Dynamic Swarm Intelligence for Time Series Forecasting in the Presence of Concept Drift

Gustavo H. F. M. Oliveira[1*†], George G. Cabral[2], Augusto C. F. M. Oliveira[3], Marília G. F. M. Oliveira[2], Leandro L. Minku[4], Adriano L. I. Oliveira[5]

[1]Sistemas de Informação, UFAL, Brazil.
[2]Departamento de Informática e Estatística, UFRPE, Brazil.
[3]Engenharia de Software, UPE, Brazil.
[4]Centro de Informática, UFPE, Brazil.
[5]School of Computer Science, University of Birmingham, UK.

*Corresponding author(s). E-mail(s): gustavo.oliveira@penedo.ufal.br;
Contributing authors: george.gcabral@ufrpe.br; augusto.oliveira@upe.br; marilia.gfmo@gmail.com; L.L.Minku@cs.bham.ac.uk.; alio@cin.ufpe.br;
†L. Minku and A. Oliveira have jointly supervised this work.

Abstract

A time series is a sequence of numerical data arriving sequentially as a data stream. Typically, data streams' generating distributions change over time (i.e., concept drifts), potentially degrading predictive model performance. Despite extensive literature on time series forecasting and concept drift handling separately, few studies address both problems together. Recently, Particle Swarm Optimization (PSO) has been applied to dynamic optimization problems and training neural networks for forecasting. Building on these PSO applications, we propose the first approach using dynamic swarm intelligence to address concept drift in time series forecasting. Our approach includes three strategies: (i) using swarm models to detect concept drifts; (ii) managing swarm models to adapt to new concepts; and (iii) utilizing past swarm models for recurrent drifts. Experiments with six synthetic and four real-world datasets demonstrated that the proposed approach was competitive in concept drift detection, achieving the best forecasting errors compared to existing methods. This suggests dynamic swarm intelligence is a promising method for handling concept drift in time series forecasting.

Keywords: Swarm Intelligence, Time Series Forecasting, Concept Drift, Artificial Neural Network.

1 Introduction

A time series is a sequence of numerical observations recorded over time, arriving in the form of a data stream [1]. One of the major challenges in handling data streams consists in the change over time of their underlying distributions, causing past knowledge potentially no longer useful for accurately predicting future observations [2]. Such changes are known as concept drifts [3].

Most time series forecasting studies are based on off-line learning approaches (i.e., batch learning) and overlook the concept drift problem [4]. For real scenarios in the presence of concept drift over time, this can lead to degraded models[5]. Nonetheless, studies that take into account concept drift in time series forecasting typically rely on a concept drift detection mechanism that resets the off-line predictive models [4, 6, 7]. Although it helps, forecasting performance is still hindered by the delay in collecting recent observations for learning the new concept (i.e., an outdated model keeps in use until the new concept is learned by the new model.

The Concept Drift phenomenon has been extensively investigated for classification problems [3, 8], however, most existing approaches cannot be easily adapted for time series forecasting problems. Existing classification approaches that actively detect concept drift are typically based on the classification error [3] and cannot be applied to time series forecasting since the regression error is based on an approximation of the correct numeric value. Moreover, if these approaches were adapted to time series forecasting, since (typically) the models are reset after a drift detection, they would still struggle with the delay for learning the new concepts after the drift. Passive ensemble approaches that react to concept drift without relying on explicit drift detection methods have also been proposed [9]. However, they still suffer from delays in reacting to drifts case the new concept has not been seen before in the stream - given that they need to train many models in the same concept to obtain good modeling of the current data distribution.

Dynamic Optimization Problems (DOPs) [10] provide potential solutions to the delay for learning new concepts. This area is concerned with optimizing problems that evolve over time [11], often moving optimal solutions to different regions of the search space [12]. As such, it is related, but not the same as learning in the presence of concept drift. Dealing with concept drift means ensuring that the classifier learns the distribution of data whenever it changes over time. Although predictive model training can also be seen as an error optimization problem, there is no guarantee that a past concept can help find the optimum point of the new concept's error function. For this reason, dynamic optimization algorithms are not often used to deal with concept drift in real-world problems.

Population-based meta-heuristics used in DOPs could serve as inspiration for new approaches aimed at improving the time series forecasting quality in the presence of concept drift, given that there are diverse approaches, and few of them have been explored in this field. Strategies used to maintain diverse populations in such meta-heuristics [13, 14] may be used to train *diverse* forecasting models, i.e., models that learn different perspectives, and that may be used to adapt to a new concept more rapidly. When monitored for drift detection, some of these models may also detect concept drifts earlier or more reliably since they have different perspectives on the problem. Thus, increasing the chances of having a model more appropriate for the current situation, potentially accelerating and improving drift detection. Moreover, memory strategies from DOP literature [15] may be helpful for dealing with recurrent concepts, i.e., concepts that have been seen in the past.

This work proposes a method able to improve concept drift detection quality while reduces the delay for learning new concepts. The proposed method is called Swarm Intelligence for Series with Concept Drift (SISC). SISC enabled this work to have two contributions (i) the application of dynamic swarm intelligence in a real problem and not in an optimization function; and (ii) a proposal to handle concept drift in time series forecasting which it was little investigated. SISC adopts Particle Swarm Optimization (PSO) [16] as this type of meta-heuristic since it has been widely used in the literature for training artificial Neural Networks (NN), and has two possible strategies to adapt to concept drifts: (i) SISC-P, that employs swarm models to better react to concept drift; and (ii) SISC-M, that stores past best swarm models to be reused especially for recurring concepts. SISC investigation was guided be the following research questions (RQs):

RQ1) How could dynamic swarm intelligence improve concept drift detection in time series forecasting? In theory, the several particles of a dynamic swarm intelligence approach can be used to monitor a large area of the search space leading to a more reliable detection of a change in the function to be optimized. Inspired by that, we can monitor each NN trained by PSO using drift detectors, instead of only one, leading to a more reliable drift detection.

RQ2) To what extent using dynamic swarm intelligence improves the adaptation of time series forecasting models to different concepts? In dynamic swarm intelligence, when the function to be optimized changes, instead of reoptimize another swarm from scratch, an old swarm is reevaluated in the new concept and the best solutions are used as an initial swarm in order to accelerate the search for an acceptable solution in this new concept. Therefore, we could maintain many swarms of models to select the one that presents better forecasting performance to make the next predictions on the new concept instead of retraining an entirely new swarm.

RQ3) To what extent using past swarms of models accelerate adaptation to recurring concepts in time series forecasting? In dynamic swarm intelligence, long-term memory is used to store all the best particles found over time, with the aim of reusing them when a similar concept takes place. Thus, in time series forecasting, we could use the best solutions found so far to support the adaptation to recurrent concepts (i.e., the solutions with best performances can be selected as predictive models to the current concept).

This paper is further organized as follows. Section 2 presents the problem definition. Section 3 overviews related works. Section 4 presents the proposed approach (SISC), and partially answers RQ1, RQ2 and RQ3. Section 5 depicts the experimental setup. Section 6 presents the results and provides an analysis of the SISC performance in several aspects, complementing the answer for RQ1, RQ2 and RQ3. Section 7 concludes the paper and presents some directions for further related research.

2 Problem Definition

A time series can be defined as a potentially infinite sequence of observations $S = \{a_1, ..., a_t, ...\}, a_n \in \mathbb{R}$. At any time step $t-1$, a time series forecasting model must be able to provide a prediction $\hat{y}_t$, which is an approximation of $y_t = a_t$ [17].

Each predicted value $\hat{y}_t$ is a regression of an input $\vec{x}_t = \{a_{t-k}, ..., a_{t-1}\}$, i.e., the last k time series observations. This regression is performed by a model $f_{t-1} : \mathbb{R}^k \to \mathbb{R}$, trained with off-line supervised learning based on a set of previously received training examples of the format $(\vec{x}_{t'}, y_{t'}) = (a_{t'-k}, ..., a_{t-1'}, a_{t'})$. Each incoming example $(\vec{x}_t, y_t)$ comes from a specific joint probability distribution $P_t(\vec{x}, y) = P_t(y|\vec{x})P_t(\vec{x})$, where $P_t(\vec{x})$ is the unconditional probability density function and $P_t(y|\vec{x})$ is the conditional probability of the outputs given the inputs.

So, the challenge faced by supervised learning algorithms is that observations produced at distinct time steps t and $t + \Delta$ may come from different joint probability distributions $P_t(\vec{x}, y) \neq P_{t+\Delta}(\vec{x}, y)$, i.e., the time series may be affected by concept drift [18], making the model obsolete and potentially ineffective for forecasting future observations.

3 Related Work

Although time series suffer from concept drift, there are not many works investigating this phenomenon. Thus, to understand better concept drift we refer reader to classification surveys such as [8], [3], and [9]. But, to understand the existing time series forecasting approaches that deal with concept drift, we describe them in single-based and swarm-based approaches.

3.1 Single-Based Approaches

Cavalcante et al. [19], proposed two approaches to deal with concept drift. The approaches combine the neural network Extreme Learning Machine (ELM) [20] with the drift detectors ECDD [21], and DDM [22]. ELM is trained with off-line supervised learning with m training examples to make one-step ahead forecasts. ECDD and DDM are approaches that monitor the model's forecasting performance over time in an online way to detect variations to detect drifts. When the model's performance worsens and the drift threshold (parameter c in ECDD, and σ in DDM) is reached, m examples of the new concept are collected to retraining the model. The problem is waiting for these observations, because an obsolete model is used in the meantime, compromising the system forecasting performance. Although this work used drift detectors, their detection performance was not evaluated.

