Data-Driven Evolutionary Computation under Continuously Streaming Environments: A Drift-Aware Approach

Yuan-Ting Zhong and Yue-Jiao Gong, *Senior Member, IEEE*

***Abstract*—Streaming Data-Driven Evolutionary Algorithms (SDDEAs) have emerged as a crucial paradigm in the area of data-driven optimization. However, current methods face critical limitations when handling unpredictable concept drifts in continuously evolving environments. To address this research gap, we propose DASE, a drift-aware streaming evolutionary algorithm that features two key innovations. First, we introduce a hierarchical confidence drift detector that operates on a moving window over continuous data streams, identifying concept drifts by evaluating statistical deviations in model accuracy. Second, we propose a context-aware warm start mechanism that adaptively transfers knowledge from historical environments to the new environment using environmental similarity-based weighting. These dual innovations not only enables automatic segmentation of streaming data into coherence environments but also enhances optimization performance with the real-time responsiveness. Experimental evaluations on benchmark problems demonstrate that DASE significantly outperforms state-of-the-art algorithms across various drift scenarios, establishing it as a powerful method for addressing challenges in continuously streaming environment.**

***Index Terms*—Streaming data-driven evolutionary algorithm, dynamic optimization, streaming data, surrogate model, concept drift.**

I. Introduction

EVOLUTIONARY Algorithms (EAs) [1] are a class of algorithms inspired by the principles of evolution. They solve optimization problems by representing solutions as a population of individuals, which evolve iteratively based on the fitness values, following the "survival of the fittest" principle. Over the past several decades, EAs have demonstrated their effectiveness in addressing a wide range of optimization problems [2]–[5]. Nevertheless, many real-world industrial applications lack well-defined objective functions and rely on limited data obtained from costly physical experiments and numerical simulations for optimization [6]–[10]. This makes it impractical to directly evaluate fitness values at each iteration in EAs. To address these issues, Data-Driven Evolutionary Algorithms (DDEAs) have been proposed [11]. DDEAs leverage data and employ machine learning techniques to construct surrogate(s) that approximate the real fitness values, thereby assisting the optimization process in EAs and reducing the computation cost. Researches have demonstrated that DDEAs can achieve high-quality solutions with only relying on data [12]–[16].

However, in many real-world applications, data is generated as continuous streams, which poses unique challenges for traditional approaches. For instance, in modern smart cities, traffic monitoring systems generate continuous data streams, such as vehicle counts and traffic flow patterns from sensors [17], [18]. Storing this real-time data in memory is often impractical due to its volume [19]. Moreover, traffic patterns can shift unpredictably due to factors like accidents or weather [20], causing concept drift, where data distribution evolves over time [21]. Offline processing such huge amount of data demands growing storage capacity and may cause delayed analyses. Consequently, DDEAs designed for static environments are not equipped to handle the additional challenges posed by data streams with concept drift. First, keep tracking promising solutions is crucial in dynamic environments, as solving from scratch for each new environment is inefficient and fails to leverage correlations between data concepts [22]. Second, timely detection and segmentation of concept drift are essential to maintain surrogate accuracy and prevent misleading EAs. Moreover, effective data management is needed to balance data quantity, as too little data causes overfitting, while too much hinders surrogate's quick adaptation to environment changes [23].

So far, only a few attempts [24]–[27] have been proposed considering difficulties inherent in challenges mentioned above in dynamic environments with concept drifts, named Streaming DDEAs (SDDEAs) [23]. Typically, they focus on the correlation between different environments in data streams and manage the continuous growth of data through designing data management strategies. While existing methods have shown promising results in dynamic environments, the following issues remain unresolved:

1) Nearly all existing SDDEAs assume that concept drift between environments is known a priori. As a result, they treat each distinct batch of data points in the stream as a separate environment, assuming that concept drift happens whenever a new batch is introduced. However, in reality, the onset of concept drift is often unpredictable, and the changes in data distributions may be subtle or gradual, making it difficult to detect and handle using batch-based assumptions.

This work was supported in part by Guangdong Natural Science Funds for Distinguished Young Scholars (Grant No. 2022B1515020049), in part by National Natural Science Foundation of China (Grant No. 62276100), in part by Guangzhou Science and Technology Elite Talent Leading Program for Basic and Applied Basic Research (Grant No. SL2024A04J01361), and in part by the Fundamental Research Funds for the Central Universities. (Corresponding author: Yue-Jiao Gong)

Y.-T. Zhong and Y.-J. Gong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: {ytalienzhong, gongyuejiao}@gmail.com).

2) Many methods presume that all data from the current environment is available at once. This does not reflect the continuously streaming nature of complex scenarios, which requires real-time and incremental adaptation.
3) Some SDDEAs rely on active sampling for solution-fitness pairs during the optimization process, which is generally not feasible in many complex scenarios that only have passive data available [22], [28]–[30].

To fill this huge research gap, we propose a drift-aware streaming evolutionary algorithm, called DASE. The main contributions of this research are as follows.

- We propose a drift-aware framework designed to handle unknown drifts in continuous streaming environments. The framework continuously monitors the data stream and identifies when concept drift occurs. When a drift is detected, relevant information from the previous environment is archived, and an adaptive knowledge transfer mechanism is employed to warm-start the optimization process. This allows for faster and more efficient adaptation to the new environment. If no drift is detected, the framework continues optimization within the current environment.
- We propose a hierarchical confidence drift detector (HCDD), which operates on a moving window to handle continuously streaming data. It monitors model's error-rate deviations for incoming data using multiple confidence levels of warning intervals. This hierarchical approach enables more precise and adaptive detection of various types of concept drifts.
- For new environments, we design an adaptive knowledge transfer mechanism to enable warm-start and mitigate the inefficiencies of optimization from scratch. It adaptively reuses relevant information from past environments at both the surrogate and population levels, ensuring the algorithm's effectiveness by focusing on the most pertinent knowledge while avoiding overload from irrelevant information.
- Besides, as mentioned, some SDDEAs rely on active sampling to select specific solution-fitness pairs during optimization, while our algorithm uses passively incoming data. In other words, it processes continuously streaming data without the ability to actively select which solution positions to sample for fitness, making it more suitable for real-world environments.

We compare our proposed algorithm with state-of-the-art (SOTA) SDDEAs and DDEAs on benchmarks with different types of drift, and the experimental results verify the effectiveness and superiority of our DASE.

The remainder of this paper is organized as follows. Section II provides a summary of the background and related work. Section III presents the details of our proposed DASE. The experimental comparisons and analyses are discussed in Section IV. Finally, Section V concludes the paper and outlines directions for future work.

II. Background and Related Work

A. Problem Definition

The Streaming Data-Driven Optimization (SDDO) problem involves optimizing a time-varying, agnostic objective function based on sequentially arriving data, where the problem landscape and global optima evolve over time due to concept drift. At each time step $t \in \{1, \cdots, T\}$, a new data point $(\boldsymbol{x}, y_t)$ is observed, with no access to the explicit form of the objective function F_t. The goal is to continuously track the dynamically changing optimum $\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x}} F_t(\boldsymbol{x})$, relying solely on passively observed data. In this study, we utilize SDDObench [23] to generate such evolving data streams. Due to space limitation in the manuscript, the detailed formulation of the SDDO problem definition is presented in the Section I-A of the supplementary material.

B. From DDEAs to SDDEAs

In DDEAs [31]–[34], the problem information is derived from available data, which is used to construct surrogate models that approximate real fitness values to assist the optimization process. During the optimization process, some DDEAs actively query a fixed number of decision variables $(\boldsymbol{x}_i)$ to obtain their corresponding objective values (y_i) [35], [36]. These newly sampled data points are incorporated into the dateset, enriching the problem landscape representation and enhancing the surrogate model accuracy [37]–[39]. In contrast, other DDEAs acquire data passively, relying solely on off-the-shelf data from related applications/tasks without collecting new data during the optimization process [13], [15].

Building upon the foundation of DDEAs, SDDEAs extend these approaches to streaming environments, where data arrives in an ongoing fashion [23]. Unlike static contexts, optimization in continuously streaming environments presents additional challenges. Firstly, as data streams grow indefinitely, memory constraints prevent storing all the data for future use. Furthermore, streaming data often exhibits spatial and temporal correlations, such as periodicity and time-dependence, making it inefficient to optimize each new environment from scratch. Moreover, SDDEAs require concept drift detection mechanisms to identify and segment distinct environments within the infinite data streams. This mechanisms ensure the optimization process remains adaptive, efficient, and responsive to the evolving problem landscape.

Despite significant progress in DDEAs, research on SDDEAs remains relatively limited. Leveraging knowledge from past environments in continuously streaming data can substantially enhance optimization in the current environment. One class of methods involves enriching the current dataset with the best-found solutions from past environments. For instance, SAEF-1GP [24] augments the current dateset with best-found solution from the mos recent environment, relying heavily on real fitness evaluations. Similarly, SAEA/MPCP [40] employs clustering and differencing techniques to predict new data based on the best-found solutions from the two most recent environments. SAEA-TL [41] selects historical based on models with lower root mean square error (RMSE) in the current environment. Additionally, DETO [25] clusters past

model parameters and integrates solutions from environments nearest to the cluster centroid.

Another class of SDDEA methods focuses on leveraging past surrogate models to improve current performance. DSE-MFS [26] maintains a model pool, updating existing models and adding a new one for each environment, then combines them via weighted integration. MLO [27] adopts a meta-learning framework, using parameters from past models to initialize models configuration for optimization in the new environment.

These approaches highlight the potential of utilizing past knowledge, whether through data or model transfer, to improve optimization in dynamic environments under data streams. However, several limitations remain in existing methods. Firstly, existing methods do not explicitly address unknown concept drifts in data streams, as they assume concept drift is known a priori and occurs whenever a new batch of data is introduced. Moreover, they typically assume simultaneous access to all data in the new environment, a condition incompatible with continuously streaming data that arrives incrementally. In addition, some methods rely on only the most recent high-quality solutions for memory-based population initialization, which is fragile in the presence of recurrent drifts. Furthermore, some methods depend on active sampling of specific solutions during the optimization process, which can be impractical. These challenges underscore the urgent need for further research and innovation in SDDEAs to develop algorithms capable of handling streaming data, unknown concept drifts, and real-time adaptability effectively.

C. Concept Drift in Data Streams and Handling Approaches

In this paper, we are going to propose a novel drift-aware SDDEA. Before proceeding, it is important to review the basics of concept drift and traditional methods for handling it in streaming data mining [42]. Concept drift refers to changes in the conditional distribution of objective values given static decision variables [43], and is typically classified into three types: i) Sudden drift, characterized by abrupt changes. ii) Incremental drift, involving progressive transitions. iii) Recurrent drift, where past concepts reappear after a prolonged period. In many real-world applications, data streams are subject to one or more types of concept drift, with unpredictable timing [44]. For example, weekday rush-hour traffic patterns can be disrupted by accidents, weather, or other unforeseen events, impacting traffic speeds and congestion levels [45]. While financial markets fluctuate due to market trends, policy shifts, and other external factors [46].

Concept drift can degrade model performance, making timely and accurate detection crucial. A common strategy monitors prediction error rates, identifying drift when statistically significant changes occur [47]. The Drift Detection Method (DDM) [48], a widely used technique, employs two thresholds: surpassing the warning threshold triggers current model retraining while exceeding the drift threshold triggers model replacement. Over the years, various extensions have since been proposed. EDDM [49] enhances detection sensitivity by analyzing the distance between consecutive misclassifications. HDDM [50] applies Hoeffding's inequality to set error bounds, while FW-DDM [51] performs incremental drift detection using fuzzy time windows.

These methods work well for classification tasks where predictions are discrete [52]. However, in optimization problems, the objective values typically span a complex optimization space with infinite possibilities, often resulting in a regression-type problem that violates the boundedness assumptions of these methods [53]. The wide variability in objective values leading to large differences in prediction errors, making the model's error highly sensitive to individual points. As a result, traditional "one-hit-then-detect" methods, which trigger drift detection after a single large error, are more prone to false positives due to this sensitivity. This highlights the need for drift detection mechanisms specifically tailored to the context of SDDEAs.

