Neural Predictive Monitoring for Collective Adaptive Systems

Francesca Cairoli[1(✉)], Nicola Paoletti[2,3], and Luca Bortolussi[1]

[1] Department of Mathematics and Geosciences, University of Trieste, Trieste, Italy
francesca.cairoli@units.it
[2] Department of Computer Science, Royal Holloway, University of London, London, UK
[3] Department of Informatics, King's College London, London, UK

Abstract. Reliable bike-sharing systems can lead to numerous environmental, economic and social benefits and therefore play a central role in the effective development of smart cities. Bike-sharing models deal with spatially distributed stations and interact with an unpredictable environment, the users. Monitoring the trustworthiness of such a collective system is of paramount importance to ensure a good quality of the delivered service, but this task can become computationally demanding due to the complexity of the model under study. Neural Predictive Monitoring (NPM) [5], a neural-network learning-based approach to predictive monitoring (PM) with statistical guarantees, can be employed to preemptively detect violations of a specific requirement – e.g. a station has no more bikes available or a station is full. The computational efficiency of NPM makes PM applicable at runtime even on embedded devices with limited computational power. The goal of this paper is to demonstrate the applicability of NPM on collective adaptive systems such as bike-sharing systems. In particular, we first analyze the performance of NPM over a collective system evolving deterministically. Then, following [7], we tackle a more realistic scenario, where sensors allow only for partial observability and where the system evolves in a stochastic fashion. We evaluate the approach on multiple bike sharing network topologies, obtaining highly accurate predictions and effective error detection rules.

1 Introduction

As the urban population grows there is an increasing need for innovative technologies that will allow cities to reach a good and sustainable quality of life with an equitable distribution of resources. Given a service and an urban framework, a developer should design a solution that guarantees the quality of the service delivered. Systems with decentralised and distributed designs, comprised

of many autonomous and interacting entities, are known as collective adaptive systems (CAS). In CAS, the user becomes part of the system design. Formal models provide detailed descriptions of the design choices of the system under study, whereas formal methods are used to analyse the effects of these choices on the safety and reliability of such a system. In general, the goal of formal verification is to check if the system satisfies a certain requirement, e.g. avoiding an undesirable or dangerous region of the state space. It is straightforward to frame verification as a reachability checking problem. Similarly, predictive monitoring (PM) focuses on the online analysis of such reachability. PM is preemptive, meaning that it aims at predicting, at runtime, if a future violation of the requirement can be reached from the current state of the system within a given time-bound. PM is invoked periodically and typically at high frequencies. Therefore, reachability needs to be determined rapidly so that the response is provided before the eventual failure occurs. Any solution to the PM problem involves a trade-off between the *accuracy* of the reachability prediction and its computational *efficiency*. The analysis must execute within strict real-time constraints and typically with limited hardware resources. Exact formal methods suffer well-known scalability issues. The general goal of this paper is monitoring the reliability of a CAS to ensure good quality of service. This is an extremely challenging task as the state space is typically large and spatially distributed. Moreover, having humans in the loop makes the behavioural analysis even more complex. In this paper, we present NPM-CAS, an adaptation of Neural Predictive Monitoring (NPM) [5] to CAS. NPM is a machine-learning-based approach to PM that builds on Conformal Predictions (CP) to provide highly accurate predictions in a highly efficient manner together with statistical guarantees over its predictions and a principled method for detecting potential prediction errors, which significantly enhances the reliability of PM estimates.

In summary, the main contributions of this paper are the following:

- We extend Neural Predictive Monitoring of [5] so that it can be applied to CAS, where the reliability of multiple agents can be synchronously monitored. The classification problem becomes a multi-output problem instead of a single-output one as in [5]: each output predicts the reliability of a single agent.
- We extend the CP framework to work under multiple-output classification problems so that we can have an agent-specific error detection rule and serve-specific statistical guarantees.
- We extend NPM to allow for stochastic dynamics. In [5] only deterministic and non-deterministic dynamics where considered. In such a scenario, the classification problem becomes a multi-class problem as states cannot be deterministically labelled as safe or unsafe.
- We evaluate the method on three different bike-sharing systems having network geometries with increasing complexities.

The paper is structured as follows. Section 2 describes the details of the bike-sharing model. Section 3 formally states the problems solved by NPM for a

generic CAS. Section 4 provides the theoretical background on CP, used to quantify the predictive uncertainty and to have statistical guarantees. The results of the experimental evaluation are then presented in Sect. 5.

2 Bike Sharing System

Bike-sharing systems (BSSs) are becoming important for urban transportation. In these systems, users arrive at a station, pick up a bike, use it for a while, and then return it to another station of their choice. Each station has a finite capacity and it cannot host more bikes than its capacity. Stochasticity is due to the randomness of user choices.

2.1 Model of the System

The BSS is modeled as a Markovian system with M stations and a fleet of N bikes. Each bike can be either locked at a station i, for $i \in \{1, \dots, M\}$, or in transit between two stations i and j, for $i, j \in \{1, \dots, M\}$ and $i \neq j$. Station i can host at most K_i bikes. We can frame this system as a population model where individuals, the bikes, can belong to M^2 different species: *stationary bikes* $S = \{S_1, \dots, S_M\}$ for bikes locked in a station and *transitioning bikes* $T = \{T_{i,j} \mid i \neq j\}$ for bikes moving between stations. The total number of species is thus $|S \cup T| = |S| + |T| = M^2$. The state of the system $x(t) \in \mathbb{N}^{M^2}$ counts the number of bikes in each species at time t. At each station, new users arrive at a rate λ_i, independently of the number of bikes present in that station. However, if the station has no bike available, the unhappy user leaves the system. Instead, if the station is not empty, the user picks up a bike at this station and joins the pool of riding users and the bikes move from a species in S to a species in T. We can summarize these events with a transition from S_i to $T_{i,j}$, given that the bike is moving from station i towards station j, happening with rate $\lambda_i \cdot \mathbb{I}(x_{S_i} > 0)$. The trip time between the two stations is exponentially distributed with mean $1/d_{ij}$, where d_{ij} is the distance between the two stations. After this time, the riding user wants to return the bike. If the destination has fewer than K_j bikes, the user returns the bike to this station and leaves the system. If the station has already K_j bikes, meaning if it is full, no more bikes can be returned. In this case, the user waits for a slot of that station to become available. This transition can be summarized as moving from species $T_{i,j}$ to species S_j with rate $x_{T_{ij}} \cdot \mathbb{I}(x_{S_j} < K_j)/d_{ij}$. The dynamics of the system is thus fully determined by $2M(M-1)$ reactions of the form:

$$
\begin{aligned}
R_1^{i,j}: \quad & S_i \longrightarrow T_{i,j} && \text{with rate } \lambda_i \mathbb{I}(x_{S_i} > 0), \\
R_2^{i,j}: \quad & T_{i,j} \longrightarrow S_j && \text{with rate } x_{S_j} \cdot \mathbb{I}(x_{T_{ij}} < K_i)/d_{ij}
\end{aligned}
$$

for every $i \neq j \in \{1, \dots M\}$. Let $\mathcal{R}$ denote the set of all possible reactions. The topology of the network of stations strongly influences the dynamics of the system. Figure 2 shows a very simple topology where all bikes are equidistant but each station can have a different arrival rate λ_i, meaning that some stations can be more popular than others, and a different capacity K_i.

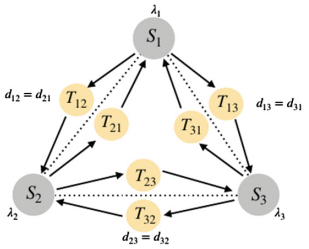

Fig. 1. BSS network with triangular topology: all bikes are equidistant but each station can have a different arrival rate λ_i and a different capacity K_i.

2.2 Dynamics of the System

The time evolution of the population model presented above can be described by the deterministic evolution of its probability mass. Let $\mathbb{P}_{x_0}(x(t) = x)$ denote the probability of finding the system in state x at time t given that it was in state x_0 at time t_0. This probability satisfies a system of ODEs known as Chemical Master Equation (CME):

$$\frac{\partial}{\partial t}\mathbb{P}_{x_0}(x(t) = x) = \sum_{j=1}^{|\mathcal{R}|} \left[f_{\mathsf{R}_j}(x - \nu_j)\mathbb{P}_{x_0}(x(t) = x - \nu_j) - f_{\mathsf{R}_j}(x)\mathbb{P}_{x_0}(x(t) = x) \right], \quad (1)$$

where ν_j is the update vector associated with reaction $\mathsf{R}_j \in \mathcal{R}$. The equation above is the Kolmogorov equation for a population process, considering the inflow and outflow probability at time t for a state x. Since the CME is a system in general with countably many differential equations, its analytic or numeric solution is almost always infeasible.

In this regard, *approximate solutions* become the only viable approach to analyse the dynamics of a complex stochastic population model. In particular, we can resort either to stochastic simulation algorithms or to deterministic fluid approximations.

Gillespie Simulation. The Gillespie stochastic simulation algorithm (SSA) [8] generates trajectories that are exact realizations of the CME (Eq. (1)). Given a certain initial state, one can take a large number of samples (trajectories) that serves as an empirical estimate of the CME that can be used to extract information about the process via statistical methods. For example, one can consider an upper and a lower quantile and obtain a credible interval over the trajectory space (Fig. 2 (right)).

Mean Field Approximation. The deterministic approximation of a stochastic population model builds on the observation that stochastic fluctuations tend to average out as the population size grows larger, i.e. when the number of interacting individuals is very large. In particular, if the state variables are scaled, so that the state evolution is independent of the population size, the dynamics

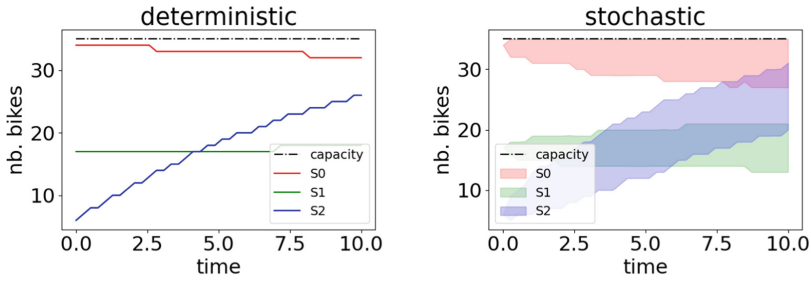

Fig. 2. Deterministic (left) and stochastic (right) trajectory for stationary bikes over the triangular topology of Fig. 1. In the stochastic version, we show the 95% credible interval over the trajectory space.

of the stochastic models is very similar to a deterministic one, described by an ODE, the well-known mean-field (MF) approximation [2,3,6,9]. Thus, in the BSS, as the number of bikes present in the system increases, the dynamics of the system tends to the following fluid ODE:

$$\frac{d\hat{x}}{dt} = \sum_{i \neq j} \frac{\nu_1^{i,j}}{M} \lambda_i \mathbb{I}(\hat{x}_{S_i} > 0) + \frac{\nu_2^{i,j}}{d_{ij}} \mathbb{I}(M \cdot \hat{x}_{S_j} < K_i), \tag{2}$$

where $\hat{x} = \frac{x}{M}$ is the scaled state and $\nu_1^{i,j}$ and $\nu_2^{i,j}$ are the original update vectors respectively for reaction $R_1^{i,j}$ and $R_2^{i,j}$. Therefore, MF trajectories have a deterministic evolution (Fig. 2 (left)). The formalism and dynamics of the Markovian population model, presented here specifically for a BSS, can be easily applied to a generic CAS.