Cavalcante et al. [6] proposed the Feature Extraction Drift Detector (FEDD), a method for detecting concept drift designed for time series. FEDD create a feature vector with 16 statistical characteristics from a batch of m examples of the time series.

For each new observation, the feature vector is updated and the distance between the new and old vectors is calculated. If the distance between the new and old vectors increases by more than a c threshold, a concept drift is informed. In this work, only the performance of concept drift detection was evaluated, i.e., FEDD was not investigated as part of an approach for time series forecasting.

3.2 Swarm-Based Approaches

In a preliminary study of the current work [7], we applied ELM trained by the Self-Adaptive Particle Swarm Optimization (IDPSO) [23] algorithm to make predictions in time series, and also to detect drifts. The best ELM model generated by IDPSO was used to make predictions, and the each one generate were monitored by an ECDD for drift detection. Two drift detection rules were proposed: (i) IDPSO-ELM-B, which monitors the ELM models' average forecasting error; and (ii) IDPSO-ELM-S, which monitors all ELM models separately, and if all models agree that there is a drift, a drift is informed. IDPSO-ELM-B led to the best forecasting performance, but its ability to detect drifts was not so good. IDPSO-ELM-S led to the most reliable detections, but the delay of such detections resulted in a delay to retrain the model (wait for m training examples), which in turn impaired its forecasting performance since an obsolete model was used until that happened.

Rakitianskaia et al. [14] proposed an approach using swarm intelligence to deal with concept drift. The approach was proposed and evaluated for classification problems, even though it could potentially be used for time series forecasting. The authors proposed the Reinitialize Particle Swarm Optimization (RPSO) approach, which uses PSO to learn the weights of Multilayer Perceptrons (MLPs) based on a sliding window with m training examples. The average training accuracy of the model is monitored on-line to detect concept drift. If the model error increases σ standard deviations, the weights of the NN are reset and the NN is retrained on the current sliding window. Retraining on the current window are very costly and can hinder adaptation to drifts, as it may contain training examples that belong to the old concept. Moreover, the drift detection strategy is highly sensitive to outliers when the training error is low.

Overall, although there are some works in the literature using PSO for concept drift [7, 14] no existing work took advantage of *dynamic* swarm intelligence to improve concept drift handling. SISC uses *dynamic* swarm intelligence to overcome the aforementioned problems, aiming to improve concept drift detection, and time series forecasting.

4 SISC – Proposed Approach

This section introduces the proposed approach (SISC) which will support the answers of the aforementioned research questions. Fig. 1 depicts the overall operation of the proposed method which comprises four main blocks. First block contains an offline model trained by the IDPSO algorithm. The second one performs online forecasting on the incoming observations. The third block contains the drift detection method SISC-D, which monitors multiple ELM models (particles) in order to detect drifts in the time series. Finally, the fourth block provides two different mechanisms for tackling concept drift. These mechanisms estimate new models able to operate effectively in the new concept: SISC-P, which uses the current swarm to accelerate adaptation to

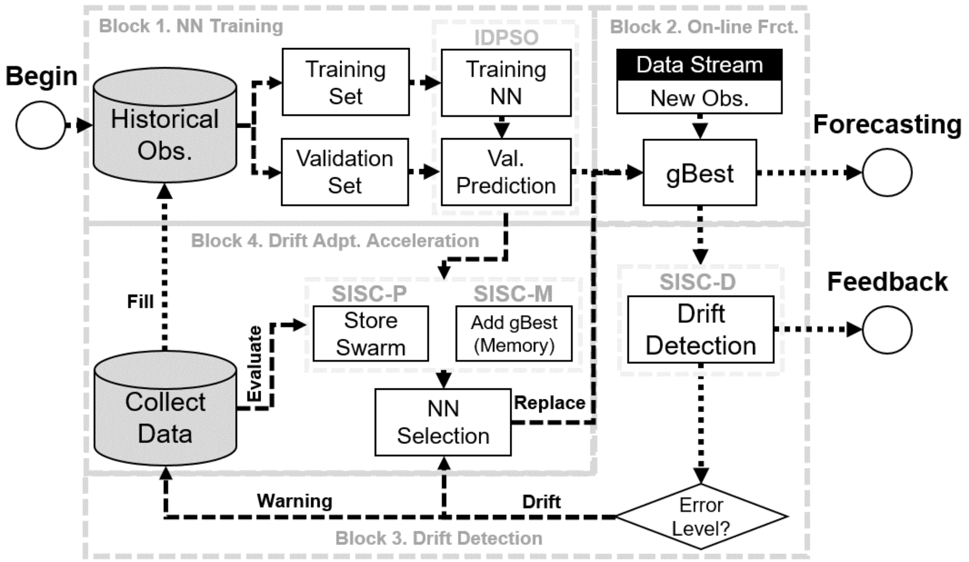


Figure 1: SISC general procedure. The choice between SISC-P and SISC-M in Block 4 is mutually exclusive.

drifts, and SISC-M, which uses best models from past swarms to accelerate adaptation to drifts. Both use previously trained models to adapt to the drift.

4.1 NN Training

As shown in the first block of Fig. 1, the predictive model is trained using offline supervised learning on a set of m training examples from the data stream. These examples are divided into 80% for training (τ) and 20% for validation (ν). The validation set is used to avoid model overfitting when training the predictive model, as will be explained later on this section.

As regression model, this work adopted the Extreme Learning Machine neural network [20] due to its fast training procedure and acceptable predictive performance [24] demonstrated in past studies involving swarm intelligence [7, 25]. In addition, the ELMs analytical training helps to guarantee a stable and fast training time - important characteristics to the SISC framework. As depicted in Fig. 2, the input weights and hidden biases of each ELM model are selected using a swarm intelligence approach, namely IDPSO [26], while the output weights are determined analytically by the ELM typical procedure through the Moore-Penrose (MP) generalized inverse. This arrangement introduces two benefits: (i) optimization of few parameters in the input layer, which results in a quick optimization, and (ii) fast training in the second layer with good generalization [20]. Other neural networks types, such as MLP, are not applicable in the SISC framework due to their higher training computational cost.

Alg. 1 depicts the training phase (i.e., learning the input weights of an ELM particle) using the IDPSO optimization algorithm [23].

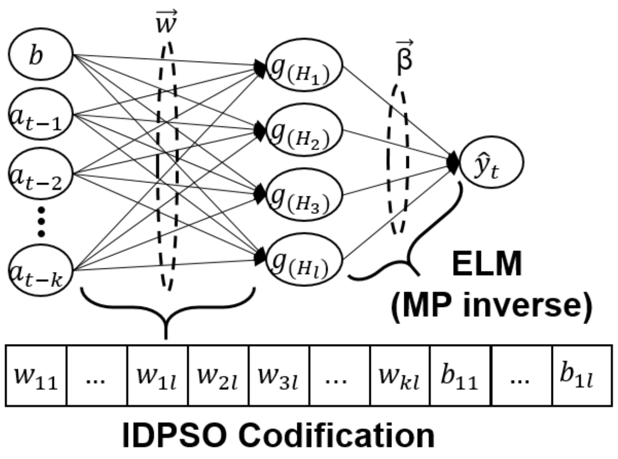


Figure 2: ELM network configuration. $\vec{w}$ represents the input weights, k represents the number of inputs, and l represents the number of hidden neurons. The input b stands for the bias. H_l is the induced local field provided to the non-linear activation function (g(·)). $\vec{\beta}$ represents the vector of output weights. In addition, the vector representing the IDPSO particle codification is also depicted.

In Alg. 1, lines 3 and 4 randomly assign the positions and velocities values for each particle. Each position and velocity consist of a D-dimensional vector, where: $D = (k+1) * l$; k is the number of inputs; and l is the number of hidden neurons. Line 9 indicates the analytical procedure aimed at computing the ELM's output weights ($\vec{\beta}$). $\vec{\beta}$, depicted in Eq. 2, is obtained by solving the following linear system: $B = H^{+}Y$. H^{+} refers to the Moore-Penrose generalized inverse of the matrix H. H is a matrix containing the products between the input instances and the weights of the input layer activated by a non-linear function $g(\cdot)$ (Eq. 1). Finally, Y represents the respective outputs for each input instance (Eq. 2).

