Lightweight Preprocessing for Agent-Based Simulation of Smart Mobility Initiatives

Carlo Castagnari[1], Jacopo de Berardinis[1], Giorgio Forcina[1], Ali Jafari[3], and Marjan Sirjani[2,3]

[1] University of Camerino, Division of Computer Science, Smart Mobility Lab
Via Madonna delle Carceri 9, Camerino (MC), 62032, Italy
[2] Malardalen University, School of Innovation, Design and Engineering
Hogskoleplan 1, Vasteras, 72123, Sweden
[3] Reykjavik University, School of Computer Science
Menntavegur 1, Reykjavik, 101, Iceland

Abstract. Understanding the impacts of a mobility initiative prior to deployment is a complex task for both urban planners and transport companies. To support this task, Tangramob offers an agent-based simulation framework for assessing the evolution of urban traffic after the introduction of new mobility services. However, Tangramob simulations are computationally expensive due to their iterative nature. Thus, we simplified the Tangramob model into a Timed Rebeca (TRebeca) model and we designed a tool-chain that generates instances of this model starting from the same Tangramob's inputs. Running TRebeca models allows users to get an idea of how mobility initiatives affect the system performance, in a short time, without resorting to the simulator. To validate this approach, we compared the output of both the simulator and the TRebeca model on a collection of mobility initiatives. Results show a correlation between them, thus demonstrating the usefulness of using TRebeca models for unconventional contexts of application.

Keywords: Agent-Based Simulations, Actor-Based Modeling Languages

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 [14], 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 will jeopardize the quality of life of citizens, accentuating many problems like traffic congestion, high cost of personal mobility, land use inefficiencies as well as the environmental impacts.

Urged by these threats, urban planners are now shifting their attention to Smart Mobility, defined as "a complex set of projects and actions, different in

The work of the 4th and a part of that of the 5th author are supported by "Self-Adaptive Actors: SEADA" (project nr.163205-051) from Icelandic Research Fund.

goals, contents and technology intensity" [2] focused on mobility issues. Examples of smart mobility services range from carsharing and bikesharing services to more advanced ones like self-driving taxis and dynamic ridesharing systems.

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. In fact, the current approach of using heuristics and best-practises might end up being risky for decision makers, since the resulting mobility initiatives may be unaccepted by the population [6, 9].

Driven by these motivations, *Tangramob* [4] offers a simulation environment for mobility assessment. This open source tool allows urban planners and transport companies to understand 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 mobility initiative is accepted by the population and how it affects both citizens and 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, a Tangramob user 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) [10] model in which the RL-based learning process and the representation of traffic were both 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 a tool chain that generates instances of the TRebeca model from the same input files of Tangramob scenarios. Despite the many simplifications, comparing the results obtained from the two models on different mobility initiatives for the same scenario shows that the TRebeca model behaves similarly to Tangramob. This correlation makes it possible for users to 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.

The structure of this paper is as follows: Section 3 gives an overview of Tangramob, as well as a brief description of its ABM. In Section 4 we introduce the Rebeca and Timed Rebeca modeling languages and we present the simplified Timed Rebeca model derived from the original one. Section 4.3 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 5 we describe how we designed the experiment to prove both the similarity among the two models and the utility of the simplified TRebeca model. Finally, in Section 6 we show the experimental results and we discuss about their implications in order to confirm the former hypothesis.

2 Related work

To the best of our knowledge, Tangramob is the first simulator supporting intermodality and multimodality in a context-independent architecture. This section thus provides the reader with an overview of common approaches aimed at handling large-scale scenarios within reasonable computational time. Indeed, this is one of the most faced challenges in Agent-Based (AB) traffic simulations. The state of the art provides different solutions to this problem, which can be classified into two groups: *technical approaches* and *model-based ones.*

The first group collects all those alternatives which keep the integrity of the AB traffic model, 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 [7] 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, [13] achieved a speedup of up to 67 times by means of a CUDA re-implementation of MATSim.