3 Neural Predictive Monitoring for CAS

In this section, we describe the Neural Predictive Monitoring technique for a generic CAS evolving with either deterministic or stochastic dynamics.

3.1 Deterministic Dynamics

Consider the model $\mathcal{M}_{det}$ of a CAS with state space X evolving deterministically over discrete time with time steps of width Δt. Consider a temporal horizon H, the dynamics can be described by a function $F_{det} : X \to X^H$, mapping a state $x(t)$ to a trajectory $F_{det}(x(t)) = x(t_1) \cdots x(t_H)$, where $t_j := t + j\Delta t$. The measurement process is instead modeled by a deterministic function μ mapping a state x into its observable part y, $y = \mu(x)$. The CAS is composed of N different agents and we aim at monitoring the reliability of service for each of these N agents. For instance, in the BSS the agents are the N bike stations. Reliability is modeled by considering a region D of the state space that we want to avoid, referred to as the *unsafe* or *dangerous region*. Predictive monitoring of such a

system corresponds to deriving a function that approximates a given reachability specification for all the N agents, $\mathsf{Reach}(D, x, H) \in \{-1, 1\}^N$: given a state x and a set of unsafe states D, establish whether agent $i \in \{1, \ldots, N\}$ admits a trajectory starting from x that reaches D in a time H. If such a trajectory exists, $\mathsf{Reach}^{(i)}(D, x, H)$ evaluates to 1, -1 otherwise, where $\mathsf{Reach}^{(i)}(D, x, H)$ denotes the i-th component of $\mathsf{Reach}(D, x, H)$. The approximation is w.r.t. some given distribution of states, meaning that we can admit inaccurate reachability predictions if the state has zero probability.

Full Observability. We now illustrate the PM problem under the ideal assumption of full observability (FO).

Problem 1. (PM under FO). Given a CAS $(\mathcal{M}_{det}, F_{det})$ with N agents, state space X, a distribution $\mathcal{X}$ over X, a time bound H and set of unsafe states $D \subset X$, find a function $h : X \to \{-1, 1\}^N$ that minimizes the probability

$$Pr_{x \sim \mathcal{X}}\Big(h(x) \neq \mathsf{Reach}(D, x, H)\Big).$$

A state $x \in X$ is called *positive* for agent i w.r.t. a predictor h if the i-th component of $h(x)$ evaluates to 1, $h^{(i)}(x) = 1$. Otherwise, it is called *negative*.

As discussed in the next section, finding h, i.e., finding a function approximation with minimal error probability, can be solved as a supervised multi-output classification problem, provided that a reachability oracle is available for generating supervision data. The predictor h is indeed solving N classification problems at once. In [5] such a classification problem is solved using deep neural networks, which demonstrated the best performance across several other machine learning models.

Partial Observability. The problem above relies on the assumption that full knowledge about the state is available. However, in most practical applications, state information is only partial. Under partial observability (PO), we only have access to a sequence of past observations $\bar{y}_t = (y_{t-H_p}, \ldots, y_t)$ which can be generated by applying the observation function μ to the *unknown* state sequence $x_{t-H_p}, \ldots, x_t$, evolving according to F_{det}. In the following, we consider the distribution $\mathcal{Y}$ over Y^{H_p} of the observations sequences $\bar{y}_t = (y_{t-H_p}, \ldots, y_t)$ induced by state $x_{t-H_p} \sim \mathcal{X}$, dynamics given by F_{det} and observations given by μ.

Problem 2. (PM under PO). Given the system and reachability specification of Problem 1, find a function $g : Y^{H_p} \to \{-1, 1\}^N$ that minimizes

$$Pr_{\bar{y}_t \sim \mathcal{Y}}\Big(g(\bar{y}_t) \neq \mathsf{Reach}(D, x_t, H)\Big).$$

In other words, g should predict reachability values given in input only for a sequence of past observations, instead of $x(t)$, the true state at time t. In particular, we require a sequence of observations for the sake of identifiability. Indeed, for general non-linear systems, a single observation does not contain enough information to infer the state [7].

Error Detection. The predictors h and g provide approximate solutions and, as such, they can commit safety-critical prediction errors. Building on [4], we endow the predictive monitor of Problem 1 and 2 with an error detection criterion Rej. This criterion should be able to *preemptively* identify – and hence, reject – inputs where the prediction is likely to be erroneous (in which case Rej evaluates to 1, 0 otherwise). Rej should also be optimal in that it has minimal probability of errors in detection. The rationale behind Rej is that uncertain predictions are more likely to lead to prediction errors. Hence, rather than operating directly over inputs, $s \in \{x, \bar{y}\}$, the detector Rej receives in input a measure of predictive uncertainty of $f \in \{h, g\}$ about s.

Problem 3. (Uncertainty-based error detection). Given an approximate reachability predictor $f \in \{h, g\}$ for the system $(\mathcal{M}_{det}, F_{det})$ and reachability specification of Problem 1 and 2, and a measure of predictive uncertainty $u_f : S \to U^N$ over some uncertainty domain U and over a space $S \in \{X, Y^{H_p}\}$ with distribution $\mathcal{S} \in \{\mathcal{X}, \mathcal{Y}\}$, find an optimal error detection rule, $Rej_f : U \to \{0,1\}^N$, that minimizes the probability

$$Pr_{s_t \sim \mathcal{S}}\Big(\mathbf{1}\big(f^{(j)}(s_t) \neq \mathsf{Reach}^{(j)}(D, s_t, H)\big) \neq Rej_f^{(j)}(u_f^{(j)}(s_t)) \mid j \in \{1, \dots N\}\Big).$$

In the above problem, we consider all kinds of prediction errors, but the definition and approach could be easily adapted to focus on the detection of a specific type of error, e.g. on false negatives (the most problematic errors from a safety-critical viewpoint).