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_l \cdot x_1 + b_l) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_m + b_1) & \cdots & g(w_l \cdot x_m + b_l) \end{bmatrix}_{m \times l} \quad (1)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_l^T \end{bmatrix}_{l \times m}, Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_m^T \end{bmatrix}_{l \times m} \quad (2)$$

During the training phase (lines 7 to 20), the particles move in a D-dimensional hyperspace seeking for the best fitness (line 10). The fitness of each particle is its

Algorithm 1 ELM weights optimization using IDPSO

```
1: Input: SwarmSize, #iterations, c1, c2, $\omega_{initial}$, $\omega_{end}$, $\tau$, $\nu$
2: for each i from 1 to SwarmSize do
3:     $v_j \leftarrow$ random()                              ▷ random velocities
4:     $p_j \leftarrow$ random()              ▷ random positions (weight and bias)
5:     Swarm.append($p_j$)
6: end for
7: for each $i$ in #iterations do
8:     for each Particle ($p_j$) in Swarm do
9:         $h_j \leftarrow$ ELM($p_j$,$\tau$)                    ▷ output weights training
10:        $f_j \leftarrow$ Fitness($p_j$, $h_j$, $\nu$)                          ▷ Eq. 3
11:        pBest $\leftarrow$ PBest($f_j$)
12:        gBest $\leftarrow$ GBest($f_j$)
13:        $v_j \leftarrow$ Velocity($p_j$, ..., $c2_i$)                       ▷ Eq. 4
14:        $p_j \leftarrow$ Position($p_j$, $v_j$)                          ▷ Eq. 5
15:        $\varphi_i \leftarrow$ Phi(pBest, gBest, $p_j$)                   ▷ Eq. 8
16:        $\omega_i \leftarrow$ Inertia($\varphi_i$, $\omega_{initial}$, $\omega_{end}$)               ▷ Eq. 6
17:        $c1_i \leftarrow$ PersonalFactor($\varphi_i$, $c1_i$)                ▷ Eq. 7
18:        $c2_i \leftarrow$ SocialFactor($\varphi_i$, $c2_i$)                  ▷ Eq. 7
19:    end for
20: end for
21: Return gBest, Swarm
```

Mean Absolute Error (MAE) computed on the validation set (ν), which can be easily obtained after determining the ELM's output weights. The fitness is calculated according to Eq. 3, where: o is the number of examples in ν; y_j^{ν} is the true output for a validation example j; and $\hat{y}_j^{\nu}$ is the predicted output for a validation example j.

$$fitness^{(\nu)} = \frac{1}{o}\sum_{j=1}^{o}\left|y_j^{(\nu)} - \hat{y}_j^{(\nu)}\right| \tag{3}$$

The solution with best fitness value (lowest MAE) found by each particle so far is stored in pBest (Line 10), while the best solution found so far by the whole swarm is stored in gBest (Line 11). Both pBest and gBest are used to update the velocity in line 13 using Eq. 4. Velocity represents the distance that each particle will fly in the weight and bias space, being used to update the position of each particle in line 14 using Eq. 5:

$$\begin{aligned}\mathbf{v}_j(i+1) = \omega * v_j(i) + c_1 \cdot rnd() \cdot (pBest_j - p_j(i)) \\ +c2 \cdot rnd() \cdot (gBest - p_j(i))\end{aligned} \tag{4}$$

$$p_j(i+1) = p_j(i) + v_j(i+1) \tag{5}$$

where ω, $c1$, $c2$, and rnd() represent inertia, control factors, and a random number in [0,1], respectively. These parameters are iteratively modified for each particle so that each one can decide for itself to change for a global or local search (lines 15, 16, 17, and 18) by Eqs. 6 to 8:

$$\omega(i) = \frac{\omega_{initial} - \omega_{end}}{1 + e^{\varphi(i)\cdot(i-((1+ln(\varphi(i)))\cdot K_{max})/\mu)}} + w_{end} \tag{6}$$

$$c_1(i) = c_1 \cdot \varphi^{-1}(i), c_2(i) = c_2 \cdot \varphi(i) \tag{7}$$

$$\varphi(t) = \frac{\|gBest - p_j(i-1)\|}{\|pBest_j - p_j(i-1)\|} \tag{8}$$

where $\omega_{initial}$ and ω_{end} represent the initial value and the final value of ω, respectively, K_{max} is the maximum number of iterations, i is the current iteration, and μ is an adjustment factor to ensure that ω will decrease.

4.2 On-line forecasting

Finally, as shown in the second block of Fig. 1, the best ELM model encountered during training (**gBest**) in terms of validation error (MAE) is retrieved and used to predict observation a_t of the time series based on the last k observations as input features, i.e, $\{a_{t-k}, ..., a_{t-1}\}$. Once the true target ($a_t$) is obtained, the error for this observation is computed and passed to the drift detector as explained in Section 4.3. It is important to mention that there is no incremental learning or on-line retraining in this part; there is only a batch retraining triggered when a drift is detected, where new m instances of the new concept are collected and used for this according to described in NN training.

4.3 SISC-D – Detecting Concept Drift with Swarm Intelligence in Time Series Forecasting

This section proposes an approach to improve concept drift detection in time series forecasting using swarm intelligence, answering RQ1. The motivation is that existing approaches based on drift detection usually monitor the error of a single forecasting method. If this single model was built as a result of a poor training process, drift detection may be inaccurate, generating several false alarms [7]. Dynamic swarm intelligence literature has investigated the use of several particles to monitor a larger area of the optimization search space to be more consistent in stating when the function to be optimized changed [15].

Based on that, we propose an extension of our preliminary IDPSO-ELM-S [7] (see Section 3). In IDPSO-ELM-S, each particle is an ELM that has its own drift detector (ECDD). This combination allowed to monitor the drift from various perspectives giving the best drift detections. Given that, we also adopt in this work the ECDD [21] as our drift detector. ECDD is a method that detects the drift based on the error obtained from the ELM model for each new observation. The error can be measured by the thresholds warning level (w), and drift level (c). Note that no prior drift information is needed, only the model error iteratively. Any drift detector could potentially be used as long as it is compatible with the time series forecasting error. In future works, we intend to investigate other types of more accurate drift detectors.

At this point, we want to study only the impact that monitoring of various models can have.

IDPSO-ELM-S has a rule that a drift is detected only if all ECDDs agreed that there is a drift. This rule achieved promising results in detecting drifts, but it greatly increases the detection delay, thus also increasing the delay in retraining. To prevent that, we propose a change to this rule. Given the result of each ECDD drift detector S $= (s_1, s_2, \cdots, s_{swarmSize})$, a drift is detected using $mode(S)$, meaning that the majority agree that there is a drift. So, this new rule is used in the third block of Fig. 1.

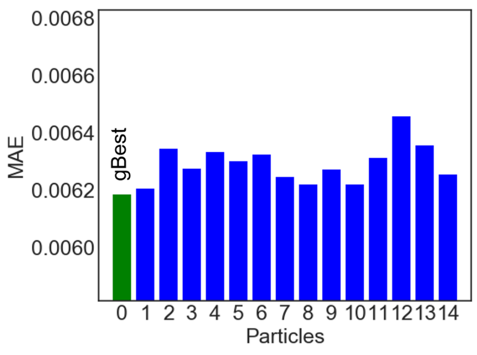
If most models have a *normal* error level, gBest remains untouched. If most models have a *warning* error level, the system begins to collect incoming observations. Each observation received is stored in a batch as shown in the gray box **Collect data** in Fig. 1. **Historical Observations** in the first block in Fig. 1 is only full filled with m new observations when most models have a *drift* error level, informing that it is time to retrain the model according to discussed in Section 4.1. While the batch is not filled, these observations are used in the **NN selection** method to accelerate the adaptation to the drift, according explained in Sections 4.4 and 4.5.

4.4 SISC-P – Reevaluating and Reusing Swarm Models to Improve Adaptation to New Time Series Concepts

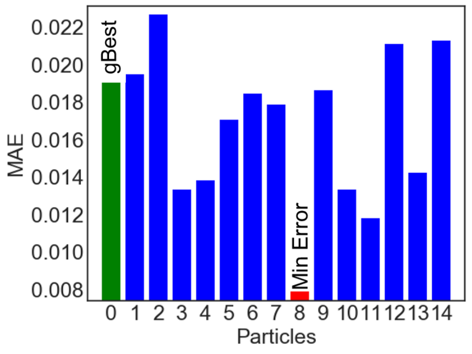
This section proposes our strategy SISC-P (Adaptation by Particles) to answer RQ2. SISC-P is in the fourth block of Fig. 1, and is triggered upon a SISC-D drift detection and remains active while the system is collecting the batch of m observations for retraining. SISC-P takes inspiration from the dynamic optimization literature as follows. Fouladgar et al. [13] create several swarms to find the best points of the dynamic optimization function. When the function changes, old solutions become obsolete, and a new search needs to be done. To avoid this, all swarms are reevaluated in the new function, and the best solutions are used as a starting point for a new search.

Since in this work a swarm of models also is adopted, and the particles represent different ELM models with distinct capabilities to adapt to a new concept, we developed a mechanism to recognize which ELM model (particle) of this swarm better fits the new concept. To illustrate the relevance of that in the context of time series forecasting, Fig. 3 depicts the performances of a swarm with 15 particles in two different moments: (i) when the concept remains with the same characteristics of the training phase (Fig. 3a), and (ii) when a new concept appears (Fig. 3b). In Fig. 3a, all particles perform similarly, i.e., the error standard deviation is low. In Fig. 3b, as a new concept appears, the errors of the particles vary a lot, i.e., the error standard deviation highly increases. In particular, particle 8 seems much better suited to the new concept than the other particles, including the gBest particle obtained by IDPSO for the previous concept indicating that it needs to be replaced.

Therefore, using this idea of replacing gBest with more suited particles, SISC-P applies the **NN Selection** mechanism when a drift takes place, as shown in Fig. 4. This method receives the instances collected after the drift (gray instances in Fig. 4), and returns the ELM model in the swarm that has the minimum MAE (Eq. 3) for these observations. This process is repeated for each new observation incrementally stored, which means that possibly a different model is chosen to predict the next observation.



(a) Before concept drift.



(b) Upon Concept Drift.

Figure 3: Particles error distribution on the linear time series with abrupt drifts. In Fig. (a), the green particle has the best performance in a recently trained concept. In Fig. (b), the red particle presents the best performance on the new concept.

