Towards Automatic Decision Support for Bike-Sharing System Design*

Maurice H. ter Beek, Stefania Gnesi, Diego Latella, and Mieke Massink

ISTI–CNR, Via G. Moruzzi 1, Pisa, Italy
{terbeek,gnesi,latella,massink}@isti.cnr.it

Abstract. Public bike-sharing systems are a popular means of sustainable urban mobility, but their successful introduction in a city stands or falls with their specific designs. What kind of bikes and docking stations are needed, how many and where to install them? How to avoid as much as possible that stations are completely empty or full for some period? Hence, a bike-sharing system can be seen both as a highly (re)configurable system and as a collective adaptive system. In this paper, we present two complementary strategies for the evaluation of bike-sharing system designs by means of automated tool support. We use the Clafer toolset to perform multi-objective optimisation of attributed feature models known from software product line engineering and the recently developed mean field model checker FlyFast to assess performance and user satisfaction aspects of variants of large-scale bike-sharing systems. The combined use of these analysis approaches is a preliminary step in the direction of automatic decision support for the initial design of a bike-sharing system as well as its successive adaptations and reconfigurations that considers both qualitative and performance aspects.

1 Introduction

More and more cities are deploying public bike-sharing systems (BSS) as a sustainable urban mode of transportation [22]. The concept is simple: a user arrives at a docking station, rents a bike, uses it for a while and returns it to a station close to their destination; payment is either per trip or by subscription. BSS potentially offer multiple benefits, among which the reduction of vehicular traffic, pollution, noise and energy consumption. To improve the efficiency and user satisfaction of BSS, the load between different stations should be balanced, e.g. by using incentive schemes that influence the behaviour of users but also by efficient redistribution of bikes among stations.

The current third generation technology-based BSS are very different from the first generation free BSS introduced in Amsterdam roughly half a century ago. *Bicing*, the well-known and successful BSS of the city of Barcelona, currently consists of over 6,000 bikes and 420 stations. There are now similar BSS

* Research partly supported by the EU FP7-ICT FET-Proactive project QUANTICOL (600708) and by the Italian MIUR project CINA (PRIN 2010LHT4KM).

in more than 500 cities worldwide. The largest can be found in China with upto 90,000 bikes and over 2,000 stations, one every 100 meters. Fourth generation BSS are already being developed. These include movable and solar-powered stations, electric bikes and smartphone real-time availability applications [22]. In the context of QUANTICOL (`www.quanticol.eu`) we collaborate with PisaMo S.p.A., an in-house public mobility company of the Municipality of Pisa, which introduced the BSS *CicloPi* in Pisa two years ago. This BSS, which currently consists of roughly 140 bikes and 15 stations, was supplied by Bicincittà S.r.l.

The design of a BSS is multi-faceted and complex. First of all, BSS are composed of many components, among which bikes and stations, but also human users. The latter form an intrinsic part of the BSS and their individual patterns of behaviour have a decisive impact on the collective usability and performance of a BSS, which is highly dynamic. Furthermore, there are questions concerning costs of installing and running a BSS, maintenance, specific user preferences and needs and specificities of the city architecture. Hence, BSS can be seen as highly (re)configurable systems and collective adaptive systems[1] (CAS). Our long-term goal is to be able to provide automatic decision support for the initial design of a BSS to be deployed in a city, as well as for successive adaptations and reconfigurations that consider both qualitative and performance aspects. In this context, it is important to realise that the design and behaviour of the individual entities from which a BSS is composed, may exhibit variability not only in the kind of features but also in the quantitative characteristics of features.

In [3–6], we studied product lines of BSS and their bikes, respectively. In this paper, we make use of these product lines to analyse different configurations. Software product line engineering (SPLE) is an engineering approach aimed at cost-effectively developing a variety of (software-intensive) products from a common reference model or architecture, i.e. that together form a (software) product line. Commonalities and differences are defined in terms of *features* and variability models encode exactly those combinations of features forming valid products. Actual product configuration during application engineering is thus reduced to selecting desired options in the variability model. We extend the product lines from [3, 4, 6] by explicitly taking feature attributes and feature cardinalities into account. The former enrich variability models with quantitative constraints, while the latter allow us to explicitly distinguish BSS configurations by the specific number of stations and bikes of each variant that are used.

We will first perform multi-objective optimisation of a BSS variability model with the recently developed toolset Clafer [1], after which we will study the use of the recently developed on-the-fly mean field model checker FlyFast [18, 20] to analyse behavioural and performance aspects of BSS configurations. Clafer-MOO(Visualizer) [23] allows to compare system configurations (variants) with respect to various quality dimensions (e.g. cost), select the most desirable one and analyse the impact of reconfigurations on a variant's quality dimensions.

[1] These are systems consisting of a large number of spatially distributed heterogeneous entities with decentralised control and varying degrees of complex autonomous behaviour able to adapt to changing circumstances.