III. Proposed Algorithm

A. Framework

The framework of DASE is depicted in Fig. 1. Initially, the first environment is built using the initial dataset. As data streams arrive, the data at each time point is evaluated by the hierarchical confidence drift detector (HCDD) (Section III-B) to determine whether it belongs to current environment. When the detector signals a concept drift, a new environment is constructed. Dynamic environments often exhibit correlations between successive environments, making optimization from scratch inefficient without leveraging previous knowledge [4].

To address this, we maintain an archive (Section III-C), that stores historical environmental information. This archived knowledge facilitates the warm-up construction of new environments (Section III-E) through the adaptive knowledge transfer mechanism (Section III-D).

The pseudo code of DASE is provided in Algorithm 1, with its building blocks detailed in the subsequent subsections. The source code is available at https://github.com/YTALIEN/DASE.

B. Hierarchical Confidence Drift Detector

In the continuously streaming environment, data arrives sequentially, the distributional relationships between data points are typically unknown, and concept drift cannot be identified a priori. When the data distribution changes, the accuracy of models constructed in the previous environment deteriorates, leading to a decline in algorithmic performance.

Considering a sequence of instances represented as pairs $X_i = (\boldsymbol{x}_i, y_i)$. The error rate predicted by models can be calculated as

$$ER_i = \begin{cases} \left| \dfrac{y_i - \hat{y}_i}{y_i} \right|, & y_i \neq 0 \\ |y_i - \hat{y}_i|, & \text{otherwise} \end{cases} \tag{1}$$

where $\hat{y}_i$ denotes the object value predicted by models.

For an objective space, the error rate is treated as a Gaussian-like random variable (an empirical validation is provided in the Section III of the supplementary material). In a static environment, the probability distribution of the data

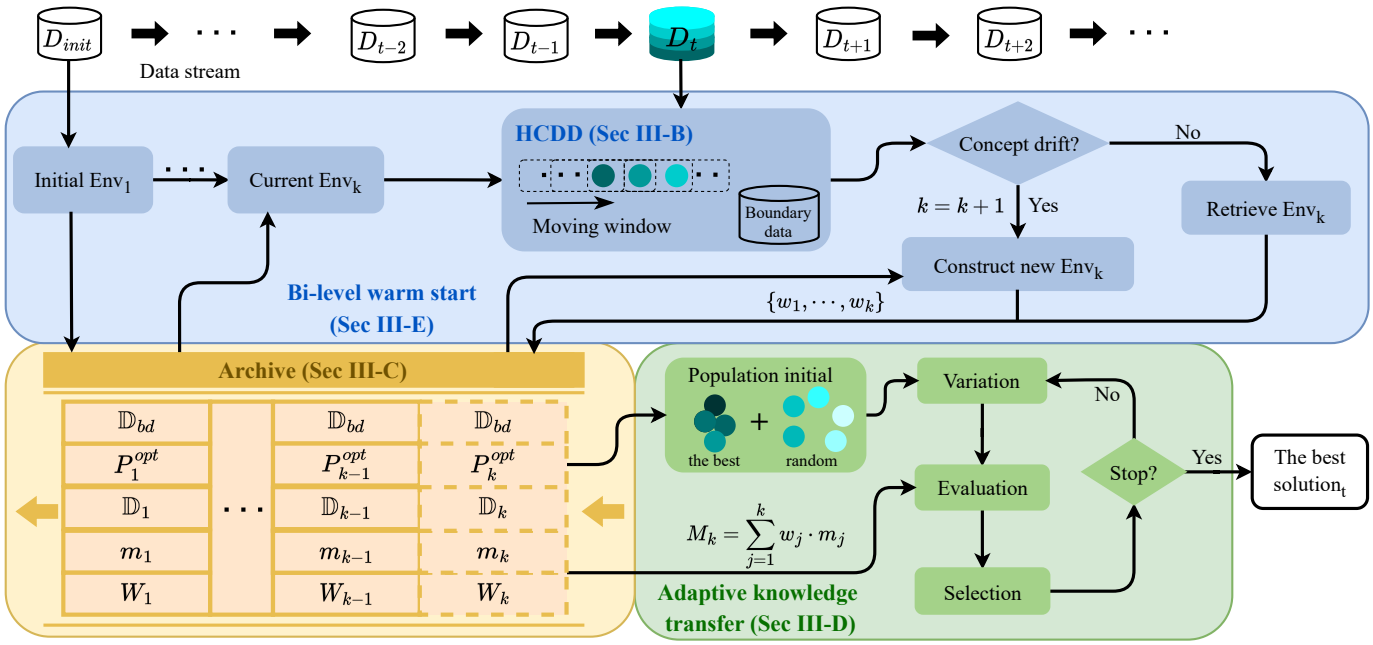


Fig. 1. The framework of the DASE.

Algorithm 1 DASE

Input:
$\mathbb{D} = \{\cdots, X_t, \cdots\}$: Data streams; n^{init}: The sample size of the initial environment; n^{p}: The population size; n^{opt}: The optimal population size; $\text{arc}_{\max}$: The maximum number of environments stored in the archive; mhL_i: The maximum hit limit for confidence level L_i, $i = 1, 2, 3$; u: The model update period.

Output: The optimal solution for time t

$Arc \leftarrow \varnothing$, $k \leftarrow 1$, $\mathbb{D}_{\text{init}} \leftarrow \mathbb{D}(1, n^{\text{init}})$
$P \leftarrow$ Latin hypercube sampling ▷ Initialization
m_k, W_k, $P_k^{\text{opt}} \leftarrow$ InitialEnv($\mathbb{D}_{\text{init}}$) ▷ Build env 1
$Arc \leftarrow Arc \cup \{\{\mathbb{D}_{\text{init}}, m_k, W_k, P_k^{\text{opt}}\}\}$ ▷ Store env 1
while not the end of $\mathbb{D}$ **do**
  $X_t \leftarrow \mathbb{D}(t)$
  $\mathbb{D}_k, m_k, W_k, P_k^{\text{opt}} \leftarrow Arc(k)$ ▷ Retrieve current env
  $Flag_{\text{drift}}$, $\mathbb{D}_{\text{bd}} \leftarrow$ HCDD(mhL_i, m_k, $\mathbb{D}$, $\mathbb{D}_k$) ▷ Drift detect
  if $Flag_{\text{drift}}$ = True **then**
    $k \leftarrow k + 1$
    m_k, W_k, $P_k^{\text{opt}} \leftarrow$ NewEnv(Arc, $\mathbb{D}_{\text{bd}}$) ▷ Build new env
    $n \leftarrow |\mathbb{D}_k| + 1$
  else if n mod u = 0 **then**
    $m_k \leftarrow$ UpdateModel($\mathbb{D}_k$) ▷ Update model
  end if
  $P_k \leftarrow$ ReusePopulation(Arc) ▷ Pop level's warm-start
  $M_k \leftarrow$ Eq. 5 ▷ Model level's warm-start
  $P_k^{\text{opt}} \leftarrow$ EvolvePopulation(P_k, M_k) ▷ Optimization iteration
  $\mathbb{D}_k \leftarrow \mathbb{D}_k \cup \{X_t\}$ ▷ Update dataset
  $Arc \leftarrow Arc \cup \{\{\mathbb{D}_{\text{bd}}, \mathbb{D}_k, m_k, W_k, P_k^{\text{opt}}\}\}$
  if len(Arc) > $\text{arc}_{\max}$ **then**
    $Arc \leftarrow Arc$ / $\{Arc(1)\}$ ▷ Update archive
    $k \leftarrow k - 1$
  end if
  return $P_k^{\text{opt}}(1)$
end while

remains stable, and the model's prediction errors will fluctuate around a fixed mean. We characterize the current environment ($\mathbb{D}_k = \{X_1, \cdots, X_q\}$) by the mean and standard deviation of the model error rates as

$$\mu = \frac{1}{q}\sum_{i=1}^{q} ER_i\ , \quad \sigma = \sqrt{\frac{1}{q}\sum_{i=1}^{q}(ER_i - \mu)^2} \tag{2}$$

where q is the number of data points collected for the current environment.

To detect concept drift, we assess whether there has been a significant change in the error rate. Accordingly, a warning interval is defined as $[\mu + \lambda \times \sigma, \infty)$, where λ is a constant determined by the desired confidence level. By monitoring whether the error rate of incoming data frequently falls inside this interval, we can detect significant changes and identify when concept drift occurs.

Multiple concept drifts of varying magnitudes can occur in complex data streams, ranging from incremental to sudden, as mentioned in Section II-C. Smaller degrees of change may go undetected if the warning interval is too narrow, while overly broad warning intervals can lead to false positives, misidentifying stable distributions as drifting. To address these challenges and account for varying degrees of drift, we adopt a hierarchical confidence strategy with tri-level confidence strategy. The first confidence level L_1 is set at $67\% + (1 - 67\%)/2 = 83.5\%$, with a warning interval of $[\mu + \sigma, \infty)$. The second confidence level L_2 is set at $95\% + (1 - 95\%)/2 = 97.5\%$, corresponding to a warning interval of $[\mu + 2\sigma, \infty)$. Lastly, the third confidence level L_3 is set at $99\% + (1 - 99\%)/2 = 99.5\%$, with a warning interval of $[\mu + 3\sigma, \infty)$. This hierarchical confidence strategy enables effective detection of drifts with smaller magnitudes while maintaining sensitivity to larger changes, thus minimizing false positives and improving overall detection accuracy.

In addition, the complex distribution of the fitness landscape in the objective space poses challenges for models' accuracy

across all regions. In well-fitted regions, the approximation error rate is relatively low. However, in more challenging areas, such as near-peak regions where fitness values exhibit high variability, models struggle to accurately approximate the true landscape. Consequently, an elevated error rate for a single instance at one time dose not necessarily indicate concept drift, as it may stem from model approximation errors. This issue is particularly problematic in "one-hit-then-detect" methods, which may falsely identify drift based on isolated errors, without accounting for the inherent variability of complex optimization landscape. To enhance drift detector reliability, the proposed HCDD employs a moving window (MW) and a multiple-hit confirmation strategy. The MW continuously updates with new data points while discarding the oldest, ensuring the detector reflects the most recent data distribution rather than relying on single instance evaluations. Additionally, each confidence level is assigned a hit counter (hL_i for $i = 1, 2, 3$), and drift is confirmed when the corresponding hit count reaches its predefined maximum. This method effectively mitigates false positives caused by model approximation errors, enhancing the robustness of the drift detection mechanism.

The steps of HCDD are summarized in Algorithm 2. Initially, the distribution of the current environment is calculated using all data in the dataset $\mathbb{D}_k$. More specifically, as data continuously arrives, $\mathbb{D}_k$ expands until drift is detected. Upon detecting a new environment, $\mathbb{D}_k$ is reset to the boundary data $\mathbb{D}_{\text{bd}}$, which includes data points falling inside the warning intervals identified by the drift detector. Otherwise, $\mathbb{D}_k$ continues expanding. Then tri-level confidence thresholds are established, with their corresponding hit counts initialized to zero. As streaming data continuously arrives, the MW updates by adding the newest data point and removing the oldest. The relative mean ($\hat{\mu}$) and standard deviation ($\hat{\sigma}$) of MW are computed to represent the most recent data distribution. The updated distribution is sequentially compared against the tri-level confidence thresholds to determine if it falls inside any of the corresponding warning intervals. If it dose, the hit count associated with the respective confidence level is increased. When the hit count for any confidence level reaches its predefined maximum hit limit (mhL_i for $i = 1, 2, 3$), the drift is considered to have occurred. For instance, if $\hat{\mu}+\hat{\sigma} \geq \mu+3\sigma$, then $hL_3 \leftarrow hL_3 + 1$. Drift is detected when hL_3 reaches mhL_3. Incorporating the moving window and multiple hit strategies reduces false positives caused by inaccuracies of the model approximations in complex optimization problem spaces. Furthermore, the hierarchical confidence strategy enhances sensitivity and enables more accurate detection of varying degrees of drift. Finally, data points falling inside the warning intervals are added to a boundary data zone and stored in the archive for future use as training data to build new models when a new environment emerges.

