A Tool-chain for Statistical Spatio-Temporal Model Checking of Bike Sharing Systems*

Vincenzo Ciancia[1], Diego Latella[1], Mieke Massink[1], Rytis Paškauskas[1], and Andrea Vandin[2]

[1] Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy
[2] IMT School for Advanced Studies Lucca, Italy

Abstract. Prominent examples of collective systems are often encountered when analysing smart cities and smart transportation systems. We propose a novel modelling and analysis approach combining statistical model checking, spatio-temporal logics, and simulation. The proposed methodology is applied to modelling and statistical analysis of user behaviour in bike sharing systems. We present a tool-chain that integrates the statistical analysis toolkit MultiVeStA, the spatio-temporal model checker `topochecker`, and a bike sharing systems simulator based on Markov renewal processes. The obtained tool allows one to estimate, up to a user-specified precision, the likelihood of specific spatio-temporal formulas, such as the formation of clusters of full stations and their temporal evolution.

Keywords: Collective Adaptive Systems; Spatio-temporal Model Checking; Statistical Model Checking; MultiVeStA

1 Introduction

This paper studies the application of *statistical model checking* techniques to spatio-temporal verification, in the context of smart transportation systems. Statistical model checking (e.g., [29, 28]) permits the quantitative estimation of the likelihood of events in a simulated system. In this paper we use a boolean model checker to evaluate qualitative properties over single runs of a probabilistic simulator, and exploit statistical model checking to estimate, via repeated simulations, the probability that such properties hold for the model. Spatio-temporal verification is a recent development in Computer Science, inspired by spatial logics for topological spaces [7]. The modal logics and model-checking perspective is enhanced with spatial information, such as proximity or reachability properties. This methodology is able to capture subtle differences in behavioural analysis, such as "the points that are *now close* to a point that will be green *tomorrow*" vs. "the points that *tomorrow* will be *close* to a point that is green *now*". In [10], the logic STLCS (Spatio-Temporal Logic for Closure Spaces) was introduced. The topological approach of spatial logics is retained, but models are generalised to *closure spaces*, in order to include *finite graphs* in the landscape of the considered models.

* Research partially funded by the EU project QUANTICOL (nr. 600708).

STLCS model checking has been explored in the context of smart cities and smart transportation, with applications in smart public bus services [12] and smart bike sharing systems (BSS) [15]. The latter have recently become a popular public transport mode in many cities [17, 31] operating from a few (e.g. Pisa) up to several hundreds of docking stations (e.g. Hangzhou, Paris, or London[3]). The BSS concept is quite simple. A number of stations with docks partially filled with bicycles are placed throughout a city. Users of the service may hire any bicycle at any station at any time, and must return it at some station of their choice. The initial period of, typically, thirty minutes is free of charge, after which an hourly fee is charged. To maintain a high level of usage of the system it is important to keep the service attractive to its users. User satisfaction is difficult to evaluate quantitatively using only data obtained from real systems, as such data does not assess predictability of the service from a users point of view. A model-based approach was presented in [30] using Markov Renewal Processes (MRP) as the underlying probabilistic model. The model provides insight in the frequency and plausible causes for undesired delays in returning bikes and in the efficiency of bike sharing from a user's point of view. The model includes spatial aspects related to the presence of large groups of commuters in the morning and afternoon that go to a limited number of specific areas. Including commuters in the model turned out to be crucial to reproduce, up to a certain level of accuracy, actually observed cycling duration data for a large city such as London. In [15] we applied STLCS model checking on single simulation traces of the model. As expected, the introduction of commuter populations led to a larger number of stations being completely full in some places and empty in others. Spatio-temporal model checking also showed a number of more complex properties such as the emergence and persistence of regions in which all stations were full for some time (full clusters) and the development over time of such clusters. However, the results were only shown for individual simulation traces.

In this paper we generalise the approach of [15] to infer statistical properties of the system behaviour, rather than purely quantitative observations. We propose a methodology to quantify the likelihood of spatio-temporal properties in the system. We introduce a tool chain that integrates the simulator of [30], the spatio-temporal model checker topochecker ([13]) and MultiVeStA [35], a statistical model checker for discrete event simulators. Using MultiVeStA, *multiple* (spatio-temporal) properties can be analysed *simultaneously*, i.e. all estimators are updated at once for all points of the space during a single simulation, instead of performing one simulation for each point. In this paper this is used to obtain *separate* observations on all points of the space, using *the same* set of simulations. The obtained performance speed-up is directly responsible for the feasibility of statistical spatio-temporal model checking.

2 Bike Sharing Simulation Model

We briefly recall the main aspects of the bike sharing model that was introduced in [30] and that forms the basis for the stochastic simulator that we use here in combination

[3] Pisa: `http://www.pisamo.it`, Hangzhou: `http://www.publicbike.net`; Paris: `http://www.velib.paris.fr`, London: `https://tfl.gov.uk/modes/cycling/santander-cycles`

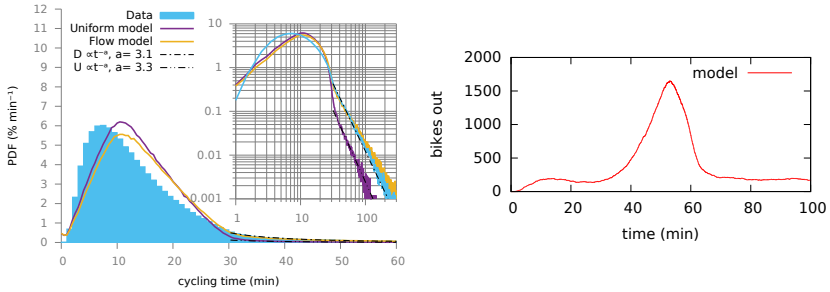

Fig. 1. Left: cycling duration histograms (Data) in London, using 831,754 trip records in October 2012, and results of simulation of the uniform model (dark lines) and the flow model (light lines). Maintenance trips are not considered. Right: total bike rentals over 100 minutes.

with the spatio-temporal model checker briefly described in the next section. The bike sharing model is intended to serve as an explanatory model for some of the salient aspects of the distribution of cycling times observed in real bike sharing systems. In particular, such distributions show a considerable number of surprisingly long cycling trips that cannot be attributed to maintenance events.