Statistical Guarantees. The general goal of Problems 1, 2 and 3 is to minimize the risk of making mistakes in predicting reachability and in predicting prediction errors, respectively. We are also interested in establishing probabilistic guarantees on the expected error rate, in the form of prediction regions guaranteed to include the true reachability value with arbitrary probability.

Problem 4. (Probabilistic guarantees). Given the system and reachability specification of Problem 1 and 2 find, for every output $j \in \{1, \dots, N\}$, a function $\Gamma^{\epsilon}_{f^{(j)}} : S \to 2^{\{-1,1\}}$, mapping an input s_t into a prediction region for the corresponding reachability value, i.e., a region that satisfies, for any error probability level $\epsilon \in (0, 1)$, the *validity* property below

$$Pr_{s_t \sim \mathcal{S}}\Big(\mathsf{Reach}^{(j)}(D, s_t, H) \in \Gamma^{\epsilon}_{f^{(j)}}(s_t)\Big) \geq 1 - \epsilon.$$

Among the maps that satisfy validity, we seek the most *efficient* one, meaning the one with the smallest, i.e. less conservative, prediction regions.

3.2 Stochastic Dynamics

We now consider a CAS $\mathcal{M}_{stoch}$ evolving stochastically over a state space X and over discrete time. Function $F_{stoch} : X \to \mathcal{X}^H$ describes the dynamics, over a temporal horizon H, mapping a state $x(t)$ to a random variable over the trajectory space X^H, $F_{stoch}(x(t)) = \mathbf{x}(t_1) \cdots \mathbf{x}(t_H)$. A sample

$\xi \sim F_{stoch}(x(t))$ is nothing but a trajectory over X^H. The distribution of $F_{stoch}(x(t))$ can be empirically approximated by taking a large number, P, of samples, $\bar{\xi} := (\xi_1, \ldots, \xi_P) \sim F_{stoch}(x(t))$. We evaluate the safety of state x through a function $\mathsf{StochReach}(D, x, H) \in \{-1, 0, 1\}^N$, which outputs 1 if the trajectories starting from x eventually reach D with probability higher than $(1 - \alpha)$ (x safe), -1 if D is reached with probability below α (x unsafe), 0 otherwise. These probabilities can be derived with Monte-Carlo or numerical probabilistic model checking techniques [13,14]. Predictive monitoring of such a stochastic system $(\mathcal{M}_{stoch}, F_{stoch})$ corresponds to deriving a function that approximates $\mathsf{StochReach}(D, x, H)$ w.r.t. some given distribution for x.

Problem 5. (Stochastic PM). Given an system $(\mathcal{M}_{stoch}, F_{stoch})$ with state space X, a distribution $\mathcal{X}$ over X, a time bound H and set of unsafe states $D \subset X$, find a function $h_s : X \to \{-1, 0, 1\}^N$ that minimizes the probability

$$Pr_{x \sim \mathcal{X}}\Big(h_s(x) \neq \mathsf{StochReach}(D, x, H)\Big)$$

The uncertainty-based error detection rule of Problem 3 and the statistical guarantees of Problem 4 are defined very similarly in the stochastic scenario. The main differences are that the predictive errors of Problem 3 are now defined as $\mathbf{1}\big(h_s^{(i)}(x) \neq \mathsf{StochReach}^{(i)}(D, x, H)\big)$ for $i \in \{1, \ldots, N\}$ and the predictive region $\Gamma^{\varepsilon}_{h_s}$ of Problem 4 is a function $\Gamma^{\varepsilon}_{h_s} : X \to 2^{\{-1,0,1\}^N}$.

3.3 Predictive Monitoring for BSS

Given a BSS modeled as in Sect. 2, we aim at predicting, from the current state of the system, if a station i is about to get full, $x_{S_i} = K_i$, or if it is soon going to be empty, $x_{S_i} = 0$. The goal of predictive monitoring is to access this information in advance, so that one can try to prevent undesirable events from happening, e.g. by using a truck to transport bikes from one station to another.

Different scenarios, with increasing complexity, can be considered. Each station constantly monitors the number of bikes available, so that measuring the number of stationary bikes is straightforward. On the other hand, when a bike is in transit, we have no exact information about where it is directed to.

We start by considering, a simplified scenario where we assume to have complete knowledge about the state of each bike. We then consider the more realistic setting in which no information about the state of transitioning bikes is available, so we must predict the future reliability of the service only from partial information.

In terms of system dynamics, we start by predicting the service reliability based on the deterministic evolution of the system, using the MF approximation. In this scenario, a state x is labelled as unsafe if the deterministic trajectory, starting from x, violates the requirement. It is labelled as safe otherwise.

We then move to a more complex scenario, where the stochasticity of the dynamics is preserved. Under these circumstances, a state x can be classified as safe, unsafe or risky. It is safe if both the lower and upper bound trajectories

satisfy the requirement, unsafe if they both violate it and risky if only one of the bounds violates the requirement. Notice that, potentially, one could extend this approach to an arbitrary number of quantiles by adding a label for each quantile.

By doing so we can create a synthetic dataset by randomly sampling a pool of n initial states, $x^1(t_0) \dots, x^n(t_0)$, and by letting the system evolve from each of these states for a time H. We then use the obtained trajectories to label them as safe, unsafe or risky. As we have N stations, we are going to consider N different requirements, each state thus is associated with N labels. In other words, we separately monitor the future reliability of each station from the current state of the system. The dataset can be summarized as

$$Z' = \{(s_i, \ell_i)\}_{i=1}^{n}, \tag{3}$$

where $s = x(t_0)$ in case of full observability (FO) and $s = x_S(t_0)$ in case of partial observability (PO), whereas $\ell_i = (\ell_i^1, \dots, \ell_i^N)$. If the dynamics is deterministic $\ell_i^j \in \{safe, unsafe\}$. If the dynamics is stochastic $\ell_i^j \in \{safe, risky, unsafe\}$.

