11. CO_2 emissions.

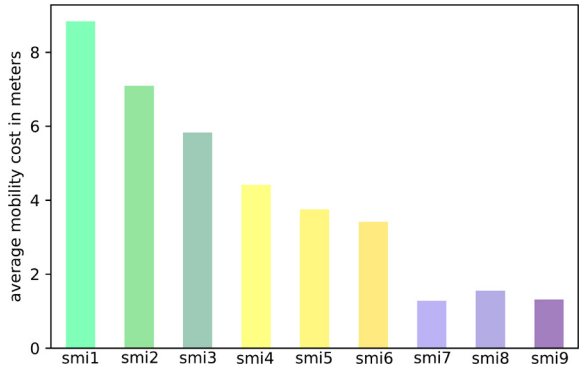

Fig. 12. Mobility costs.

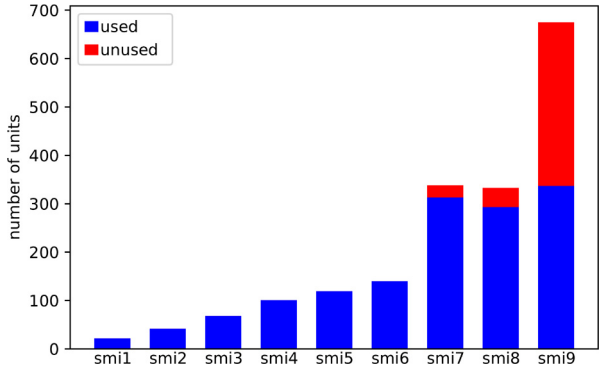

Fig. 13. Mobility fleet usage.

5.1.3. Mobility resources usage

Fig. 13 shows the proportion of used and unused vehicles for each initiative. Light and medium SMIs are well configured, since there are no unused vehicles that would otherwise result in a waste of resources. Conversely, as the number of subscribers gets closer to 100%, the distribution of resources becomes tougher (all the massive-SMIs have unused vehicles).

5.2. Validation: comparing ToolTRain with Tangramob

The selection process of a *light* and a *medium* SMI is not trivial, since their performance is quite similar in scale. Thus, for each of these groups we selected the SMI with the lowest deployment of resources, i.e. smi-1 and smi-4. For what concerns the massive-SMIs group, we selected the one with the lowest unused resources, (i.e. smi-7), since it is more efficient.

As shown in Figs. 14, 15, 17 and 19, number of subscribers, travel distances, CO_2 emissions and resources usage are almost the same between Tangramob and ToolTRain. For the mobility costs parameter, Fig. 18 shows that both smi-1 and smi-4 are very similar; whereas for smi-7 the difference is less than one euro, which is acceptable. On the other hand, travel

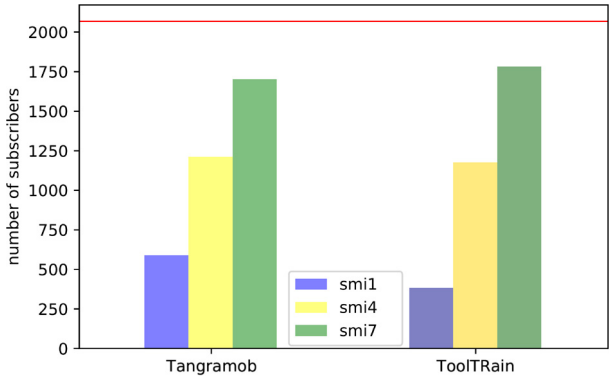

Fig. 14. Subscriptions.

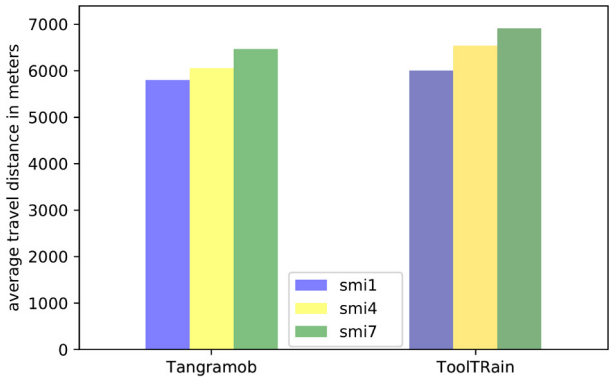

Fig. 15. Travel distances.