An illustration of such a distribution for the bike sharing systems in London is provided in Fig. 1 (Data). There, 7% of all cycling trips are longer than thirty minutes, some extending up to two hours, which is more than the time necessary to traverse the complete service area in London (about fifteen kilometres). This range coincides with the so-called 'algebraic tail' of the distribution, the range in which the probability density function (PDF) is well approximated by $\propto t^{-a}$ with some exponent $a > 0$ (Fig. 1, inset). Such "algebraic tails" were found in data from all considered cities. Simulation results of the bike sharing model of [30] suggest that this phenomenon is a consequence of a form of risk-taking behaviour of users of bike sharing systems. Most users use bike sharing to reach a planned destination at a planned time and use an estimate of the time it will take them from their origin to their destination to know when to leave. Users risk, of course, that no parking place is found at or close to the destination in which case they would have to extend the travel itinerary to find another station where to deposit their bike. Such risk-taking behaviour can be shown to actually reduce the mean trip duration when considering the *overall system* [30]. The bike sharing model takes this risk-taking user behaviour explicitly into account, as well as other human factors such as speed of walking and biking.

The model is composed of two populations: a population of stations and a population of agents, the latter representing relevant user behaviour. Both can schematically be represented as automata as shown in Fig. 2. Users can take or return bikes from/to a station via the actions 'take' and 'return', respectively. Each station has a particular capacity c and a number of bikes parked in it n, as well as a position. To keep the model simple, stations are situated on a regular grid as shown in the left panel of Fig. 3. Users are modelled as agents that pass repeatedly through four different states as shown in Fig. 2. Each agent is parameterised by two addresses on the grid in an area at walking distance from a station which we will denote by origin and destination, respectively.

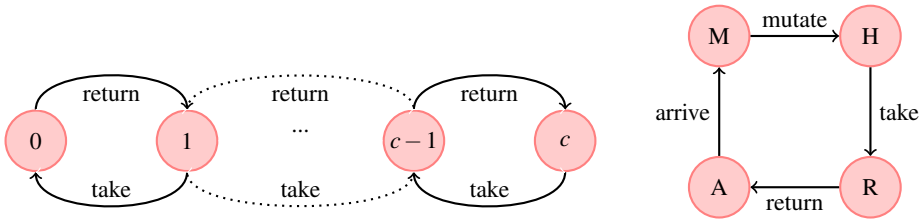

Fig. 2. Models of a bicycle station (left) and user agent (right).

Their behaviour is as follows. From the origin they walk to the nearest station where they take a bike (H), then they bike to the station close to the destination, return the bike (R) and walk to the destination. Upon arrival, the user process is re-instantiated (M).

The mathematical framework is that of Markov Renewal Processes (MRP), which are a generalisation of Continuous Time Markov Chains allowing for non-Markovian events and non-exponential distributions of inter-event times [9]. This approach was chosen in particular to reflect more accurately trip durations and agent's decisions. In particular, in MRPs the sojourn time has a distribution that depends both on the origin and the destination. In the model a user always finds a parking place, but this may not be in the preferred station if there are no places available. This then is reflected in a longer trip duration for the 'return' transition. The 'take' and 'return' transitions between users and stations are synchronised. The 'arrive' and 'mutate' transitions are not synchronised with stations but re-initialise the agent's states. For more details about the model, the Reader is invited to consult [30].

A model for station utility perception is used in which agents that want to take little risk tend to search for suitable stations in a larger area surrounding their target destination, whereas agents that take a higher risk search in a smaller area, risking not to find a parking place. Higher risk should lead to shorter trips in general when there are enough parking places available, but occasionally to much longer trips when this is not the case (see [30] for further details). The cycle time distribution obtained via simulation (Fig. 1) shows that such events affect only a small fraction of all trips if the distribution of agents' origins and destinations is *spatially homogeneously distributed*, as in Fig. 1, the 'uniform model' ($\Pr\{\text{cycling trip} > 30\text{min} \mid \text{uniform}\} = 0.01$) which increases sevenfold if there are larger destination concentrations as in the 'flow model' ($\Pr\{\text{cycling trip} > 30\text{min} \mid \text{flow}\} = 0.07$). The flow model reflects the presence of areas that attract more users than other areas at certain times of the day. This is a reasonable assumption about real cities. An obvious consequence is that also the areas of full stations will be, as a rule, larger. As shown in Fig. 1 the flow model approximates rather closely the actual distribution of cycling times in London.

Fig. 3 shows a spatial simulation set-up for a grid of stations of the size of that of the London area. The panels in the middle and on the right show an artificially introduced probability distribution for the request for bikes and parking places, respectively.