Model checking is a widely used, powerful approach to the automatic verification of concurrent, distributed systems, including performance aspects. It is an efficient procedure that, given an abstract system model $\mathcal{M}$, decides whether $\mathcal{M}$ satisfies a (temporal) logic formula Φ. Currently, the integration of mean field and fluid approximation techniques with model-checking [8, 9, 17, 18, 20] is receiving increased attention as a way to obtain highly scalable formal methods supporting the design of *large-scale* CAS for which performance aspects are essential to their desired behaviour. Mean field approximation techniques originate in statistical physics and biochemistry where they are used to analyse large-scale phenomena like particle interaction and chemical reactions between molecules of different substances. The key idea of such approximation techniques is to replace the actual stochastic or probabilistic interactions in a system, which often lead to a combinatorial explosion of possibilities, by an approximation of the average system behaviour over time in terms of the numbers (fractions) of elements present in a population. In our setting, we assume that the elements have a small number of local states, and we consider the number (fraction) of elements (agents) that are in a particular local state [16, 21]. The change over time of these fractions can be defined as the solution of a set of ordinary differential equations or, in our case, difference equations, given an initial state of the overall system, and approximate its evolution over time. Informally speaking, the larger the populations in the system, the better the quality of the approximation. Typically their size is in the order of hundreds, thousands or even better, millions.

A more technical introduction to mean field and fluid approximation can be found in [10]. For what concerns mean field model checking, the idea is to analyse the properties of the behaviour of a single agent in the context of the overall system behaviour (approximated as described before). The difficulty in model-checking the properties of such an agent is that the probabilities involved in its behaviour are not constant but may, e.g., depend on the changing fractions of specific agents in the system over time. So the probabilities in the model of the agent under study are time-dependent or time-inhomogeneous, or more formally time-inhomogeneous discrete time Markov chains (IDTMC). In [20], we analysed performance aspects of a BSS with homogeneously distributed resources.

An important characteristic of BSS is the probability that stations are empty or full, since these situations generate distress among users and discourage them to use the BSS. In an online survey conducted by Froehlich [13] on the *Bicing* BSS in Barcelona in 2009 about the experience of users with bike sharing, it turned out that 75% of the users stated commuting as a motivation to sign up for membership. Moreover, the same users identified "finding an available bike and a parking slot" as the two most important problems encountered (76% and 66% of the 212 respondents, respectively). These problems should therefore be addressed in the best possible way within the obvious budget constraints of cities, and at the same time keeping the number of kilometres made by vans that are involved in the (unavoidable) redistribution of bikes as small as possible.

We thus use the on-the-fly mean field model checker FlyFast [18, 20] for the analysis of properties of selected stations in a BSS and for a mean field analysis of the BSS [21]. This prototypical tool was developed for the analysis of discrete time Markov population models. This differs from fluid model-checking approaches developed for the analysis of *continuous time* Markov population models as in [8, 9, 17] which moreover use a global model-checking approach, i.e. an approach in which a formula is analysed for *all* states. The latter approach always requires a full state space search, whereas an on-the-fly approach generates only as much of the state space as necessary to analyse the property. Especially in case of *conditional* reachability properties this may be much more efficient. On-the-fly mean field model checking on discrete time Markov population models can also be used to approximate global fluid model checking of continuous Markov population models under certain conditions (see [19] for further details).

The paper is organised as follows. In §2 we define a variability model of a BSS and analyse it with ClaferMOOVisualizer in §3. In §4 we capture BSS behaviour in a Markov population model and analyse it with FlyFast in §5. §6 outlines how to combine these approaches in a future decision support system for BSS design.

2 Attributed Feature Model of BSS

The de facto standard variability model in SPLE is a *feature model* [24]. A *feature* characterises a stakeholder visible piece of functionality of a product or system and a feature model provides a compact representation of all possible products of a product line or configurable system in terms of their features (behaviour is not captured). However, there may be hundreds of features or configurable options, which easily leads to superfluous or contradictory variability information (e.g. 'false' optional or 'dead' features). There is a lot of work on computer-aided analyses of variability models to extract valid products and detect anomalies [7].

Graphically, features are nodes of a rooted tree and relations between them regulate their presence in products: *optional* features may be present provided their parent is; *mandatory* features must be present provided their parent is; exactly one *alternative* feature must be present provided their parent is; at least one *or* feature must be present whenever their parent is; a *requires* constraint indicates that the presence of a feature requires that of another; an *excludes* constraint indicates that two features are mutually exclusive. We identify a product $\mathcal{P}$ from the product line with a non-empty subset $\mathcal{P}_{\mathcal{F}}$ of the set $\mathcal{F}$ of features. Deciding whether a product satisfies a feature model can be reduced to Boolean satisfiability (SAT), which can be effectively computed with SAT solvers [2].

In this paper we consider a BSS with three types of stations.We assume stations with capacity 15 to be located in the city centre (C), those with capacity 5 in the periphery (P) and those with capacity 10 in between (M, for middle). We define configuration 1 of a size in the order of that of the BSS in Barcelona in 2009 [13]: 330 stations of which 100 of type P, 150 of type M and 80 of type C. Subsequently, we present analyses of reconfiguring it into 397 stations of which 200 of type P, 150 of type M and 47 of type C (defined as configuration 2).

Once we equip features with attributes (e.g. `capacity(Centre) = 15`) we obtain an *attributed feature model* [7]. Now a product $\mathcal{P}$ is a non-empty subset $\mathcal{P}_{\mathcal{F}} \subseteq \mathcal{F}$ that moreover satisfies the additional quantitative constraints over feature attributes (e.g. `capacity(DockingStation)` $\leq$ `10`). Complex quantitative constraints require satisfiability modulo theories (SMT) solvers like Z3 [11].