Note that these models are not re-optimized for each new observation, they are only reevaluated using MAE. The models in the reevaluation are those previously trained during the training phase (Section 4.1).

4.5 SISC-M – Reusing Past Best Swarm Models to Deal with Recurring Concepts in Time Series Forecasting

This section proposes our strategy SISC-M (Adaptation by Memory) to answer RQ3. SISC-M is performed in the fourth block in Fig. 1 and is also triggered upon a SISC-D drift detection. SISC-M takes inspiration from the dynamic optimization as follows. Nasiri et al [15] uses a long-term memory to store all gBests found over time. In this work, a filter is used for the storage of different solutions to increase the chances of having at least one efficient solution when the drift takes place. Then, the memory solutions are reevaluated in the new function, and the best one is used. This strategy was shown to be efficient in dynamic optimization problems with recurring drifts.

So, SISC-M relies on the assumption that good past solutions that are not currently good might become good again in the future. The expectation is that old concepts may reappear and then can be treated by a pre-trained model. For this, whenever a swarm is trained, the gBest is stored in a **Memory** in the fourth block of Fig. 1.

Alg. 2 presents the procedure to add different ELM models. In lines 2–3, the new particle $gBest$ is stored in the $Memory$. If there is no available space anymore ($MaxMemorySize$), the nearest neighbor of $gBest$ in the memory is determined (line 5). The nearest neighbor $Neigh$ is the model with the smallest Euclidean distance to $gBest$ in the space of model input weights (Fig. 2). If $Neigh$ is very close to $gBest$, i.e., if the distance is smaller than $ExclusionThreshold$, $Neigh$ is replaced by $gBest$ (lines 5 to 11). The reason for this is to avoid redundancy and to supply the system with ELM models that have different perspectives.

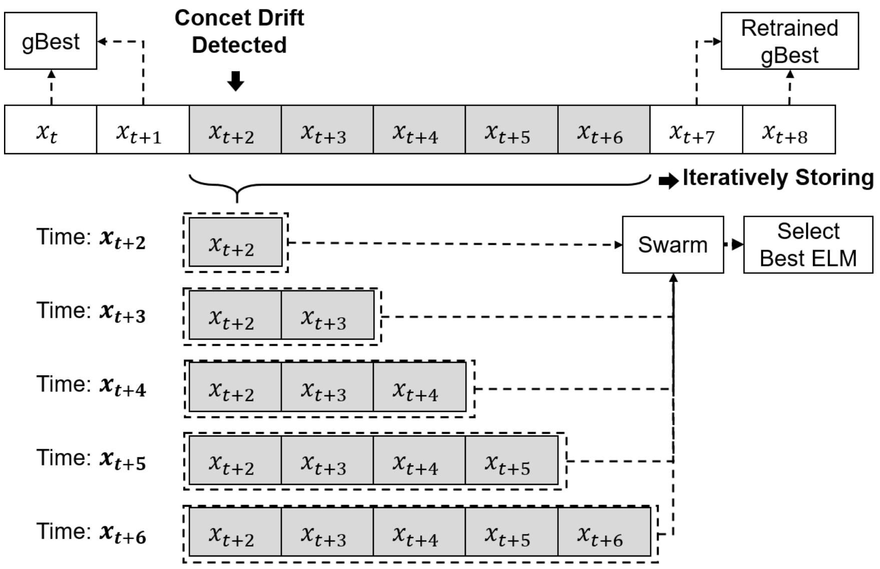


Figure 4: The procedure of NN estimation. In periods without drift (white boxes), the gBest is always applied to predict the next instances. In periods of drift (gray boxes), the method stores iteratively the drifted instances and selects for the new group of data, the ELM in the swarm with the best performance for these observations. The ELM selected is used to predict the new instance. This procedure remains active until a new batch is fulfilled for a retrain.

Algorithm 2 Storing Particles for Future Reuse

```
1: Input: gBest, $MaxMemorySize$, $ExclusionThreshold$
2: if (length(Memory) < $MaxMemorySize$) then
3:     Memory ← Add(gBest)
4: else
5:     Neigh ← NearestNeighbor(Memory, gBest)
6:     Dist ← Distance(Neigh, gBest)
7:     if (Dist < $ExclusionThreshold$) then
8:         Memory ← Remove(Neigh)
9:         Memory ← Add(gBest)
10:    end if
11: end if
```

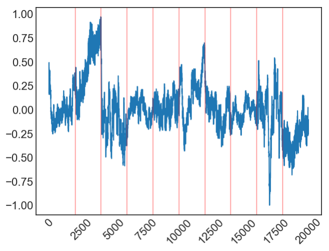
Finally, to handle with the concept drift problem, we use the generated *Memory* to get the new *gBest* in **NN Selection** (Fig. 4) similarly to discussed in Section 4.4.

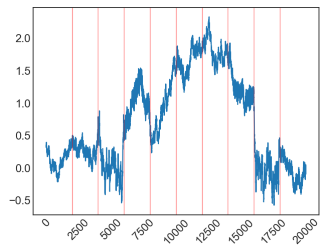
5 Experimental Setup

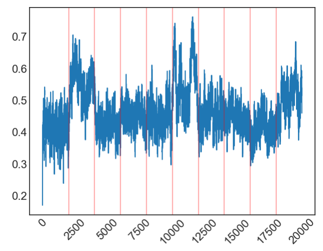
The datasets used in the experiments are presented in Section 5.1. The setup to investigate the drift detection capability of the SISC-D approach (RQ1) is presented in Section 5.5, and the setup to evaluate the forecasting ability obtained when using SISC-P (RQ2) and SISC-M (RQ3) is presented in Section 5.11.

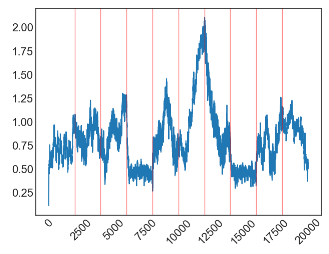
5.1 Data Sets

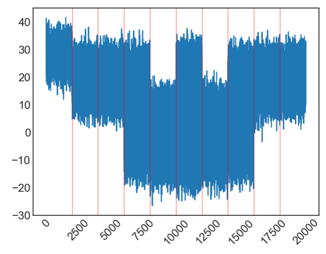
We used three groups of time series in our experiments: (i) artificial time series (5.2), (ii) real-world time series with known concept drifts (5.3), and (iii) real-world financial time series for which there is no prior knowledge on whether there is drift or not(5.4). To enable comparison, all series were normalized in the range of 0 to 1.


(a) Lin. Grad.


(b) Lin. Abt.


(c) Nl. Grad.


(d) Nl. Abt.


(e) Sazonal

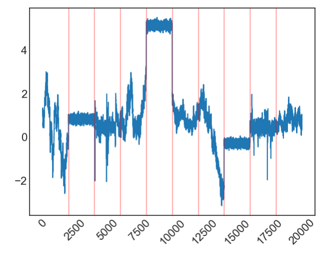

(f) Hybrid

Figure 5: Representation of each time series with known drifts. Figs 5a, 5b, 5c, 5d, 5e, and 5f are the artificial time series. Vertical red lines represent known concept drifts.

5.2 Artificial Time Series

As in the real world it is difficult to really know where the drift happen, we elaborated 6 types of artificial (synthetic) time series with known drifts to assess the model's ability to find these drifts, and they can be seen in Fig. 5.

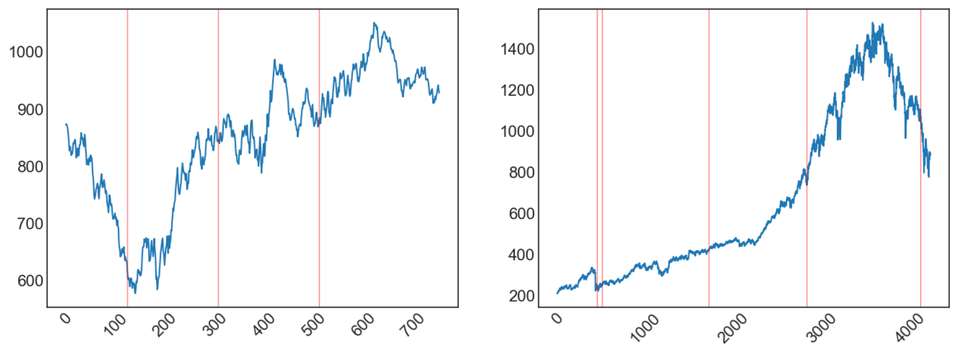
So, to guarantee the effectiveness and efficiency of the method in different variations of drifts, we generate thirty examples of each type, totaling 180 artificial times series. The performance of these series is always presented by the average obtained from the group. In general, each time series is composed of 10 concepts with 2,000 of observations. From the sixth concept, the first, second, third, and fourth concepts are repeated, making all of these series recurring. The procedure to generate and their characteristics are discussed below, and the parameters used are shown in Table 1.

The linear series were simulated using an autoregressive (AR) process, defined by: $x_t = a_1x_{(t-1)} + a_2x_{(t-2)} + .. + a_px_{(t-p)} + w_t$, where $a_1, .., a_p$ are parameters of the AR model and $w_t : t = 1, .., n$ is a white Gaussian noise; the variables w_t belong to a normal distribution.