Algorithm 2 HCDD

Input:
$\mathbb{D} = \{\cdots, X_t, \cdots\}$: Data streams; $\mathbb{D}_k$: The data for the current environment; m_k: The model for current environment; mhL_i: The maximum hit limit for confidence level L_i, $i = 1, 2, 3$;
Output: $Flag_{\text{drift}}$, $\mathbb{D}_{\text{bd}}$

$Flag_{\text{drift}} \leftarrow$ False
$\mu, \sigma \leftarrow \text{GetDistribution}(m_k, \mathbb{D}_k)$ ▷ Via Eq. 2
$\mu + i \times \sigma \leftarrow \text{GetThresholds}(\mu, \sigma), \quad i = 1, 2, 3$
Initialize MW, $hL_i \leftarrow 0$, $\mathbb{D}_{\text{bd}} \leftarrow \varnothing$ ▷ Initialization
while not the end of $\mathbb{D}$ **do**
  MW.moving(X_t) ▷ Add newest data point, remove oldest
  $\hat{\mu}, \hat{\sigma} = \text{GetDistribution}(m_k, MW)$ ▷ Via Eq. 2
  if $\hat{\mu} + \hat{\sigma} > \mu + 3\sigma$ **then**
    $hL_3 \leftarrow hL_3 + 1$ ▷ Data falls inside L_3 interval
    $\mathbb{D}_{\text{bd}} \leftarrow \mathbb{D}_{\text{bd}} \cup \{X_t\}$
  else if $\hat{\mu} + \hat{\sigma} > \mu + 2\sigma$ **then**
    $hL_2 \leftarrow hL_2 + 1$ ▷ Data falls inside L_2 interval
    $\mathbb{D}_{\text{bd}} \leftarrow \mathbb{D}_{\text{bd}} \cup \{X_t\}$
  else if $\hat{\mu} + \hat{\sigma} > \mu + \sigma$ **then**
    $hL_1 \leftarrow hL_1 + 1$ ▷ Data falls inside L_1 interval
    $\mathbb{D}_{\text{bd}} \leftarrow \mathbb{D}_{\text{bd}} \cup \{X_t\}$
  end if
  if $hL_1 = mhL_1$ or $hL_2 = mhL_2$ or $hL_3 = mhL_3$ **then**
    $Flag_{\text{drift}} \leftarrow$ True ▷ Concept drift detected
    break
  end if
end while
return $Flag_{\text{drift}}$, $\mathbb{D}_{\text{bd}}$

C. Archiving Strategy

The necessity of an archiving strategy in optimization under data streams arises from three primary reasons. First, continuous data arrival and limited memory capacity make it impractical to store all incoming data. A representative subset, reflecting the current distribution, is retained to construct high-fidelity models for the current environment. Second, storing information from previous environments prevents restarting optimization from scratch during transitions to new environments, enabling faster adaptation to new environments. Third, outdated information becomes irrelevant to the current environment due to the time-sensitive nature in data streams. Allocating limited storage to more relevant and timely information minimizes computational overhead, enabling the algorithm to adapt rapidly to changing environments and improving real-time responsiveness.

Given these considerations, in our DASE, the archive for the current environment k retains the following key information:

1) *Boundary Data ($\mathbb{D}_{\text{bd}}$)*: This dataset includes data points that fall inside the warning intervals identified by HCDD and are used, along with incoming data, to build new environments.
2) *Dataset ($\mathbb{D}_k$)*: This dataset comprises data collected within the current environment and is incrementally updated as new data streams arrive. It is reset upon detecting environmental changes and serves as the foundation for constructing and updating the model for the current environment.
3) *Models (m_k)*: The models are constructed using the dataset $\mathbb{D}_k$ for the current environment, which capture the distribution of the current environment and the approximate fitness values for optimization. They are periodically updated with newly available data after every u data points to balance real-time responsiveness and stable performance.

4) *Weights (W_k)*: The set of weights quantify the similarity of past environments to the current one, guiding the warm-start strategy of reusing of historical knowledge to enhance optimization efficiency.
5) *The best-found Population (P_k^{opt})*: The best-found population is derived through optimization at each time point in the current environment, with a size of n^{opt}. The best-found population is incorporated into the initial population for the subsequent time point. This approach accelerates optimization and maintains diversity to avoid premature convergence.

At each time point, incoming data associated with the current environment archive information are used to detect drift and identify the emergence of a new environment. If a new environment is detected, historical information is retrieved to facilitate the warm-start phase, accelerating optimization in the new environment. Conversely, if no drift is detected, the current environment's information is retrieved to continue optimization without models reconstruction. The archive is updated accordingly. To manage storage constraints, when the maximum capacity of the archive is exceeded, the oldest environment's information is discarded to make space for the new environment.

D. Adaptive Knowledge Transfer

In continuously streaming environments, limited data available at each time point makes it impractical to ignore previous knowledge. Notably, recurring concepts may appear over time, allowing valuable information from past environments to be reused. Building on the concerns identified, we propose a warm-start strategy for building new environments via an adaptive knowledge transfer mechanism. This section introduces the adaptive knowledge transfer mechanism, with the warm-start strategy detailed in the following Section III-E.

The adaptive knowledge transfer mechanism assigns weights to past environments, prioritizing information from environments that exhibit higher relevance to the current environment. In essence, the similarity between environments j and k is calculated by differences in predicted objective values and approximation errors, as defined in Eq. 3.

$$\begin{aligned} MD_{j,k} &= \sum_{i \in P^{\text{rnd}}} \frac{1}{|P^{\text{rnd}}|} (\hat{y}_i^j - \hat{y}_i^k)^2 \\ AE_{j,k} &= \sum_{i \in \mathbb{D}_k} \frac{1}{|\mathbb{D}_k|} (\hat{y}_i^j - y_i)^2 \\ Sim_{j,k} &= \frac{1}{MD_{j,k} + AE_{j,k} + \delta} \end{aligned} \tag{3}$$

where P^{rnd} is a randomly sampled population from solution space, $\mathbb{D}_k$ contains archived data for environment k with known ground-truth fitness values, and $\hat{y}_i^j$ is the predicted objective value of models in environment j for solution i. A small constant $\delta = 10^{-5}$ is added to avoid division by zero.

The similarity calculation involves two main components. The first is the mapping distance ($MD_{j,k}$), which compares predicted objective values differences from environments j and k on a random population. The second is the approximation error ($AE_{j,k}$), which measures how well the model in environment j approximates the true fitness values in environment k. By combining these two components, a comprehensive quantitative measure of similarity between two environments is obtained.

Once the similarity is determined, the weight of each past environment is calculated using Eq. 4.

$$\begin{aligned} w_k &= \max(0.5, 1 - \frac{k}{arc_{\max}}) \\ w_j &= (1 - w_k) \cdot \frac{Sim_{j,k}}{\sum_{i=1}^{k-1} Sim_{i,k}}, \quad j = 1, \cdots, k-1 \end{aligned} \tag{4}$$

where $Sim_{j,k}$ denotes the similarity between the environment k and j, and $\text{arc}_{\max}$ is the maximum capacity of the archive. The weight of current environment w_k smoothly decreases from 1 to 0.5 as environmental index k approaches the archive capacity, ensuring the newest environment retains at least 50% influence. The residual weight $(1 - w_k)$ is distributed proportionally to environment similarities $Sim_{j,k}$, assigning higher weights to more similar environments.

The adaptive knowledge transfer mechanism reduces the influence of less relevant past environments, prioritizing more similar historical information. This ensures the optimization process efficiently leveraging relevant knowledge, improving performance in dynamic environments.

E. Bi-Level Warm Start

Based on adaptive knowledge weighting, the warm-start strategy utilizes information from past environments at two levels, which contains the models and the best-found populations.

Specifically, the two levels for reusing historical knowledge in the warm-start strategy are as follows:

1) **Weighted Ensemble of Models:** Models from past environments with greater similarity to the current one offer valuable insights into the problem landscape. This enables smoother adaptation to the current problem landscape by utilizing knowledge from different regions of the solution space, captured by models from distinct environments. This enhances the fitting ability of current models, which may otherwise be poorly adapted due to limited data. In cases of recurrent drift, similar environments are likely to reappear. By keeping past models unaltered (i.e., not fine-tuning them with new data of current environment), we prevent catastrophic forgetting of environment-specific patterns. These frozen models serve as "knowledge anchors", preserved for reuse when the corresponding environments reappear.

Based on the adaptive knowledge weights calculated in Section III-D, the ensemble of models for the current environment k is constructed as:

$$M_k = \sum_{j=1}^{k} w_j \cdot m_j \tag{5}$$

2) **Weighted Reuse of the Best-found Populations:** Dynamic environments often display correlations between successive conditions [4], making it advantageous to reuse the

best-found populations from past environments to initialize the current optimization process. The best-found populations from past environments encapsulate valuable information about promising regions in the solution space, which can be leveraged to accelerate the exploration for the the best-found solutions in the current environment.

Hence, we use the best-found populations from past environments as part of the initial population for the new environment. At first, the top n_j individuals from each past the best-found population are selected. For each past environment, $n_j = \min(n^{\text{opt}}, \lfloor n^{\text{p}} \cdot w_j \rfloor)$, where n^{p} represents the population size, w_j denotes the weight of environment j, and the $\min$ operator ensures that the number of reused individuals dose not exceed the size of the best-found population (n^{opt}). These selected individuals are then combined, with the top n^{opt} individuals chosen to form the initial population for the new environment. Additionally, random individuals are included in the initial population to increase diversity and improve the algorithm's exploratory capabilities. The second part of the initial population consists of $n_k = n^{\text{p}} - n^{\text{opt}}$ random individuals.

F. Complexity Analysis

Considering a data stream with N total points processed over N time steps, the overall time complexity of DASE can be analyzed as follows. The time complexity of the HCDD is $O(N)$. For the adaptive knowledge transfer mechanism, the time complexity is dominated by similarity calculation by $O\left(\sum_{k=1}^{N_{\text{env}}} N_k\right)$, where N_k is the dataset size in the k-th environment, and N_{env} indicates the number of environments determined by HCDD. The relationship among N, N_k, and N_{env} is given by: $N \leq \sum_{k=1}^{N_{\text{env}}} N_k \leq |MW| \cdot N$. When the HCDD divides the data streams completely without any overlapping elements, it holds that $\sum_{k=1}^{N_{\text{env}}} N_k = N$. Conversely, in the other extreme scenario where the HCDD starts a new environment to each incoming data point, it results in $\sum_{k=1}^{N_{\text{env}}} N_k = |MW| \cdot N$, since the data within the moving window MW becomes the dataset for the new environment. Consequently, the time complexity of the adaptive knowledge transfer step remains $O(N)$, as $|MW|$ is a constant and can be disregarded. For the bi-level warm start process, the surrogate model construction during the warm ensemble stage is dominated by the construction of RBFN of the new environment, exhibits a time complexity of $O\left(\sum_{k=1}^{N_{\text{env}}} (k_c^2 \cdot N_k)\right)$ for all environments, where k_c represents the number of RBF centers. Typically, k_c is on the order of $\sqrt{N}$, making the complexity of all surrogate construction $O(N^2)$. The population reuse complexity is linear to the n, where n is the population size. Finally, population evolution maintains a time complexity of $O(nN)$, with the number of generations associated with the data stream length N. In summary, the overall time complexity of DASE is $O(N^2 + nN)$.

IV. Experiments

A. Experiment Setup

In the experiments, we utilize SDDObench [23], a benchmark specifically designed for SDDEAs, to evaluate the performance of our proposed algorithm. This benchmark consists of two sets of objective functions combined with five distinct types of concept drifts. In total, it comprises eight diverse problem instances, denotes F1 to F8. The evaluation of SDDObench demonstrates that existing SDDEAs still face significantly challenges in effectively addressing these benchmark instances [23]. A more comprehensive description of the benchmark instances can be found in the Section I-A of the supplementary material.