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, [3] introduces MacroSim, a MATSim's module for macroscopic mobility simulations. In MacroSim, agents are handled sequentially and decoupled from each other, as well 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 [5] 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).

Although scaling the model to a smaller yet representative population is certainly helpful to save time, this approach is still not enough to cope with both large-scale scenarios and shared mobility services. Concerning the exploitation of GPU computing, 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 purposes. On the other hand, the event-based approach suggested in [5] can be even improved by modeling other traffic dynamics as messages.

3 The Tangramob simulator

Tangramob [4] is an agent-based simulation framework supporting urban planners and transport companies in shaping Smart Mobility Initiatives (SMIs) within urban areas. Users can thus 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. Technically, a Tangramob simulation requires the following inputs:

- **road network** of the urban 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),
- **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.

Once the above inputs are provided, Tangramob can start a simulation which returns several output files in order 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 added 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.

The detailed description of Tangramob is out of the scope of this work; nevertheless, in order to present the ABM as well as the derivation process for the TRebeca model, we introduce the following key concepts to the reader: *tangrhub*, *smart mobility initiative* and *commuting pattern*.
A **tangrhub** is a geo-located container of mobility services (e.g. carsharing, bikesharing), each of which in turn manages a fleet of vehicles (e.g. cars, bikes).
A **Smart Mobility Initiative (SMI)** is thus about placing a number of *tangrhubs* within the urban area of interest, adding one or more mobility services to each of them, and providing a characterization for each service. In particular, to define a smart mobility service for a *tangrhub*, the user has to specify the service type (i.e. intra-hub or inter-hub), the initial number of vehicles and the service charge (i.e. cost per km, per hour and fixed cost). 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 **commuting pattern** is the intermodal representation of how a person moves from an origin location to his destination. Such a trip can be either simple, e.g. the commuter directly travel from origin to destination by either walking or car; or more complex, e.g. the commuter will use more than one means of transport, for instance: walking to the closest bus station, travel by bus, then reach a metro line and so on. A clear example of a commuting pattern is the route provided by the trip planner of Google Maps. However, how will the commuting patterns of citizens change as the smart mobility initiative is introduced? In Tangramob, the complexity of commuting patterns is limited to three schemes: direct path, 2-trip path and 3-trip path (shown in Figures 1, 2, 3, respectively). In particular, nodes O and D represent respectively the commuter origin and destination; 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.

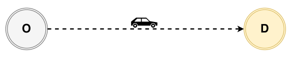

Fig. 1. Direct path

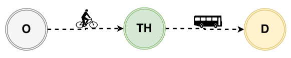

Fig. 2. 2-trip path

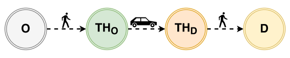

Fig. 3. 3-trip path

3.1 Tangramob Agent-Based Model (ABM): an overview

This section introduces the formalization of Tangramob, that will be helpful to understand the translation of the original agent-based model into the simplified TRebeca one (presented in Section 4.2).

The Tangramob ABM has two different 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 smart mobility services' state space. 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 Figure 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 (a traveling alternative can be thought of as a combination of one up to three legs, each of which can involve a smart mobility service and it is based on the Tangramob commuting patterns). Next, 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.

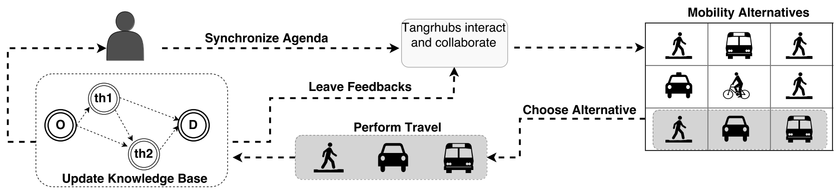

Fig. 4. Commuter-Tangrhub interaction loop

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, traveled distance and travel comfort. The use of a feedback is dual: on one hand, it allows a person to make more informed decisions in the future; on the other hand, it enables *tangrhubs* to improve their mobility services.