4 Uncertainty Quantification and Statistical Guarantees

In the following, we provide the necessary background on Conformal Prediction (CP), the technique used to quantify the uncertainty and to obtain statistical guarantees over the predictions, the two ingredients needed to solve Problem 3 and Problem 4 in both the deterministic and the stochastic scenario. In the following, we provide an intuitive explanation; we refer the interested reader to [7] for a more detailed description of the procedure. The main difference is that CP is now addressing a multi-output multi-class classification problem rather than a simple binary classification problem.

4.1 Conformal Predictions for Multi-output and Multi-class Classification

Conformal Prediction (CP) [1] is a very general approach that associates measures of reliability to any traditional supervised learning problem. NPM for CAS, presented in Sect. 3, deals with multi-output classification problems (Problem 1, 2 and 5). We thus present the theoretical foundations of CP in relation to a generic multi-class multi-output classification problem.

Let S be the input space, $L = \{l^1, \dots, l^c\}$ be the set of labels (or classes), and define $Z = S \times L^N$, where N is the number of outputs. The classification model is represented as a function $f : S \to [0,1]^{c \times N}$ mapping inputs into N vectors of class likelihoods. For each output j, the class predicted by $f^{(j)}$ corresponds to the class with the highest likelihood. In the context of NPM for CAS, the input space S can be either X, under FO, or Y^{H_p}, under PO, whereas labels L indicates the possible reachability values ($c = 2$ in the deterministic version and $c = 3$ in the stochastic version), and $f \in \{h, g, h_s\}$ is the predictor.

For a generic input s_i, we denote with $\ell_i = (\ell_i^1, \ldots, \ell_i^N)$ the vector of true labels for s_i and with $\hat{\ell}_i$ the vector of labels predicted by f. Test points, whose true labels are unknown, are denoted by s_*. The main ingredients of CP are: a set of labelled examples $Z' \subseteq Z$, a classification model f trained on a subset of Z', a *nonconformity function* $ncm_{f^{(j)}} : S \times L \to \mathbb{R}$ and a statistical test. The nonconformity function $ncm_{f^{(j)}}(s_i, \ell_i^j)$ measures the "strangeness" of an example (s_i, ℓ_i^j), i.e., the deviation between the label ℓ_i^j and the corresponding prediction $f^{(j)}(s_i)$. For ease in the notation, let ncm_j denote the nonconformity function $ncm_{f^{(j)}}$.

CP Algorithm for Multi-output Classification. Given a set of examples $Z' \subseteq Z$, a test input $s_* \in S$, and a significance level $\varepsilon \in [0, 1]$, CP computes $\Gamma_{\varepsilon,*}$, a set of N prediction regions.

1. Divide Z' into a training set Z_t, and calibration set Z_c. Let $q = |Z_c|$ be the size of the calibration set.
2. Train a model f using Z_t.
3. Define a nonconformity function $ncm_j(s_i, \ell_i^j))$ for every output $j \in \{1, \ldots, N\}$.
4. Apply the nonconformity measure to each example in Z_c

$$A_c = \Big\{ \big\{ \alpha_{ij} = ncm_j(s_i, \ell_i^j) \mid j \in \{1, \ldots, c\} \quad \mid (s_i, \ell_i) \in Z_c \Big\}$$

   and, for each output $j \in \{1, \ldots N\}$, sort the nonconformity scores in descending order: $\alpha_{1j} \geq \cdots \geq \alpha_{qj}$.
5. For a test input s_*, compute the nonconformity scores w.r.t each output and w.r.t. each possible class:

$$A_* = \Big\{ \big\{ ncm_j(s_*, l^k) \mid k \in \{1, \ldots, c\} \quad \mid j \in \{1, \ldots, N\} \Big\}.$$

   Then, for $j \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, c\}$ compute the respective smoothed p-value

$$p_*^{(j,k)} = \frac{|\{z_i \in Z_c : A_c^{(i,j)} > A_*^{(j,k)}\}|}{q+1} + \theta \frac{|\{z_i \in Z_c : A_c^{(i,j)} = A_*^{(j,k)}\}| + 1}{q+1}, \quad (4)$$

   where $\theta \in \mathcal{U}[0, 1]$ is a tie-breaking random variable. Note that $p_*^{(j,k)}$ represents the portion of calibration examples whose j-th outputs are at least as nonconforming as the tentatively labelled test example (s_*, l^k).
6. Return a set of N prediction regions (one per output)

$$\Gamma_{\varepsilon,*} = \Big\{ \big\{ l^k \in L : p_*^{(j,k)} > \varepsilon \quad \mid j \in \{1, \ldots, N\} \Big\}. \quad (5)$$

   together with the p-values.

Note that in this approach, called inductive CP [11], steps 1–4 are performed only once, while Steps 5–6 are performed for every test point s_*.

Statistical Guarantees. The CP algorithm outputs *prediction regions*, instead of single point predictions: given a significance level $\varepsilon \in (0,1)$ and a test point s_*, its prediction region with respect to output j, $\Gamma^{(j)}_{\varepsilon,*} \subseteq L$, is a set of labels guaranteed to contain the true label ℓ^j_* with probability $1-\varepsilon$. The rationale is to use a statistical test, more precisely the Neyman-Pearson theory for hypothesis testing and confidence intervals [10], to check if (s_*, l^k) is particularly nonconforming compared to the calibration examples. The unknown distribution of nonconformity scores, referred to as $\mathcal{Q}$, is estimated by applying ncm_j to all calibration examples, set A_c (step 4). Then the scores A_* (step 5) are computed for every possible label and every output in order to test for the null hypothesis $A_* \sim \mathcal{Q}$. The null hypothesis is rejected if the p-values associated with A_* are smaller than the significance level ε. If a label l^k is rejected for output j, meaning if it appears unlikely that $ncm_j(s_*, l^k) \sim \mathcal{Q}^{(j)}$, we do not include this label in $\Gamma^{(j)}_{\varepsilon,*}$. Therefore, given ε, the prediction region for each output contains only those labels for which we could not reject the null hypothesis. In the stochastic setting, our approach guarantees that there is a probability (w.r.t. sampling) of $1-\varepsilon$ that our prediction region includes the correct StochReach value, i.e., whether the (stochastic) system will reach D with probability above $1-\alpha$, below α or in-between.

