Exploring Spatio-temporal Properties of Bike-sharing Systems

Vincenzo Ciancia, Diego Latella, Mieke Massink, Rytis Paškauskas

Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo", Italy

***Abstract*—In this paper we explore the combination of novel spatio-temporal model-checking techniques, and of a recently developed model-based approach to the study of bike sharing systems, in order to detect, visualise and investigate potential problems with bike sharing system configurations. In particular the formation and dynamics of clusters of full stations is explored. Such clusters are likely to be related to the difficulties of users to find suitable parking places for their hired bikes and show up as surprisingly long cycling trips in the trip duration statistics of real bike sharing systems of both small and large cities. Spatio-temporal analysis of the pattern formation may help to explain the phenomenon and possibly lead to alternative bike repositioning strategies aiming at the reduction of the size of such clusters and improving the quality of service.**

I. Introduction

Smart bike sharing systems (BSS) have become recently a popular public transport mode in hundreds of modern cities [1], [2] operating from a few (e.g. Pisa) up to thousands of docking stations (e.g. Hangzhou, Paris, or London[1]). The principle of a BSS 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 for private use, 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. The operator assumes the responsibility to maintain a high level of usage of the system.

Operating a BSS raises multiple issues such as efficiency of fuel-consuming repositioning services [3], or integration with other public transport modes. Not the least, it is important to make the service attractive to its users. An indication that user satisfaction should be addressed seriously can be found in a survey of user experience with the *Bicing* BSS in Barcelona, conducted by Froelich et al. [4]. It reports that 75% of the users who used it for commuting between their home and study or work, stated that 'finding an available bike and parking slot' were the two most important problems, encountered in 76% and 66% cases of 212 respondents, respectively. Indeed, stations have finite capacities, and finding a completely full station close to a destination can be annoying. Since a bicycle must be returned to one of the designated stations lest the user should incur a fine, she must find another drop off station that is not full. This feature distinguishes bike-sharing from, e.g., the taxi service. The additional searching time, and the associated risk of undesired and unpredictable delays, and fees, are likely to affect her satisfaction with the system. However, user satisfaction due to such temporarily disabled stations is difficult to evaluate quantitatively from the available data. This is so because the cycling data alone is not sufficient to describe neither the intentions of its users, nor the predictability of the service. To investigate this issue from a different angle, a model-based approach was presented by some of the authors [5], in which the point of view of 'rational agents' who participate in bike-sharing, is assumed. Our study suggests that cycling times in different cities, and among pairs of stations within cities, are similarly distributed[2], suggesting a possibility of a generic interpretation. The rational agent model, based on minimal assumptions about travelling and decision making, reproduces rather well the cycling time distributions in London and Pisa [5]. This observation encourages extending the application of the model and investigating the performance of a system with respect to agents' intentions. The analysis suggests that some features of the cycling time distribution can be related to the predictability of a travel process which, as just discussed, can be related to the users' satisfaction with a system. In particular, it suggests extending the notion of a problematic full station to a problematic area in which all stations are full: a full-station *cluster*. A full station represents an area that cannot efficiently serve its customers. Full station clusters increase the size of unserviceable area and with it, the distance and time that a user with her objective in this area is likely to waste before finding a suitable station for parking. Indeed, formation of such clusters and their evolution is evident from the existing bike-sharing visualisations[3].

The main advantage of a modelling approach over data analysis is a possibility to study hypothetical cases where station configurations, traffic flows, or incentives are altered to explore the efficiency of proposed solutions to the aforementioned issues. In this paper, we present a first exploratory study from the ongoing work, that also serves to gain a better understanding of which properties of a system can be currently expressed in a succinct matter, and which of these properties could suggest fruitful insights into the flow management. Traces generated by a simulation model are studied using a novel spatio-temporal model checker based on *closure spaces*, called `topochecker`[4].

Spatio-temporal model-checking extends the classical approach to fully automated verification of software systems [7] to accommodate also spatial information. In our case, we extend the branching-time temporal logic CTL (*Computation Tree Logic)* adding the spatial operators defined in [8], ob-

[1]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

[2]see also [6]

[3]See, e.g. http://bikes.oobrien.com/london

[4]See https://github.com/vincenzoml/topochecker.