The set-up of the model follows the principle that the total service area, the number and capacities of stations, the number of bicycles, and the average number of hourly

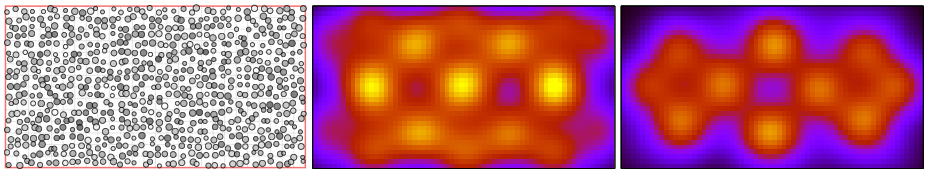

Fig. 3. Spatial set-up to simulate the London data-set. Left panel: The map of randomly generated stations and a snapshot of their filling degree (circle size $\propto c$, shade $\propto n$). Middle, right: distributions of the demand origin and destination locations, respectively.

trips should be close to those in London, but without pursuing a photographic accuracy of the underlying topography of the city. The result is a 7×13 km^2 area with a 19×38 array of stations with randomly perturbed locations, random capacities between 15 and 40 docks. Two groups of agents are injected into this model. The first group of 400 agents are sampled with uniformly distributed spatial and temporal demand profiles, simulating the homogeneous component of the overall demand. The second group of 2000 agents are sampled from topical spatial demand distributions that contribute to the visible peaks in Fig. 3. This group is requested to honour a kind of appointment that requires the agent to arrive at a destination by a certain epoch (time). These arrival times are sampled from $U(50,60)$. The actual arrival time does not coincide with the appointment time, almost certainly, because the travel process is stochastic (cf. Fig. 1).

MRPs can be simulated using the methodology of 'exact stochastic simulation' of chemical reaction networks, substituting agents for 'molecules' [21], adapting it to a non-Markovian modelling framework. Among the known simulation methods, the highest efficiency was achieved by adapting a version of the 'next reaction method' [20]. Such simulations generate a stochastic trajectory $Y(\omega) = \{T_i, X_i, i \in \{0,\dots,N\}\}$ where $\{T_i\}$ is a series of epochs of events, and the state space of $X_i = \{\otimes_{a\in\mathscr{A}}\alpha_a(i), \otimes_{s\in\mathscr{S}}\sigma_s(i)\} \in \mathscr{X} = \{H,R,A,M\}^{\mathscr{A}} \times \otimes_{s\in\mathscr{S}}\{0,\dots,c_s\}$ is a product space of the agent states $\{H,R,A,M\}$ for all agents $\mathscr{A}$, and the number of parked vehicles $\sigma_s \in \{0,\dots,c_s\}$, where c_s is the capacity of station s, for each bike-sharing station $s \in \mathscr{S}$. This trajectory is then transformed into a snapshot sequence. A snapshot sequence $\Sigma(\omega)$ is defined as a projection of $Y(\omega)$ to the station-only component, cut at regular time intervals $\Delta > 0$. Thus $\Sigma(\omega) = \{\otimes_{s\in\mathscr{S}}\hat{\sigma}_s(j), j = 0,\dots\}$, where $\hat{\sigma}_s(j) = \{\sigma_s(i) : T_i \le \Delta j < T_{i+1}\}$. Each sequence $(\hat{\sigma}_s(0), \hat{\sigma}_s(1), \dots)$ is interpreted as a sequence of independent random numbers, which are integral and bounded by 0 and c_s (see Fig. 4 for a graphical illustration). The interpretation of all stations' sequences is clearly a complicated task for analysis. In the following sections, we will show the application of *statistical* spatio-temporal model checking to identify problematic stations and areas. Note that by virtue of the Hoeffding's theorem [36], such sequences are Monte Carlo-compatible [25], justifying the deployment of SMC, as described in Sect. 4.1.

3 Spatio-temporal Model Checking

Spatio-temporal model checking is a variant of classical model checking where *spatial* logical reasoning is combined with classical temporal operators. In this work we use

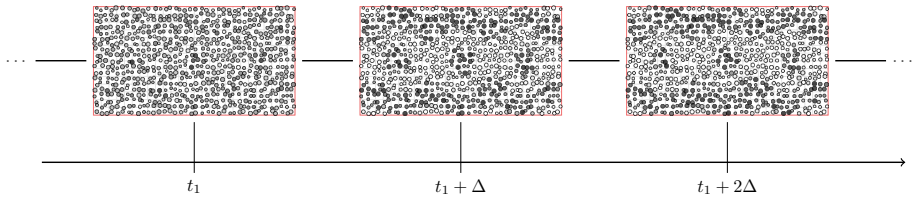

Fig. 4. Linear snapshot model based on single-simulation traces.

$$
\begin{array}{lll}
\Phi ::= \mathtt{TT} & & \text{[TRUE]} \\
\quad | \ [\mathtt{p}] & & \text{[ATOMIC PREDICATE]} \\
\quad | \ \mathtt{!}\,\Phi & & \text{[NOT]} \\
\quad | \ \Phi \,|\, \Phi & & \text{[OR]} \\
\quad | \ \Phi \,\&\, \Phi & & \text{[AND]} \\
\quad | \ \mathtt{N}\,\Phi & & \text{[NEAR]} \\
\quad | \ \Phi\,\mathtt{S}\,\Phi & & \text{[SURROUNDED]} \\
\quad | \ \mathtt{A}\,\varphi & & \text{[ALL FUTURES]} \\
\quad | \ \mathtt{E}\,\varphi & & \text{[SOME FUTURE]}
\end{array}
\qquad
\begin{array}{lll}
\varphi ::= \mathtt{X}\,\Phi & & \text{[NEXT]} \\
\quad | \ \mathtt{F}\,\Phi & & \text{[EVENTUALLY]} \\
\quad | \ \mathtt{G}\,\Phi & & \text{[GLOBALLY]} \\
\quad | \ \Phi\,\mathtt{U}\,\Phi & & \text{[UNTIL]}
\end{array}
$$

Fig. 5. STLCS syntax

the *spatio-temporal logic of closure spaces* (STLCS) of [13]. The temporal fragment of the logic consists in *Computation Tree Logic* [16], whereas the spatial fragment is that of [10], comprising a spatial *near* modality, expressing topological proximity, and a binary spatial *surrounded* operator.