To do this, following a Reinforcement Learning approach, Tangramob simulations are iterative: each iteration corresponds to a typical day in which commuters try the new mobility services and record their experience.

At the end of a simulation, commuters may change their original mobility habits in favour of those mobility services that can better accommodate their needs. In case the simulated SMI does not suit some commuters, those agents will then restore their initial traveling habits (e.g. traveling by private car).

4 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 simplified TRebeca model, and we argue about its derivation process from the Tangramob ABM.

4.1 Rebeca and Timed Rebeca

Reactive Object Language (Rebeca) [11] 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, 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 behaviour 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 behaviour 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 [11], 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 behaviour of a Rebeca model is hence defined as the parallel execution of the released messages of the rebecs.

Timed Rebeca (TRebeca) [8, 10, 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.

As it emerges from its features, TRebeca is the right modeling language for 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.

4.2 The simplified Tangramob model in TRebeca

Tangramob is a very 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 AB model as it is into a TRebeca model, since its executions would result into an unmanageable state explosion. Thus, the designed TRebeca model is simplified, but still keeping the core features. In detail, the TRebeca model is composed of the two following rebecs: *CommuterGenerator* and *Tangrhub*. In the simplified model, commuters do not have a rebec counterpart. They are modeled as packets to be delivered within a network. Each commuter is characterized by a data structure representing its daily mobility agenda (introduced in Section 3.1). This data structure holds the following information: the commuter identifier (id), the nearest tangrhub to his home (th_h), the time units to elapse before leaving home ($time_{h,w}$), the time units for walking from home to th_h ($time_{fm}$), the nearest tangrhub to its workplace (th_w), the time units spent working ($time_{work}$) and the time units for walking from th_w to his workplace ($time_{lm}$).

Modeling the so-described "commuter", required many assumptions. Indeed, a careful reader may notice that providing each commuter with two tangrhubs, implies that all of them are only expected to travel by inter-hub mobility services, thus adopting the 3-path commuting pattern (Figure 3). Therefore, with this simplification, in the TRebeca model commuters have two commuting patterns:

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

Another strict assumption regards the fixed time units associated to each sub-trip: traveling times are always the same and traffic congestion is just emulated by adding random delays during the model-generation phase (Section 4.3).

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 in the TRebeca model, 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 made by *Tangrhubs*. Every time a commuter is scheduled, a *Tangrhub* releases a vehicle of the selected service with the best trade-off between its current fleet and its priority value. In particular, we associate each service with a priority value, which is meant to represent people's preferences: the higher the value, the more the service will be preferred.

Concerning the other rebec, the *CommuterGenerator* is in charge of monitoring the progress of commuters and their scheduling. In particular, it checks whether commuters are leaving home or they have just performed the last sub-trip. Moreover, this rebec is notified every time a commuter experienced a service disruption, i.e. it did not find any available service for an inter-hub sub-trip.

In order to describe how actors interact in the the simplified TRebeca model, we provide the reader with its event graph (Figure 5), which gives an intuitive and highly abstracted view of events and their causality relations. The model showed in this graph is composed of labeled nodes which represent events and their owner rebec. Edges show the causality relations among vertices, and can be either conditional (thick edges) or mandatory (thin edges).