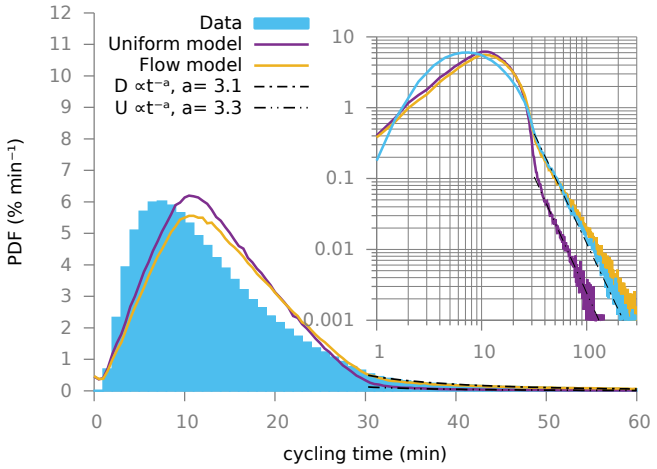

Fig. 1. Cycling duration histograms (Data) in London, using 831,754 trip records in October 2012, and results of simulation of the uniform model (dark lines) and the flow model (light lines). Maintenance trips are not considered.

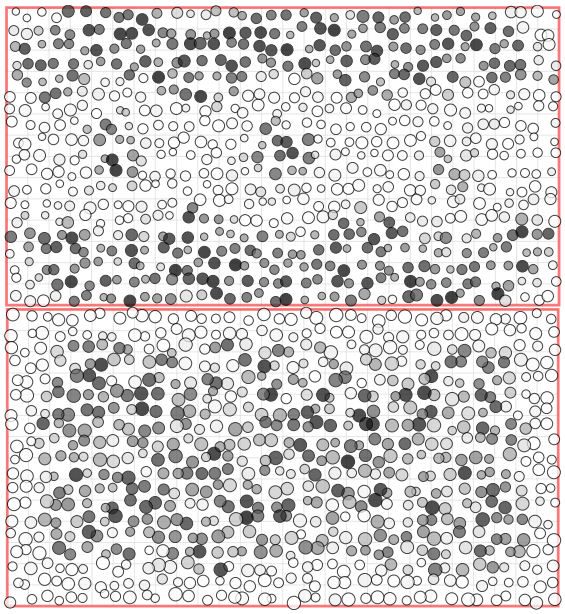

Fig. 2. Snapshots of the flow (top), and the uniform (bottom) models. The sizes of circles are proportional to station capacities, and the shades are proportional to the normalised available bicycles (NAB). The number of stations and bicycles are equal in both models, and the number of trips per hour and station capacities are similar.

taining the *spatio-temporal logic of closure spaces* (STLCS) [9]. The model-checking algorithm assigns to each formula of the logic a set of *points in space-time*, that is, a pair of a state and a point in space. Formulas featuring arbitrary nesting of spatial and temporal operators can be expressed and automatically verified. In this paper, we build on the existing simulator and spatio-temporal model checker. We show that the combination of the two tools can be used to verify formulas that study various spatial and temporal aspects of clusters (note that clusters are, in turn, a *spatial* phenomenon). In particular, STLCS is able to predicate about *formation*, *persistence*, and *propagation* of clusters, among other related phenomena.

The outline of the paper is as follows. In Sect. II we briefly recall the bike sharing model of [5] and in Sect. III we recall the spatio-temporal logic STLCS and related model-checker. In Sect. V we illustrate relevant spatio-temporal properties of the bike-sharing model. In Sect. VI and Sect. VII we discuss related work and preliminary conclusions, respectively.

II. From cycling times to user satisfaction

The bike-sharing model presented in [5] describes the dynamics of a population of rational agents, coupled to the dynamics of bicycle stations in a two-dimensional rectangle representing a city. The model parameters are calibrated so as to reproduce cycling times of a particular real BSS. Once satisfactory correspondence with a BSS of interest is established, its other features are inferred from the model. Ideally, they should provide additional insights about a system not available directly, or readily, from the available data.