STLCS is interpreted over so-called *snapshot models* [27]. A snapshot model is a triple consisting of a *Kripke frame* $(S,\mathscr{R})$ with states in S, accounting for the temporal evolution of a system, a *closure space*[4] $(X,\mathscr{C})$ that represents space, and a valuation function $\mathscr{V} : X \times S \to 2^P$ assigning to each pair of a point in X, and a state in S, the boolean valuation 2^P of a finite set of atomic propositions P. Although using closure spaces for spatial logics is relevant in order to link it to the topological interpretation of [2], for the purpose of this paper, looking at so-called *quasi-discrete closure spaces* [19] is sufficient. In other words, the reader may consider X to be the nodes of a finite directed graph G, and, given $A \subseteq X$, let $\mathscr{C}(A)$ be A itself, plus the nodes b in X such that there is a node a in A, with $a \to b$ an edge of G.

The formal syntax of formulas is described by the grammar in Fig. 5, where $\mathtt{p}$ ranges over a finite or countable set of *atomic propositions*. The truth value of formula ϕ is defined at a point in space x and state s, written $(x,s) \models \phi$. The full semantics of the logic is provided in [13], whereas a tutorial-type introduction to spatial (and spatio-temporal) logics and their model checking can be found in [11]. We briefly comment on the spatial operators, that are less known. A pair (x,s) satisfies ϕ_1 *surrounded by* ϕ_2 (written $\phi_1 \mathscr{S} \phi_2$) whenever, in the graph associated to $(X,\mathscr{C})$, it is not possible to find a path p

[4] A *closure space* is a pair $(X,\mathscr{C})$ where X is a set, and the *closure operator* $\mathscr{C} : 2^X \to 2^X$ assigns to each subset of X its *closure*, obeying to the following laws, for all $A,B \subseteq X$: 1) $\mathscr{C}(\emptyset) = \emptyset$; 2) $A \subseteq \mathscr{C}(A)$; 3) $\mathscr{C}(A \cup B) = \mathscr{C}(A) \cup \mathscr{C}(B)$. We refer to [10] for an introduction.

from x to a point y, with $(x,s) \not\models \phi_1$, unless path p passes first by point z with $(z,s) \models \phi_2$. For interpreting the temporal operators, one uses the traditional interpretation of CTL, so that, for example, $(x,s) \models \mathtt{EX}\phi$ whenever there is a path p in the Kripke frame $(S,\mathscr{R})$ with $(x,p(1)) \models \phi$. This simple, orthogonal definition of space and time is typical of snapshot models (an example of a linear[5] snapshot model for BSS is shown in Fig. 4). However, arbitrary nesting of spatial and temporal formulas allows one to express quite complex assertions (e.g. a point x at state s being surrounded by points that *will eventually satisfy* a certain property). In STLCS, the temporal and spatial fragment can be freely nested; the computational complexity of the *global* model checking algorithm of [13] is linear in the product of the size of S, X, and the number of sub-formulas of the checked formula.

As an example, consider the STLCS formula $\mathtt{EF}\,[\mathtt{full}]\mathtt{S}\,(\mathtt{A}\,\mathtt{X}\,[!\mathtt{full}])$ where atomic proposition $[\mathtt{full}]$ is satisfied by full stations. The formula is satisfied by a point (station) x in state s if the point x possibly ($\mathtt{E}$) satisfies $[\mathtt{full}]$ in some future ($\mathtt{F}$) state s', and in that state, it is not possible to leave the area of points satisfying $[\mathtt{full}]$ unless passing by a point that will necessarily ($\mathtt{A}$) satisfy $[!\mathtt{full}]$ (not full station) in the next ($\mathtt{X}$) time step. In other words, a situation in which there is a contiguous area of full stations surrounded by stations that are not full.

4 A Tool-chain for Statistical Spatio-Temporal Model Checking

In this section we detail the implementation of our tool-chain. One interesting aspect of MultiVeStA [35] is its modularity. By implementing specific plugins, the tool acts as an *orchestrator* for running a simulator and observing its results. In statistical spatio-temporal model checking, MultiVeStA invokes several runs of the simulator of [30], which is deployed as a separate executable. The simulator outputs a spatio-temporal model in the format of `topochecker`. A special functionality has been added to the model-checker, permitting MultiVeStA to invoke it several times over the same model, while keeping `topochecker` running, to avoid reloading of the model and recomputation of the intermediate results. This permits one to define a large number of statistical observations in MultiVeStA, corresponding to the truth value of spatio-temporal formulas at each point of space, in an efficient way. We remark that, since the simulator is a separate executable, it is straightfoward to reuse the same tool-chain for simulations coming from other domains, as long as the simulation process formats its results using the input language of `topochecker`.

4.1 Multivesta

We briefly present the tool for distributed statistical model checking MultiVeStA[6]. The tool can be easily integrated with any existing discrete event simulator, or formalism that provides probabilistic simulation. It has been successfully used in the analysis of many scenarios, including public transportation systems [22], volunteer clouds [34],

[5] Note that snapshot models may also be branching models.

[6] Available at `http://sysma.imtlucca.it/tools/multivesta/`

crowd-steering [33], swarm robotics [6], opportunistic network protocols [3], contract-oriented middlewares [4], and software product lines [5]. Here MultiVeStA is used to estimate quantitative spatio-temporal properties of bike sharing systems. The integration is performed by instantiating a Java Interface exposing simple methods used by MultiVeStA to interact with the considered simulator (such as *reinitialize the simulator to perform a new simulation*, *perform one step of simulation*, or *perform a whole simulation*, depending on the specific use case).