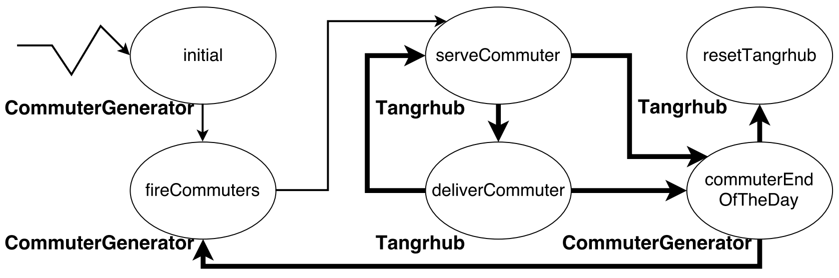

Fig. 5. Event Graph of the TRebeca Model

As shown in Figure 5, the *CommuterGenerator* 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 commuters and sends a *serveCommuter* message to every th_h after a specific time unit, which is computed as follows: $time_{h,w} + time_{fm} + randomDelay$. When a *Tangrhub* receives such a message, one of the following three actions is possible:

1. it sends a message to the next *Tangrhub* (i.e. th_w), in case there is an available mobility service and the commuter is reaching his workplace,
2. it sends a message to the *CommuterGenerator*, which informs it that the commuter is coming back home,
3. it sends a message to the *CommuterGenerator*, which informs it that the commuter did not find an available mobility service (i.e. service disruption).

The first message triggers a deliverCommuter event, which represents the travel of a commuter by means of a mobility service. Then, 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, messages 2 and 3 will trigger a commuterEndOfTheDay event. Every time this event occurs, the *CommuterGenerator* updates the number of arrived commuters (i.e. the ones who finished their activities), and eventually registers a service disruption. 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 *resetSmarthub* message to each *Tangrhub* and restarts the run of the scenario by sending a fireCommuters message to itself.

The pseudo-code of the TRebeca model, together with a complete runnable example model, can be found at [1].

4.3 ToolTRain: infer, generate, run, infer and collect

Since we aim at using the TRebeca model as a lightweight simulation tool, we should be able 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 learning process of Tangramob's commuters and its queue-based traffic simulation.

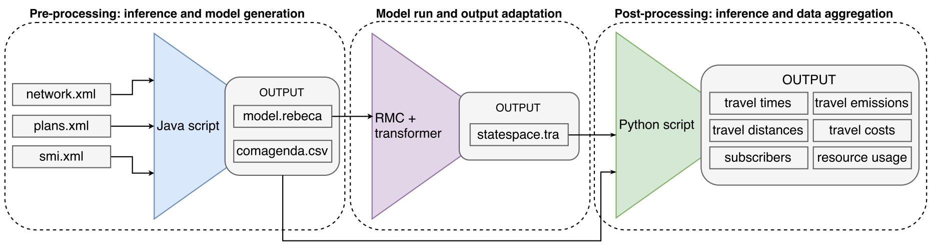

Fig. 6. The architecture of ToolTRain

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. Figure 6 shows the 3 building blocks of ToolTRain:

Model inference and generation Starting from the input files of Tangramob (Section 3), 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 to the tangrhubs, are assumed to travel by car or walk.
- For each potential user, from now on *potential subscriber*, the tangrhubs closest to his home and to his workplace are chosen, and the corresponding 3-path commuting pattern (Figure 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 is computed as euclidean distance from point to point. This is done for all the daily trips of each commuter, since travel time is derived from travel distance according to a reference walk speed.
- 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.

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 Rebeca model in RMC results in the generation of the whole state space of the model. Next, in ToolTRain, the so-generated statespace is converted into a tinier representation in which only a list of pre-defined state variables are reported for each state for further analysis. Therefore, it is worth remarking that we actually use RMC for performance evaluation rather than correctness.

Data inference and aggregation Since we are interested in the results of a smart mobility initiative, the converted statespace 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.

4.4 Comparing the two models

All the previously outlined abstraction rules are meant to simplify the original AB model, 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 Reinforcement Learning iterative process, used for modeling people's acceptance, is replaced by decision-rules. These rules are both encoded in the TRebeca model and achieved by the commuter filtering process of ToolTRain. 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 6.3, 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.

Concerning the modeling techniques, Tangramob lies on an agent-based model which is conceptually similar to the actor-based one of the TRebeca counterpart. 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.

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. Therefore, considering that the output data is also presented in the same way (graphs and tables), using the TRebeca model is as intuitive as using Tangramob.