The non-linear time series used in the experiments were generated by two smoothing models [6]: NL1: $x_t = [a_1x_{(t-1)} + a_2x_{(t-2)} + a_3x_{(t-3)} + a_4x_{(t-4)}] * [1 - exp(-10x_{t-1})]^{-1} + w_t$ and NL2: $x_t = a_1x_{(t-1)} + a_2x_{(t-2)} + \left[a_3x_{(t-1)} + a_4x_{(t-2)}\right] * \left[1 - exp\left(-10x_{t-1}\right)\right]^{-1} + w_t$.

The seasonal linear time series were simulated by a linear model using variables (s) indicating the number of periods in the series. This model contains s seasons defined by $x_t = m_t + \zeta_{(1+mod(t-1,s))} + w_t$ where m_t represents the trend, $\zeta_{(1+mod(t-1,s))}$ the seasonal factor, and w_t the Gaussian noise.

Gradual drifts refers to the transition phase where the probability of observations from the old concept decreases while the probability of the new one increases [18]. Abrupt drifts occur when at a moment in time t, the source distribution $P(x, y)_t$ is suddenly replaced by a different distribution in $P(x, y)_{t+1}$ [9].



(a) Down Jones (b) S&P500

Figure 6: Real-world time series with known concept drifts. Vertical red lines represent known concept drifts.

5.3 Real-World Time Series with Known Concept Drifts

The two real-world time series with known concept drifts described in this section were first studied in papers about change point detection, where time series are analyzed in an off-line way (after the whole time series is collected) to determine the location of the drifts (see Fig. 5).

The first series is Dow Jones Industrial Average daily series studied in [27], and illustrated in Fig. 6a. It has 735 points and abrupt drifts caused by the following historical moments: **124** (condemnation of the advisors of former President Nixon, G. Gordon Liddy and James W. McCord Jr. on January 30, 1973); **307** (initiation of the OPEC embargo against the United States on October 19, 1973); and **510** (resignation of former President Nixon on August 9, 1974).

The second series is the financial S&P500 studied in [28], and illustrated in Fig. 6b. It has 4229 points and gradual drifts caused by the following historical moments: **448 and 508:** Increase and decrease in volatility following the October 1987 stock crash; **2826:** Asian Summer Crisis of 1997; and **4119:** May 2002, an increase in volatility due to the collapse of the internet bubble and the fraudulent bankruptcies of Enron and WorldCom.

5.4 Real World Financial Time Series

Three other real-world time series available at https://finance.yahoo.com were used: (i) the Dow Jones Industrial Average, a daily series, containing 8139 points, with the data collected from January 29, 1985 until May 12, 2017; (ii) Nasdaq, also a daily series, containing 11667 points, with data collected from May 2, 1985 until May 12, 2017; (iii) S&P 500, a daily series, containing 16,857 points, with data collected from May 15, 1950 through May 12, 2017. As there is no prior knowledge about drifts in these time series, they were used to evaluate only the forecasting ability of the proposed approach.

5.5 Drift Detection Evaluation

This section explains the instruments to investigate RQ1, which concerned improving concept drift detection by monitoring swarm models.

5.6 Comparison approaches

SISC-D was compared with six drift detectors from the literature: DDM [19], ECDD [7, 19], FEDD [6, 7], IDPSO-ELM-B [7], IDPSO-ELM-S [7], and RPSO [14]. The first five methods were chosen because they have been applied in the context of time series forecasting, representing baselines and the state-of-the-art in this area. The last method was chosen because it is based on swarm intelligence.

5.7 Parameter Setup

For a fair comparison, all drift detectors were combined with the same predictive model (ELM), which was executed 30 times for each time series with known drifts. To assign an appropriate value for the drift detectors to learn the ELM training error, we evaluated $m = [100, 200, 300, 400, 500]$. The value that produced the best forecasting error was $m = 300$. As the real-world time series with known concept drifts are short, we fixed $m = 100$. The parameters used in the experiments were: RPSO-ELM: $\sigma = [1, 2, 3, 4]$; ELM-DDM: $\sigma = [2, 4, 6, 8]$; ELM-FEDD: $c = [0, 0.25, 0.5, 0.75]$; ELM-ECDD, IDPSO-ELM-B, IDPSO-ELM-S and SISC: $c = [0.25, 0.5, 0.75, 1]$.

Table 1: Parameters used to generate the artificial time series. From the sixth concept, the first, second, third, and fourth concepts are repeated.

Time Series	**Concept**	α	**p**
	1	{0.007,-0.253,0.855,0.391}	4
	2	{-0.443,0.447,1.352,-0.356}	4
	3	{0.003,-0.328,0.146,1.172}	4
Linear Gradual	4	{0.333,-0.113,0.054,0.715}	4
	5	{-0.634,0.335,1.36,-0.074}	4
	6	{-0.441,0.074,1.257,0.108}	4
		α	**p**
	1	{0.149, 0.051, 0.433, 0.367}	4
	2	{-0.318, 0.413, 1.148, -0.245}	4
	3	{0.003, -0.328, 0.146, 1.172}	4
Linear Abrupt	4	{-0.443, 0.447, 1.352, -0.356}	4
	5	{-0.027, 0.22, -0.038, 0.845}	4
	6	{-0.479, 0.856, 0.025, 0.598}	4
		α	**Model**
	1	{0.02, 0.149, 0.122, 0.691}	NL 1
	2	{0.214, 0.175, 0.256, 0.349}	NL 1
	3	{0.675, 0.04, 0.129, 0.141}	NL 1
Non-linear Gradual	4	{0.259, 0.187, 0.251, 0.291}	NL 1
	5	{0.333, -0.113, 0.054, 0.715}	NL 1
	6	{0.178, -0.091, 0.363, 0.545}	NL 1
		α	**Model**
	1	{-0.067, 0.234, 0.155, 0.677}	NL 1
	2	{-0.507, 0.259, 1.397, -0.15}	NL 1
	3	{-0.439, 0.375, 1.333, -0.269}	NL 1
Non-linear Abrupt	4	{0.07, -0.052, 0.635, 0.334}	NL 1
	5	{-0.443, 0.447, 1.352, -0.356}	NL 1
	6	{-0.276, 0.334, 0.41, 0.532}	NL 1
		ζ	**s**
	1	{34, 32, 30, 28, 26, 24, 22, 24, 26, 28, 30, 32}	12
	2	{34, 26, 18, 10, 18, 26, 10}	7
	3	{34, 26, 18, 10, 18, 26}	6
Seasonal	4	{34, 26, 18, 10, 2, -6, -14, -6, 2, 10, 18, 26}	12
	5	{34, 10, -14, 10}	4
	6	{38, 28, 18, 8, 0, -8,-18,-8, 0, 8, 18, 28}	12
		α, ζ	**Model**
	1	{0.003, -0.005, 1.0}	AR
	2	{Last 3 observations}	Seasonal
Hybrid	3	{0.059, 0.086, 0.62, 0.21}	NL 2
	4	{0.018, 0.95, 0.032}	AR
	5	{Last 3 observations}	Seasonal
	6	{0.55, 0.024, 0.41, 0.009}	NL 1

5.8 False Positive Detection (FPR)

It consists of the number of times that a real drift was detected, and after that a false detection occurred.

5.9 Detection Delays (DD)

It is the amount of instances the approach needed until it can detect the occurrence of a known drift. We compute it as the sum of the approach's delays considering all drifts that exist in the time series. When a *miss-detection* occurs, all the time series observations that belong to the concept are counted as the method's delay.

5.10 Drift Detection Curve

These curves show the behavior of the approaches for all parameter settings investigated [29]. Each point on the graph is formed by the coordinate (FPR, DD) for a specific parameter executed for a time series. So, the methods with the best detection performance are those with curves nearest to the point (0,0) of the graph, which are those with small DD and few FPR.

To assign an adequate value to the number of observations used to train the ELM, the values $m = 100$, 200, 300, 400 and 500 were tested. The value that produced the best forecasting error on the test was $m = 300$. For the real-world time series with known concept drifts, we assign $m = 100$, since the series are short and values greater than 100 cover some concept drifts.

5.11 Time Series Forecasting Evaluation

This section explains the instruments to investigate both RQ2 and RQ3. RQ2 requires to check whether the reuse of swarm models can improve forecasting ability (SISC-P). RQ3 requires to check whether the reuse of past best swarm models can improve forecasting ability when there are recurring concepts (SISC-M).

5.12 Comparison approaches

In addition to the approaches discussed in 5.5, we added the SISC-D method, which uses only the drift detection (without SISC-P and SISC-M), to evaluate whether SISC-P and SISC-M improve the system's predictive performance.

5.13 Parameter Setup

We performed preliminary experiments in the artificial time series to select the best forecasting parameters. The grid search done was presented in Section 5.5. For ELM's parameters we fixed $k = 5$ for input neurons, $l = 10$ for hidden neurons, and the sigmoid as activation function. For the approaches parameters: RPSO-ELM: $\sigma = 1$; ELM-DDM: $\sigma = 8$; ELM-FEDD, ELM-ECDD, IDPSO-ELM-B, IDPSO-ELM-S: $c = 0.25$; SISC-P, SISC-M: $c = 0.25$ and $w = 0.1$. For SISC-M, we fixed 30 slots for solution allocation ($Memory$) and 3 for the solution exclusion threshold ($exclusionThreshold$). For the methods that use IDPSO or PSO: 50 iterations (K_{max}), 30 particles (Swarm size), 3 iterations without improvement as stop criterion. For IDPSO: $\omega_{initial} = 0.8$; $\omega_{end} = 0.4$; $c_1 = c_2 = 2$; $max\_position = max\_velocity = 0.3$; and $min\_position = min\_velocity =$ -0.3, $\mu = 100$. For PSO, $\omega = 0.8$.