Model specification is delegated to the integrated simulator, while MultiVeStA offers a simple and flexible property specification language, MultiQuaTEx (which extends QuaTEx [1]). MultiQuaTEx consists of a few ingredients: (i) real-valued observations on the system states, such as the number of bikes in a bike station at a certain point in time, its current full/empty status, or the truth value of a spatio-temporal property (0 for false and 1 for true); (ii) arithmetic expressions and comparison operators; (iii) a one-step next operator (which triggers the execution of one step of a simulation); (iv) if-then-else statements; (v) recursion. MultiQuaTEx is used to define random variables, associating a real value to each simulation. Then, MultiVeStA estimates the expected value of such random variable. Note that in case we get 0 or 1 upon the occurrence of a certain event (e.g., when considering the truth value of a spatio-temporal property), we get a Bernoulli random variable, and MultiVeStA hence estimates the probability of such an event. An in depth discussion of MultiVeStA's architecture and of MultiQuaTEx is provided in [35, 33]. Estimations are computed according to a user specified confidence interval (CI) (α, δ). In particular, the mean value of n samples is computed, with n minimal but large enough to guarantee that the size of the $(1-\alpha)\times 100\%$ CI is bounded by δ. In other words, if a MultiQuaTEx expression is estimated as $\bar{x} \in \mathbb{R}$, then its actual expected value belongs to the interval $(\bar{x}-\frac{\delta}{2}, \bar{x}+\frac{\delta}{2})$, with probability $(1-\alpha)$. In all the experiments discussed in the next section we focus on the probabilities of bike station properties, fixing $\alpha = 0.1$ and $\delta = 0.05$. A single MultiQuaTEx query may address many different properties simultaneously, such as the number of bikes in each bike station at a certain point in time, or even at the varying of time. All such properties are analysed reusing the same simulation traces, leading to huge analysis speed ups. Note that the estimation of each property might require a different number of simulations. MultiVeStA performs only n simulations, with n the maximum number of simulations required by each individual property (see [33] for full details).

4.2 Statistical Spatio-temporal Model Checking using `topochecker`

Statistical spatio-temporal model checking assumes an underlying simulation model of a spatio-temporal system (that can be described as a snapshot model, see Section 3). The methodology is aimed at estimating the likelihood, at each point of space, that a given formula (with boolean valuation) is true, with a user-specified *global confidence interval* – that is, the same interval is used for all points. By this, a *heat-map* is produced that associates to each point of space a probability value. In principle, to achieve this, standard techniques from statistical model checking might be used. For each pair (x,s) to be observed and each formula ϕ, a series of simulations of a system should be executed, computing the (boolean) satisfaction value of $(x,s) \models \phi$ (see Section 3 for the meaning of this notation) in each specific simulation. A probability estimate can then be

computed by keeping the cumulative account of the number of times the formula is satisfied, until the specified confidence interval is reached. However, such naive approach is not feasible on all but the simplest models, due to the already *cpu-hungry* simulation and model checking processes being iterated not only for the number of simulations that are necessary to achieve the required confidence, but also for each point of space. The proposed tool-chain turns the theoretical approach of statistical spatio-temporal model checking into a feasible analysis methodology. Input to the tool-chain are: (i) The parameters of the simulation, describing relevant features of a bike sharing system, such as the position and capacity of stations, and the number of users etc.; (ii) A set of qualitative or quantitative spatio-temporal formulas, characterising features of interest of the behaviour of the system, such as the formation of clusters of full stations; (iii) A set of quantitative queries whose evaluation is based on the outcome of the spatio-temporal model checking process. The approach used in this paper exploits the "multi" in MultiVeStA, by using an observation *for each point of the space*, resulting in a large number of random variables – one for each point of space and formula – being analysed at once reusing the same simulations. Since each observation in MultiVeStA corresponds to a different query in its internal language, we also adapted the spatio-temporal model checker `topochecker` to be run as a server for each bike sharing simulation. The server receives queries from MultiVeStA in the form of pairs (x, ϕ) where x is a point of space, and ϕ is a spatio-temporal formula. The value of $(x,0) \models \phi$, where 0 is the first point of the trace obtained from the simulator, is computed and returned to MultiVeStA. The sophisticated *global* model checking algorithm of topochecker uses a cache that stores the intermediate computations of the model checker for each formula. As a result, the time required to compute the satisfaction value (x, ϕ) *for all points of space* x is just a fraction more of the time required to compute the same value on one point. Such machinery speeds up statistical model checking of a factor which is proportional to the number of points of the space. In our case, such speed-up is the key to actually be able to run our experiments. The output from MultiVeStA consists in a list of estimates of all the queries used (as we mentioned above, one for each formula and point of space). To actually produce a heat-map, the result is transformed by a simple *rendering* script, that colours the graph representing space, using the results from MultiVeStA. The resulting collaboration pattern is depicted in Figure 6. We remark that the total execution time for all the properties we consider is in the order of around five hours on a standard laptop; this hints at the importance of observing multiple points at the same time (exploiting the specific capabilities of MultiVeStA); the size of the considered space is 722 points, and running the statistical model checking sequentially for each point would multiply our execution times accordingly, changing the approach from "feasible" to "unfeasible".