The experimental section of our study consists of three main experiments. First, we compare our proposed DASE with state-of-the-art (SOTA) DDEAs, including DDEA-SE [54], BDDEA-LDG [55] and TT-DDEA [56] (Section IV-D). Second, we evaluate the performance of our DASE against some SOTA SDDEAs, including DSE-MFS [26], DETO [25], and SAEF-1GP [24] (Section IV-E). Moreover, we conduct some ablation studies to access the contributions of individual components in DASE (Section IV-F). Finally, we perform parameter analysis of DASE, with the results presented in Section IV-G and in Section IV-C of the supplementary material.

To ensure the validity and fairness of our experiments, we adopt the following experimental settings.

- Data Streams: The total number of data points in the data streams is given by $N_{\text{e}} \cdot es$, where N_{e} is the number of environments in each independent run, set to 60, and es is the number of data points in each environment, set to 200. Samples in the data streams are generated using Latin hypercube sampling (LHS) [57]. Most existing methods assume access to batch data and treat each batch as a distinct environment. In our experiments, the data arrives as a continuous stream. To ensure a fair comparison, we introduce an update interval u, restricting existing methods to update their solutions only after accumulating u new data points. Naively setting $es = u$ would unfair benefit these methods by revealing the precise environment shift timing (by aligning u with the true environment boundary), while our method needs to detect it without prior knowledge. On the other hand, setting es independently of u aligns with practical situations with unknown drifts, but it could cause the previous methods to become very unstable due to mismatched batch expectations. Thus, we set es as a multiple of u, which ensures that the previous methods can still perform stably while avoiding to leak knowledge of drift timing (providing a relatively fair test surrounding for DASE). Fig. 1 in the supplementary material provides a comparative illustration of the data usage protocols between DASE and the compared methods.
- Evolutionary Optimizers: The number of optimal iterations, i, at each time point is set to 30, and the population size is uniformly set to 100.
- Other Parameters: All other parameters remain consistent with those outlined in the original literature to ensure optimal performance and a fair comparison.

Each problem instance is carried out in 10 independent runs, and the results are reported as the average and standard

deviation.

Additionally, we employ the Kruskal-Wallis test [58] followed by post hoc Dunnett's test [59], with a Bonferroni correction strategy [60]. The Kruskal-Wallis test is used to assess whether there are significant differences among the compared algorithms. If the test indicates a significant effect, post hoc Dunnett's test is then used to perform multiple comparisons between every two algorithms. The Bonferroni correction is applied to adjust the p-values, reducing the risk of type I errors due to multiple testing. The significance level is set to 0.05. To indicate performance comparisons, we use "+" to denote that DASE outperforms a competing methods, "≈" to indicate similar performance, and "−" to denote underperformance. For each problem instance, the best-performing algorithm is highlighted in **bold**. The average rank of each algorithm across all problem instances is also provided in the table, offering a comprehensive comparison of their performance.

The experiments are conducted on an environment with following specifications: Intel(R) Xeon(R) E5-2696 v3 @ 2.30GHz with 128G of RAM. The implementation of our code is done in Python. It is worth noting that the algorithms used for comparison were implemented by their respective original authors.

B. Algorithmic Settings

An overview of the parameter settings for DASE is presented in the Table I and further elaborated below. For the surrogate models in the current environment, we use radial basis function network (RBFN) due to its excellent performance in numeric approximation and low computational cost. The number of RBF centers is set to $\sqrt{N}$, where N represents the size of the data used to construct the models. Gaussian function is chosen as the basis function, and the centers are determined using K-means clustering. The weights of the linear layer in the RBFN are calculated through pseudo-inverse method [15].

In the module of HCDD, the moving window size $|MW|$ is set to 50, which is at least equal to the model update interval, ensuring statistical reliability. The maximum hit limits for three levels, mhL_i, $i = 1, 2, 3$, are defined based on the level factor β, which is set to 10. Specifically, mhL_1 is set to 3β, mhL_2 to 2β, and mhL_3 to β.

The maximum archive size in DASE, as $\mathrm{arc}_{\max}$, is set to 30, striking a balance between adequately capturing recurring drifts and minimizing excessive storage requirements. In the bi-level warm start strategy, the proportion of the best-found individuals r^{opt} reused in the initial population is set to 0.2. This implies that the initial population consists of $n^{\mathrm{opt}} = \lfloor n^{\mathrm{p}} \cdot r^{\mathrm{opt}} \rfloor$ the best-found individuals reused from past environments, with the remaining $n^{\mathrm{p}} - n^{\mathrm{opt}}$ individuals being randomly generated.

For the evolutionary optimization process, we adopt the DE/current-to-best/1/bin algorithm, known for its strong global optimization capabilities. The scalar parameter F is set to 0.5 and the crossover rate Cr is set to 0.9, enhancing the population's diversity and exploration ability.

TABLE I
PARAMETER SETTINGS OF DASE

Module	Parameter	Value
HCDD	Moving window size	50
	Level factor β for maximum hit limits	10
Surrogates	Number of RBF centers k_{c}	$\sqrt{N}$*
Archive	$\mathrm{arc}_{\max}$	30
Warm start	r^{opt}	0.2
Optimization process	DE/current-to-best/1/bin F	0.5
	DE/current-to-best/1/bin Cr	0.9

*N is the size of the data used to construct the surrogates

C. Performance Metrics

Since the landscape and optimum solution change across different environments, it is essential to evaluate how algorithms track the the best solution under such dynamic environments. In this paper, we assess the performance of SDDEAs using two widely-used and effective performance measures, as described in [23].

- **Online Error** ($E^{(t,i)}_{\mathrm{online}}$): This metric evaluates the error between the best-found solution and the global optimal solution for specific iteration. It is formally defined as:

$$E^{(t,i)}_{\mathrm{online}} = f(x^{*(t,i)}) - f(x^{\star(t)}) \tag{6}$$

  where $x^{*(t,i)}$ is the best solution found after i-th iteration in the t-th time point, and $x^{\star(t)}$ is the global optimal solution in the t-th time point.
- **Offline Error** (E_{offline}): It computes the average error between the best-found solution and the global optimal solution across iterations. It is defined as:

$$E_{\mathrm{offline}} = \frac{1}{TI}\sum_{t=1}^{T}\sum_{i=1}^{I}\left[f(x^{*((t-1)I+i)}) - f(x^{\star(t)})\right] \tag{7}$$

  where T is the total number of time points, given by $N_{\mathrm{e}} \cdot es$, I is the iterative number in the current environment, $x^{*((t-1)I+i)}$ is the best solution found at the i-th iterative evaluation in the t-th time point, $x^{\star(t)}$ has the same definition as in Eq. 6.

These metrics provide a comprehensive evaluation of the algorithms ability of tracking for optima in dynamic environments. The E_{offline} metric is used to compare the overall performance of the algorithms (as shown in Table II and Table III), while E_{online} illustrates the convergence trajectories of the compared algorithms (as shown in Fig. 2).

D. Comparison with DDEAs

In this subsection, we compare our proposed DASE with several SOTA DDEAs (DDEA-SE, BDDEA-LDG and TT-DDEA). Although these algorithms are primarily designed for solving problems in static environments, they incorporate advanced techniques such as surrogate ensembles and semi-supervised learning, which enhance their search capabilities in optimization tasks. Therefore, we conduct this experimental comparison to further illustrate the effectiveness of our DASE in comparison to these DDEAs. A brief description of each algorithms is provided in the Section IV-A of the supplementary material. These algorithms introduce various enhancements to