A representative probability density function (PDF) of cycling times in London is shown in Fig. 1 (Data). One of its salient features is that 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 service area in London (about fifteen kilometres). This range coincides with the so-called 'algebraic tail' of the distribution, the range in which the PDF(t) is well approximated by $\propto t^{-a}$ with some exponent $a > 0$ (Fig. 1, inset). The rational agent model based on a Markov Renewal Process (MRP [5]) suggests that the algebraically tapered PDF is a generic consequence when the agents are willing to assume the risk of limiting the radius of the searched area. Such risk is rational, because it reduces the median trip duration [5]. Agents risk, of course, that a suitable station within the area is not found. These 'bad' events affect only a small fraction of all trips if the distribution of agents' origins and destinations is spatially homogeneous, as in Fig. 1, the 'uniform model' ($\Pr\{\text{trip} > 30\text{min} \mid \text{uniform}\} = 0.01$) but become more relevant if there are larger destination concentration ($\Pr\{\text{trip} > 30\text{min} \mid \text{flow}\} = 0.07$). The latter is called the 'flow model' and according to Fig. 1 it describes rather well the actual distribution in London. Presence of areas that attract more users than other areas is a reasonable assumption about real cities. An obvious consequence is that also the areas of full stations will be, as a rule, larger. However, identification and analysis of problematic areas is not so obvious. In Fig. 2, example snapshots of both types of models are shown.

In setting up the model we followed the principle that the total service area, the number and capacities of stations, the number of bicycles, and the average number of hourly trips should be close to those in London. However, we were not pursuing photographic accuracy of the underlying topography of the city, as our objective is only to illustrate the general idea of cluster identification. Thus for example, the nearest neighbours of stations are defined only topologically. 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, and 500 agents that make, on average, 900 trips per hour. The agent behaviour is sampled randomly, however, to introduce flows, a superposition of Gaussian distributions for the origin and destination locations is used for the flow model, and some counter-current flows are added to improve the balance of the flow. Numerical simulation of this model

generates traces, each trace consisting of snapshots, each snapshot representing a system's state at a particular instance of time. The agent distributions are not visible in Fig. 2 but they have an effect on the station occupations (see [5] for more details). Clearly, the distribution of stations and their temporal dependence is a complicated task for analysis. In the following sections, we will discuss the application of a spatio-temporal model checker to identify problematic stations. Among the station's properties that are commonly used to describe bike-sharing systems, we will use the number of bikes parked at a station (n), the number of vacancies (v; the capacity of a station is $c = n + v$), the normalised available bicycles ('NAB'$= \frac{n}{n+v}$, [4]), and the station bike and slot congestion averages $p^{\pm}(t)$ (see [5, Eq. 8]). The latter are equal to the fraction of the time up to t that a particular station is either full or empty.

III. Spatio-temporal model checking

In this section we briefly recall the *spatio-temporal logic for closure spaces*[5] (STLCS) and the model checking algorithm that was introduced in [9]. STLCS extends the spatial logic SLCS [8], [10] with temporal operators from the *computation tree logic* CTL (see e.g. [7]). The algorithm permits evaluation of spatio-temporal properties of points of space, using valuations that depend on the temporal state of execution.

First, we show the formal syntax of formulas, described by the following grammar, where `p` ranges over a finite or countable set of *atomic propositions*.

$$
\begin{array}{lcll}
\Phi & ::= & \texttt{TT} & \text{[True]} \\
 & | & [\texttt{p}] & \text{[Atomic predicate]} \\
 & | & !\Phi & \text{[Not]} \\
 & | & \Phi \,|\, \Phi & \text{[Or]} \\
 & | & \Phi \,\&\, \Phi & \text{[And]} \\
 & | & \texttt{N}\,\Phi & \text{[Near]} \\
 & | & \Phi \texttt{S} \Phi & \text{[Surrounded]} \\
 & | & \texttt{A}\,\varphi & \text{[All futures]} \\
 & | & \texttt{E}\,\varphi & \text{[Some future]} \\
 & & & \\
\varphi & ::= & \texttt{X}\,\Phi & \text{[Next]} \\
 & | & \texttt{F}\,\Phi & \text{[Eventually]} \\
 & | & \texttt{G}\,\Phi & \text{[Globally]} \\
 & | & \Phi\,\texttt{U}\,\Phi & \text{[Until]}
\end{array}
$$