5 Properties and Results

In this section we revisit some of the spatio-temporal properties of bike sharing systems that were presented in [15]. Therein, some of the authors used spatio-temporal model checking on single traces of the BSS simulator. A regular grid representing bike sharing stations was coloured with two colours, representing the boolean satisfaction value of properties. As discussed, using statistical spatio-temporal model checking we

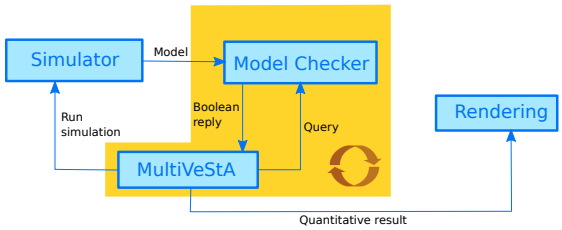

Fig. 6. Collaboration in the tool-chain used for statistical spatio-temporal model checking.

can collect information about single simulations to assess the *probability* with which each station satisfies the property of interest in the entire system behaviour. We will visualise such probability by means of a colouring of the stations in a grid according to a sequential colour palette of 10 uniform steps ranging from light grey (denoting low probability) to dark red (denoting high probability). We use this visualisation to facilitate the quick analysis of the results. Detailed values of the probabilities, variance and size of the confidence interval δ are indeed produced by MultiVeStA. Let us first recall some basic spatio-temporal properties of bike sharing systems. Note that, throughout this section, all simulations start from an initial state in which all stations are half full.

Full stations and clusters. We characterise stations that are *full*, that is, with no vacant parking places, and *clusters* of full stations, that is, stations that are full, and are only connected to adjacent stations that are full in turn. These two (purely spatial) properties are formalised in STLCS below:

$$\begin{aligned} \texttt{full} &= [\texttt{vacant}{==}0] \\ \texttt{cluster} &= \texttt{I}\,(\texttt{full}) \end{aligned}$$

Connectivity between stations is expressed using the derived *interior* operator $\texttt{I}\,\Phi = !\,(\texttt{N}\,(!\,\Phi))$. Informally speaking, in an undirected graph, points satisfying $\texttt{I}\,\Phi$ are only connected to points satisfying Φ. The smallest possible cluster in the regular grid that was used for the simulation is therefore composed of a full station such that its direct neighbours in the north, south, east and west directions (also called its *von Neumann neighbourhood*) are also full. Note that the definition of `cluster` only identifies (on purpose) these "inner" full stations and not their direct full neighbours. The abbreviation `full` uses a boolean predicate (equality), applied to the quantitative value of the atomic property $[\texttt{vacant}]$.

Let us now consider probability that a station will eventually be full, formalised as:

$$\texttt{eventuallyFull} = (\texttt{EF}\,\texttt{full})$$

MultiVeStA evaluates the property for all stations simultaneously, using the same set of generated simulations. As discussed in Section 4.1, we used $\alpha = 0.1$ and $\delta = 0.05$. The simulations cover a period of 100 minutes in steps of 2 minutes each. This includes the morning period in which there is a peak of requests for bikes and parking places due to a large group of commuters leaving from home. The results are shown in Figure 7

that depicts the grid of stations, and for each station a colour indicating the approximate probability with which the property holds according to the colour scale shown on the right of the grid. The results clearly show that the stations that have a high probability to get full during this period of the simulation correspond to the areas in the model that have been assigned a high attractiveness for commuters as shown in Figure 3 in a pattern that is easy to recognise.

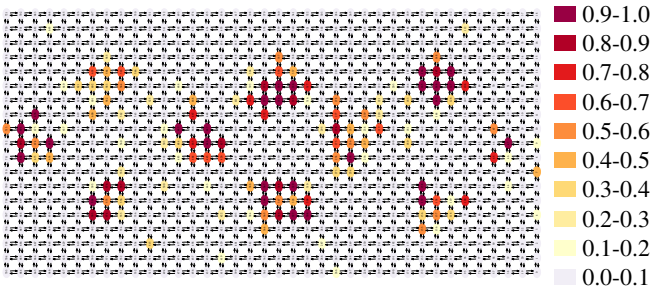

Fig. 7. Probability of stations to be eventually full within the maximal length of the simulations considered (100 minutes), starting from an initial situation in which all stations are 50% full.

We can identify stations belonging to clusters that *persist* for some amount of time, that is, they last for a specific number of time steps. The following formulas specify the persistence of such a situation for two and three time steps:

$$\texttt{cluster2steps} = \texttt{cluster} \,\&\, (\texttt{AX}\, \texttt{cluster})$$
$$\texttt{cluster3steps} = \texttt{cluster} \,\&\, (\texttt{AX}\, \texttt{cluster2steps})$$

By combining these formulas with the *eventually* operator, as before, we can assess the probability of stations to eventually become a cluster and remain so for 3 consecutive steps. The results are shown in Figure 8.

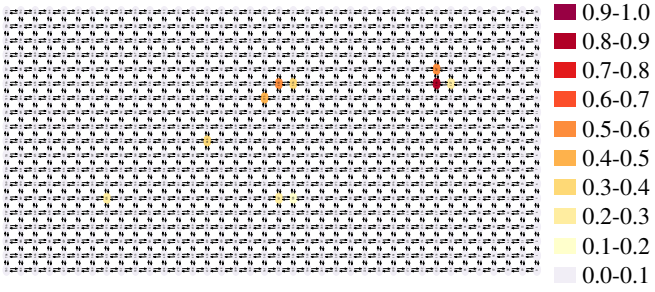

Fig. 8. Probability of stations that eventually become a cluster and remain a cluster for 3 steps.