Nonconformity Function. A nonconformity function is well-defined if it assigns low scores to correct predictions and high scores to wrong predictions. In multi-output classification problems, a natural choice for ncm_j, based on the underlying model f, is

$$ncm_j(s_i, l^k) = 1 - P_{f^{(j)}}(l^k|s_i), \tag{6}$$

where $P_{f^{(j)}}(l^k|s_i)$ is the likelihood of class l^k for output j when the model f is applied on s_i. If $f^{(j)}$ correctly predicts ℓ^j_i for input s_i, the corresponding likelihood $P_{f^{(j)}}(\ell^j_i|s_i)$ is high (the highest among all classes) and the resulting nonconformity score is low. The opposite holds when $f^{(j)}$ does not predict ℓ^j_i. The nonconformity measure chosen for our experiments, Eq. 6, preserves the ordering of the class likelihoods predicted by $f^{(j)}$ for every output j.

Confidence and Credibility. Observe that, for significance levels $\varepsilon_1 \geq \varepsilon_2$, the corresponding prediction regions are such that $\Gamma_{\varepsilon_1} \subseteq \Gamma_{\varepsilon_2}$. It follows that, given an input s_* and an output j, if ε is lower than all its p-values, i.e. $\varepsilon < \min_{k=1,\ldots,c} p^{(j,k)}_*$, then the region $\Gamma^{(j)}_{\varepsilon,*}$ contains all the labels. As ε increases, fewer and fewer classes will have a p-value higher than ε. That is, the region shrinks as ε increases. In particular, $\Gamma^{(j)}_{\varepsilon,*}$ is empty when $\varepsilon \geq \max_{k=1,\ldots,c} p^{(j,k)}_*$.

The *confidence* of a point $s_* \in S$ w.r.t. output j, $1-\gamma^{(j)}_*$, measures how likely our prediction for s_* is compared to all other possible classifications (according to the calibration set). It is computed as one minus the smallest value of ε for which the conformal region is a single label, i.e. the second largest p-value γ_*:

$$1 - \gamma_*^{(j)} = \sup\{1 - \varepsilon : |\Gamma_{\varepsilon,*}^{(j)}| = 1\}.$$

Informally, the confidence of a prediction can be interpreted as the probability that a prediction corresponds to the true label.

The *credibility* w.r.t. output j, $\kappa_*^{(j)}$, indicates how suitable the training data are to classify that specific example. In practice, it is the smallest ε for which the prediction region is empty, i.e. the highest p-value according to the calibration set, which corresponds to the p-value of the predicted class:

$$\kappa_*^{(j)} = \inf\{\varepsilon : |\Gamma_{\varepsilon,*}^{(j)}| = 0\}.$$

Intuitively, credibility quantifies how likely a given state is to belong to the same distribution of the training data.

Uncertainty Function. The higher $1 - \gamma_*^{(j)}$ and $\kappa_*^{(j)}$ are, the more reliable the prediction $\hat{\ell}_*^j$ is. Therefore, our uncertainty-based rejection criterion relies on excluding points with low values of $1 - \gamma_*^{(j)}$ and $\kappa_*^{(j)}$. We stress, in particular, the following statistical guarantee: the probability that the true prediction for s_* is exactly $\hat{\ell}_*^j$ is at most $1 - \gamma_*^{(j)}$.

The uncertainty map u_f used to quantify the predictive uncertainty of a predictor f, introduced in Problem 3, is thus defined as

$$u_f(s_*) = \{(\gamma_*^{(j)}, \kappa_*^{(j)}) \mid j \in \{1, \ldots, N\}\}.$$

4.2 Uncertainty-Based Rejection Rule

Confidence and credibility measure how much a prediction can be trusted. Our goal is to leverage these two measures of uncertainty to identify a criterion to detect errors of the reachability predictor. The rationale is that every new input s is required to have values of confidence, $1 - \gamma$, and credibility, κ, sufficiently high in order for the classification to be accepted. However, determining optimal thresholds is a non-trivial task.

In order to automatically identify optimal thresholds, we proceed with an additional supervised learning approach. For this purpose, we introduce a *cross-validation strategy* to compute values of confidence and credibility, using Z_c as validation set. For every output j, the cross-validation strategy consists of removing the i-th score, $A_c^{(i,j)}$, in order to compute $\gamma_i^{(j)}$ and $\kappa_i^{(j)}$, i.e. the p-values at $s_i \in S_c$ w.r.t. output j, where $S_c = \{s \mid (s, \ell) \in Z_c\}$. In this way, we can compute confidence, $1 - \gamma^{(j)}$, and credibility, $\kappa^{(j)}$, for every point in the calibration set. For each output j, the points of the calibration set are then labelled with 1 or 0 depending on whether the classifier $f^{(j)}$ makes a prediction error over that calibration point or not. We then solve N binary classification problems by training N separate Support Vector Classifiers (SVCs) over the calibration set. These SVC optimally solve Problem 3.