We consider the attributed feature model of a BSS depicted in Fig. 1, which we obtained by adding attributes to the feature model of the bike-sharing product line of [3, 4] (ignoring, mainly, the user feature) and replacing its `Bike` feature with the attributed feature model of the bikes product line of [6] (ignoring the computational unit feature). We extracted all features and values for the cost (in euros) and capacity attributes from documents received from Bicincittà. The values for customer satisfaction (`c_sat`) instead are estimates based on discussions with PisaMo and Bicincittà. Our research aims to replace them by more realistic values obtained from performance analyses like those we perform in §5.

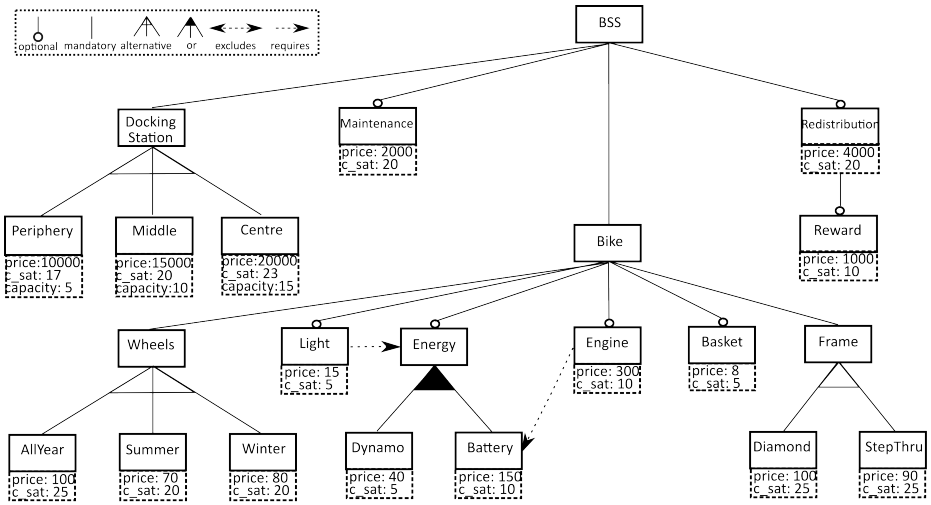

Fig. 1. Attributed feature model of a BSS.

3 Variability Analysis of BSS

In this section, we present initial variability analyses over the above BSS model. We use the attributed feature modelling capabilities of the lightweight textual SPL modelling language Clafer [1] and its extension ClaferMOO(Visualizer) [23]. Each feature can have attributes and quality constraints can be specified globally or in the context of a feature. Hence we can associate a price to each feature and a global constraint that only allows products (feature configurations) whose total

costs remain within a predefined threshold value. If more than one attribute is associated with a feature this may result in multiple such optimisation objectives.

The ClaferMOO extension of Clafer was specifically introduced to support attributed feature models with complex multi-objective optimisation goals. The latter have a set of solutions, known as the Pareto front, that represents the trade-offs between several conflicting objectives. Intuitively, a Pareto-optimal solution is such that no objective can be improved without worsening another. A set of Pareto-optimal variants generated by ClaferMOO can be visualised (as a multi-dimensional space of optimal variants) and explored in the interactive online tool ClaferMOOVisualizer, which was specifically designed for SPL scenarios.

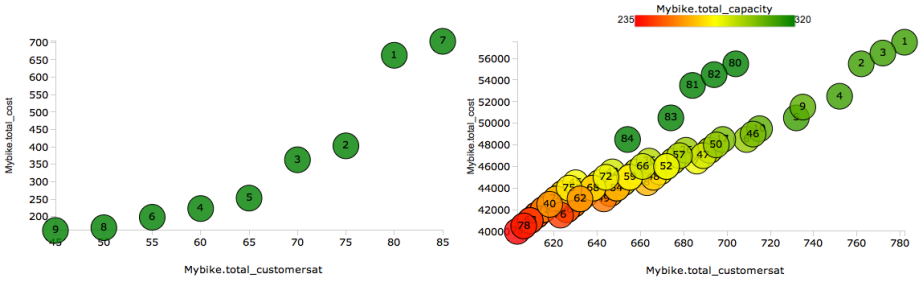

Fig. 2. Bubble graphs of Pareto fronts for Bike configurations (left) and the BSS (right).

We first focus on the bike feature in Fig. 1 in isolation. Figure 2(left) depicts the result of optimising it by minimising the cost of a product while at the same time maximising customer satisfaction. If we inspect the nine resulting variants in detail, we see that variants 1 and 7 are very costly, since these concern electric bikes. We conclude that variants 2–5 offer reasonable customer satisfaction at an affordable cost. These are exactly the variants without `Engine` but with `Energy`.

Now that we selected the bike configuration(s), we turn to the rest of the BSS in Fig. 1. Recall that we are interested in comparing BSS configurations with different distributions of docking stations over a city. Therefore, we make use of the possibility to add feature cardinalities to a feature model specified in Clafer. We allow between 33 and 39 docking stations,[2] upto 10 stations of type P, 15 of type M and 8 of type C for the first configuration, upto 20 stations of type P, 15 of type M and 4 of type C for the second configuration and we use additional constraints to prohibit mixing the two configurations of docking stations. For this to work, we actually clone the features `Periphery`, `Middle` and `Centre` thus allowing us to distinguish the stations used in one configuration from those used in the other. This is a novel use of feature cardinalities and feature cloning.