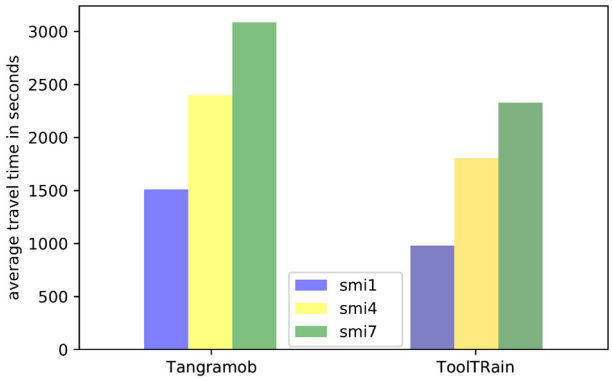

Fig. 16. Travel times.

times (Fig. 16) are different, but at least they follow the same upward trend. Nevertheless, even though the TRebeca model still lacks a realistic representation of traffic, we can conclude that H1 is verified.

5.3. Computational performance statistics

In order to compare Tangramob and ToolTRain in terms of computational time required for a single experiment, Table 3 reports the CPU time of each selected SMI for both a Tangramob simulation and a ToolTRain run. More precisely, the experiments are performed on a Manjaro Linux desktop with an i7-4790S CPU @ 3.20 GHz and 16 GB RAM. Each Tangramob simulation is configured for 110 iterations. It is also worth remarking that the time needed for pre-computing the values of the model, as well as the model generation itself, explained in section 3.4.1, is negligible since it is very low. Moreover, since this operation is done once, such a time is not considered as a relevant factor for the overall time.

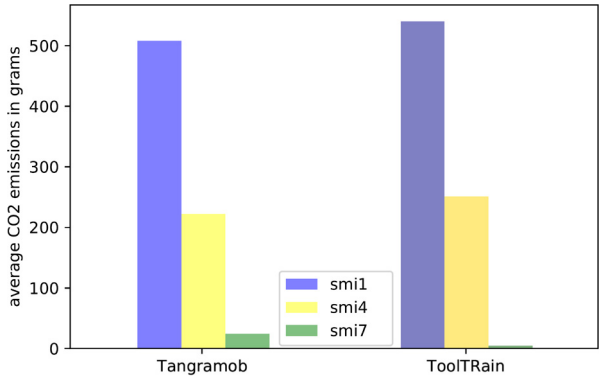

Fig. 17. CO_2 emissions.

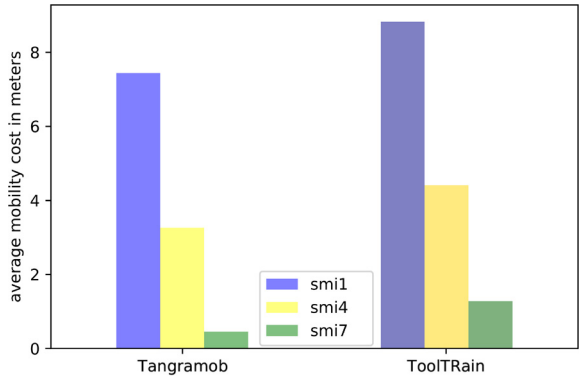

Fig. 18. Mobility costs.

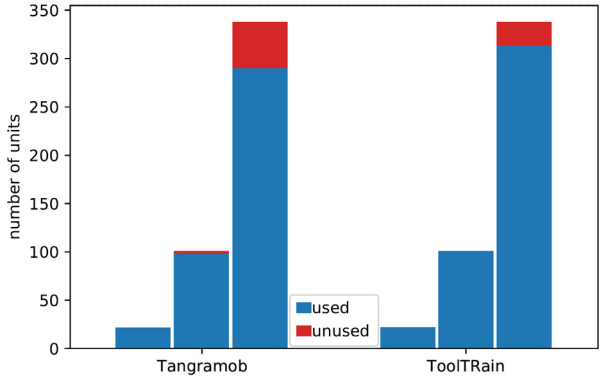

Fig. 19. Mobility fleet usage.

Table 3
Computational times.