TABLE II
THE MEAN AND STANDARD DEVIATION OF E_{offline}

Instance	Drift	DDEAs			SDDEAs			DASE
		TT-DDEA	DDEA-SE	BDDEA-LDG	DETO	DSE-MFS	SAEF-1GP	
F1	D1	6.64e+01±3.03e-02 (+)	6.55e+01±1.26e-01 (+)	6.56e+01±1.02e-01 (+)	6.75e+01±8.04e-01 (+)	6.72e+01±3.21e-02 (+)	6.55e+01±3.43e-02 (+)	**6.05e+01±2.83e-02**
	D2	6.54e+01±2.59e-02 (+)	6.53e+01±5.88e-02 (+)	6.54e+01±1.13e-01 (+)	6.64e+01±2.64e-02 (+)	6.27e+01±7.15e-02 (≈)	6.86e+01±6.29e-02 (+)	**6.30e+01±4.25e-02**
	D3	6.43e+01±4.96e-02 (+)	6.31e+01±1.04e-01 (+)	6.32e+01±8.64e-02 (+)	6.53e+01±1.55e-01 (+)	6.49e+01±2.09e-01 (+)	6.32e+01±7.08e-02 (≈)	**6.26e+01±3.55e-02**
	D4	6.27e+01±2.26e-01 (+)	6.21e+01±5.39e-02 (+)	6.21e+01±1.19e-01 (+)	6.45e+01±1.84e+00 (+)	6.46e+01±6.72e-02 (+)	6.26e+01±3.55e-02 (+)	**5.48e+01±7.16e-02**
	D5	6.34e+01±5.99e-02 (+)	6.22e+01±7.38e-02 (+)	6.24e+01±5.28e-02 (+)	6.46e+01±6.67e-01 (+)	6.47e+01±3.28e-02 (+)	6.30e+01±4.20e-02 (+)	**5.96e+01±4.27e-02**
F2	D1	6.64e+01±4.39e-02 (+)	6.55e+01±8.85e-02 (+)	6.57e+01±1.10e-01 (+)	6.75e+01±1.34e+00 (+)	6.72e+01±5.99e-02 (+)	6.54e+01±4.17e-02 (+)	**6.25e+01±3.69e-02**
	D2	6.62e+01±3.93e-01 (+)	6.45e+01±6.58e-01 (≈)	6.52e+01±1.50e-01 (+)	6.69e+01±5.75e-01 (+)	6.49e+01±7.63e-01 (+)	6.86e+01±8.61e-02 (+)	**6.46e+01±1.01e-02**
	D3	6.55e+01±4.11e-01 (+)	6.44e+01±4.83e-01 (+)	6.46e+01±3.53e-01 (+)	6.61e+01±1.22e+00 (+)	6.59e+01±2.36e-01 (+)	6.32e+01±3.77e-02 (≈)	**6.30e+01±3.46e-02**
	D4	6.53e+01±1.16e+00 (+)	6.48e+01±4.38e-01 (+)	6.38e+01±7.51e-01 (+)	6.63e+01±2.74e+00 (+)	6.59e+01±6.39e-01 (+)	6.26e+01±6.15e-02 (+)	**6.00e+01±1.01e-02**
	D5	6.65e+01±1.01e+00 (+)	6.51e+01±5.24e-01 (+)	6.45e+01±2.64e-01 (+)	6.79e+01±1.80e+00 (+)	6.62e+01±3.58e-01 (+)	6.31e+01±3.26e-02 (≈)	**6.23e+01±8.50e-02**
F3	D1	6.64e+01±4.45e-02 (≈)	6.55e+01±1.11e-01 (-)	6.56e+01±6.11e-02 (-)	6.75e+01±2.82e+00 (+)	6.71e+01±4.53e-02 (≈)	**6.55e+01±3.55e-02 (-)**	6.67e+01±1.12e-02
	D2	6.49e+01±7.74e-02 (+)	6.43e+01±1.44e-01 (+)	6.41e+01±2.51e-02 (+)	6.55e+01±3.86e-02 (+)	6.30e+01±3.11e-02 (≈)	**5.77e+01±1.14e-01 (-)**	6.24e+01±4.82e-02
	D3	6.22e+01±1.03e-01 (+)	6.12e+01±1.81e-01 (≈)	6.12e+01±2.94e-02 (≈)	6.27e+01±2.18e+00 (+)	6.29e+01±8.72e-02 (+)	**5.97e+01±5.86e-02 (-)**	6.07e+01±5.79e-02
	D4	6.12e+01±1.92e-01 (+)	6.09e+01±1.81e-01 (+)	6.10e+01±7.84e-02 (+)	6.26e+01±1.22e+00 (+)	6.28e+01±8.31e-02 (+)	6.02e+01±6.74e-02 (+)	**5.37e+01±1.05e-02**
	D5	6.16e+01±9.66e-02 (+)	6.08e+01±1.34e-01 (+)	6.08e+01±1.48e-01 (+)	6.25e+01±9.62e-01 (+)	6.26e+01±7.38e-02 (+)	6.05e+01±3.85e-02 (+)	**5.83e+01±1.12e-02**
F4	D1	5.27e+00±4.00e+00 (+)	1.90e+00±8.86e-02 (+)	8.64e-01±2.74e-02 (+)	4.87e+01±1.69e+00 (+)	5.01e+01±2.45e+00 (+)	8.94e+00±1.07e-01 (+)	**6.46e-04±4.85e-05**
	D2	4.12e+01±1.23e+01 (+)	8.25e+00±2.66e-01 (+)	6.67e+00±5.96e-02 (+)	5.86e+01±9.53e-01 (+)	2.32e+01±9.04e-01 (+)	9.56e+00±1.63e-01 (+)	**3.20e+00±1.57e-01**
	D3	7.11e+01±4.32e+01 (+)	8.52e+00±2.86e-01 (+)	6.42e+00±1.51e-01 (+)	4.79e+01±1.13e+00 (+)	1.72e+01±1.32e+00 (+)	9.65e+00±1.22e-01 (+)	**1.55e+00±2.63e-01**
	D4	2.68e+01±2.89e-01 (+)	1.40e+01±2.74e-01 (+)	1.44e+01±1.37e-01 (+)	4.22e+01±9.34e-01 (+)	2.30e+01±1.89e+00 (+)	1.02e+01±2.71e-01 (+)	**5.19e+00±3.21e-01**
	D5	4.34e+01±2.08e+01 (+)	1.43e+01±3.45e-01 (+)	1.34e+01±2.85e-01 (+)	4.36e+01±1.15e+00 (+)	2.56e+01±2.55e+00 (+)	1.17e+01±1.91e-01 (+)	**4.99e+00±8.57e-01**
F5	D1	5.44e+01±1.60e+01 (+)	5.83e+01±2.27e+00 (+)	4.29e+01±5.86e-01 (+)	1.65e+03±1.46e+01 (+)	2.90e+03±7.20e+01 (+)	1.60e+02±1.29e+00 (+)	**4.21e+01±7.54e+00**
	D2	1.63e+03±9.63e+02 (+)	2.09e+02±6.47e+00 (+)	1.80e+02±8.98e-01 (+)	1.37e+03±7.01e+01 (+)	1.07e+03±8.86e+01 (+)	1.96e+02±4.81e+00 (≈)	**1.33e+02±5.37e+00**
	D3	3.88e+03±5.52e+02 (+)	2.10e+02±6.74e+00 (+)	1.61e+02±5.35e+00 (≈)	9.71e+02±9.10e+00 (+)	9.26e+02±2.40e+02 (+)	1.92e+02±2.07e+00 (+)	**1.63e+02±1.33e+01**
	D4	3.81e+03±1.25e+03 (+)	4.77e+02±1.13e+01 (+)	4.58e+02±9.32e+00 (+)	1.56e+03±1.26e+01 (+)	1.21e+03±9.16e+01 (+)	2.83e+02±6.03e+00 (+)	**1.69e+02±1.60e+01**
	D5	1.25e+03±4.42e+02 (+)	4.83e+02±1.24e+01 (+)	4.45e+02±1.16e+01 (+)	1.56e+03±1.58e+01 (+)	1.25e+03±6.50e+01 (+)	3.14e+02±4.08e+00 (≈)	**2.06e+02±3.60e+01**
F6	D1	2.11e+03±9.36e+02 (+)	1.20e+01±9.76e-02 (+)	1.13e+01±7.63e-02 (+)	2.02e+01±4.09e-02 (+)	2.11e+01±1.01e-02 (+)	1.81e+01±2.24e-02 (+)	**6.65e+00±1.06e-01**
	D2	3.99e+01±2.65e+00 (+)	1.61e+01±4.09e-02 (≈)	1.60e+01±4.62e-02 (≈)	2.04e+01±1.68e-02 (+)	1.96e+01±3.99e-02 (+)	1.82e+01±7.82e-02 (+)	**1.22e+01±1.11e-01**
	D3	3.62e+01±3.67e+00 (+)	1.71e+01±1.03e-01 (≈)	1.70e+01±1.21e-01 (≈)	2.04e+01±4.70e-01 (+)	1.92e+01±7.54e-02 (+)	1.82e+01±2.73e-02 (+)	**1.77e+01±6.02e+00**
	D4	3.98e+01±1.61e+00 (+)	1.77e+01±8.53e-02 (+)	1.75e+01±1.98e-02 (+)	2.07e+01±6.30e-01 (+)	2.02e+01±8.80e-02 (+)	1.87e+01±1.58e-02 (+)	**1.58e+01±3.39e-01**
	D5	4.08e+01±6.92e+00 (+)	1.75e+01±1.04e-01 (+)	1.74e+01±3.73e-02 (+)	2.06e+01±3.38e+00 (+)	2.02e+01±6.82e-02 (+)	1.87e+01±2.61e-02 (+)	**1.57e+01±3.82e-01**
F7	D1	4.96e+01±2.57e+00 (+)	9.27e-01±1.03e-02 (+)	8.51e-01±1.70e+00 (+)	9.29e-01±2.90e+00 (+)	1.01e+00±1.84e+00 (+)	6.58e-01±7.79e-02 (+)	**9.86e-02±1.27e-03**
	D2	1.93e+00±2.65e-01 (+)	9.21e-01±9.54e-01 (+)	9.21e-01±2.78e+00 (+)	8.98e-01±1.08e-01 (+)	1.01e+00±1.97e+00 (+)	5.79e-01±2.04e+00 (+)	**1.02e-01±1.52e-02**
	D3	1.95e+00±4.59e-01 (+)	9.62e-01±1.34e+00 (+)	9.13e-01±4.63e+00 (+)	8.95e-01±2.02e-01 (+)	1.01e+00±6.93e-01 (+)	6.11e-01±1.48e+00 (+)	**1.00e-01±1.36e-02**
	D4	2.18e+00±5.18e-01 (+)	9.53e-01±4.40e+00 (+)	9.13e-01±1.50e-01 (+)	8.61e-01±3.71e-01 (+)	1.01e+00±1.91e+00 (+)	5.87e-01±5.19e-01 (+)	**9.91e-02±1.42e-03**
	D5	1.71e+00±2.83e-01 (+)	9.51e-01±1.12e+00 (+)	9.10e-01±1.32e+00 (+)	8.87e-01±1.22e+00 (+)	1.01e+00±3.73e+00 (+)	7.89e-01±5.90e+00 (+)	**1.01e-01±6.72e-02**
F8	D1	5.04e+01±9.35e-01 (+)	5.30e+01±6.56e-01 (+)	5.22e+01±2.77e-01 (+)	9.22e+01±4.15e+00 (+)	1.00e+02±9.26e-01 (+)	4.75e+01±7.34e-02 (+)	**4.10e+01±1.50e-02**
	D2	2.77e+02±1.34e-01 (+)	6.12e+01±7.01e-01 (+)	6.13e+01±2.74e-01 (+)	8.91e+01±1.26e+00 (+)	8.09e+01±9.90e-01 (+)	5.06e+01±4.60e-01 (≈)	**5.09e+01±2.22e-02**
	D3	1.41e+02±1.76e+01 (+)	6.22e+01±1.70e-01 (+)	6.15e+01±2.38e-01 (+)	8.12e+01±4.14e+00 (+)	7.88e+01±1.19e+00 (+)	**4.92e+01±8.92e-02 (-)**	5.86e+01±3.07e-02
	D4	1.35e+02±1.08e+01 (+)	6.55e+01±2.32e-01 (+)	6.65e+01±1.85e-01 (+)	8.29e+01±1.85e+00 (+)	8.32e+01±1.15e+00 (+)	5.29e+01±1.06e-01 (+)	**5.00e+01±6.99e-02**
	D5	1.22e+02±1.56e+01 (+)	6.70e+01±3.62e-01 (+)	6.72e+01±8.48e-01 (+)	8.35e+01±1.04e+00 (+)	8.39e+01±1.01e+00 (+)	6.67e+01±1.93e-01 (+)	**5.60e+01±1.15e-02**
+/≈/-		39/1/0	35/4/1	35/4/1	40/0/0	37/3/0	30/6/4	NA
Average Rank		5.80	3.17	2.70	6.10	5.32	3.20	1.07

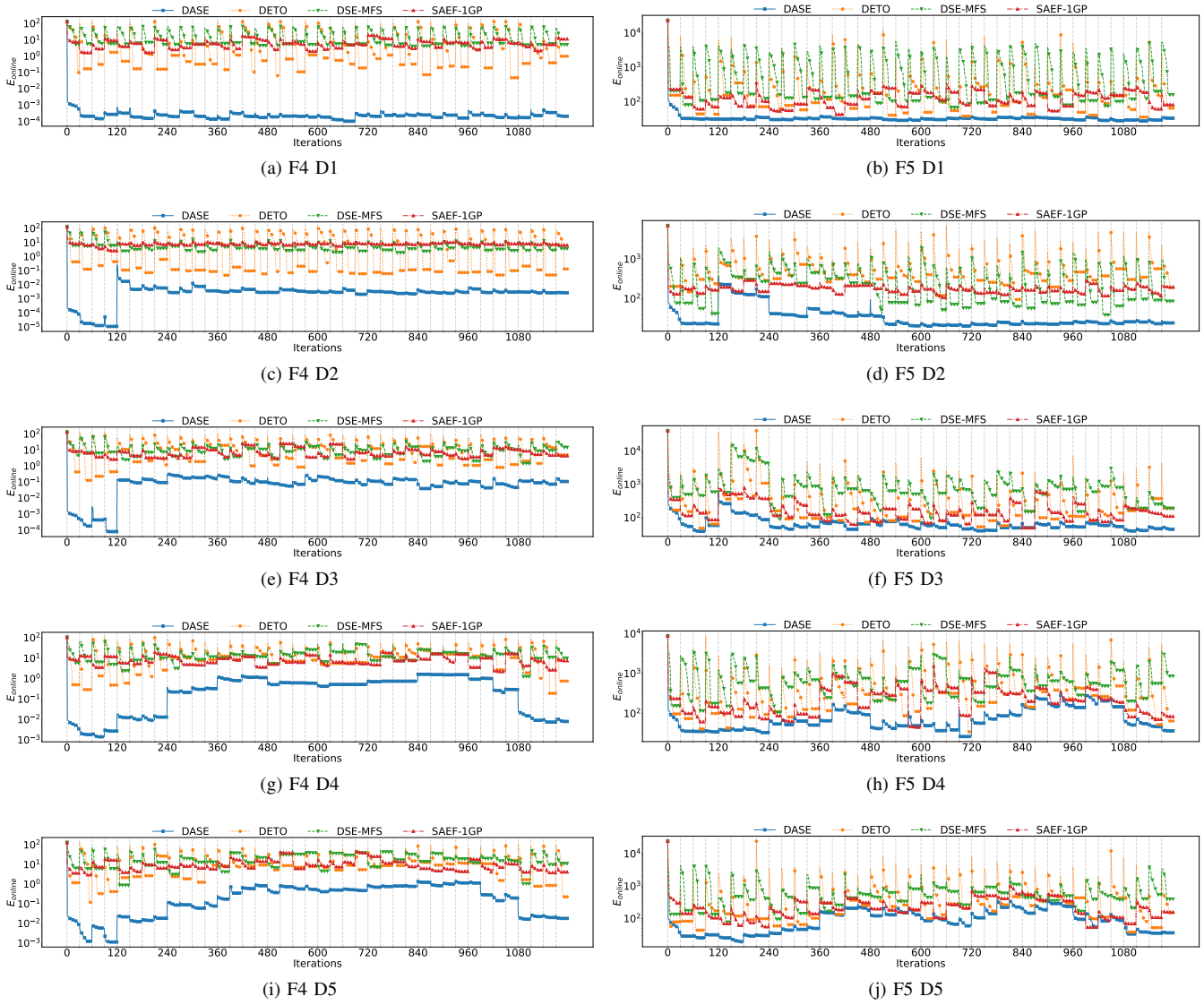
DDEAs, which have been reported in the literature to perform well on problems in static environments.