Figure 2(right) depicts the result of optimising the model just described by minimising a product's cost while at the same time maximising both customer

[2] There is no technical limitation, but for ease of presentation and to speed up the computation we divided the number of stations and all costs by a factor 10.

satisfaction and capacity. This results in 84 variant configurations, coming from the three optional features `Maintenance`, `Redistribution` and `Reward` and all possibilities to reconfigure configuration 1 (33 stations: 10 of type P, 15 of type M, 8 of type C) into a variant configuration with upto 39 stations, of which upto 20 of type P, upto 15 of type M and upto 4 of type C, thus including configuration 2. Variants 1–5 all concern configuration 2, while variants 80–84 all concern configuration 1. If we inspect these variants in detail, it turns out that configuration 2 has slightly less capacity than configuration 1 (310 vs. 320), offers much higher user satisfaction (732–782 vs. 654–704) at a generally higher cost (50500–57500 vs. 48500–55500). Note that the actual cost of a BSS with configuration 1 or 2 must be multiplied by 10 and, moreover, the cost of the bikes must be added. For capacities of 3100–3200 parking slots, some 1550–1600 bikes are needed [12]. Bike variants 2–5 cost between 223 and 403 euro per bike.

4 BSS as a Markov Population Model

For the analysis of performance-related aspects we now define the BSS as a Markov population model. Our model is inspired by the bike-sharing model of Fricker and Gast [12], which is a continuous time model based on homogeneous space where all locations are equally accessible to the users. We however use *discrete time* variants of this model. In the simple case, the model consists of N stations, each with a capacity of K parking slots. The number of bikes in the system is constant at s bikes per station on average, amounting to sN bikes in total. In every time step each station has the same probability that a user requests a bike. The probability that a bike is returned to a station depends on the number of bikes in circulation (i.e. not parked). Figure 3 shows a graphical representation of the model of a single bike station with K parking slots.

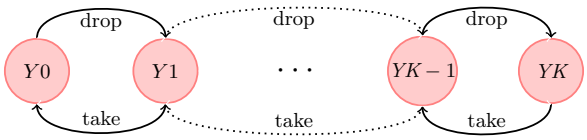

Fig. 3. A bike station.

Following [12], we denote the stochastic process composed of N stations in parallel by $Y^N(i) = (Y_0^N(i), \ldots, Y_K^N(i))$, but in our case this denotes a discrete time Markov process where i denotes the discrete time step. Each element $Y_k^N(i)$ in the vector denotes a random variable that gives the fraction of the total number of stations that have k bikes parked in it at time step i. In the following we address the request and return rates, assuming that we reached the system state $(y_0, y_1, \ldots, y_K)$.

Bikes requested. Assume there is one request of a bike per time unit per station on average, and for simplicity, assume that one time unit corresponds to 1 hour. Then we have in total, for the whole system, on average, N requests for bikes per hour. The arrival of requests to the set of stations in which exactly k bikes are parked is then Ny_k. If such a request indeed arrives at a station with k parked bikes, then the fraction of stations y_k decreases with $1/N$ and the fraction of stations y_{k-1} increases with $1/N$. For technical reasons a time unit of 1 hour is too long to obtain accurate results using a mean field approximation in discrete time in the context of this BSS model. In what follows we therefore rescale the time units to smaller ones, adjusting the probabilities of occurrences of requests per time unit accordingly. In particular, we use λ to denote the number of requests per hour and assume time units of two minutes, obtaining the probability of a request for a bike in a time unit as $\lambda/30$.

Bikes returned. The number of bikes that can be returned depends on the number of bikes in use, which can be obtained as the total number of bikes minus those that are parked in the stations. The total number of bikes is sN, where s is the average number of bikes per station and N the total number of stations. The average number of bikes parked in the stations is $\sum_{k=1}^{N} ky_k$, so the number of bikes in transit can be expressed as $N(s-\sum_{k=1}^{N} ky_k)$. We further assume that the average travel times, i.e. the time during which a bike is used for one trip, is $1/\mu$ and that stations to which bikes are returned are selected at random. Under these assumptions, assuming for simplicity that $\mu = 1$, a bike is returned to a station with $k \leq K-1$ bikes with rate $y_k \mu N(s-\sum_{k=1}^{N} ky_k)$ per hour. If such a bike is indeed returned to a station with $k \leq K-1$ bikes, then y_k is decreased by $1/N$ and y_{k+1} is increased by $1/N$. As before, we will work with time steps of two minutes, which means that we work with $\mu/30$ and that the probability to return a bike to a station with k bikes in one time step is $y_k(\mu/30)N(s-\sum_{k=1}^{N} ky_k)$. In all experiments in the sequel, one time unit corresponds to 1/30th of an hour (i.e. 2 minutes).