	SMI-1	SMI-4	SMI-7	Iterations
Tangramob	693413 ms	841782 ms	947465 ms	110
ToolTRain-old	45641 ms	57882 ms	69242 ms	–
ToolTRain-new	71868 ms	104279 ms	126633 ms	–

5.4. Comparison of the models' outcomes

Comparing the results in Figs. 14–19 and the corresponding ones presented in [8], it emerges that the model presented in this paper yields more convincing results especially regarding the travel times, which were still not well-aligned to that of Tangramob. Concerning the other performance measures, there are no significant changes in the outputs, so we are omitting

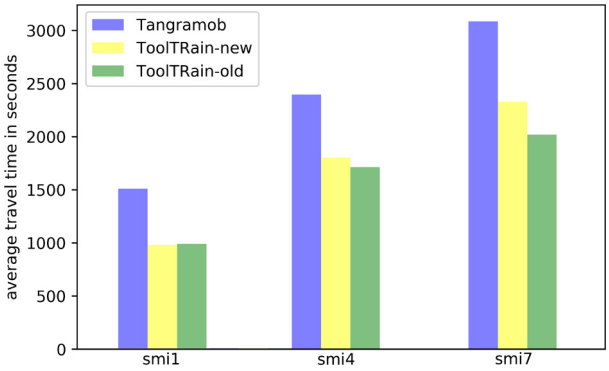

Fig. 20. Comparison of travel times in ToolTRain versions.

their comparison in this paper. While it is true that such a simplified non-iterative model cannot represent urban traffic as accurately as Tangramob, ToolTRain's travel times are now closer to those of the simulator, as can be seen in Fig. 20. More precisely, the travel times of ToolTRain are about 4/5 of those resulting from the simulator.

6. Towards other application contexts

The scope of this section is to show how the approach behind our work, and more technically the one offered by ToolTRain, can be applied in contexts other than smart mobility simulation. For this purpose, first we present and describe a new case study, highlighting both the objectives and the main difficulties behind the problem, then we suggest a similar methodology to address the problem and meet the goals of the project. Moreover, we want to emphasize that specifying a model with TRebeca and the ToolTRain approach are enough flexible to be exploited in different contexts, whereas Tangramob is not so adaptable to other scenarios since it has a specific goal and logic. Furthermore, due to its complexity, a possible redesign and reimplementation would require very big efforts.

6.1. The Electric Work Site Project

The case study chosen for this section is the Electric Work Site Project by Volvo [15], for which a first study has been proposed in [16]. Basically, it involves operating an electrified quarry site and managing a fleet of haulers navigating autonomously and carrying out predefined tasks. In this problem, haulers are intended to work in a fleet manner for performing tasks such as material transport, loading, unloading, charging etc. in a cyclic manner. The whole simplified process is depicted in Fig. 21 and can be described as follows:

1. The materials primarily demolished at a quarry site are *loaded* into a Primary Crusher (PC) where they are crushed and remnants are *loaded*. On a technical viewpoint, haulers are *loaded* using a wheel loader.
2. The materials are then *transported* and *unloaded* to a Secondary Crusher (SC).
3. There is a charge station (i.e. a number of chargers) for the operating vehicles: every hauler returning from the unloading point *shall* be charged to full battery.

In this scenario, each hauler is assigned unique tasks like getting loaded at PC, unloading at SC, charging at charge stations, navigating or waiting.

The main goal of this project is to find an efficient and effective schedule of activities among haulers in such a way to optimize the following performance measures:

- amount of material transported;
- idle time of the haulers;
- number of chargers;
- travel times;
- travel distances;
- energy consumed,

while consistently respecting a predefined set of constraints and desired properties like:

- no hauler is allowed to load materials at the loading point while the PC is in operation;
- limitations on the speed of haulers apply in some areas for safety concerns;
- haulers in competition of shared resources do not collide with each other.