5 Experimental design and setup

So far, we outlined the ABM of Tangramob and the simplification process for the derivation of the corresponding TRebeca model. However, using ToolTRain as lightweight preprocessing for Tangramob requires us to prove this hypothesis:

H 1 *Given a network, a population and a SMI, ToolTRain can approximate Tangramob, i.e. there is a positive correlation between their outputs.*

Because Tangramob returns several output files, testing H1 is equivalent to testing different sub-hypothesis, one for each output variable of the simulator. Therefore, with the exception of urban traffic, which is not represented in the TRebeca model, we need to show that ToolTRain can return similarly to Tangramob:

- travel times
- travel distances
- CO_2 emissions
- mobility costs
- number of subscribers
- mobility resource 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 defined the following partitions: *light-SMIs* (VPC $\leq$ 0.05); *medium-SMIs* (0.05 $\leq$ VPC $\leq$ 0.10); *massive-SMIs* (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, travel 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 this experiment, we chose a subarea of Ascoli Piceno (Italy), a mid-sized town of 50K inhabitants, and we we scaled down the scenario in order to deal with 2068 commuters. The 9 SMIs to be investigated, outlined in Table 5, share the same number and location of *tangrhubs*, each of which is provided with the same type of mobility services. Charges are also fixed for each service type (Table 5). A more detailed description of the experimental setup is provided in [1].

Table 1. Cost and priority values per mobility service

	Cost per hour	Cost per km	Fixed Cost	Priority (only for ToolTRain)
Bikesharing	0.5 €	0 €	0.01 €	30
Carsharing	13 €	0.1 €	0.01 €	40
Scootersharing	2.5 €	0.1 €	0.01 €	35

Table 2. The investigated Smart Mobility Initiatives (SMIs)

		light-SMIs			medium-SMIs			massive-SMIs		
Tangrhub	Service Type	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 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

6 Experimental results

In this section, we first show the results of the 9 SMIs runs with ToolTRain, then we select 3 of them. 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.

6.1 9 SMIs experimental results

The output variables that we are going to discuss, presented in Section 5, can be gathered into three categories: (i) number of subscriptions, (ii) commuters' travel performance measures and (iii) mobility resources usage.

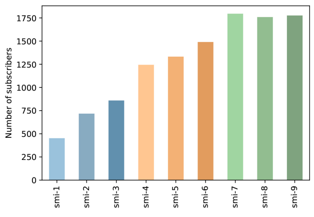

Fig. 7. N. of subscribers

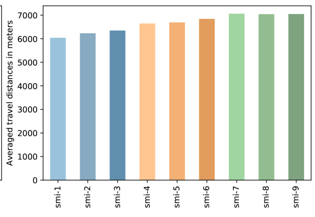

Fig. 8. Travel distances

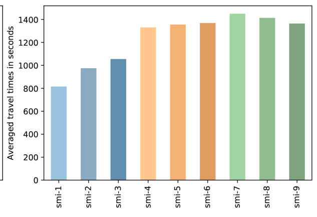

Fig. 9. Travel times

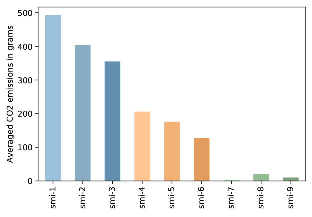

Fig. 10. CO2 emissions

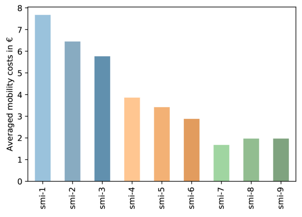

Fig. 11. Mobility costs

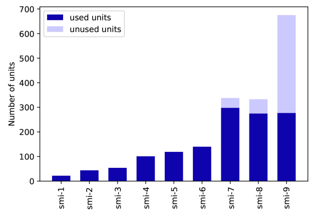

Fig. 12. Mobility fleet usage

A subscriber is a person who, at the end of the simulation, is expected to change his traveling habits in favour of the new mobility services. Thus, the number of subscribers is a measure of people acceptance of an SMI. Figure 7 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 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.