We extend this basic model by introducing the populations P, M and C of stations, as anticipated in §2. Recall that next to a different capacity, each also has a different location in the city. These locations are characterised by different usage patterns of bikes by commuters. Such patterns were observed in data about usage in real BSS such as the one in Barcelona [13]. Most commuters live in the suburbs and take the bike in the morning to go to their work or to school in the centre and go homewards somewhere in the afternoon or towards the evening. This leads to clearly distinguishable usage patterns that show complementary behaviour: stations in the periphery have a high request rate in the morning whereas those in the centre, with some delay, have a high return rate. These flows are reversed in the afternoon and towards the evening. The stations in the middle show a more stable pattern as requests and returns are more balanced.

We model the flow of commuters between the periphery and the centre during daytime by a combination of oscillating functions. The latter are modelled

as populations within the same specification framework[3]. Their definition is inspired by the oscillatory process defined in [15]. Figure 4(left) shows three periods of 12 hours each, reflecting periods during which the BSS is most used (we omit night time). The early morning of the first day period starts around time step 100. The curves show the change in request and return rates in the periphery and centre stations over the period, with a clear peak in the morning and a smaller, more distributed peak during the afternoon and evening hours. The rates oscillate around the average request rate $\lambda = 1$ and return rate $\mu = 4$. For request rate 1 this means $330 (= N)$ requests per hour for the total system. When modulated by the oscillations, the number of requests varies between $0.5 \times N = 115$ and $1.3 \times N = 440$ per hour, assuming that each request corresponds to finding a bike. In data about the BSS in Barcelona (which has twice as many bikes and total capacity than in our model) the number of requests goes up to 1000 per hour in the morning. So given the capacity and number of bikes used in our model, these numbers seem in line with those in the literature.

Besides the modulation of request and returns during day in the various locations, also the number of stations located in the various parts of the city is of importance. Recall that we defined two configurations in §2. We furthermore assume that initially each station of type P is filled with 3 bikes, those of type M with 5 bikes, and those of type C with 7 bikes, leading to an average number of $s = \frac{3\times100 \,+\, 5\times150 \,+\, 80\times7}{330} = \frac{1610}{330} \approx 4.8788$ bikes per station.

Figure 5 shows the variation of the fraction of stations of type P with 0, 1, 2, 3, 4 and 5 bikes parked, respectively, over a time horizon of 300 time units, comparing the average of 500 stochastic simulation runs in Fig. 5(left) with a mean field approximation of the model in Fig. 5(right). The results show a good correspondence. For the precise conditions of the model under which this happens we refer to [21]. In the case of the BSS model these conditions are satisfied as long as the population sizes used in the model are sufficiently large[4]. Recall that mean field models provide an approximative result on the *average* behaviour of a system. Individual simulations, in particular of models with relatively small populations, may show varying behaviour due to stochastic variability.

5 Examples of Performance Features of BSS

This section contains exemplary performance analyses over the BSS model of §4.

5.1 Normalised Activity/Bicycle Data

One way to characterise usage patterns of stations is by their average filling degree over time. Froehlich et al. call this normalised activity/bicycle (NAB)

[3] The input language of the FlyFast model-checker used for the analysis is described in [18, 20] and consists of a simple high level language for Markov population models from which the mathematical structures on which the model-checking algorithms are based are automatically generated in an on-the-fly fashion.

[4] In the simulation we used population sizes multiplied by a factor 10, so considering 3300 stations instead of 330. This has nothing to do with the factor 10 scaling in §3.

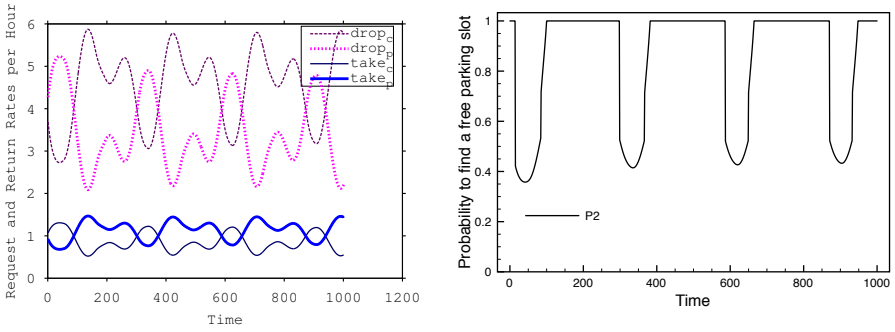

Fig. 4. (left) Request and return rates per hour for stations of types P and C for three cycles of one day (night omitted) and (right) time dependent probability of property P2 for a model with $\lambda = 1$ and $\mu = 4$ per hour and of type configuration 1, i.e. with $N = 330$ stations, of which 100 with capacity 5 in P and each holding 3 bikes initially; 150 with capacity 10 in M and 5 bikes; 80 with capacity 15 in C and 7 bikes (cf. §5.2).