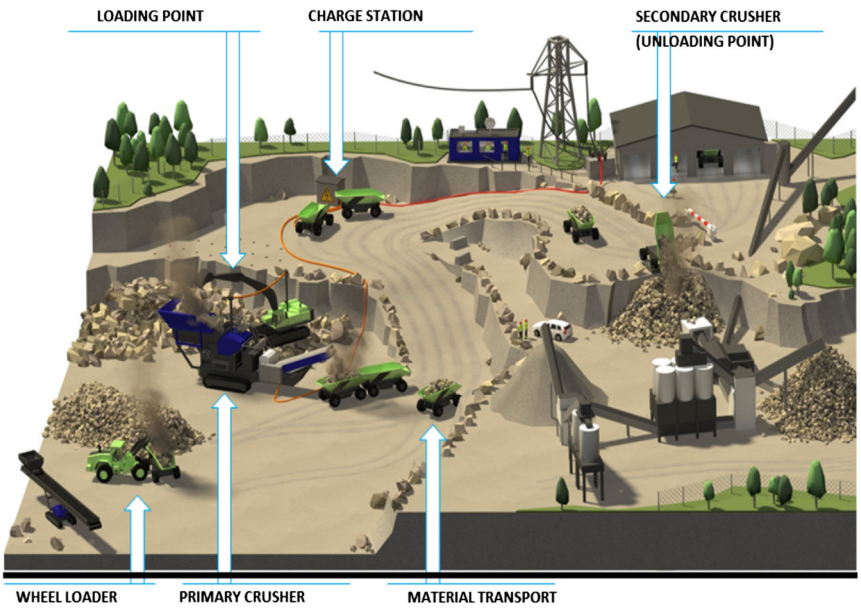

Fig. 21. Representation of the Electric Site Project.

As it emerges from these constraints, the path segment through which transportation takes place is a collision prone area, since haulers tends to meet together.

Analogously to the methodology described in Section 3, the approach we propose in this regard consists in capturing, at a proper level of detail, all the relevant aspects of problem in a Timed Rebeca model. Once such a model has been properly formalized and developed, it can be used as a reference architecture for instantiating executable models starting from a set of input files describing the configuration of the quarry site. The role of the reference model is thus similar to a class definition in an object-oriented programming language: it serves like a mould for baking cookies, but the whole preparation process is fully automated. In particular, the second and final step of our approach is to develop a chain of software components which together implement the idea of ToolTRain. More precisely, as discussed in Section 3.4, such a tool-chain would provide the following components: i) a program for instantiating a Timed Rebeca model from the configuration of the quarry site; ii) a routine for running the resulting model instance; iii) a post-processing script for extracting and inferring the performance measures discussed above from the resulting state space.

Although it is out of the scope of this paper to provide a formal and detailed description of both model and tool-chain of this new problem, the following subsections are intended to provide more information on the approach. The purpose of this investigation is to encourage the reader to critically evaluate a similar methodology for other contexts of application.

6.2. From the problem to a reference model: a brief investigation on the actors

Modeling the Electric Work Site problem in an actor-based modeling language, like Timed Rebeca, requires first to identify the actors involved in order to further describe their behavior by means of message servers and message exchange.

In this problem, we consider the following entities as rebec types: loading points (e.g. primary crushers), unloading points (e.g. secondary crushers), charging stations and the network describing the accessible paths of the quarry site. All these rebecs share the same goal, i.e. optimizing the performance measures discussed in Section 6.1, and each of them is responsible for certain predefined tasks. The key to success is behind the cooperation among rebecs by means of message passing. This will guarantee that undesired actions shall not occur during the execution of the model.

Haulers are the actual core of the model: they are responsible for moving materials from loading points to unloading points; traveling safely on the accessible areas of the quarry site; and charging their battery once a loading-unloading cycle has been completed. Analogously to commuters, haulers can be modeled as messages, and their flow among the other rebecs describes both their movement and the execution of their tasks. There is indeed a close similarity with the characterization of commuters described in Section 3.3. In fact, while commuters follow a personal plan of daily activities and legs (Section 2.1), we can think to organize and assign a schedule (i.e. a sequence of tasks) to each hauler rebec. Moreover, it is possible to group these tasks within the same categories of plan elements: activities and legs. More precisely, we can think of *idle*, *loading*, *unloading* and *charging* as activities, whereas a single movement from one activity to the next one is still a leg with its own route (i.e. a sequence of path segments to traverse). Therefore, considering schedules as plans makes it possible to keep the same characterization of commuters and use similar input files to describe the number and the configuration of each hauler (Listings 4 and 5).

```
<person id="0" age="30" employed="no">
  <plan selected="yes">
    <act type="home" x="0.0" y="0.0" link="3466" end_time="08:00:00"/>
    <leg mode="car" route="3466 7874 9588 1395"></leg>
    <act type="work" x="0.0" y="200.0" link="1395" end_time="17:00:00"/>
    <leg mode="car" route="1395 9588 7874 3466"></leg>
    <act type="home" x="0.0" y="0.0" link="3466" end_time="24:00:00"/>
  </plan>
</person>
```

Listing 4: Example of commuter's plan.

```
<hauler id="HX0" battery−capacity="100Wh" status="operative">
  <schedule selected="yes">
    <act type="idle" x="0.0" y="0.0" link="150" end_time="06:00:00"/>
    <leg route="150 237 656 321"></leg>
    <act type="load" x="30.0" y="20.0" link="321" duration="00:20:00"/>
    <leg route="321 478 942"></leg>
    <act type="unload" x="40.0" y="25.0" link="942" duration="00:10:00"/>
    <leg route="942 656 321 276"></leg>
    ...
  </schedule>
</hauler>
```

Listing 5: Example of hauler's schedule.