The results in Table II with dimension 5 (due to space constraints, comparison results for dimensions 10 and 20 are presented in the Section IV-B of the supplementary material) highlight the performance differences among these algorithms. Notably, DASE significantly outperforms all other algorithms across the eight instances of five drift scenarios given the same data streams. Specifically, DASE achieves the lowest average ranking, significantly outperforming all other DDEAs, followed by BDDEA-LDG, DDEA-SE, and finally TT-DDEA in descending order. The performance advantage of DASE is particularly prominent on functions F1, F2, F4, and F8, where the achieved values of E_{offline} are several orders of magnitude lower than those obtained by its competitors. A fundamental strength of DASE lies in its ability to retain and reuse knowledge from past environments. By integrating historical populations and surrogate models into the optimization of the current environment, DASE effectively mitigates the data scarcity problem inherent to data stream environments. This strategy enables DASE to construct more accurate surrogate models at the early stages of a newly detected environment, thus facilitating rapid and reliable convergence. In contrast, algorithms such as BDDEA-LDG and DDEA-SE, though also ensemble-based but only exploit information within the current environment. This limited scope of ensemble construction often results in sparse or biased data distributions, especially under sudden drifts, thereby degrading model accuracy and solution quality.

E. Comparison with SDDEAs

In this subsection, we compare our proposed DASE with several SOTA SDDEAs (SAEF-1GP, DSE-MFS, and DETO). A brief description of each algorithm is provided in the Section IV-A of the supplementary material. These algorithms provide various enhancements to the performance of SDDEAs, achieving promising results in dynamic environments, while overlooking the challenges posed by unknown concept drift in the literature. In this study, we examine their performance in dynamic environments characterized by unknown concept drifts within data streams. In the experiment, all methods are provided with the same number of data points to ensure a fair comparison. Besides, note that our proposed algorithm obtain data through passive method, whereas DETO and SAEF-1GP actively select which data points to query, requiring new queries for each iteration. To adapt these active-query based SDDEAs to passive-query framework, modifications were made to their algorithms. Specifically, for DETO, evaluations of the best-found solution using the real fitness function were replaced with evaluations performed using models. For SAEF-1GP, surrogate model is updated only at the beginning of the optimization process, with updates during the optimal iterations removed, which aligns with [26].

1) Performance Comparison: The comparison results presented in Table II reveal the performance disparities among the SDDEAs. As shown in the table, DASE achieves the smallest average ranking compared to all other algorithms. Moreover, DASE significantly outperforms the other SDDEAs in several instances (F4, F5, F6, F7 and F8) under drift scenarios (D2,



(a) F4 D1 (b) F5 D1

(c) F4 D2 (d) F5 D2

(e) F4 D3 (f) F5 D3

(g) F4 D4 (h) F5 D4

(i) F4 D5 (j) F5 D5

Fig. 2. The convergence trajectories of E_{online} on the first 40 time points of instance F4 and F5

D3, D4 and D5). The observed performance differences, which are several orders of magnitude, demonstrate the efficiency of our proposed algorithm in solving problems in continuously streaming environments.

Among the compared algorithms, SAEF-1GP achieves the second-lowest average performance. This can be attributed to its memory-based initialization, where recent high-quality solutions are used to generate a partially converged initial population. As a result, SAEF-1GP performs well in F1, F2 and F3 under drift scenarios D1, D2 and D3. However, this approach neglects scenarios where concepts from earlier environments reappear in recurrent drift scenarios. Consequently, the models become less reliable when limited data available, leading to poor performance on complex problem landscapes, particularly in F4, F5 and F7 under drifts D4 and D5. DSE-MFS employs a model ensemble strategy, integrating models from past environments. This approach proves effective in drift scenarios D2, D3 and D5 for instances F4, F5, F6 and F8. However, its weighting strategy for the ensemble relies solely on the approximation errors of models in the current environment, disregarding the underlying prediction similarity between different environments. Consequently, DSE-MFS performs poorly under drift D4 across all instances. While DETO considers the prediction similarity between past and current environments based on clustering model parameters, it overlooks the surrogate approximation error in the current environment. Excessive reliance on data from similar environments can lead to inaccuracies in landscape fitting, further exacerbated by older data derived from clustering. Consequently, DETO fails to effectively track the dynamical best-found solution, resulting in worse performance compared to DSE-MFS.

To provide a more intuitive comparison of the convergence of DASE and other SDDEAs during the optimization process in every iteration, Fig. 2 illustrates some convergence trajectories of the average E_{online} (over 10 runs) with respect to the number of iterations for each algorithm on instances F4 and F5 across all drifts scenarios. It is evident from the

plot that, in most time points, DASE is capable of obtaining better solutions with faster convergence compared to the other algorithms under continuously changing environments.

After initialized, DASE starts processing the incoming data arriving as data streams. In the case of drift D1, the drift detection mechanism ensures that DASE identifies the absence of a new environment. In the light of this, it continues to use the preserved information from the current environment for optimization. This approach maintains a smooth convergence curve without sharp oscillations when new data arrives, offering a distinct advantage over the other algorithms. In contrast, for drifts D2, D3, D4 and D5, when a concept drift is detected, DASE leverages the best-found populations from past environments to initialize the optimization process for the new environment. This strategy significantly accelerates convergence within each environment. On the other hand, the compared algorithms do not incorporate a drift detection mechanism for data streams. Instead, they assume by default that the environment changes when a new batch of data arrives, and in that case, they perform some random initialization strategies. As a result, their convergence curves display sharp fluctuations at the arrival of each data batch, often leading to deterioration in the current best-found solutions. Overall, DASE consistently outperforms the other algorithms by achieving better solutions in fewer iterations across all time points and drifts scenarios.

2) Runtime Comparison: We also conduct a practical runtime comparison of different algorithms. As showed in Fig. 3, DASE achieves the shortest average processing time for all time points in the data streams among the compared SDDEAs. Furthermore, under drift D1, DASE exhibits the lowest time expenditure compared to other drift scenarios. This is due to its drift detector mechanism, which allows the algorithm to continue optimizing using the archived models when no drift is detected, avoiding the need to construct new models. This highlights the real-time responsiveness of DASE. For drift scenarios D2 and D3, the time of DASE is slightly higher compared to D1. This is because more abrupt drifts are more likely to trigger the detection mechanism, necessitating the frequent construction of new models to adapt to the changes. As a result, the average processing time for these scenarios increases slightly. Among the other algorithms, SAEF-1GP is the second fastest. Its lower running time is primarily due to the simplicity of its operations, as it trains only a single GP for each batch and reuses information only from the most recent environment. DSE-MFS, on the other hand, requires significantly more time because it updates all the models in its model pool based on the current data batch, a step that is inherently time-intensive. DETO is the slowest among the compared algorithms. Its runtime is heavily influenced by the size of the data streams, as it employs a multi-input GP with a large number of parameters. Additionally, DETO involves saving and clustering models parameters from all past environments, further increasing its computational burden.

F. Ablation Study

In this section, we analyze the contributions of the key components introduced in DASE.

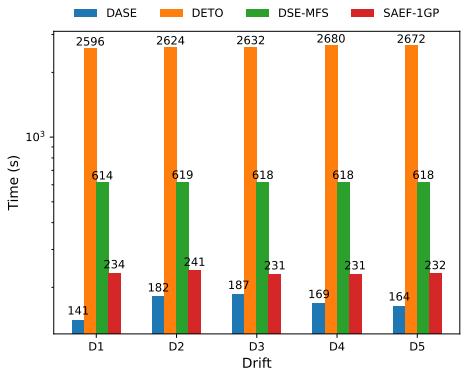


Fig. 3. Runtime comparison among SDDEAs.

1) Key Components Analysis: At first, we conduct a comparative study between DASE and its several variants, detailed as follows.

- **DASE-w/o-L_3**: This variant excludes the confidence level L_3 from the drift detector in DASE. In the drift detect module, only confidence levels L_1 and L_2 are used as the warning intervals for drift detection.
- **DASE-w/o-L_3&L_2**: This variant excludes both confidence levels L_3 and L_2 from DASE. Only confidence level L_1 is employed to calculate the warning interval for identifying concept drift in data streams.
- **DASE-w/o-AdaK**: This variant excludes the adaptive knowledge transfer mechanism (AdaK) and assigns the weights for each past environment in equal.
- **DASE-w/o-WEM**: In this variant, the module for the weighted ensemble of models (WEM) from past environments, based on their similarity to the current environment, is removed. Only the model constructed in the current environment is used as the surrogate for the optimization process.
- **DASE-w/o-WRP**: This variant omits the weighted reuse of populations (WRP) from past environments as the initial population for the current environment is removed. Consequently, only a randomly sampled population is used as the initial population for the optimization process in the current environment.

The result of the experiment is shown as Table III, and our discussion of the experimental results is presented below.

DASE outperforms all of other variants in almost all instances across different drift scenarios. This demonstrates the effectiveness of our approach, which combines drift detection with a warm-start strategy for the current environment. By reusing information stored in the archive from past environments, our method significantly enhances SDDEAs for solving problems in dynamic environments under data streams.

Both DASE-w/o-L_3 and DASE-w/o-L_3&L_2 perform worse than DASE in most cases. This indicates that the HCDD, which leverages three confidence levels, is effective at detecting varying degrees concept drift under diverse data stream characteristics. The absence of L_2 and L_3 reduce sensitivity to larger distribution shifts, making it detrimental for detecting significant drifts. When only L_3 is removed, the HCDD strug-