Commuters' performance measures are related to the number of subscribers. Concerning travel times and travel distances (Figures 8 and 9), their averages grow as the number of subscribers of the SMI increases. These trends are due to the fact that subscribers will extend their trips since they pass through two *tangrhubs* instead of making a direct trip. Moreover, subscribers perform their first-mile and last-mile trips by walk, which is considerably time-consuming.

Even mobility costs and CO2 emissions (Figures 10 and 11) 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 travelling is just less than €8 per commuter; in smi-7 it is 4 times less.

Mobility resources usage Figure 12 shows how vehicles are daily used. Light and medium SMIs are well configured, since there are no unused vehicles. Conversely, as the number of subscribers is close to 100%, the distribution of resources becomes tougher. Indeed, all the massive-SMIs have unused vehicles.

6.2 ToolTRain vs Tangramob: comparing the 3-SMIs

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 SMIs groups we selected the SMI with the lowest deployment of resources, i.e. smi-1 and smi-4. On the other hand, 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 Figures 13, 14, 16 and 18, number of subscribers, travel distances, CO_2 emissions and resources usage are almost the same between Tangramob and ToolTRain. For the mobility costs parameter, Figure 17 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 times (Figure 15) 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, with some precautions concerning the travel times output.

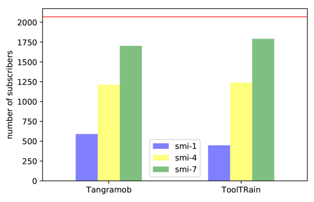

Fig. 13. Subscriptions

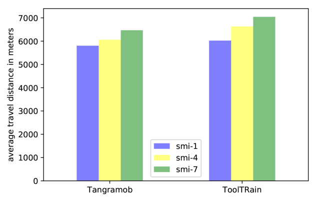

Fig. 14. Travel distances

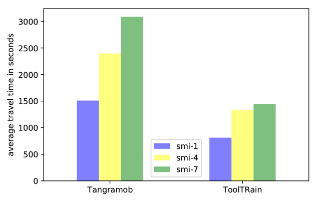

Fig. 15. Travel times

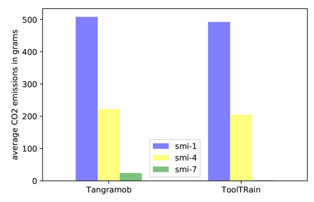

Fig. 16. CO2 emissions

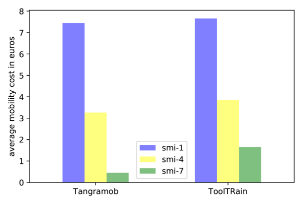

Fig. 17. Mobility costs

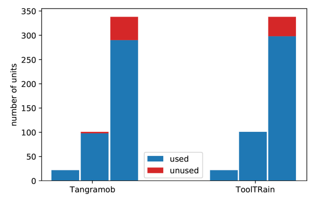

Fig. 18. Mobility fleet usage

6.3 Computational performance statistics

In order to compare Tangramob and ToolTRain in terms of computational time, Table 6.3 reports the CPU time of each selected SMI for both a Tangramob simulation and a ToolTRain run. More in detail, the experiments are performed on a Manjaro Linux desktop with an i7-4790S CPU @ 3.20GHz and 16GB RAM. Each Tangramob simulation is configured for 110 iterations.

Table 3. Computational times

	SMI-1	SMI-4	SMI-7	Iterations
Tangramob	693413 ms	841782 ms	947465 ms	110
ToolTRain	45641 ms	57882 ms	69242 ms	-

7 Conclusions and Future Work

Assessing the effects of smart mobility initiatives is a complex and risk-bearing task in urban planning. A Decision Support System (DSS) like Tangramob can support urban planners and transport companies in such a duty, even though the computational requirements might be considerable in case of large scenarios.