Achieving a collision-free scenario is the reason why the road/path network of the quarry site, from now on just network, is modeled as separate rebecs. In particular, the network can be modeled as a collection of rebecs, one for each critical and potentially dangerous sections of the quarry site, i.e. road/path intersections. These are indeed the areas where two or more paths overlap and collisions are thus more likely to occur. As can be seen in Listing 5, each leg of a hauler's schedule is specifically assigned to a route. Unlike those of commuters, we describe a route as a sequence of intersections that the hauler has to traverse in order to reach the location of the next activity. Therefore, intersection rebecs manage the flow of haulers (messages) to guarantee mutual exclusion and avoid collisions in critical areas of the quarry site. These rebecs thus act as referees, responsible for ensuring a safe and fair allocation of space resources. Traveling among intersections without collisions is among the autonomous capabilities of the HX machines used as haulers in this project [15]. Nevertheless, splitting the original network of the quarry site in lots of intersections would provide a more granular control on the movement of haulers. As a consequence, all the runs of the model would be completely collision-free by design, at the cost of overloading communications among rebecs.

Technically, there is no difference between the road network characterization of Tangramob and that of the quarry site, and this allows to use the very same representation given in Section 2. The resulting network is thus a weighted multigraph with links denoting path segments and nodes representing intersections or, more generally, strategic points in which an interaction with the network rebec shall occur for the sake of travel progress.

Loading points, unloading points and charging stations can be considered as the main facilities for the operation of haulers and, together with the network, they describe the configuration of a quarry site. Each of them can thus be modeled as a rebec type with its own features and behavior, according to the project requirements. For instance, we can imagine a loading point rebec as an entity defining the rules and the behavior (i.e. the protocol) that haulers should follow for: approaching the crusher; getting filled of materials by a wheel loader; managing the collaboration with other haulers that are performing the same operation; handling exceptional situations (e.g. local incidents and malfunctions); and leaving the area. Moreover, each loading point has its own features such as the location on the network (i.e. a reference to a link), the number of wheel loaders available, the minimum and the maximum amount of materials that should be loaded every day and so forth.

6.3. A fully-automated approach for simulation

The previous section provided a few insights on the actors involved in the model we are planning to design and develop. Once a clear definition of the model is complete, we will be able to implement the before mentioned tool-chain in order to get the most from the model and improve its usability. For this purpose, we use the architecture of ToolTRain.

As outlined in Fig. 22, the first two modules of the pipeline are very similar to those of the ToolTRain architecture used for Tangramob. Basically, the first module takes the configuration of the quarry site (i.e. network and facilities) and the schedules of the available haulers as input, and generates an instance of the Timed Rebeca reference model as output. Moreover, in order to exploit the pre-processing capabilities of this module, the user can also provide the `schedules.xml` input file with just the declaration of the available haulers, thus giving no detail on the actual schedules (no task is assigned). In

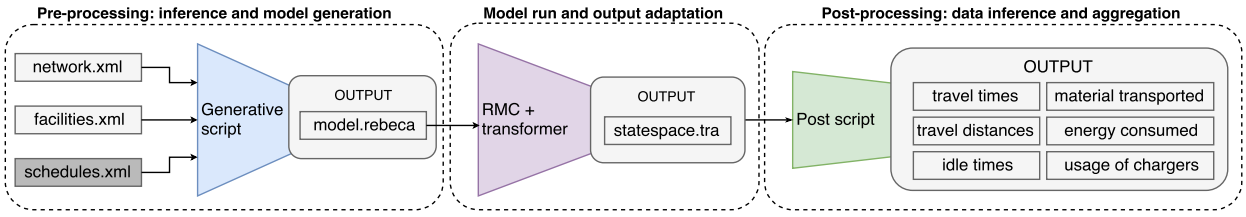

Fig. 22. The ToolTRain-like architecture of the new pipeline.

this case, the script will attempt to build a number of schedules by means of heuristics and custom strategies, similarly to the approach described in Section 6 (if n schedules emerge from this step, n different models will be generated). Next, the second module is responsible to run RMC on the model resulting from the previous step; if more than one model is generated, the module will run each of them in parallel. Finally, once the computation of the model(s) is over, the last module will exploit the resulting state space(s) to collect and infer the output variables (i.e. the performance measures outlined in Section 6.1) that summarize the performance of the scenario(s) in light of user's objectives.