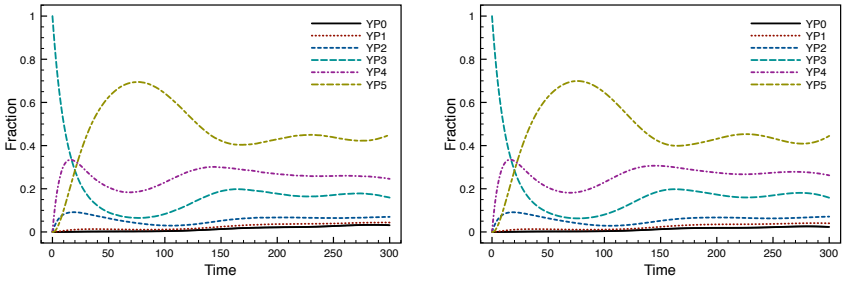

Fig. 5. Fractions of stations of type P with 0 to 5 bikes, indicated respectively by $YP0$ to $YP5$: (left) simulation average over 500 runs and (right) mean field approximation.

data [13]. These filling degrees are usually collected from data observed in real systems, but we obtain them from the mean field analysis of the model as follows:

$$NAB = \frac{\sum_{k=1}^{K_i} yi_k k}{K_i} \times \frac{N}{N_i}$$

where $i \in \{\mathrm{P}, \mathrm{M}, \mathrm{C}\}$ denotes the type of stations, K_i the capacity of stations of type i, yi_k the fraction of type-i stations with k bikes parked in it of the total number of stations N and N_i the number of stations of type i. The multiplication with $\frac{N}{N_i}$ is because we are interested in the filling degree of type-i stations and not in the total number of stations N, whereas yi_k gives the fraction with respect to the *total* number of stations N. For instance, let $YP1, \ldots, YP5$ denote the fractions of the total number of stations N of stations of type P with 1, 2, 3, 4 and 5 bikes, respectively, where $K_p = 5$ is the capacity of such stations. Then $BP = YP1 \times N + 2 \times YP2 \times N + 3 \times YP3 \times N + 4 \times YP4 \times N + 5 \times YP5 \times N$ is the total number of bikes parked in stations of type P. This gives on average

$ABP = BP/N_P$ bikes parked per station, where N_P is the number of stations of type P, and ABP/K_p is the filling degree of such a station of type P.

We now compare the NAB of configurations 1 and 2 defined in §2. The NAB of configuration 1, for an average request rate of one bike per station per hour and an average trip duration of 15 minutes, is shown in Fig. 6(left) for a period of approximately 3 days of 12 hours each (night time is omitted). We clearly see the pattern of fluctuations of request rates also appear in the fluctuations in the filling degrees of the stations. Furthermore, a decrease in filling degree in the stations in the periphery in the morning corresponds to a slightly delayed increase in filling degree in the stations in the centre. The delay is due to the time it takes to cycle from the periphery to the centre. The reverse is happening in the evening when commuters go back home. Interestingly, the stations in the middle area show a more stable filling degree than those in the centre and in the periphery. Both patterns have been observed also in real data from the BSS in Barcelona [13].

The NAB of configuration 2 is shown in Fig. 6(right). In order to compare the configurations under the same assumptions on bike requests we have to correct for this fact in the model of configuration 2 because the request of bikes in the model are expressed per station. So we divide the requests (and returns) in the periphery for each station by 2, and double the requests (and returns) per station for those in the centre. Under these assumptions, the figure shows that in configuration 1 the stations in the periphery are in general less empty, but that the filling degree of those in the centre now fluctuates much more. Larger fluctuations might require more frequent interventions from BSS operators to rebalance the distribution of bikes. The situation in the periphery seems improved, but perhaps there are now more stations and bikes than really necessary. Many other BSS configurations and situations can be investigated. For example the effect on the average filling degree of an increase in requests for bikes when the density and number of stations and bikes is increased or the effect on resource usage and related implications on user satisfaction and costs of different distributions of stations over the various areas of the city. We limited ourselves here to only some examples due to space limitations.

Mean field based models provide an approximation of the average behaviour in a computationally efficient way compared to a simulation based approach. Moreover, the analysis time is independent of the number of stations in the BSS, meaning that the approach easily scales to BSS with a size in the order of those found in Chinese cities. The approximations are actually better for BSS with more stations because this reduces the stochastic variability of system behaviour. Just as an indication of the evaluation times we would like to mention that the analysis in Fig. 6 takes less than a second to perform.

5.2 Properties of Individual Stations in the Context of the System

In the previous section, we analysed performance aspects of the BSS as a whole. However, also the behaviour of individual stations in the context of the overall BSS can provide valuable insight into the level of service that is provided to its

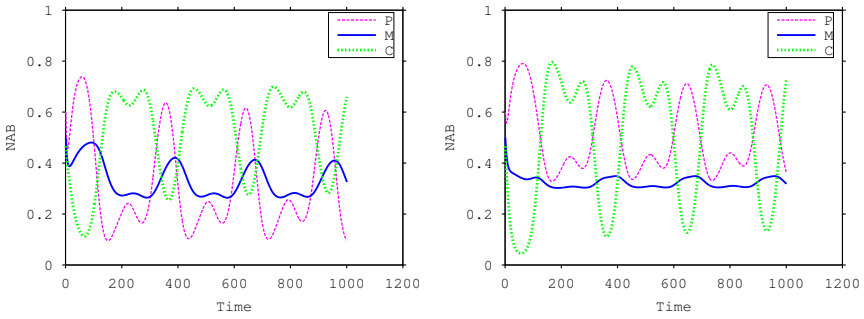

Fig. 6. NAB for station types P (pink dashes), M (blue line) and C (green dots), with $\lambda = 1$ and $\mu = 4$ per hour: (left) for configuration 1 (see §2), with P-stations each holding 3 bikes initially; M-stations 5 bikes; and C-stations 7 bikes, and (right) for configuration 2 (see §2), with the same initial distribution of bikes over stations.