Rejection Rule Refinement. As already observed in [4], predictors with very high accuracy result in over-conservative rejection rules. Intuitively, the reason is that since the number of non-zero calibration scores is limited, the p-values are less sensitive to changes in the nonconformity score. We here propose an output-specific refinement, meaning that, for each output j, we add to $A_c^{(j)}$ the points where $f^{(j)}$ predictions where rejected by $Rej^{(j)}$. By doing so, we add calibration points with informative non-zero calibration scores. However, in doing so we modify the data generation distribution of the calibration set. Thus the statistical guarantees, meaning the prediction regions, are computed w.r.t. the original calibration set.

Active Learning. The rejection rule defined above can be used as an uncertainty-based query strategy for an active learning approach, allowing the user to select the points where the predictor f is performing poorly and then add them to the training set to improve the performances of f.

5 Experiments

Implementation. The workflow can be divided into steps: (1) define the BSS models for different architectures, (2) generate the synthetic datasets Z' for both the deterministic (both under FO or PO) and the stochastic version, (3) train the NPM-CAS, (4) train the CP-based error detection rules and (5) evaluate NPM-CAS on a test set. The technique is fully implemented in Python[1]. In particular, PyTorch [12] is used to craft, train and evaluate the neural networks used to solve Problem 1, 2 and 5. The source code for all the experiments can be found at the following link: https://github.com/francescacairoli/CAS_NPM.git.

Datasets Generation. We set the number of bikes in the system to $M = 100$ for each configuration. The training set consists of $20K$ points, the calibration set consists of $10K$ points and the test set consists of $5K$ points. In the stochastic version, the upper and lower bounds are computed over samples of 200 trajectories per point. We define BSS networks with three different topologies with increasing dimensions, i.e. larger number of bike stations, and thus increasing complexity.

- *Triangular network* – Fig. 1: 3 bike stations, departure rates $\lambda_1 = 0.25$, $\lambda_2 = 0.2$, $\lambda_3 = 0.15$, distances are set to 10, station capacity K is set to 35 for each station.
- *Diamond network* – Fig. 3 *(left)*: 5 bike stations, departure rates $\lambda = 0.25$, $\lambda_a = 0.2$, $\lambda_b = 0.15$, distances are set to $a = 10$ and $b = 12$, station capacity K is set to 25 for each station.
- *Hexagon network* – Fig. 3 *(right)*: 7 bike stations, constant departure rates $\lambda = 0.2$, distances $d = 10$, constant station capacity $K = 20$.

[1] The experiments were performed on a computer with a CPU Intel x86, 24 cores and a 128 GB RAM and 15 GB of GPU Tesla V100.

Training Details. A grid-search approach has been used to find the best performing hyper-parameters under each configuration. In FO scenarios (both deterministic and stochastic), we use a feed-forward neural network composed of five layers with 50 neurons each, LeakyReLU activations and drop-out with probability 0.1. The last layer has a ReLU activation so to obtain positive likelihood scores that are fed into a cross-entropy loss. The training is performed for 1000 epochs over batches of size 256, using Adam optimizer with a learning rate of 0.0005.

In the PO scenario, we use one-dimensional convolutional neural networks with N channels, 128 filters of size 5, LeakyReLU activations and drop-out with probability 0.1. As before, the last layer has a ReLU activation and a cross-entropy loss. The training is performed for 400 epochs over batches of size 256, using Adam optimizer with a learning rate of 0.0005.

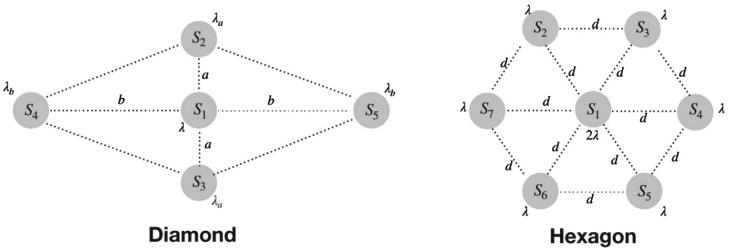

Fig. 3. Diamond and hexagon geometries.

Computational Costs. NPM-CAS is designed to work at runtime which translates into the need for high computational efficiency together with high reliability. The time needed to generate the dataset and to train both methods does not affect the runtime efficiency of the NPM-CAS, as it is performed only once (offline). Once trained, the time needed to analyse the reachability of the current sequence of observations is the time needed to evaluate the trained neural networks, which is almost negligible (in the order of microseconds on GPU). On the other hand, the time needed to quantify the uncertainty depends on the size of the calibration set. It is important to notice that the percentage of points rejected, meaning points with predictions estimated to be unreliable, affects considerably the runtime efficiency of the methods. Therefore, we seek a trade-off between accuracy and runtime efficiency. The training phase takes from 3 to 10 hours, whereas computing a single prediction takes less than 1 microseconds. Training each SVC takes from 1 to 10 s, whereas computing values of confidence and credibility for a single point takes from 0.01 to 0.08 s.

Measures of Performance. The measures used to quantify the overall performance of the NPM-CAS are: the *accuracy* of the reachability predictor, the *error detection rate* and the *rejection rate*. We seek high accuracies and detection rates without being overly conservative, meaning keeping a rejection rate

as low as possible. We also check if and when the statistical guarantees are met empirically, via values of coverage and efficiency. Efficiency is measured as the percentage of singletons in the prediction regions. We analyse and compare the performances of NPM-CAS on the three different BSS network configurations – the triangular, the diamond and the hexagon network. For each configuration, we compare the results of the deterministic version under the full observability assumption (Det-FO), the deterministic version under the partial observability assumption (Det-PO) and the stochastic version assuming full observability (Stoch).

5.1 Results

Table 1 summarizes the experimental results over the three different BSS topologies – triangular, diamond, and hexagon – and under the three different experimental settings – Det-FO, Det-PO and Stoch. The first column (**Acc.**) shows how NPM-CAS provides extremely accurate predictions, accuracies are always greater than 95%. In particular, in Det-FO the accuracy is always greater than 98%, in Det-PO it is always greater than 96% and in Stoch it is always greater than 95%.