Besides classical Boolean connectives, STLCS features:

Temporal operators: the CTL path quantifiers `A` ("for all paths"), and `E` ("there exists a path"). As in CTL, such quantifiers must necessarily be followed by a path-specific operator. Our algorithm uses a minimal set of path operators[6], namely `X` ("next"), `F` ("eventually"), `G` ("globally"), `U` ("until");

Spatial operators: the SLCS spatial operator `N` ("near"), expressing that a point in space is near another point in the spatial neighbourhood relation satisfying a certain formula, and the binary operator `S` ("surrounded"), denoting the fact that a point has "no way out" from a region of points satisfying a property, unless passing by points satisfying another property.

[5] 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 [8] for an introduction.

[6] All standard CTL operators can be derived from this set (see e.g. [7]).

A model $\mathscr{M}$ of STLCS is composed of a Kripke structure $(S, \mathscr{R})$, where S is a non-empty set of *states*, and $\mathscr{R}$ is a non-empty *accessibility relation* on states, and a closure space $(X, \mathscr{C})$, where X is a set of points and $\mathscr{C}$ the closure operator. For the purposes of this paper it is sufficient to consider the sub-class of closure spaces generated by graphs; a graph (V, E), with V the set of vertices and $E \subseteq V \times V$ uniquely characterises the closure space $(V, \mathscr{C}_E)$ where, for any $A \subseteq V$, $\mathscr{C}(A) = A \cup \{v \in V \,|\, \exists a \in A . aEv\}$. Such closure spaces belong to the class of *quasi-discrete* closure spaces (see [8]). Every state s has an associated valuation $\mathscr{V}_s$, making $((X, \mathscr{C}), \mathscr{V}_s)$ a *closure model* according to Definition 6 of [8]. Equivalently, valuations have type $S \times X \to 2^P$, where P is the set of atomic propositions, thus, the valuation of atomic propositions depends both on states and points of the space.

The truth value of a formula is defined at a point in space x and temporal state s. The mutual nesting of spatial and temporal operators permits one to express rather complex spatio-temporal properties. Let us proceed with a few examples. Consider the STLCS formula

$$\texttt{EG}[\texttt{green}]\texttt{S}[\texttt{blue}]$$

Point x satisfies such formula in state s if there exists (`E`) a temporal path rooted at s, such that in all states (`G`), i.e. at any point in time, x satisfies atomic property [`green`], and it is not possible to start from x, following edges of the spatial graph, and leave the region of points satisfying [`green`] in which x is located, unless passing by a point satisfying [`blue`].

A further example exhibiting nesting of spatio-temporal operators is the STLCS formula

$$\texttt{EF}[\texttt{green}]\texttt{S}(\texttt{AX}[\texttt{blue}])$$

This formula is satisfied by a point x in state s if point x possibly (`E`) satisfies [`green`] in some future (`F`) state s', and in that state, it is not possible to leave the area of points satisfying [`green`] unless passing by a point that will necessarily (`A`) satisfy [`blue`] in the next (`X`) time step.

For space reasons, we omit the formal semantics of the logic, which can be found in [9].

IV. The spatio-temporal model checker

To verify our formulas, we used a newly developed spatio-temporal model checker named `topochecker`. The tool[7] is a *global model checker* using a dynamic programming algorithm to verify STLCS formulas on finite models. Models are composed of a temporal part, which is a Kripke structure, and a spatial part, which is a finite, quasi-discrete closure space (see [8]). The Kripke structure and the spatial structure are given in the `dot` graph description language[8], and valuations of atomic propositions are provided by a *comma-separated-values* file associating to each point in space-time a list of propositions. The tool enriches basic STLCS by allowing users to specify that some atomic properties have an associated

[7] The tool is available at https://github.com/vincenzoml/topochecker under an open source license, and is meant to be an optimized and user-oriented implementation of the algorithm that was studied in [11], therefore superseding previously developed prototypes.