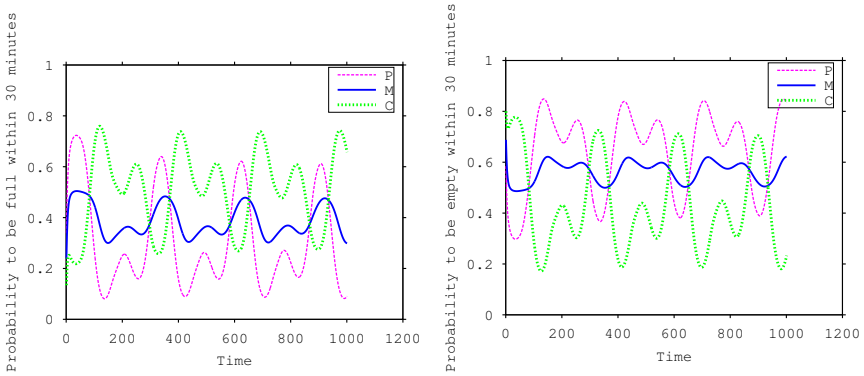

Fig. 7. Time dependent probability to be full (left) or empty (right) within 30 minutes for the station types P (pink dashes), M (blue line) and C (green dots), with $\lambda = 1$ and $\mu = 4$ per hour, for configuration 1.

customers. We provide some examples of properties of individual stations that may be relevant and that have been obtained by applying the mean field model checker FlyFast, developed by some of the authors of the current paper [18, 20].

Probability of a station getting full or empty within the time of arrival An example of a property of an individual station, operating in the context of the global system, is the question of how likely it is that a station gets full (or empty, respectively) within approximately 30 minutes (corresponding to 15 time steps) corresponding to the usual maximal free allowance in a BSS, i.e. no fee is due for trips shorter than 30 minutes. In other words, this property could provide some insight in the likelihood that a station selected as destination by a user gets full between the time the user departs and the maximal free allowance. For a station in the periphery with a capacity of 5 parking slots, this property

can be formalised in PCTL [14] as

$$\mathcal{P}_{=?}(\ \mathrm{tt}\ \mathcal{U}^{\leq 15}\ \ YP5\ ) \tag{P1}$$

where $YP5$ a periphery station that is full, i.e. having 5 bikes parked. A similar property can be formulated for a station getting empty replacing $YP5$ by $YP0$. Figure 7(left) shows the results for a selected destination station in state $YP3$, so there are still $5-3=2$ parking slots free at the destination at the time the user departs. Note that this probability depends on the time at which property P1 is evaluated because of the evolution of the system over time and that the figure shows its evaluation at times ranging from 0 to 1000 time units. The interested reader is referred to [20] for further details on the mean field model-checking algorithms for such properties. Similarly, Fig. 7(right) shows the probabilities for a destination station to get *empty* within 30 minutes when it currently has two bikes parked in its slots.

A more sophisticated nested property, P2, considers the situation in which, within 30 minutes a destination station in the periphery: (i) gets full, but then, with high probability ($\geq$0.99) has a free slot within 6 minutes, or (ii) does not get full, but then, with low probability ($\leq$0.01) gets full within 6 minutes.

This property can be formalised in PCTL as follows:

$$\begin{aligned}\mathcal{P}_{=?}(\ \mathrm{tt}\ \mathcal{U}^{\leq 15}\ \ ((YP5\ \wedge\ \mathcal{P}_{\geq 0.99}(YP5\ \mathcal{U}^{\leq 3}\ \neg YP5))\ \vee \\ (\neg YP5\ \wedge\ \mathcal{P}_{\leq 0.01}(\neg YP5\ \mathcal{U}^{\leq 3}\ \ YP5)))\ )\end{aligned} \tag{P2}$$

The result of checking P2 against a BSS with an initial state of the selected individual destination station and which has two empty parking slots available (i.e. the local state is $YP3$) is shown in Fig. 4(right). Property P2 is evaluated at times ranging from 0 to 1000 time units. We clearly see that there are periods during which the likelihood to find a free slot at the destination station in the periphery is 1, which thus indicates a very good level of service indeed. But there are also periods during which this likelihood reduces considerably. These are peak times when the level of service for what concerns finding free slots to park a bike in the periphery drops to a much lower level. Note that P2 formalises the probability with which a user can expect to find a free parking slot at a station in the periphery within 30 minutes from departure. Similar properties can be verified for stations in other parts of the city. Property P2 took just a few seconds to be analysed, which is a very good result given the size of the system and the nested form of the formula.

6 Discussion and Future Work

In this paper, we presented two approaches for the evaluation of BSS designs by means of automated tools in a rather orthogonal way. We performed variability analysis (by multi-objective optimisation) on system configurations and performance analysis (by mean field model checking) on behavioural models.