In this paper, we show how the Agent-Based Model (ABM) of Tangramob can be translated into a simplified one in TimedRebeca (TRebeca), which is an actor-based formal language. To make this model as useful as Tangramob, we also introduced ToolTRain, a tool-chain designed for generating an instance of the corresponding TRebeca model from the same input files of Tangramob; running the resulting model; and inferring the output from its run. This tool thus allows users to get an idea of a smart mobility initiative in a shorter time.

The comparative experiment designed to validate this approach, shows a positive correlation between the output variables of both Tangramob and ToolTRain.

Also, the computational time comparison reported in Table 6.3 confirms the validity of the proposed approach, since a ToolTRain run requires less than 10% of time needed for its corresponding Tangramob simulation. Considering such promising results, we would like to emphasize that the conceptual organization and architecture of ToolTRain can be reused in other AB contexts. This would even allow to exploit the power and the expressiveness of actor-based formal languages like Rebeca in order to reproduce the behavior of a certain phenomenon with an acceptable fidelity and few implementation efforts.

As future work, we are planning to improve the TRebeca model in order to introduce new mobility services as well as finding a better way to emulate traffic, due to its implication on travel times. 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).

References

1. Timedrebeca example page of tangramob, `https://goo.gl/HBacfs`
2. Benevolo, C., Dameri, R.P., DAuria, B.: Smart mobility in smart city. In: Empowering Organizations, pp. 13–28. Springer (2016)
3. Bosch, P.M., Ciari, F.: Macrosim - a macroscopic mobsim for MATSim. Procedia Computer Science 109 (2017), `https://doi.org/10.1016/j.procs.2017.05.406`
4. Castagnari, C., De Angelis, F., de Berardinis, J., Forcina, G., Polini, A.: Tangramob: an agent-based simulation framework for validating smart mobility solutions, `https://www.tangramob.com/docs/SmartHub_Thesis.pdf`
5. Charypar, D., Axhausen, K., Nagel, K.: Event-driven queue-based traffic flow microsimulation. Transportation Research Record: Journal of the Transportation Research Board (2003), 35–40 (2007)
6. Fehrenbacher, K.: Another failed attempt to make ride sharing work in the U.S., ridejoy to shut down, `https://goo.gl/3ITYce`
7. Horni, A., Nagel, K., Axhausen, K.W.: The Multi-Agent Transport Simulation MATSim. Ubiquity-Press, London (2016)
8. Khamespanah, E., Sirjani, M., Kaviani, Z.S., Khosravi, R., Izadi, M.J.: Timed rebeca schedulability and deadlock freedom analysis using bounded floating time transition system. Science of Computer Programming 98, 184–204 (2015)
9. Mamiit, A.: Why the ride-sharing company failed to conquer china and what it means for everyone else (2016), `https://goo.gl/cHuC9n`
10. Reynisson, A.H., Sirjani, M., Aceto, L., Cimini, M., Jafari, A., Ingolfsdottir, A., Sigurdarson, S.H.: Modelling and simulation of asynchronous real-time systems using timed rebeca. Science of Computer Programming 89, 41–68 (2014)
11. Sirjani, M.: Rebeca: Theory, applications, and tools. In: Formal Methods for Components and Objects, 5th International Symposium. pp. 102–126 (2006)
12. Sirjani, M., Khamespanah, E.: On time actors. In: Essays Dedicated to Frank De Boer on Theory and Practice of Formal Methods - Volume 9660. pp. 373–392. Springer-Verlag New York, Inc. (2016)
13. Strippgen, D., Nagel, K.: Multi-agent traffic simulation with cuda. In: High Performance Computing & Simulation, 2009. HPCS'09. IEEE (2009)
14. United Nations: The world's cities in 2016. New York, United (2016)