[8] Further information on the `dot` notation can, for example, be found at http://www.graphviz.org/Documentation.php.

floating-point value; therefore, atomic propositions can be basic comparisons between the name of an atomic property and a floating point value. The tool permits parametric macro abbreviations, that we use in Section V.

The model checker is written in the programming language OCaml[9], and carefully optimised. In particular, we use native arrays and memory management (through the OCaml library `bigarray`); the algorithm uses a table of $k \times s \times f$ memory words, where k is the number of temporal states, s is the number of spatial locations, and f is the number of subformulas of a formula. One pass is executed over this table, filling it with a truth value for every cell, running in $O(k \times s \times f)$ steps.

Execution times are interesting for our experiments: even if we tested a quite large number of complex formulas (not all of which have been discussed here in detail), all our experiments are completed in negligible amounts of time, of the order of one second, on a standard laptop; this is because our graph is rather small (722 stations) and the number of instants of time we consider is in the order of 100-1000 states. Despite such small numbers, our preliminary experiments show that spatio-temporal model-checking can give useful insights on interesting behavioural features of bike sharing systems, even for large ones, like the one of London.

V. Properties and results

In this section the main examples will be presented in the form of logical formulas, expressing several properties of a bike-sharing model. These formulas are interpreted as signals of possible interest to the operator, such as the presence, or persistence of problematic clusters of stations. It is intended as a demonstration of how a spatio-temporal model-checker may be used to encapsulate such signals in a completely formal way, without attempting to explore the rich repertoire of possibilities (see [12] for more examples of spatio-temporal formulas). We are concerned with the full-station clusters as more salient to the discussion, although the extension of the same kind of formulas to empty station clusters is a straightforward matter.

The spatial structure is added to the simulation model in the form of an undirected graph, whose vertices are stations, and edges are the nearest station connections. The parameters are detailed in section II. A single trace of the simulation model is used as input to the model-checker. It represents the evolution of a model at specific time intervals, truncated after a given number of steps. For all the experiments except the last one (related to user satisfaction), the duration of an interval is 10 minutes, and the number of time steps is 101. In the last experiment, we considered a time interval of 1 minute and 301 time steps[10]. Spatio-temporal model-checking is performed on a single simulation trace. Starting with simple expressions of a system's state, we proceed to develop more complex formulas that nest spatial and temporal operators.

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

[9]See http://www.ocaml.org.

[10]The results can be reproduced using the data and scripts, provided with the source code of the tool.

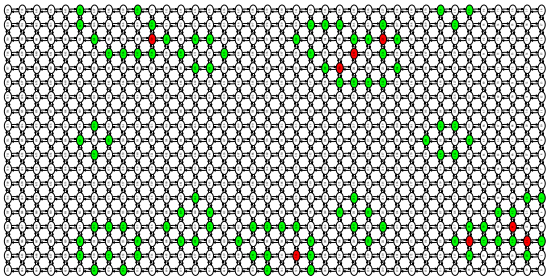

Fig. 3. Stations of time step 80 of our simulation that are on the boundary of the region of points that will eventually become a cluster (green), and stations that, whenever they are full, stay full and become part of a cluster (red).

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

Connectivity is expressed by 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 is therefore composed of a full station such that its direct neighbours in the north, south, east and west directions are also full. Note that the definition of `cluster` only identifies (on purpose) these "inner" full stations and not their direct full neighbours. The macro abbreviation `full` uses a boolean predicate (equality), applied to the quantitative value of the atomic property $[\texttt{vacant}]$.

Formation of clusters: a point evolves into a cluster when it becomes full, and stays full until it becomes part of a cluster. This may be detected by the following formulas:

$$\begin{aligned} \texttt{implies}(\texttt{f},\texttt{g}) &= (!\texttt{f})\,|\,\texttt{g}; \\ \texttt{nextCluster} &= (\texttt{EF full})\,\&\,(\texttt{AG implies}(\texttt{full}, \texttt{A full U cluster})) \end{aligned}$$

Here, `implies` is standard logical implication. The definition of `nextCluster` characterises points that will eventually become full and, for every future state, whenever full, they will stay full until becoming part of a cluster. This is a very strong property, that few points possess. Such points are central in cluster formation, as they represent stations that always form a cluster when they become full. In Fig. 3, these points are shown in red, in a state[11] of the simulation where there are many of them. For comparison, the boundary of the points that will become a cluster are shown in green, that is, those points satisfying $(!\texttt{EF cluster})\,\&\,(\texttt{N EF cluster})$.