5.14 Prequential Evaluation

After batch training on an initial fraction of the series, we use each instance first to test the model, and then to train the model. In our proposed method, we do not retrain in each new observation, only when a complete batch is stored after the drift. To measure the predictions made, we use MAE (Eq. 3), because it calculates only the subtraction of the predicted by true value, allowing to analyze in detail the variation caused by the drift.

5.15 Runtime

Difference in seconds between the end time and the start time of an approach. Machine used has 8GB of ram and processor Intel Xeon E3-1220 v5.

5.16 Friedman and Nemenyi Tests

We use the Friedman's non-parametric test with significance level $\alpha = 0.05$ across time series, following the advice in [30]. If the null hypothesis that there is no statistical difference between the approaches is rejected, the pairwise comparison is used with Nemenyi test with a significance level $\alpha = 0.05$.

6 Results and Discussions

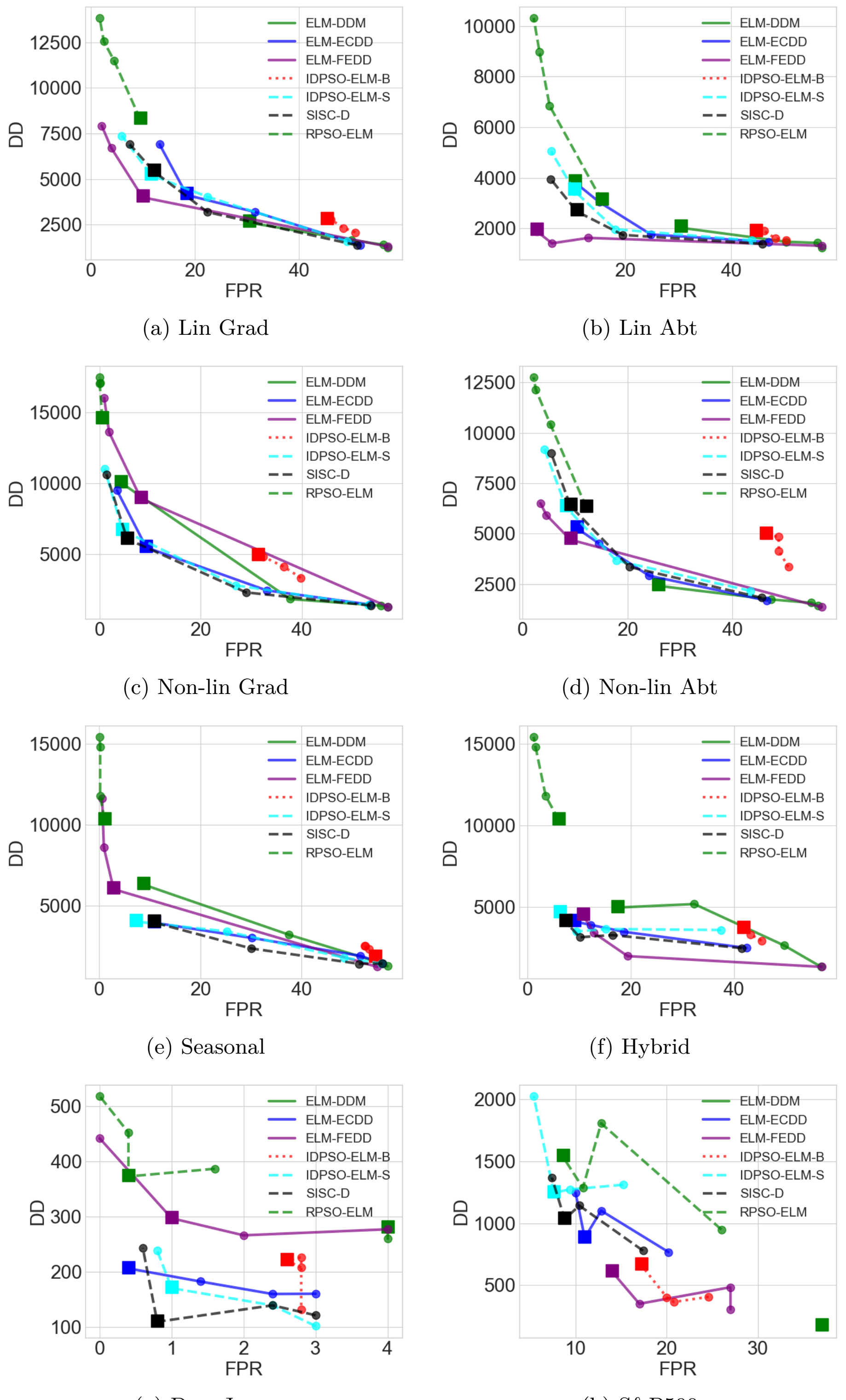
Sections 6.1, 6.2 and 6.3 present the analyses to validate the answers to RQ1, RQ2, and RQ3, respectively.

6.1 RQ1 – Impact of Drift Detection with Swarm Intelligence

Artificial Time Series

Figs. 7a, 7b, 7c, 7d, 7e and 7f show the drift detection curves for all drift detection methods for artificial time series. We first compare the results obtained by IDPSO-ELM-S, IDPSO-ELM-B, SISC-D, and ELM-ECDD, as they are all based on ECDD. The difference between them is that IDPSO-ELM-S, IDPSO-ELM-B and SISC-D monitor several swarm models, whereas ELM-ECDD monitors a single model. As can be seen in Fig. 7a, 7b, 7c, 7d, IDPSO-ELM-B obtained a low detection performance because its detection rule is based on the average of the models' performance. The average is sensitive to outlier points, which has led to high FPR. The curves for IDPSO-ELM-S and SISC-D are closer to the optimal point (0,0), indicating that it is worth to monitor swarms models separately. SISC-D achieved better results than IDPSO-ELM-S, because it uses the majority vote for detecting, instead of requiring a drift detection for all swarm models. The use of majority vote reduced DD without affecting FPR too much, confirming the advantage of our proposed use of majority vote.

ELM-DDM and RPSO-ELM obtained the worst results, with their curves far from the optimal point (0, 0). Although RPSO-ELM is a swarm-based method, its drift detector only monitors the average performance of the best model, not benefiting from the swarm as a whole for drift detection purposes. Both of these methods use mean and standard deviation to detect a concept drift. These metrics are very sensitive to outlier points, and this can generate either a very long DD or high FPR, depending on the error obtained by ELM in training.



(a) Lin Grad (b) Lin Abt

(c) Non-lin Grad (d) Non-lin Abt

(e) Seasonal (f) Hybrid

(g) Dow Jones (h) S&P500

Figure 7: Drift Detection Curves. Each point on the graph is formed by the coordinate (FPR, DD) for a specific parameter executed for a time series. The parameters presented here were presented in the Section 5.5. So, the methods with the best detection performance are those with curves nearest to the point (0,0) of the graph, which are those with small delays and few false positive detections. The squared symbols are the points closest to (0,0) for each approach.

When comparing SISC-D against ELM-FEDD, ELM-FEDD achieved better detection performances for the following kinds of time series: (i) linear gradual (Fig. 7a), (ii) linear abrupt (Fig. 7b) and (iii) non-linear gradual (Fig. 7c). These are the time series with less variability. ELM-FEDD works better for this type of time series because it analyzes the statistical characteristics of the time series (e.g. time series variance, partial autocorrelation function, correlation function and others), enabling it to identify smaller concept drifts better than the other methods. On the other hand, SISC-D achieved better scores for (i) non-linear abrupt time series (Fig. (7d), (ii) seasonal time series (Fig. 7e), and (iii) hybrid time series (Fig. 7f), where a better balance between DD and FRP was achieved. These time series has severe drifts, which are large variation in the distribution of the data from one concept to another. SISC-D works better for this type of time series because error-based methods depend on the model degradation to identify a concept drift, thus the more severe the drift, the faster the model wears off, thereby favoring its detection tests.

Real-world Time Series

Figs. 7g and 7h show the drift detection curves for real-world time series. In Fig. 7g, it can be observed that the SISC-D reached the best detection performance. It occurs because these series present severe drifts, thus degrading the model performance quickly which favor SISC-D as it is an error-based method. Observing Fig. 7h, ELM-FEDD obtained the best detection performance, while methods based on error obtained the worst. It occurs because this series has less variability, which does not affect the performance of the models. Therefore, as ELM-FEDD finds small variations in the characteristics of the series, it was able to better identify the drifts.

RQ1 Answer: SISC-D achieved better detection performance in time series with severe drifts, as this type of drift degrades the model faster, favoring error-based methods. SISC-D reached better detection performance than other error-based methods because it (i) monitors a group of models rather than a single model, reducing the number of false alarms; and (ii) triggers a detection based on the majority of models rather than all, reducing the detection delay.