Problematic user experience. The next property we consider is related to problematic user experience, namely not to find a parking place in a suitable station. When a user wants to leave a bike at a specific station, and such station is full, she may try to find a nearby station with available parking slots, or she may wait for some time in the same station hoping that someone is needing a bike. This behaviour may be typically sufficient to solve the problem, at the expense of a longer trip duration. One may want to check how effective this procedure is. In the following formula, we check whether it is possible that, in three time steps of 2 minutes each, the user is still unable to leave the bike in the same or a nearby station because they are full when she arrives. The formula `tripEnd` characterises this situation. It expresses a nested spatio-temporal situation where the user arrives at a full station, and in the next step, while possibly moving to another neighbouring station, finds it full again, being unlucky this way for three consecutive attempts. In terms of the STLCS logic, this is expressed as follows:

$$\texttt{tripEnd} = \texttt{full} \,\&\, (\texttt{N}\,(\texttt{A}\,\texttt{X}\,(\texttt{full} \,\&\, (\texttt{N}\,(\texttt{A}\,\texttt{X}\,(\texttt{full} \,\&\, \texttt{N}\,(\texttt{A}\,\texttt{X}\ \texttt{full})))))))$$

Combining this formula with the *eventually* operator provides an overview of the probability that such an unlucky series of events may happen to a user at a particular station. The results are shown in Figure 9. The resulting probabilities for the stations are very close to those for property `eventuallyFull`, but they are slightly lower.

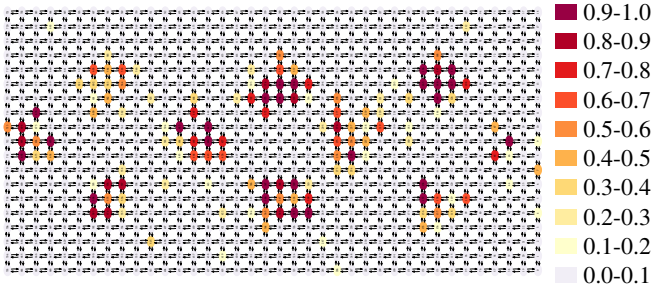

Fig. 9. Probability that a user willing to park her bike in a station finds it full and cannot find a parking place within three consecutive attempts in neighbouring stations.

As a hint on the feasibility of the approach, we remark on the execution times. On a high-end laptop, the computations of the example of Figure 9 take around 5 hours, analysing 77 batches of 20 simulations each.

6 Related Work

The field of spatial logics is as old as modal logics itself, with early logicians such as Tarski already laying the foundations of a topological interpretation of modal operators and of the completeness of the logic *S*4 for the class of topological spaces [7]. Research

efforts in spatial and spatio-temporal model checking are far more recent, and often tailored to specific applications. In [23] a linear spatial superposition logic is defined for the specification of emergent behaviour. The logic is applied to pattern recognition in the context of medical image analysis. The Mobile Stochastic Logic (MoSL) [18] has been proposed to predicate on mobile processes in models specified in StoKLAIM, a stochastic extension of KLAIM based on the tuple-space model of computation. Other variants of spatial logics concern the symbolic representation of the contents of images, and, combined with temporal logics, for sequences of images [8]. In [24], the approach of [23] has been further extended, defining the spatio-temporal logic SpaTeL, and a statistical model checking algorithm. The algorithm estimates the probability of events that relate different regions of space at different times. Regions are identified by spatial partitioning using *quad trees*. In SpaTeL, spatial formulas can only be nested below temporal formulas. In contrast, STLCS can arbitrarily nest spatial and temporal formulas, at the expenses of using simpler models that do not explicitly describe regions, but only deal with points. The spatio-temporal logic STLCS used in the current paper addresses properties of discrete, graph-based models that, in our case study, reflect the geographical position of docking stations in a city. The spatial fragment of STLCS, and related model-checking algorithms, were introduced in [10] and have also inspired the work on Spatial Signal Temporal Logics in [32], where a linear time logic is introduced to reason about properties of signals, considering both their truth values and their robustness in the presence of local perturbations of the signals. The spatial fragment has also been used to analyse aspects of public bus transportation systems [12].

7 Conclusions

We have discussed the general idea of statistical spatio-temporal model checking as a form of statistical model checking applied to points of the space. A tool-chain has been developed to study the feasibility of the approach. Future work tailored to bike sharing systems analysis will extend the simulator by modelling *incentives* to analyse their usage in improving the overall performance of such systems. The effectiveness of incentives can be then captured by logic formulas and assessed statistically before their deployment.

More generally, statistical spatio-temporal model checking can be used in any kind of simulation scenario for spatio-temporal systems. As MultiVeStA can be integrated with discrete event simulators that allow for probabilistic simulation and the spatio-temporal model checker just needs spatial snapshot models in a very general format, the approach can be applied to other systems. We plan to use the approach also in modelling mitigation strategies for problems of *smart bus* networks, continuing the work of [12].

These developments are part of a more general effort in statistical spatio-temporal model checking aimed at investigating global properties of collective-adaptive systems (CAS), taking spatial aspects into account. In this light, it will be relevant to propose variants of statistical spatio-temporal model checking that operate over the semantic domains of process calculi with spatial aspects, such as [14]. A further interesting issue would be the extension of the statistical spatio-temporal model checking approach to handle rare events [26].

Acknowledgements This work is supported by the EU project *QUANTICOL* (600708). We thank Mirco Tribastone and Daniël Reijsbergen for the usage data on the London bike sharing system.