Persistence of clusters: 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. This situation, for e.g., two and three time steps, is described by the formulas

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

By combining the formulas described above with the *eventually* operator, the tool is able to detect the stations that, in any state, will eventually be part of a cluster, with specified persistence. Let us look at Figure 4, where we show the output

[11]The tool is a *global* model checker, therefore it is able to produce a graph for each state of the model, related to the truth value of formulas in that particular state, even if we only show results related to one specific state.

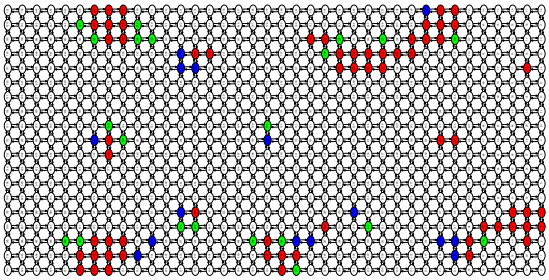

Fig. 4. Points of the initial state of the simulation that will eventually become part of a cluster (green), or of a cluster that persists for two (resp. three) time steps, coloured in blue (resp. red).

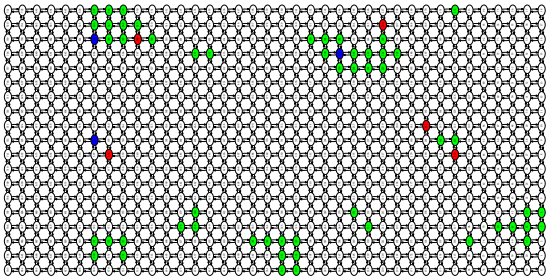

Fig. 5. Points of time step 77 of the simulation that are part of a cluster (in green), or are not full, but will become part of a cluster in one step (in red), or that are part of a cluster but will become not full in one step (in blue).

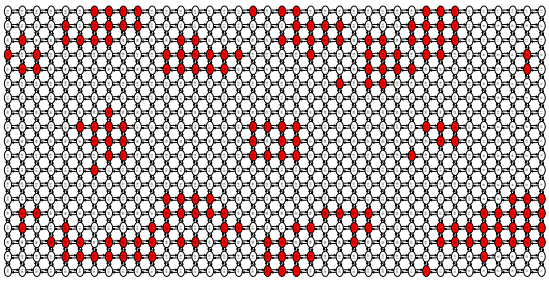

Fig. 6. Stations that are full in time step 0, where it is possible that an user applying the obvious strategy of waiting for some time, and then moving nearby, might still not find a free parking slot.

of a model checking session. The tool colours in red nodes that satisfy the formula `EFcluster3steps` (and thus also formulas describing shorter persistence times), in blue those that satisfy `EFcluster2steps`, and in green those where formula `EFcluster` is true. If we compare the results with the snapshot of the flow model shown in Fig. 2, we can see that the areas where persistent clusters arise over time are indeed corresponding to the areas where the stations are relatively full. The single snapshots such as those in Fig. 2 do not say anything about the evolution of possible clusters though. So this is additional, useful information that is very easy to obtain via spatio-temporal model checking.

Propagation of clusters: another phenomenon that can be investigated using `topochecker` is the spatial propagation of cluster. Among many possible related STLCS formulas, we show how to detect points that obey to at least one of the following: 1) they are not full, but are close to a cluster, and will necessarily become part of a cluster in the near future; or 2) they are part of a cluster, but will necessarily become not full in a short amount of time, even if still being physically close to a cluster. This is achieved by the following definitions, where the macro `bdry(f)` describes the topological boundary of the set of points satisfying f (see [8]):

$$\begin{aligned}
&\texttt{bdry(f)} && = (\texttt{!f})\,\&\,(\texttt{N f})\\
&\texttt{growingCluster} && = (\texttt{!cluster})\,\&\\
&&& \quad (\texttt{N}(\texttt{bdry}(\texttt{cluster}))\,\&\,(\texttt{AX full})\\
&\texttt{shrinkingCluster} && = \texttt{cluster}\,\&\,(\texttt{AX}(\texttt{!full}))
\end{aligned}$$