We focussed on a simplistic comparison of two different BSS configurations with respect to their cost, user satisfaction and capacity (in terms of parking slots) to illustrate the ideas. In the future, we intend to strengthen the integration of these two approaches and use the outcome of performance analyses as input for variability modeling. Think, e.g., of measuring the user satisfaction for specific configurations and feeding the resulting values into the variability model. Clearly, the probability of finding an empty (or full) docking station, based on the capacity of a BSS configuration, may directly impact user satisfaction. The resulting combined use of analysis approaches can trigger the development of automatic decision support for the design of BSS as well as for successive adaptations and reconfigurations. This is not merely an utopia, since we showed that the use of recently developed scalable performance analysis methods (i.e. on-the-fly mean field model checking in discrete time) proposed in this paper easily scale to BSS of realistic size.

Acknowledgements We thank Michele Loreti, who is the developer of FlyFast. We also thank Bicincittà S.r.l. and Marco Bertini from PisaMo S.p.A. for generously sharing with us relevant information concerning bike-sharing systems in general and Pisa's *CicloPi* bike-sharing system in particular.

References

1. K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski. Clafer: unifying class and feature modeling. *Software & Systems Modeling*, 2015.
2. D. Batory. Feature Models, Grammars, and Propositional Formulas. In J. Obbink and K. Pohl, editors, *SPLC*, volume 3714 of *LNCS*, pages 7–20. Springer, 2005.
3. M. H. ter Beek, A. Fantechi, and S. Gnesi. Challenges in Modelling and Analyzing Quantitative Aspects of Bike-Sharing Systems. In T. Margaria and B. Steffen, editors, *ISoLA*, volume 8802 of *LNCS*, pages 351–367. Springer, 2014.
4. M. H. ter Beek, A. Fantechi, and S. Gnesi. Applying the Product Lines Paradigm to the Quantitative Analysis of Collective Adaptive Systems. In *Proceedings 19th International Conference on Software Product Lines (SPLC'15)*, pages 321–326. ACM, 2015.
5. M. H. ter Beek, S. Gnesi, and F. Mazzanti. Model Checking Value-Passing Modal Specifications. In A. Voronkov and I. Virbitskaite, editors, *PSI*, volume 8974 of *LNCS*, pages 304–319. Springer, 2015.
6. M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. Statistical Analysis of Probabilistic Models of Software Product Lines with Quantitative Constraints. In *Proceedings 19th Software Product Line Conference (SPLC'15)*, pages 11–15. ACM, 2015.
7. D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated Analysis of Feature Models 20 Years Later: a Literature Review. *Information Systems*, 35(6), 2010.
8. L. Bortolussi and J. Hillston. Fluid Model Checking. In M. Koutny and I. Ulidowski, editors, *CONCUR*, volume 7454 of *LNCS*, pages 333–347. Springer, 2012.
9. L. Bortolussi and J. Hillston. Fluid Model Checking. arXiv:1203.0920v2, 2013.
10. L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective system behaviour: A tutorial. *Performance Evaluation*, 70:317–349, 2013.

11. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In C. Ramakrishnan and J. Rehof, editors, *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
12. C. Fricker and N. Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, pages 1–31, 2014.
13. J. Froehlich, J. Neumann, and N. Oliver. Sensing and Predicting the Pulse of the City through Shared Bicycling. In C. Boutilier, editor, *Proceedings 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1420–1426. Morgan Kaufmann, 2009.
14. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
15. R. Hayden. *Scalable Performance Analysis of Massively Parallel Stochastic Systems.* PhD thesis, Imperial College London, April 2011.
16. J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings 2nd Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 33–43. IEEE, 2005.
17. A. Kolesnichenko, P.-T. de Boer, A. Remke, and B. R. Haverkort. A logic for model-checking mean-field models. In *Proceedings 43rd Conference on Dependable Systems and Networks (DSN'13)*, pages 1–12. IEEE, 2013.
18. D. Latella, M. Loreti, and M. Massink. On-the-fly Fast Mean-Field Model-Checking. In M. Abadi and A. Lluch-Lafuente, editors, *TGC*, volume 8358 of *LNCS*, pages 297–314. Springer, 2014.
19. D. Latella, M. Loreti, and M. Massink. On-the-fly fluid model checking via discrete time population models. In M. Beltrán, W. Knottenbelt, and J. Bradley, editors, *EPEW*, volume 9272 of *LNCS*, pages 193–207. Springer, 2015.
20. D. Latella, M. Loreti, and M. Massink. On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination. *Science of Computer Programming*, 110:23–50, 2015.
21. J.-Y. Le Boudec, D. McDonald, and J. Mundinger. A Generic Mean Field Convergence Result for Systems of Interacting Objects. In *Proceedings 4th Conference on the Quantitative Evaluation of Systems (QEST'07)*, pages 3–18. IEEE, 2007.
22. P. Midgley. Bicycle-Sharing Schemes: Enhancing Sustainable Mobility in Urban Areas. Background Paper CSD19/2011/BP8, Commission on Sustainable Development, United Nations Department of Economic and Social Affairs, May 2011.
23. A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. Visualization and exploration of optimal variants in product line engineering. In *Proceedings 17th International Software Product Line Conference (SPLC'13)*, pages 111–115. ACM, 2013.
24. P. Schobbens, P. Heymans, and J. Trigaux. Feature Diagrams: A Survey and a Formal Semantics. In *Proceedings 14th Conference on Requirements Engineering (RE'06)*, pages 136–145. IEEE, 2006.