Table 1. Average performances over the N bike stations. **Acc.** is the accuracy, **Rej.** and **det.** denote respectively the rejection and the error detection rates, whereas **Cov.** and **Eff.** respectively denote coverage and efficiency of the prediction regions (at level $\varepsilon = 0.05$).

Topology	Version	Initial results					Refinement	
		Acc.	**Rej.**	**Det.**	**Cov.**	**Eff.**	**Rej.**	**Det.**
Triangular	Det-FO	99.20	8.47	88.93	95.17	95.26	6.87	85.14
	Det-PO	98.24	11.93	96.87	95.35	95.75	9.40	94.48
	Stoch	97.55	13.05	92.02	95.13	96.07	12.03	91.51
Diamond	Det-FO	98.85	10.10	85.22	94.90	95.09	9.39	84.87
	Det-PO	96.21	19.71	92.07	94.99	97.67	14.27	89.86
	Stoch	96.38	18.12	87.05	95.16	97.92	15.93	86.21
Hexagon	Det-FO	98.02	13.12	84.09	95.13	95.81	12.28	81.37
	Det-PO	96.47	19.67	93.56	94.79	97.13	14.39	97.07
	Stoch	95.37	26.35	88.64	95.19	98.14	20.53	81.50

In column **Cov.** we observe how the CP prediction regions meet the statistical guarantees as the empirical coverage is close to the desired value of 95%. Moreover, the prediction regions show rather high efficiencies – see **Eff.** column. Efficiencies are always greater than 95%, meaning that there is no need for the CP predictor to be over-conservative in order to meet the guaranteed coverage.

Table 1 also shows the performances of the CP-based error detection rule before and after the refinement. We observe how the refinement of the error detection rule always reduces the rejection rate but it also results in slightly lower detection rates. In particular, on average over all the case studies, the rejection rate reduces from 17.41% to 13.70%, whereas the detection rate reduces from 88.78% to 83.32%. The reduction in the rejection rate is proportional to the reduction in the detection rate. This is most likely because the number of errors is rather small, resulting in highly unbalanced datasets, even after the refinement process, making the error detection phase extremely sensitive. Moreover, the refinement process changes the data generating distribution of the calibration set, meaning that the CP statistical guarantees no longer apply. Therefore, the refined solution is more efficient but less conservative than the original one and thus, its application has to be chosen wisely knowing the criticalities of the CAS at hand.

5.2 Discussion

Our results show great promise overall: the method attains very high accuracy levels (ranging from 95.37% to 99.2%), provides statistical guarantees, and effectively identifies and reject prediction errors. As expected, the performance is affected by the complexity and the dimensionality of the problem, i.e., deterministic scenarios with few agents outperform stochastic ones with a larger number of agents. As future work, we plan to systematically evaluate the scalability of our NPM approach with respect to the complexity and the dimensionality of the CAS at hand.

Moreover, our current approach handles the stochastic setting by partitioning the range of reachability probabilities into three regions, safe ($[0, \alpha]$), unsafe ($[1-\alpha, 1]$), and "indifference" $(\alpha, 1-\alpha)$, and predicting one of these segments (a classification problem). While the method can be easily extended to support arbitrary probability partitions, a next step will be to develop a quantitative approach that directly predicts reachability probabilities rather than categorical values (a regression problem). Another open problem is dealing with partial observability in stochastic systems where state identifiability remains an issue.

Finally, a natural extension would be to apply NPM-CAS to more realistic BSS topologies, e.g., the London BSS or any other cities that make such data available.

6 Conclusions

In this paper, we presented NPM-CAS an extension of the neural predictive monitoring technique to collective adaptive systems with variable complexity. In particular, NPM-CAS works both on CAS with deterministic dynamics, under either full or partial observability, and on CAS with stochastic dynamics. The technique is experimentally tested on a bike-sharing system with network topologies with increasing complexity. Results are promising, predictions are extremely

accurate and computationally efficient, thereby enabling the deployment of predictive monitoring at runtime on embedded devices with limited computational power.

References

1. Balasubramanian, V., Ho, S.S., Vovk, V.: Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications. Newnes, Oxford (2014)
2. Bortolussi, L.: Hybrid limits of continuous time Markov chains. In: 2011 Eighth International Conference on Quantitative Evaluation of Systems, pp. 3–12. IEEE (2011)
3. Bortolussi, L.: Hybrid behaviour of Markov population models. Inf. Comput. **247**, 37–86 (2016)
4. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 129–147. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9_8
5. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring and a comparison of frequentist and Bayesian approaches. Int. J. Softw. Tools Technol. Transf. **23**(4), 615–640 (2021)
6. Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective system behaviour: a tutorial. Perform. Eval. **70**(5), 317–349 (2013)
7. Cairoli, F., Bortolussi, L., Paoletti, N.: Neural predictive monitoring under partial observability. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 121–141. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88494-9_7
8. Gillespie, D.T., Petzold, L.: Numerical simulation for biochemical kinetics. In: Systems Modelling in Cellular Biology, pp. 331–354 (2006)
9. Le Boudec, J.Y., McDonald, D., Mundinger, J.: A generic mean field convergence result for systems of interacting objects. In: Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007), pp. 3–18. IEEE (2007)
10. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses. Springer, New York (2006). https://doi.org/10.1007/0-387-27605-X
11. Papadopoulos, H.: Inductive conformal prediction: theory and application to neural networks. In: Tools in Artificial Intelligence. InTech (2008)
12. Paszke, A., et al.: Automatic differentiation in PyTorch. In: NIPS-W (2017)
13. Younes, H.L., Simmons, R.G.: Statistical probabilistic model checking with a focus on time-bounded properties. Inf. Comput. **204**(9), 1368–1409 (2006)
14. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to simulink/stateflow verification. In: Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, pp. 243–252 (2010)