The model checker can be used to verify these formulas. For instance, in time step 77 of our simulation, there are both stations that will join a cluster and stations that will leave a cluster in the next time step. We show the result in Figure 5, where we used the tool to colour in green states satisfying `cluster` (for comparison), and then to colour in red states satisfying `growingCluster` and in blue states satisfying `shrinkingCluster`. The results of these formulas provide insight in the dynamics of the clusters at particular time steps, in particular the directions in which the clusters are evolving. This may be important information for the development of repositioning strategies in particular when such dynamics are repeated over time in the same areas.

User experience: STLCS can also be used to identify specific problems related to user experience in BSSs. For example, when an user is willing 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. This behaviour may be typically sufficient to solve the problem, at the expenses of a longer trip duration. One may want to check whether this procedure is effective in a few time steps. In the following formula, we check whether it is possible that, in three time steps, the user still was unable to leave the bike in a station, which is full in the current state.

$$\begin{aligned}
\texttt{tripEnd} \quad = \texttt{full}\,\&\,(\texttt{N}(\texttt{AX}(\texttt{full}\,\&\\
(\texttt{N}(\texttt{AX}(\texttt{full}\,\&\,\texttt{N}(\texttt{AX full})))))))
\end{aligned}$$

The output from the model checker, colouring in red points of time step 0 where the formula is true, is given in Fig. 6.

VI. Related work on spatial logics

Different forms of spatial logic have been proposed in computer science to refer to logics expressing properties of structured objects such as processes or data structures, in particular in the context of π-*calculus* (e.g. [13]) and *mobile ambients* with the related ambient logic (e.g. [14]). For example a binary logic operator has been introduced, $\Phi|\Psi$, that holds for a process P when this process is a parallel composition of two processes Q, satisfying Φ, and R, satisfying Ψ. Our work is not directly connected to these logics, but rather to logics that are *spatial* in the topological sense (see [15] for a comprehensive reference).

In [16] 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. Furthermore, in a stochastic setting, the Mobile Stochastic Logic (MoSL) [17] 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 computations. Other variants of spatial logics concern the symbolic

representation of the contents of images, and, combined with temporal logics, for sequences of images [18]. The approach is based on a discretisation of the space of the images in rectangular regions and the orthogonal projection of objects and regions onto Cartesian coordinate axes such that their possible intersections can be analysed from different perspectives.

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 [8] and have also inspired the work on Spatial Signal Temporal Logics in [19], 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 [11].

VII. Conclusions and outlook

Spatio-temporal model-checking allows for the verification of complex properties that concern the sophisticated interplay of temporal and spatial modalities. In this paper we have explored a range of properties of large bike sharing systems to obtain a more detailed insight in specific emerging patterns that have an effect on the quality of service that a bike sharing system can provide from a user's point of view such as the possibility not to find a suitable parking slot within a reasonable distance from the planned destination.

In this explorative study we used only a fraction of the potential of the spatio-temporal model checker. In particular, we used it only on traces from a simulator in this case, whereas the logic for which it is designed is a *branching time* spatial logic. In fact, in future work we plan to extend its use to analyse branching structures. These arise naturally, for example, in the verification or comparison of the effect of different repositioning strategies for bikes. Different strategies may be available at specific times, yielding a non-deterministic choice between different options to alleviate the problem of the formation of clusters of full stations. One could also use spatio-temporal model-checking in a real-time monitoring scenario, in order to detect the emergence of particular patterns in a bike-sharing system in real-time, and to provide quick feedback on the possible effects of particular maintenance or repositioning interventions. We note that the size of bike sharing systems permits verification of rather complex properties in very short amounts of time. Therefore, the combination of simulation and model-checking that we propose may be used as a support system for real-time decision making in bike sharing systems. For this purpose, we plan to extend our spatio-temporal model checker to deal with quantitative measures such as distances or probabilities, and the introduction of bounded variants of the various logical operators. Using our tools in combination with statistical model checking is also worth exploring.