6.2 RQ2 – Impact of Reusing Swarm Models to Improve Time Series Forecasting

Artificial Time Series

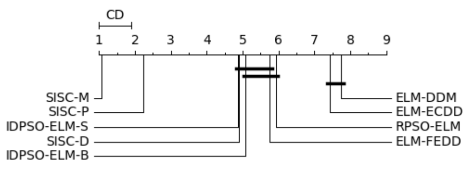
Tables 2, 3 presents the MAE, and runtime for all compared approaches. Fig. 8 presents the Friedman and Nemenyi tests for these two metrics. According to Fig. 8a, SISC-P reached the second position on Friedman rank for MAE. This implies that regardless of the type of concept drifts, the reuse of a swarm models significantly improves prediction performance over non-use (see SISC-D performance). Also, we see for runtime in Fig. 8c that the SISC-P positioned itself at the end of the ranking. This is because (i) several models are created during training, and (ii) models are reevaluated during the drift moment. Methods like ELM + drift detector are high-speed due to instant ELM training through a single calculation. However, for MAE, ELM methods trained by IDPSO reached in general better rankings than standard

ELM training (Fig. 8a). These findings show that training by swarm intelligence may improve ELM predictions.

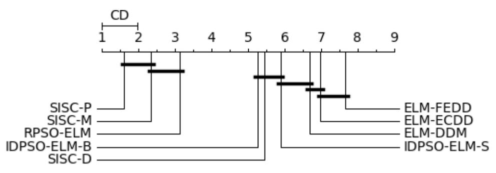
Real-world Time Series

SISC-P obtained the best MAE for all real-world time series with exception for the S&P500 series (Table 2). Fig. 8b attests statistical difference between SISC-P and all compared approaches except for SISC-M. To understand in more detail how the SISC methods work in a practical example, we analyze Fig. 9, which presents (i) SISC-D which do not select the best ELM models when drift takes place (Fig. 9a), and (ii) SISC-P which selects the best ELM models when drift takes place (Fig. 9b). Note in Fig 9a, when the SISC-D informs a concept drift, its forecasting error tend to increase. This behavior does not happen with the same intensity in Fig 9b, because SISC-P select for each new test instance a best suited model to make the next prediction.

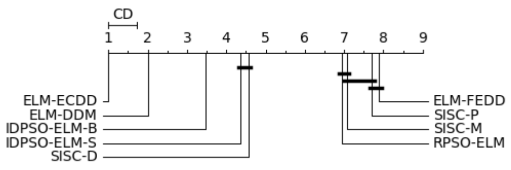
RQ2 Answer: We observed that: (i) training ELM with swarm intelligence in general improved ELM forecasting performance compared with methods that do not use it; and (ii) in the presence of concept drift, replacing the obsolete model with another model in the swarm significantly improved SISC-P performance compared with swarm-methods that do not reuse models (SISC-D); and (iii) SISC-P showed to be particularly relevant in time series with no recurrent drifts.



(a) MAE for Artificial TS



(b) MAE for Real TS



(c) Runtime for all TS

Figure 8: Friedman ranking of the approaches across time series (TS). Smaller ranks are better ranks. Friedman's p-values were 3.69E-191, 2.65E-154, 4.84E-391, respectively, rejecting the null hypothesis. Any pair of approaches whose distance between them is larger than the critical distance (CD) is significantly different according to the Nemenyi posthoc tests.

Table 2: Average and standard deviation of the MAE obtained by all methods. Values in bold represent the lowest MAEs.

Bases	ELM-DDM	ELM-ECDD	ELM-FEDD	IDPSO-ELM-B	IDPSO-ELM-S	SISC-D	SISC-P	SISC-M	RPSO-ELM
Linear Gradual	0.0257 (0.009)	0.0306 (0.018)	0.0186 (0.004)	0.0147 (0.002)	0.0147 (0.002)	0.0147 (0.002)	0.0129 (0.002)	**0.0116 (0.001)**	0.0167 (0.004)
Linear Abrupt	0.0179 (0.005)	0.0176 (0.003)	0.0135 (0.003)	0.0134 (0.003)	0.0132 (0.002)	0.0128 (0.003)	0.0105 (0.002)	**0.0091 (0.001)**	0.0124 (0.004)
Non-linear Gradual	0.0351 (0.005)	0.0353 (0.006)	0.0332 (0.004)	0.0329 (0.004)	0.033 (0.004)	0.0327 (0.004)	0.0315 (0.004)	**0.0313 (0.002)**	0.036 (0.004)
Non-linear Abrupt	0.0483 (0.035)	0.0438 (0.031)	0.0467 (0.04)	0.0179 (0.004)	0.0173 (0.003)	0.0187 (0.005)	0.0131 (0.003)	**0.0118 (0.002)**	0.0181 (0.004)
Linear Seasonal	0.0782 (0.007)	0.0726 (0.005)	0.0659 (0.002)	0.0663 (0.002)	0.0666 (0.002)	0.0668 (0.002)	0.0647 (0.002)	**0.0596 (0.002)**	0.0834 (0.004)
Hybrid	0.0511 (0.034)	0.0644 (0.049)	0.0834 (0.069)	0.0413 (0.02)	0.0328 (0.013)	0.0317 (0.011)	0.0163 (0.003)	**0.0129 (0.002)**	0.0246 (0.004)
Dow Jones-Drift	0.1905 (0.1572)	0.167 (0.1196)	0.2039 (0.1375)	0.0593 (0.0248)	0.0501 (0.0153)	0.0428 (0.02)	0.0301 (0.0027)	**0.0226 (0.0019)**	0.0356 (0.008)
S&P500-Drift	0.031 (0.0319)	0.0326 (0.0321)	0.0343 (0.0295)	0.0298 (0.0325)	0.0316 (0.0328)	0.0305 (0.0328)	**0.0063 (0.0002)**	0.0075 (0.0006)	0.0113 (0.005)
Dow Jones	0.0097 (0.0021)	0.0098 (0.0023)	0.0612 (0.0437)	0.0076 (0.0017)	0.0088 (0.0025)	0.0087 (0.0022)	**0.0033 (0.0002)**	0.0043 (0.0005)	0.0047 (0.0008)
Nasdaq	0.009 (0.0027)	0.0097 (0.0046)	0.0075 (0.0022)	0.0052 (0.0013)	0.0066 (0.0015)	0.0061 (0.0011)	**0.0026 (0.0003)**	0.0032 (0.0005)	0.0037 (0.0006)
S&P500	0.0063 (0.0026)	0.0053 (0.0011)	0.0074 (0.0024)	0.0055 (0.002)	0.0044 (0.001)	0.005 (0.0014)	**0.0018 (0.0001)**	0.0022 (0.0002)	0.0023 (0.0004)

Table 3: Average and standard deviation for runtime (seconds) for all methods. Values in bold represent the best time.

Bases	ELM-DDM	ELM-ECDD	ELM-FEDD	IDPSO-ELM-B	IDPSO-ELM-S	SISC-D	SISC-P	SISC-M	RPSO-ELM
Linear Gradual	3.9 (0.2)	**2.7 (0.1)**	40.8 (1.4)	10.9 (0.7)	15.1 (2.7)	15.4 (2.1)	32.4 (6.7)	23.1 (3.8)	27.5 (0.3)
Linear Abrupt	3.9 (0.6)	**2.6 (0.1)**	39 (1.9)	10.6 (0.7)	15.2 (1.9)	15.6 (2.3)	31.3 (5.9)	22.2 (4.1)	27.7 (0.3)
Non-linear Gradual	5.1 (0.5)	**2.5 (0.1)**	42.9 (1.8)	11 (0.7)	11.1 (1.7)	11.6 (2.2)	21.4 (4.5)	15.8 (4.6)	27.2 (0.2)
Non-linear Abrupt	4.2 (0.5)	**2.6 (0.1)**	40.7 (1.5)	10.8 (1)	13.5 (5.2)	13.5 (6.5)	28.4 (12.4)	19.6 (13.4)	27.5 (0.3)
Linear Sazonal	5.1 (0.5)	**2.8 (0.1)**	42.4 (2.8)	11.8 (0.7)	24.3 (5)	27.5 (5.9)	55.2 (12.3)	45.8 (11.8)	27.6 (0.2)
Hybrid	5.7 (1.7)	**2.5 (0.6)**	36.6 (11.3)	12.1 (2.2)	14.6 (1.6)	14.9 (1.8)	29.6 (6.3)	20.3 (2.8)	27.7 (7.4)
Dow Jones-Drift	0.1 (0)	**0.1 (0)**	0.3 (0)	1 (0.2)	1.5 (0.2)	1.3 (0.1)	2.8 (0.2)	1.9 (0.5)	1.8 (0.3)
S&P500-Drift	0.6 (0)	**0.5 (0)**	3.9 (0)	6.8 (0.8)	3 (0.4)	2.4 (0.2)	2.3 (0.8)	1.7 (0.3)	2.1 (0.7)
Dow Jones	2.2 (0.5)	**1.2 (0.2)**	12.8 (1.4)	7.8 (1.4)	11.7 (1.7)	11.5 (1.6)	17.7 (3)	20.2 (3.1)	12.3 (2.5)
Nasdaq	3.1 (0.9)	**1.6 (0.3)**	15.5 (3.6)	10.7 (4.2)	11.8 (2.8)	13.8 (2)	21.3 (4.3)	23.5 (4.5)	17.2 (3.7)
S&P500	4.7 (2)	**2.2 (0.9)**	22.6 (9.4)	19.1 (7.1)	15.8 (6.1)	16.3 (6.6)	25.7 (11.1)	29.6 (12.4)	24.3 (0.4)

6.3 RQ3 – Impact of Reusing Past Swarm Models to Deal With Recurring Concepts in Time Series Forecasting

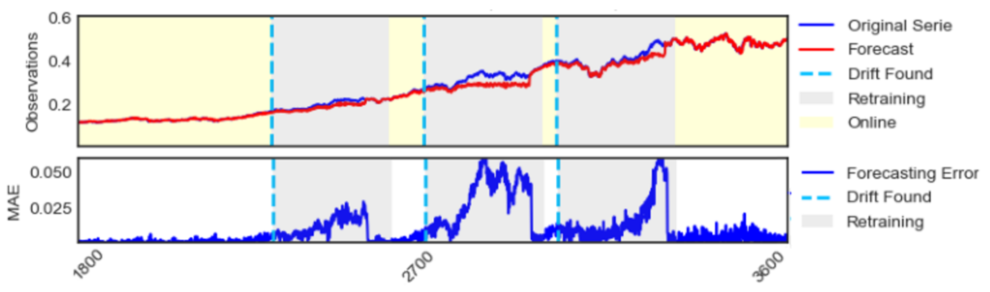
Artificial Time Series

The artificial time series have recurrent concepts from the sixth concept onwards (see Section 5.1). This allowed SISC-M to achieve the best Friedman rank for MAE (Fig 8a), due to its long-term memory, which stores models trained in past concepts, thus being able to reuse them whenever the same concept reappears.