TABLE III
THE ABLATION STUDY RESULT

Instance	Drift	DASE-w/o-L_3	DASE-w/o-L_3&L_2	DASE-w/o-AdaK	DASE-w/o-WEM	DASE-w/o-WRP	DASE
F1	D1	6.09e+01±2.68e-02 (≈)	6.19e+01±4.57e-02 (+)	6.63e+01±4.05e-02 (+)	6.26e+01±2.46e-02 (+)	6.44e+01±3.94e-02 (+)	**6.05e+01±2.83e-02**
	D2	6.31e+01±2.54e-02 (≈)	6.42e+01±4.50e-02 (+)	6.30e+01±1.03e-02 (≈)	6.44e+01±6.13e-02 (+)	6.48e+01±7.94e-02 (+)	**6.30e+01±4.25e-02**
	D3	6.27e+01±6.59e-02 (≈)	6.45e+01±8.76e-02 (+)	6.32e+01±4.69e-02 (+)	6.38e+01±3.81e-02 (+)	6.38e+01±9.03e-02 (+)	**6.26e+01±3.55e-02**
	D4	6.04e+01±6.50e-02 (+)	6.32e+01±6.22e-02 (+)	6.00e+01±7.73e-02 (+)	6.45e+01±1.07e-02 (+)	6.62e+01±1.17e-02 (+)	**5.48e+01±7.16e-02**
	D5	6.25e+01±8.25e-02 (+)	6.86e+01±4.08e-02 (+)	6.01e+01±7.52e-02 (+)	6.31e+01±7.01e-02 (+)	6.74e+01±2.29e-02 (+)	**5.96e+01±4.27e-02**
F2	D1	6.43e+01±2.35e-02 (+)	6.63e+01±3.69e-02 (+)	6.65e+01±8.42e-02 (+)	6.65e+01±2.12e-02 (+)	6.66e+01±3.66e-02 (+)	**6.25e+01±3.69e-02**
	D2	6.47e+01±5.61e-02 (≈)	6.48e+01±3.70e-02 (≈)	6.48e+01±8.02e-02 (≈)	6.52e+01±6.19e-02 (+)	6.50e+01±2.18e-02 (+)	**6.46e+01±1.01e-02**
	D3	6.33e+01±3.65e-02 (≈)	6.32e+01±9.87e-02 (≈)	6.39e+01±1.46e-02 (+)	6.46e+01±4.69e-01 (+)	6.43e+01±8.71e-01 (+)	**6.30e+01±3.46e-02**
	D4	6.19e+01±1.51e-02 (+)	6.18e+01±2.04e-02 (+)	6.21e+01±6.60e-02 (+)	6.08e+01±1.76e-02 (+)	6.22e+01±2.03e-02 (+)	**6.00e+01±1.01e-02**
	D5	6.37e+01±5.52e-02 (+)	6.42e+01±2.12e-02 (+)	6.44e+01±1.97e-02 (+)	6.29e+01±3.19e-02 (+)	6.24e+01±4.73e-02 (≈)	**6.23e+01±8.50e-02**
F3	D1	6.86e+01±1.38e-02 (+)	6.87e+01±5.16e-02 (+)	6.70e+01±1.42e-02 (≈)	6.91e+01±3.26e-02 (+)	6.69e+01±2.26e-01 (≈)	**6.67e+01±1.12e-02**
	D2	6.31e+01±1.19e-02 (+)	6.38e+01±3.87e-02 (+)	6.26e+01±4.49e-02 (≈)	6.32e+01±5.78e-02 (+)	6.28e+01±2.23e-02 (+)	**6.24e+01±4.82e-02**
	D3	6.26e+01±9.20e-02 (+)	6.31e+01±7.45e-02 (+)	6.08e+01±7.55e-02 (≈)	6.12e+01±6.68e-02 (+)	6.11e+01±2.98e-02 (≈)	**6.07e+01±5.79e-02**
	D4	6.09e+01±1.28e-02 (+)	6.14e+01±7.92e-02 (+)	5.51e+01±1.24e-02 (+)	5.88e+01±1.08e-02 (+)	5.94e+01±5.27e-02 (+)	**5.37e+01±1.05e-02**
	D5	5.89e+01±9.16e-02 (≈)	5.88e+01±3.45e-02 (≈)	5.91e+01±4.59e-02 (+)	5.88e+01±5.28e-02 (≈)	5.90e+01±2.55e-02 (+)	**5.83e+01±1.12e-02**
F4	D1	2.73e-03±1.01e-04 (+)	3.72e-03±6.92e-04 (+)	3.33e-03±5.58e-04 (+)	1.02e-03±4.83e-04 (+)	1.39e-01±1.39e-04 (+)	**6.46e-04±4.85e-05**
	D2	3.34e+00±2.55e-01 (+)	3.67e+00±1.58e-01 (+)	3.25e+00±1.28e-01 (≈)	4.32e+00±4.83e-01 (+)	3.30e+00±2.55e-01 (≈)	**3.20e+00±1.57e-01**
	D3	2.13e+00±8.07e-01 (+)	2.95e+00±3.18e-01 (+)	2.71e+00±3.81e-01 (+)	3.55e+00±7.26e-01 (+)	1.99e+00±8.23e-01 (+)	**1.55e+00±2.63e-01**
	D4	5.83e+00±6.72e-01 (+)	5.42e+00±6.30e-01 (+)	5.37e+00±9.13e-01 (+)	5.33e+00±5.00e-01 (+)	5.24e+00±5.88e-01 (≈)	**5.19e+00±3.21e-01**
	D5	5.00e+00±8.05e-01 (≈)	5.07e+00±9.22e-01 (+)	5.53e+00±4.80e-01 (+)	5.83e+00±3.56e-01 (+)	5.01e+00±5.94e-01 (≈)	**4.99e+00±8.57e-01**
F5	D1	4.90e+01±3.04e+00 (+)	4.47e+01±3.95e+00 (+)	4.26e+01±3.14e+00 (≈)	4.66e+01±3.04e+00 (+)	4.56e+01±1.77e+00 (+)	**4.21e+01±7.54e+00**
	D2	1.33e+02±1.30e+01 (≈)	1.43e+02±1.20e+01 (+)	1.48e+02±1.70e+01 (+)	1.84e+02±1.30e+01 (+)	1.57e+02±3.03e+00 (+)	**1.33e+02±5.37e+00**
	D3	1.93e+02±3.01e+01 (+)	1.79e+02±7.35e+01 (+)	1.92e+02±4.82e+01 (+)	3.24e+02±4.38e+01 (+)	2.57e+02±1.15e+02 (+)	**1.63e+02±1.33e+01**
	D4	1.95e+02±2.88e+01 (+)	1.88e+02±1.71e+01 (+)	2.48e+02±4.48e+01 (+)	2.60e+02±3.00e+01 (+)	2.07e+02±3.60e+01 (+)	**1.69e+02±1.60e+01**
	D5	2.09e+02±2.92e+01 (≈)	2.15e+02±5.03e+01 (+)	3.04e+02±1.59e+01 (+)	3.30e+02±2.96e+01 (+)	2.20e+02±2.05e+01 (+)	**2.06e+02±3.60e+01**
F6	D1	6.77e+00±5.04e-01 (+)	6.82e+00±7.44e-01 (+)	6.66e+00±5.04e-01 (≈)	8.08e+00±9.81e-01 (+)	6.66e+00±4.45e-01 (≈)	**6.65e+00±1.06e-01**
	D2	1.31e+01±4.28e-01 (+)	1.55e+01±9.08e-01 (+)	1.23e+01±4.27e-01 (≈)	1.23e+01±3.75e-01 (≈)	1.24e+01±3.17e-01 (≈)	**1.22e+01±1.11e-01**
	D3	1.89e+01±2.19e+00 (+)	2.01e+01±1.59e+00 (+)	1.98e+01±1.18e+00 (+)	1.86e+01±7.47e-01 (+)	1.85e+01±1.23e+00 (+)	**1.77e+01±6.02e+00**
	D4	1.58e+01±3.27e-01 (≈)	1.62e+01±6.58e-01 (≈)	1.60e+01±5.18e-01 (≈)	1.63e+01±2.60e-01 (≈)	1.59e+01±2.24e-01 (≈)	**1.58e+01±3.39e-01**
	D5	1.59e+01±4.32e-01 (≈)	1.69e+01±3.66e-01 (+)	1.59e+01±4.33e-01 (≈)	1.66e+01±3.27e-01 (+)	1.63e+01±6.08e-01 (+)	**1.57e+01±3.82e-01**
F7	D1	1.55e-01±6.41e-02 (+)	1.44e-01±7.11e-02 (+)	9.77e-01±3.03e-02 (+)	1.77e-01±1.02e-02 (+)	1.94e-01±2.10e-02 (+)	**9.86e-02±1.27e-03**
	D2	2.00e-01±2.34e-02 (+)	2.30e-01±1.55e-02 (+)	1.00e+00±1.55e-01 (+)	2.01e-01±1.88e-02 (+)	2.22e-01±2.40e-02 (+)	**1.02e-01±1.52e-02**
	D3	1.69e-01±1.48e-02 (+)	1.88e-01±1.80e-02 (+)	1.02e+00±1.78e-01 (+)	1.98e-01±1.49e-02 (+)	2.35e-01±1.32e-02 (+)	**1.00e-01±1.36e-02**
	D4	1.01e-01±1.87e-02 (+)	1.51e-01±1.22e-02 (+)	1.01e+00±1.65e-02 (+)	1.30e-01±4.19e-02 (+)	1.92e-01±5.18e-02 (+)	**9.91e-02±1.42e-03**
	D5	1.11e-01±2.88e-02 (+)	1.21e+00±1.29e-02 (+)	1.00e+00±7.11e-02 (+)	1.54e+00±1.17e-02 (+)	1.01e+00±3.12e-02 (+)	**1.01e-01±6.72e-02**
F8	D1	4.95e+01±4.14e-02 (+)	4.65e+01±8.90e-02 (+)	4.11e+01±3.46e-02 (≈)	4.62e+01±2.47e-02 (+)	4.87e+01±6.27e-02 (+)	**4.10e+01±1.50e-02**
	D2	5.57e+01±4.20e-02 (+)	5.72e+01±2.44e-02 (+)	5.33e+01±2.26e-02 (+)	5.48e+01±1.29e-02 (+)	5.37e+01±2.74e-02 (+)	**5.09e+01±2.22e-02**
	D3	5.96e+01±3.10e-02 (+)	6.98e+01±1.81e-02 (+)	5.87e+01±2.11e-02 (≈)	6.15e+01±2.34e-02 (+)	5.87e+01±1.27e-02 (≈)	**5.86e+01±2.22e-02**
	D4	5.34e+01±1.02e-02 (+)	5.59e+01±1.42e-02 (+)	5.13e+01±1.93e-02 (+)	5.60e+01±1.11e-02 (+)	5.12e+01±1.12e-02 (+)	**5.00e+01±6.99e-02**
	D5	5.99e+01±1.34e-02 (+)	6.10e+01±9.64e-02 (+)	5.76e+01±1.32e-02 (+)	6.23e+01±8.49e-02 (+)	6.15e+01±6.94e-02 (+)	**5.60e+01±1.15e-02**
+/≈/-		29/11/0	36/4/0	27/13/0	37/3/0	30/10/0	NA
Average Rank		2.58	4.12	3.87	3.90	4.58	1.00

gles to identify larger-magnitude drifts, leading to noticeable performance degradation.

DASE-w/o-AdaK performs worse than DASE, highlighting the effectiveness of adaptive knowledge transfer. This mechanism assigns higher weights to past environments that are more similar to the current environment, thereby facilitating the reuse of more relevant and useful knowledge.

DASE-w/o-WEM underperforms DASE, particularly under drifts D4 and D5. This underscores the value of using ensembles of models from similar past environments to improve the model accuracy. The recurrence of similar environments provides additional landscape information that significantly enhances optimization performance in the current environment.

DASE-w/o-WRP exhibits inferior performance compared to DASE. This demonstrates that reusing the best-found solutions from similar past environments boosts optimization in the current environment. By initializing populations with some degree of convergence and diversity, WRP contributes to improved performance.

2) In-Depth Analysis of HCDD: We compare HCDD with traditional methods like DDM [48] and EDDM [49] to analyze its drift detection performance more thoroughly. Since DDM and EDDM are originally designed for classification tasks rather than optimization problems, we adapt DDM for regression setting following the approach in [53]. For EDDM, we replace the classification error with a regression error rate (Eq. 1). Specifically, we define an error sample based on a predefined error threshold and then compute the distance between such error samples.

The comparison results are summarized in Table IV, where we report the precision of drift detection, defined as:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

where, TP denotes the number of true positives (correctly identified drift time points), and FP denotes the number of false positives (incorrect drift alarms). Precision is chosen as the primary evaluation metric in this study because it directly reflects the reliability of drift detection –that is, the likelihood that a detected drift corresponds to an true concept drift. In optimization problems under continuously streaming environments, false positives can trigger unnecessary model resets or update, which may hinder the optimization process. Hence, maintaining high precision is essential for preserving the stability and efficiency of the algorithm.

As shown in Table IV, both DDM and EDDM show limited effectiveness in this setting. These methods are originally tailored for classification tasks with discrete, bounded outputs and Bernoulli-distributed errors assumptions. They are well-suited for detecting changes in classification error but do not fit well to optimization tasks. In contrast, optimization problems usually involve a unbounded objective

TABLE IV
THE PRECISION OF DRIFT DETECTION RESULTS ON BENCHMARK INSTANCES F1–F8 IS REPORTED UNDER DRIFT SCENARIOS D2–D5 (SCENARIO D1 IS EXCLUDED FROM EVALUATION, AS IT CONTAINS NO TRUE DRIFT TIME POINTS).