Acknowledgements: This work is supported by the EU project *QUANTICOL: A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours*, 600708, and the IT MIUR project CINA. We thank Mirco Tribastone and Daniël Reijsbergen for London bike-sharing data.

References

[1] P. De Maio, "Bike-sharing: Its history, impacts, models of provision, and future." *Journal of Public Transportation*, vol. 12, no. 4, pp. 41–56, 2009.

[2] P. Midgley, "Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas," in *19th session of the Commission on Sustainable Development*, ser. CSD19/2011/BP8. United Nations, 2011.

[3] E. Fishman, S. Washington, and N. L. Haworth, "Bike share's impact on car use: evidence from the United States, Great Britain, and Australia," in *Proceedings of the 93rd Annual Meeting of the Transportation Research Board*, 2014.

[4] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *IJCAI*, 2009, pp. 1420–1426.

[5] M. Massink and R. Paškauskas, "Model-based assessment of aspects of user-satisfaction in bicycle sharing systems," in *Proceedings of the 18th IEEE International Conference on Intelligent Transportation Systems 2015*. IEEE, 2015, to appear.

[6] P. Borgnat, P. Abry, P. Flandrin, C. Robardet, J.-B. Rouquier, and E. Fleury, "Shared bicycles in a city: A signal processing and data analysis perspective," *Adv. Complex Syst.*, vol. 14, no. 3, pp. 415–438, 2011.

[7] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[8] V. Ciancia, D. Latella, M. Loreti, and M. Massink, "Specifying and verifying properties of space," in *8th IFIP-TCS Conference, Track B*, ser. LNCS, vol. 8705. Springer, 2014, pp. 222–235. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44602-7_18

[9] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink, "An experimental spatio-temporal model checker," in *Proceeding of VERY*SCART (workshop affiliated to SEFM 2015)*, ser. LNCS. Springer, 2015, to appear. Extended version of QC-TR-10-2014, http://milner.inf.ed.ac.uk/wiki/pages/J8N4c8/QUANTICOL_Technical_Reports.html.

[10] V. Ciancia, D. Latella, M. Loreti, and M. Massink, "Specifying and Verifying Properties of Space - Extended version," QUANTICOL, Technical Report TR-QC-06-2014, 2014.

[11] V. Ciancia, S. Gilmore, D. Latella, M. Loreti, and M. Massink, "Data verification for collective adaptive systems: spatial model-checking of vehicle location data," in *2nd FoCAS Workshop on Fundamentals of Collective Systems*, ser. IEEE Eight International Conference on Self-Adaptive and Self-Organizing Systems. IEEE Computer Society, 2014.

[12] V. Ciancia, S. Gilmore, G. Grilletti, D. Latella, M. Loreti, and M. Massink, "Spatio-temporal model-checking of vehicular movement in transport systems," *submitted.*, available from the authors.

[13] L. Caires, "Behavioral and spatial observations in a logic for the π-calculus," in *Proceedings of the 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, ser. LNCS, I. Walukiewicz, Ed., vol. 2987. Springer, 2004, pp. 72–87.

[14] L. Cardelli and A. D. Gordon, "Anytime, anywhere: Modal logics for mobile ambients," in *Proceedings of the 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, 2000, pp. 365–377.

[15] J. van Benthem and G. Bezhanishvili, "Modal logics of space," in *Handbook of Spatial Logics*, 2007, pp. 217–298.

[16] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci, "Learning and detecting emergent behavior in networks of cardiac myocytes," *Commun. ACM*, vol. 52, no. 3, pp. 97–105, 2009.

[17] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink, "Model checking mobile stochastic logic," *Theor. Comput. Sci.*, vol. 382, no. 1, pp. 42–70, 2007.

[18] A. D. Bimbo, E. Vicario, and D. Zingoni, "Symbolic description and visual querying of image sequences using spatio-temporal logic," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 4, pp. 609–622, 1995.

[19] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink, "Qualitative and quantitative monitoring of spatio-temporal properties," in *15th International Conference on Runtime Verification*, ser. Lecture Notes in Computer Science. Springer, 2015, to appear.