7. Related work

Tangramob is a simulator supporting intermodality and multimodality in a context independent architecture. This section provides the reader with an overview of common approaches aimed at handling large-scale scenarios within reasonable computational time, a crucial challenge in agent-based traffic simulations. The literature provides different solutions to this problem, which can be classified into two groups: *technical approaches* and *model-based ones*.

Besides these groups, it is also worth to remark that a mapping from complex multi-agent systems to actor-based models was also proposed in [17]. More precisely, their approach is based on actors and asynchronous message passing, and exploits the UPPAL statistical model checker (SMC) for the experiments. In their work, the approach was applied to modeling and analysis of a large and adaptive version of the Prisoner Dilemma game, where performance issues are limited by UPPAL SMC. Even tough we share the same approach, i.e. modelling multi-agent systems via an actor-based language, the purpose of our investigation is different: whereas we aim at collecting some performance criteria, Nigro et al. [17] seek to verify some properties. Moreover, since we want to represent an ABM with many collaborative entities and we consider time as an important aspect of the simulation, we preferred to use TRebeca instead of UPPAL SMC.

7.1. Technical approaches

The first group collects all those alternatives which keep the integrity of the traffic ABM, whereas trying to decrease the computational complexity by means of some expedients, such as: reducing the input dimension and optimizing the available computational resources. For instance, in [9] the practice is to scale down the model, i.e. instead of modeling the 100% of a city's population, only a representative portion is considered. It is thus possible to get comparable system dynamics with a 10% population if the transport system capacities are scaled down proportionally. Another approach is to harvest the computational power of Graphical Processing Units (GPU). For instance, [18] achieved a speedup of up to 67 times by re-implementing MATSim for Compute Unified Device Architecture (CUDA), a framework enabling programs to perform calculations using both the CPU and GPU.

7.2. Model-based approaches

On the other hand, the second group comprises alternative approaches to model traffic at a more coarse-grained level, often resulting in loss of details. Indeed, microscopic traffic simulations are much computational demanding than other models, since they track the movement of each vehicle as well as the interactions among vehicles competing for roads. In contrast, macroscopic models aggregate vehicles, and traffic is described as a continuum. For instance, [23] introduces MacroSim, a MATSim's module for macroscopic mobility simulations. In MacroSim, agents are handled sequentially and decoupled from each other, as well as from the environment, over the simulation. Their interactions are thus represented at a higher abstraction level by means of constraints in capacity and speed on each road of the network, expressed by volume-delay functions. With MacroSim, the simulation approach changes from a system-based to an individual-based one, allowing a more efficient parallelization of the mobility simulation (7 to 50 times faster).

Another modeling approach is given by [19] in which, instead of performing a microscopic traffic simulation along fixed time steps, an event-based model is used, performing only discrete actions which are relevant to the model (i.e. entering and leaving roads). This model is called Deterministic Event-Driven Queue-Based Traffic Flow Micro-Simulation (DEQSim). Compared to earlier queue-based approaches, DEQSim saves time in areas of the road network where the traffic load is small or moderate, leading to a speedup of more than 10 compared to the time-step-based approaches. In addition, [20] presents methods to increase the performance of the micro simulation model of MATSim using event-driven concepts as well as a parallel implementation. In particular, DEQSim was redesigned, giving birth to JDEQSim. This implementation consists of

the following three parts: (i) vehicles and links are the basic units of the simulation; (ii) communication among units takes place by exchanging messages via a scheduler, where each message contains a time stamp (e.g. when a vehicle is allowed to enter the next link); (iii) the scheduler contains a message priority queue, which is ordered by message time and type.

Thanks to a parallelization of JDEQSim, the authors managed to accelerate MATSim substantially, making it possible to simulate bigger runs with much fewer CPUs.

Though scaling the model to a smaller yet representative population helps saving time, this is still not enough to cope with both large-scale scenarios and the shared mobility services supported by Tangramob. Concerning the exploitation of GPU, it is worth considering that a CUDA implementation requires considerable design efforts since Tangramob is developed in Java. Moreover, the dependencies among agents and the environment, typical of macroscopic simulations, make it difficult to reach a good parallelization of the model.