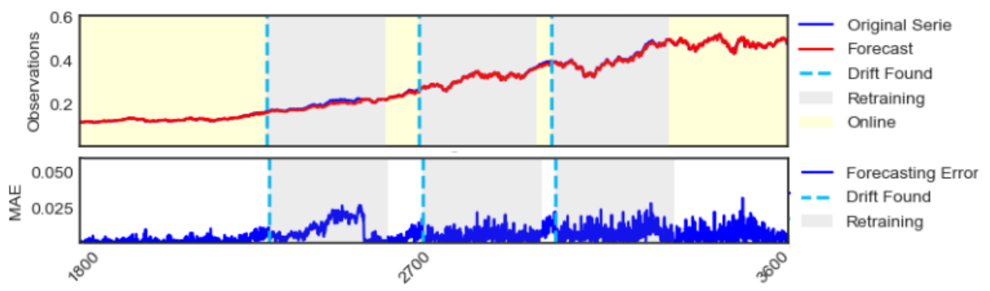
Real-world Time Series

We observe that SISC-M obtained the best MAE only for the S&P500 time series with concept drift (Table 2). This implies that this series has some segments that may be correlated, or rather, recurrent due to the performance obtained by SISC-M in recurring artificial time series. According to the statistical test (Fig. 8b), SISC-M is not statistically different from SISC-P and RPSO-ELM in terms of MAE. This indicates that in addition to recurring drifts, other types of drifts may also be favored by reusing past swarm models.

RQ3 Answer: We observed that: (i) saving the best swarm model in memory after each retraining reached the best forecasting performances for time series with recurring drifts, due to the reuse of models in memory already trained in similar distributions; and (ii) adaptation to other types of drift was also helped by that, as the nearest neighbor filter favors to store models in memory that have different capacities to respond to new concepts, increasing the chances of at least one being sufficient.



(a) SISC-D execution



(b) SISC-P execution

Figure 9: Difference in the execution of SISC-D (does not select models in the drift) for SISC-P (selects models in the drift). The upper block shows the forecast and true values in a part of the Dow Jones time series. The bottom block shows the MAE for the forecast.

7 Conclusion

We proposed SISC, the first approach that uses dynamic swarm intelligence techniques to help dealing with concept drift in time series forecasting. Dynamic swarm intelligence is usually applied to optimization functions and its use in real applications is often unviable. In this work, we were able to use its inspiration to apply it to the real problem of time series forecasting.

SISC has a drift detection strategy based on multiple swarm models (SISC-D), and two possible different strategies to adapt to concept drifts: (i) SISC-P, that employs different current swarm models to better react to concept drift; and (ii) SISC-M, that stores past best swarm models to be reused especially for recurring concepts. SISC-M and SISC-P achieved the best forecasting performances in all tested time series, whereas SISC-P achieved the best drift detections in time series with severe drifts, besides being competitive for non-severe drifts.

Future work includes: (i) how to accelerate the training process of the swarm models, e.g., by resetting only the worst particles rather than all; and (ii) combining the swarm particles into ensembles of models. We also believe that SISC can inspire other novel approaches for handling concept drifts based on dynamic swarm intelligence in both time series forecasting and classification problems.

Acknowledgment

The authors would like to thank CNPq and FACEPE (Brazilian Research Agencies) for their financial support. Leandro Minku was supported by EPSRC Grant No. EP/R006660/2.

Conflict of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

[1] Wang, H.-K., Zhang, X., Long, H., Yao, S., Zhu, P.: W-fenet: Wavelet-based fourier-enhanced network model decomposition for multivariate long-term time-series forecasting. NPL **56**(2), 1–23 (2024)

[2] Jiang, W., Ling, L., Zhang, D., Lin, R., Zeng, L.: A time series forecasting model selection framework using cnn and data augmentation for small sample data. NPL **55**(5), 5783–5810 (2023)

[3] Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: A survey. IEEE CIM **10**(4), 12–25 (2015)

[4] Ding, C., Zhao, J., Sun, S.: Concept drift adaptation for time series anomaly detection via transformer. NPL **55**(3), 2081–2101 (2023)

[5] Yang, Z., Al-Dahidi, S., Baraldi, P., Zio, E., Montelatici, L.: A novel concept drift detection method for incremental learning in nonstationary environments. IEEE TNNLS **31**(1), 309–320 (2019)

[6] Cavalcante, R.C., Minku, L.L., Oliveira, A.L.: Fedd: Feature extraction for explicit concept drift detection in time series. In: IEEE IJCNN, pp. 740–747 (2016)

[7] Oliveira, G.H.F.M., Cavalcante, R.C., Cabral, G.G., Minku, L.L., Oliveira, A.L.: Time series forecasting in the presence of concept drift: A pso-based approach. In: IEEE ICTAI, pp. 1–8 (2017)

[8] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM CSUR **46**(4), 44 (2014)

[9] Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: A survey. IF **37**, 132–156 (2017)

[10] Liu, X.-F., Zhan, Z.-H., Gu, T.-L., Kwong, S., Lu, Z., Duh, H.B.-L., Zhang, J.: Neural network-based information transfer for dynamic optimization. IEEE TNNLS (2019)

[11] Garai, S., Paul, R.K., Yeasin, M., Paul, A.: Ceemdan-based hybrid machine learning models for time series forecasting using mars algorithm and pso-optimization. NPL **56**(2), 92 (2024)

[12] Caraffini, F., Santucci, V., Milani, A.: Evolutionary Computation & Swarm Intelligence. MDPI, ??? (2020)

[13] Fouladgar, N., Lotfi, S.: A novel approach for optimization in dynamic environments based on modified cuckoo search algorithm. SC **20**(7), 2889–2903 (2016)

[14] Rakitianskaia, A.S., Engelbrecht, A.P.: Training feedforward neural networks with dynamic particle swarm optimisation. SI **6**(3), 233–270 (2012)

[15] Nasiri, B., Meybodi, M., Ebadzadeh, M.: History-driven particle swarm optimization in dynamic and uncertain environments. Neurocomputing **172**, 356–370 (2016)

[16] Bao, Y., Xiong, T., Hu, Z.: Pso-mismo modeling strategy for multistep-ahead time series prediction. IEEE TCYB **44**(5), 655–668 (2013)

[17] Kirisci, M., Cagcag Yolcu, O.: A new cnn-based model for financial time series: Taiex and ftse stocks forecasting. NPL **54**(4), 3357–3374 (2022)

[18] Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L., Petitjean, F.: Characterizing concept drift. KDD **30**(4), 964–994 (2016)

[19] Cavalcante, R.C., Oliveira, A.L.: An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection. In: IEEE IJCNN, pp. 1–8 (2015)

[20] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)

[21] Ross, G.J., Adams, N.M., Tasoulis, D.K., Hand, D.J.: Exponentially weighted moving average charts for detecting concept drift. PRL **33**(2), 191–198 (2012)

[22] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: SBIA, pp. 286–295 (2004). Springer

[23] Zhang, Y., Xiong, X., Zhang, Q.: An improved self-adaptive pso algorithm with detection function for multimodal function optimization problems. MPE **2013** (2013)

[24] Silva, C.A., Krohling, R.A.: Semi-supervised online elastic extreme learning machine with forgetting parameter to deal with concept drift in data streams, pp. 1–8 (2019). IEEE IJCNN

[25] Han, H.-G., Lu, W., Hou, Y., Qiao, J.-F.: An adaptive-pso-based self-organizing rbf neural network. IEEE TNNLS (2016)

[26] Zhang, Y., Xiong, X., Zhang, Q.: An improved self-adaptive PSO algorithm with detection function for multimodal function optimization problems. MPE **2013** (2013)

[27] Adams, R.P., MacKay, D.J.: Bayesian online changepoint detection. arXiv preprint arXiv:0710.3742 (2007)

[28] Lavielle, M., Teyssiere, G.: Detection of multiple change-points in multivariate time series. LMJ **46**(3), 287–306 (2006)

[29] Boracchi, G., Roveri, M.: Exploiting self-similarity for change detection. In: IJCNN, pp. 3339–3346 (2014). IEEE

[30] Demsar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR **7**, 1–30 (2006)