References

1. Agha, G., Meseguer, J., Sen, K.: PMaude: Rewrite- based Specification Language for Probabilistic Object Systems. ENTCS 153, 213–239 (2005)
2. Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.): Handbook of Spatial Logics. Springer (2007)
3. Arora, S., Rathor, A., Rao, M.V.P.: Statistical model checking of opportunistic network protocols. In: Proceedings of the Asian Internet Engineering Conference. pp. 62–68. AINTEC '15, ACM (2015)
4. Bartoletti, M., Cimoli, T., Murgia, M., Podda, A.S., Pompianu, L.: A contract-oriented middleware. In: Braga, C., Ölveczky, P.C. (eds.) Formal Aspects of Component Software. LNCS, vol. 9539, pp. 86–104. Springer (2015)
5. ter Beek, M.H., Legay, A., Lluch-Lafuente, A., Vandin, A.: Statistical analysis of probabilistic models of software product lines with quantitative constraints. In: 19th International Conference on Software Product Line. pp. 11–15. ACM (2015)
6. Belzner, L., De Nicola, R., Vandin, A., Wirsing, M.: Reasoning (on) Service Component Ensembles in Re- writing Logic. In: Specification, Algebra, and Software. LNCS, vol. 8373, pp. 188–211. Springer (2014)
7. van Benthem, J., Bezhanishvili, G.: Modal logics of space. In: Handbook of Spatial Logics, pp. 217–298. Springer (2007)
8. Bimbo, A.D., Vicario, E., Zingoni, D.: Symbolic description and visual querying of image sequences using spatio-temporal logic. IEEE Trans. Knowl. Data Eng. 7(4), 609–622 (1995)
9. Çinlar, E.: Introduction to stochastic processes. Prentice-Hall (1975)
10. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and Verifying Properties of Space. In: Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference. LNCS, vol. 8705, pp. 222–235. Springer (2014)
11. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Spatial logic and spatial model checking for closure spaces. In: Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems. 16th International School on Formal Methods for the Design of Computer, Communication and Software Systems, LNCS, vol. 9700, pp. 156–201. Springer (2016)
12. Ciancia, V., Gilmore, S., Latella, D., Loreti, M., Massink, M.: Data verification for collective adaptive systems: spatial model-checking of vehicle location data. In: 2nd FoCAS Workshop. International Conference on Self-Adaptive and Self-Organizing Systems, IEEE (2014)
13. Ciancia, V., Grilletti, G., Latella, D., Loreti, M., Massink, M.: An experimental spatio-temporal model checker. In: SEFM 2015 Collocated Workshops: VERY*SCART, Revised Selected Papers. LNCS, vol. 9509, pp. 297–311. Springer (2015)
14. Ciancia, V., Latella, D., Massink, M.: On-the-fly mean-field model-checking for attribute-based coordination. In: COORDINATION 2016. LNCS, vol. 9686, pp. 67–83. Springer (2016)
15. Ciancia, V., Latella, D., Massink, M., Paškauskas, R.: Exploring spatio-temporal properties of bike-sharing systems. In: SASO Workshops. pp. 74–79. IEEE Computer Society (2015)
16. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: 25 Years of Model Checking - History, Achievements, Perspectives. LNCS, vol. 5000, pp. 196–215. Springer (2008)

17. De Maio, P.: Bike-sharing: Its history, impacts, models of provision, and future. Journal of Public Transportation 12(4), 41–56 (2009)
18. De Nicola, R., Katoen, J.P., Latella, D., Loreti, M., Massink, M.: Model checking mobile stochastic logic. Theor. Comput. Sci. 382(1), 42–70 (2007)
19. Galton, A.: The mereotopology of discrete space. In: Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. LNCS, vol. 1661, pp. 251–266. Springer Berlin Heidelberg (1999)
20. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. A 104(9), 1876–1889 (2000)
21. Gillespie, D.T.: Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. 58, 35–55 (2007)
22. Gilmore, S., Tribastone, M., Vandin, A.: An Analysis Pathway for the Quantitative Evaluation of Public Transport Systems. In: IFM. LNCS, vol. 8739, pp. 71–86. Springer (2014)
23. Grosu, R., Smolka, S.A., Corradini, F., Wasilewska, A., Entcheva, E., Bartocci, E.: Learning and detecting emergent behavior in networks of cardiac myocytes. Commun. ACM 52(3), 97–105 (2009)
24. Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Gros, R., Belta, C.: Spatel: A novel spatial-temporal logic and its applications to networked systems. In: 18th International Conference on Hybrid Systems: Computation and Control. pp. 189–198. ACM (2015)
25. Hauskrecht, M.: Monte-Carlo approximations to continuous-time semi-Markov processes. Tech. Rep. CS-03-02, University of Pittsburgh (2002)
26. Jégourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Computer Aided Verification - 25th International Conference. LNCS, vol. 8044, pp. 576–591. Springer (2013)
27. Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: Spatial logic + temporal logic = ? In: Handbook of Spatial Logics, pp. 497–564. Springer (2007)
28. Larsen, K.G., Legay, A.: Statistical model checking: Past, present, and future. In: ISoLA. LNCS, vol. 8802, pp. 135–142. Springer (2014)
29. Legay, A., Delahaye, B., Bensalem, S.: Statistical Model Checking: An Overview. In: RV. LNCS, vol. 6418, pp. 122–135. Springer (2010)
30. Massink, M., Paškauskas, R.: Model-based assessment of aspects of user-satisfaction in bicycle sharing systems. In: 18th International Conference on Intelligent Transportation Systems. pp. 1363 –1370. IEEE (2015)
31. Midgley, P.: Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas. In: 19th session of the Commission on Sustainable Development. CSD19/2011/BP8, United Nations (2011)
32. Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties. In: RV. LNCS, vol. 9333, pp. 21–37. Springer (2015)
33. Pianini, D., Sebastio, S., Vandin, A.: Distributed statistical analysis of complex systems modeled through a chemical metaphor. In: International Conference on High Performance Computing & Simulation. pp. 416–423. IEEE (2014)
34. Sebastio, S., Amoretti, M., Lluch Lafuente, A.: A Computational Field Framework for Collaborative Task Execution in Volunteer Clouds. In: ICSE workshop SEAMS. pp. 105–114. ACM (2014)
35. Sebastio, S., Vandin, A.: MultiVeStA: Statistical Model Checking for Discrete Event Simulators. In: ValueTools. pp. 310–315. ACM (2013)
36. Serfling, R.J.: Approximation theorems of mathematical statistics, Probability and Statistics, vol. 162. Wiley & Sons (1980)