For what concerns model-based approaches, shifting from a micro to a macroscopic model by means of abstractions is useful in some contexts. However, Tangramob aims at modeling both intermodal trips, which follow different traveling patterns than usual ones, and the acceptance of a mobility initiative for every single person of a sample population. This last consideration is due to the fact that an urban planner should be able to find a good balance of mobility resources for a certain district according to the actual mobility needs of the nearby citizens. Thus, it turns out that a macroscopic modeling approach is not suitable for our scopes. The event-based approach suggested in [19] can be even improved by modeling other traffic dynamics as messages. In fact, the redesign presented in [20] suggests an interesting approach to follow in order to speed the computational burden of Tangramob simulations.

Another work sharing our intent is that of Crociani et al. [21] in the context of crowd management in urban scenarios. Similarly to our simulations, given the large size of these environments, as well as the high number of simultaneously present pedestrians, the computational costs of a pure microscopic simulation approach can be prohibitive. To address this problem, the authors propose a multi-scale approach which combines two simulation models of different granularity, i.e. a microscopic cellular automata (CA) based model combined with a fast mesoscopic queue based simulation model. More precisely:

1. the CA is applied to complex situations with high pedestrian interactions (e.g. high density counter-flows), and this allows the simulation system to provide a very detailed representation of parts of the scenario in which more complex behaviors can take place;
2. the queue model is employed to the wider area, where pedestrian densities are rather low. Such a mesoscopic approach can thus used to design and simulate large parts of the urban environment that are not affected by such complex dynamics but are still fundamental for the analysis of the overall scene.

The combination of these models makes it is possible to simulate large and complex scenarios in reasonable time frames [21]. Considering that we also aim at using different model with different level of granularity, we can identify a similarity with their approach. However, whereas the proposed models are used jointly to provide a complementary view of the different dynamics involved in the simulation, our simplified model aims at replacing Tangramob in the first stages of the investigation of several smart mobility initiatives.

8. Conclusions and future work

Measuring the impacts of mobility initiatives prior to their development is a complex and risk-bearing task in urban planning. A Decision Support System (DSS) like Tangramob can support both urban planners and transport companies in this task, but the computational requirements of the iterative simulations might discourage its application in large scenarios.

In the conference paper [8], we showed how the Agent-Based Model (ABM) of Tangramob can be simplified into a Timed Rebeca (TRebeca) model, which allows users to get an idea of a mobility initiative in a shorter time. To make this model more usable, we also designed ToolTRain, a tool-chain for generating an instance of the TRebeca model from the same input of Tangramob; running the resulting model; and inferring the output from its run.

The comparative experiment designed to validate this approach shows a positive correlation between the output variables of both Tangramob and ToolTRain. Also, a comparison of the computational times supports the lighter demands of ToolTRain, since a run of the model requires around 12% of the time needed for its corresponding Tangramob simulation.

In this extended paper, we provided a complete and technical description of the TRebeca reference model we used to emulate Tangramob, as well as the improvements made to represent urban traffic dynamics (e.g. travel times, preferences of commuters) at a more realistic level. Moreover, besides the results obtained from these enhancements, we emphasized that the conceptual organization and the architecture of ToolTRain can be reused in other application domains. For this purpose, we presented the Electric Work Site Project by Volvo [15], which intends to operate an automated and electrified quarry, and we investigated how the approach described in this paper can be used to address the problem and to help the user achieve the goals of the project. This investigation is intended to encourage the reader to critically evaluate a similar methodology for other contexts of application.

As future work, we are planning to further improve the TRebeca model in order to introduce new mobility services. Moreover, we will extend a fully automated tool to provide the modeling and analysis of self-adaptive urban planning systems at runtime. The resulting system would allow tangrhubs to adapt their mobility services at runtime, in response to

service disruptions, commuters' traveling experience and changes in the environment (e.g. car accidents, strikes). In this way, the urban planner will be able to observe the TRebeca model until a convergence criterion is met (e.g. 70% of subscribers).

Moreover, in order to obtain a closer (yet approximated) representation of urban traffic, we are currently designing a simplified and lightweight queue model as the one proposed in [22] that we can further integrate into ToolTRain.

Acknowledgements

The work of the 3rd and a part of that of the 4th author are supported by "Self-Adaptive Actors: SEADA" (project nr. 163205-051) from Icelandic Research Fund. Work of the 2nd and 4th author is also supported by DPAC Project at Mälardalen University.

References

[1] United Nations, The World's Cities in 2016, New York, 2016.
[2] C. Benevolo, R.P. Dameri, B. D'Auria, Smart mobility in smart city, in: Empowering Organizations, Springer, 2016, pp. 13–28.
[3] A. Mamiit, Why the ride-sharing company failed to conquer China and what it means for everyone else, https://goo.gl/cHuC9n, 2016.
[4] K. Fehrenbacher, Another failed attempt to make ride sharing work in the U.S., ridejoy to shut down, https://goo.gl/3ITYce.
[5] X. Zhou, J. Taylor, Dtalite: a queue-based mesoscopic traffic simulator for fast model evaluation and calibration, Cogent Eng. 1 (1) (2014) 961345.
[6] C. Castagnari, F. Corradini, F.D. Angelis, J. de Berardinis, G. Forcina, A. Polini, Tangramob: an agent-based simulation framework for validating urban smart mobility solutions, arXiv:1805.10906, 2018.
[7] A.H. Reynisson, M. Sirjani, L. Aceto, M. Cimini, A. Jafari, A. Ingolfsdottir, S.H. Sigurdarson, Modelling and simulation of asynchronous real-time systems using timed Rebeca, Sci. Comput. Program. 89 (2014) 41–68.
[8] C. Castagnari, J. de Berardinis, G. Forcina, A. Jafari, M. Sirjani, Lightweight preprocessing for agent-based simulation of smart mobility initiatives, in: LNCS Workshop Proceedings of SEFM, 2017.
[9] A. Horni, K. Nagel, K.W. Axhausen, The Multi-Agent Transport Simulation MATSim, Ubiquity-Press, London, 2016.
[10] M. Sirjani, Rebeca: theory, applications, and tools, in: Formal Methods for Components and Objects, 5th International Symposium, 2006, pp. 102–126.
[11] M. Sirjani, E. Khamespanah, On time actors, in: Essays Dedicated to Frank De Boer on Theory and Practice of Formal Methods, vol. 9660, Springer-Verlag Inc., New York, 2016, pp. 373–392.
[12] E. Khamespanah, M. Sirjani, Z.S. Kaviani, R. Khosravi, M.-J. Izadi, Timed Rebeca schedulability and deadlock freedom analysis using bounded floating time transition system, Sci. Comput. Program. 98 (2015) 184–204.
[13] U. Weidmann, Transporttechnik der Fussgänger, 1992.
[14] R. Axtell, R. Axelrod, J.M. Epstein, M.D. Cohen, Aligning simulation models: a case study and results, Comput. Math. Organ. Theory 1 (2) (1996) 123–141.
[15] Volvo Construction Equipment, The electric site research project, www.volvoce.com/global/en/this-is-volvo-ce/what-we-believe-in/innovation/.
[16] A. Jafari, J.J. Surendran Nair, S. Baumgart, M. Sirjani, Safe and efficient fleet operation for autonomous machines: an actor-based approach, in: Proceedings of the Symposium on Applied Computing, SAC '18, ACM, New York, NY, USA, 2018, http://doi.acm.org/10.1145/3167132.3167382.
[17] C. Nigro, L. Nigro, P.F. Sciammarella, Modelling and analysis of multi-agent systems using UPPAAL SMC, Int. J. Simul Process Model. 13 (1) (2018) 73–87.
[18] D. Strippgen, K. Nagel, Multi-agent traffic simulation with CUDA, in: High Performance Computing & Simulation, 2009, HPCS'09, IEEE, 2009.
[19] D. Charypar, K. Axhausen, K. Nagel, Event-driven queue-based traffic flow microsimulation, Transp. Res. Rec., J. Transp. Res. Board (2003) (2007) 35–40.
[20] R.A. Waraich, D. Charypar, M. Balmer, K.W. Axhausen, Performance improvements for large scale traffic simulation in MATSim, in: 9th Swiss Transport Research Conference, Ascona, 2009.
[21] L. Crociani, G. Lämmel, G. Vizzari, Multi-scale simulation for crowd management: a case study in an urban scenario, in: International Conference on Autonomous Agents and Multiagent Systems, Springer, 2016, pp. 147–162.
[22] A. Agarwal, G. Lämmel, K. Nagel, Incorporating within link dynamics in an agent-based computationally faster and scalable queue model, Transportmetrica A: Transp. Sci. 14 (5) (2018) 520–541.
[23] P. Bosh, F. Ciari, MacroSim - A macroscopic Mobsim for MATSim, Proc. Comput. Sci. 109 (2017) 861–868, Elsevier.