Ins.	Drift	HCDD	DDM	EDDM	Ins.	Drift	HCDD	DDM	EDDM
F1	D2	**0.92**	0.15	0.10	F5	D2	**0.86**	0.16	0.11
	D3	**0.99**	0.14	0.10		D3	**0.85**	0.14	0.10
	D4	**0.98**	0.14	0.10		D4	**0.85**	0.14	0.11
	D5	**0.98**	0.13	0.10		D5	**0.82**	0.14	0.11
F2	D2	**0.97**	0.16	0.10	F6	D2	**0.82**	0.16	0.11
	D3	**0.96**	0.14	0.10		D3	**0.82**	0.14	0.10
	D4	**0.94**	0.13	0.10		D4	**0.82**	0.13	0.11
	D5	**0.90**	0.14	0.10		D5	**0.78**	0.14	0.11
F3	D2	**0.90**	0.15	0.11	F7	D2	**0.79**	0.15	0.10
	D3	**0.92**	0.14	0.10		D3	**0.78**	0.14	0.11
	D4	**0.88**	0.13	0.10		D4	**0.75**	0.13	0.10
	D5	**0.89**	0.13	0.11		D5	**0.75**	0.13	0.10
F4	D2	**0.88**	0.16	0.11	F8	D2	**0.74**	0.15	0.10
	D3	**0.88**	0.13	0.11		D3	**0.74**	0.14	0.10
	D4	**0.87**	0.14	0.10		D4	**0.75**	0.14	0.11
	D5	**0.87**	0.14	0.11		D5	**0.75**	0.13	0.10

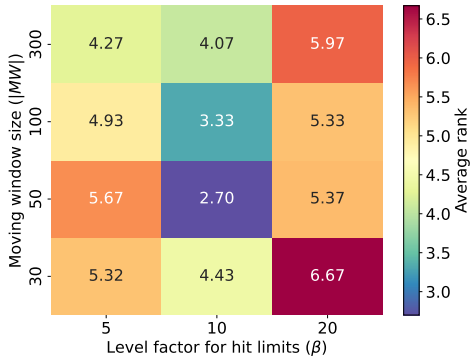


Fig. 4. Parameter sensitivity analysis of β and $|MW|$

values, making them inherently regression-like. The prediction errors in such scenarios are varying with infinite possibilities and exhibit large variability due to the complexity of the fitness landscape. As a result, these traditional methods are more sensitive to individual instances, and their "one-hit-then-detect" mechanisms often lead to false positives when applied directly to optimization tasks, resulting in low detection precision. In comparison, HCDD is specifically designed to address the limitations of traditional drift detectors in data-driven optimization in continuously streaming environments. HCDD treats prediction errors as Gaussian-like variables – better suited for regression-type problems in optimization. It employs a hierarchical tri-level confidence mechanism to detect both sudden and incremental drifts, improving accuracy while reducing false positives. Furthermore, it incorporates a moving window with a multiple-hit confirmation strategy, which improves robustness by suppressing false positives caused by approximation errors from single instance.

G. Parameter Sensitivity Analysis

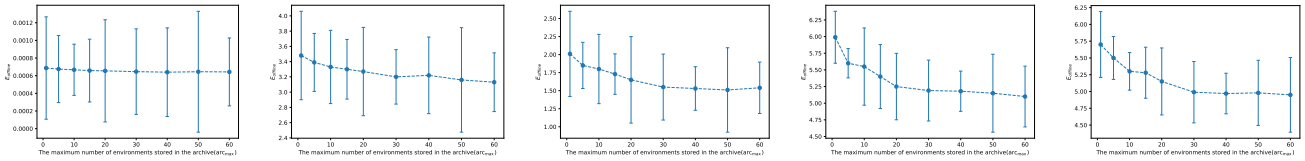
This section is dedicated to conducting a comprehensive sensitivity analysis of several key parameters related to the HCDD and warm start mechanism, aiming to offer insight into the parameter settings and their influence on performance. In addition, the analysis of parameters k_c (number of RBF centers) and n^{P} (population size) is provided in Section IV-C of the supplementary material. As for the remaining parameters, they are empirically set based on well-established conventions in the field of EAs and SDDEAs [13], [15], [25], [26], [55], [56].

1) Parameters in HCDD: $|MW|$ represents the size of the moving window, and β is a level factor that determines the maximum hit limit for the HCDD mechanism, thereby affecting its drift detection sensitivity. We conducted experiments with the following settings: $|MW| \in \{30, 50, 100, 300\}$ and $\beta \in \{5, 10, 20\}$. Our investigation results shown in Fig. 4. It can be observed that a small size of moving window may not capture sufficient statistics to characterize the current environment accurately, leading to unreliable detection. Conversely, a large size of moving window may dilute the sensitivity to change, especially when a window spans multiple environments, which makes the detected distribution less reflective of the present environment. Regarding β, a higher value leads to reduced sensitivity, potentially missing subtle drifts. In contrast, a lower value can result in an increased false positive rate due to the rise of the false positive.

2) Parameter in Bi-level Warm Start: The parameter $\mathrm{arc}_{\max}$ denotes the maximum size of the archive and directly influences the degree of information reuse from previous environments. Fig. 5 illustrates the E_{offline} values obtained when $\mathrm{arc}_{\max}$ is varied across the range $\{1, 5, 10, 15, 20, 30, 40, 50, 60\}$. As shown in the figure, E_{offline} decreases progressively as $\mathrm{arc}_{\max}$ increases up to 30, indicating improved performance due to more effective reuse of archived knowledge. However, beyond this point, the performance gain plateaus, and no significant improvement is observed with further increases. Meanwhile, increasing $\mathrm{arc}_{\max}$ implies a greater computational resources consumption in the ensemble. Hence, to strike a balance between computational resources and optimization performance, we set $\mathrm{arc}_{\max} = 30$ in DASE.

V. CONCLUSION

In this paper, we present DASE, a novel drift-aware streaming evolutionary algorithm designed to tackle optimization challenges in continuously streaming environments with unknown concept drifts. DASE is first characterized by the HCDD that operates on continuously streaming data for drift detection. It employs a tri-level confidence strategy with a multiple-hit confirmation approach, enabling precise drift detection by minimizing false positives and effectively adapting to varying degrees of concept drift. In addition, a warm-start strategy for new environments is introduced, combined with an adaptive knowledge transfer mechanism through the archive strategy. This approach provides an effective method for optimization in dynamic environment, maintaining solution diversity and enhancing the adaptability of the optimization process. To balance past and present information, only a limited amount of the most recent environment information is stored in the archive, taking storage constraints into account. Historical information from past environments is reused in two ways



(a) F4 D1 (b) F4 D2 (c) F4 D3 (d) F4 D4 (e) F4 D5

Fig. 5. Parameter sensitivity of $\mathrm{arc}_{\max}$

based on their assigned weights, which reflect their similarity to the current environment. The first is through a weighted ensemble of models, and the second is via weighted reuse of the best populations. Knowledge from past environments with higher similarity is prioritized, facilitating optimization in the current environment.

Experimental results on benchmark problems demonstrate the superiority of DASE over SOTA SDDEAs and DDEAs. Our proposed algorithm achieves faster convergence and higher solution quality across diverse problem instances and drift scenarios. The runtime analysis further highlights DASE's computational efficiency, particularly in handling problems under continuously streaming environments where timely responsiveness is critical. Overall, DASE present a promising solution to address the inherent challenges and limitations of current SDDEAs. Looking ahead, it would be valuable to extend the framework to more complex scenarios, particularly in multi-objective and high-dimensional settings, to further explore its adaptability and potential in these challenging environments.

REFERENCES

[1] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[2] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary computation*, vol. 3, no. 1, pp. 1–16, 1995.

[3] W. Li, T. Zhang, R. Wang, S. Huang, and J. Liang, "Multimodal multi-objective optimization: Comparative study of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 77, p. 101253, 2023.

[4] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades—part a," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, 2021.

[5] S. Liu, Q. Lin, J. Li, and K. C. Tan, "A survey on learnable evolutionary algorithms for scalable multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1941–1961, 2023.

[6] Z. Ma, H. Guo, Y.-J. Gong, J. Zhang, and K. C. Tan, "Toward automated algorithm design: A survey and practical guide to meta-black-box-optimization," *IEEE Transactions on Evolutionary Computation*, 2025.

[7] A. Kumar, A. Yazdanbakhsh, M. Hashemi, K. Swersky, and S. Levine, "Data-driven offline optimization for architecting hardware accelerators," in *ICLR 2022*, Feb. 2022.

[8] Q. Ye, H. Cai, and Y. Bian, "Ensemble-based offline data-driven evolutionary optimization for operation parameter of blast furnace," in *2024 20th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, 2024, pp. 1–7.

[9] H. Wang, Y. Jin, and J. O. Jansen, "Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 939–952, 2016.

[10] T. Chugh, N. Chakraborti, K. Sindhya, and Y. Jin, "A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem," *Materials and Manufacturing Processes*, vol. 32, no. 10, pp. 1172–1178, 2017.

[11] Y. Jin, H. Wang, and C. Sun, *Data-Driven Evolutionary Optimization: Integrating Evolutionary Computation, Machine Learning and Data Science*, ser. Studies in Computational Intelligence. Cham: Springer International Publishing, 2021, vol. 975.

[12] Y.-H. Sun, T. Huang, J.-H. Zhong, J. Zhang, and Y.-J. Gong, "Symbolic regression-assisted offline data-driven evolutionary computation," *IEEE Transactions on Evolutionary Computation*, 2024.

[13] H.-G. Huang and Y.-J. Gong, "Contrastive learning: An alternative surrogate for offline data-driven evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 370–384, 2022.

[14] Z. Liu, H. Wang, and Y. Jin, "Performance indicator-based adaptive model selection for offline data-driven multiobjective evolutionary optimization," *IEEE transactions on cybernetics*, vol. 53, no. 10, pp. 6263–6276, 2022.

[15] Y.-J. Gong, Y.-T. Zhong, and H.-G. Huang, "Offline data-driven optimization at scale: A cooperative coevolutionary approach," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 6, pp. 1809–1823, 2024.

[16] X.-R. Zhang, Y.-J. Gong, Z. Cao, and J. Zhang, "Island-based evolutionary computation with diverse surrogates and adaptive knowledge transfer for high-dimensional data-driven optimization," *ACM Transactions on Evolutionary Learning*, 2024.

[17] M. L. Peixoto, E. Mota, A. H. Maia, W. Lobato, M. A. Salahuddin, R. Boutaba, and L. A. Villas, "Fogjam: a fog service for detecting traffic congestion in a continuous data stream vanet," *Ad Hoc Networks*, vol. 140, p. 103046, 2023.

[18] X. Chen, J. Wang, and K. Xie, "Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning," *IJCAI*, 2021.

[19] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, Apr. 2014.

[20] E. Osekowska, H. Johnson, and B. Carlsson, "Maritime vessel traffic modeling in the context of concept drift," *Transportation research procedia*, vol. 25, pp. 1457–1476, 2017.

[21] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning*, vol. 23, pp. 69–101, 1996.

[22] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[23] Y. Zhong, X. Wang, Y. Sun, and Y.-J. Gong, "Sddobench: A benchmark for streaming data-driven optimization with concept drift," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Melbourne VIC Australia: ACM, Jul. 2024, pp. 59–67.

[24] W. Luo, R. Yi, B. Yang, and P. Xu, "Surrogate-assisted evolutionary framework for data-driven dynamic optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 137–150, 2018.

[25] K. Li, R. Chen, and X. Yao, "A data-driven evolutionary transfer optimization for expensive problems in dynamic environments," *IEEE Transactions on Evolutionary Computation*, 2023.

[26] C. Yang, J. Ding, Y. Jin, and T. Chai, "A data stream ensemble assisted multifactorial evolutionary algorithm for offline data-driven dynamic optimization," *Evolutionary Computation*, vol. 31, no. 4, pp. 433–458, 2023.

[27] H. Zhang, J. Ding, L. Feng, K. C. Tan, and K. Li, "Solving expensive optimization problems in dynamic environments with meta-learning," *IEEE Transactions on Cybernetics*, 2024.

[28] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu, "Tencentrec: Real-time stream recommendation in practice," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. Melbourne Victoria Australia: ACM, May 2015, pp. 227–238.

[29] M. Fedoryszak, B. Frederick, V. Rajaram, and C. Zhong, "Real-time event detection on social data streams," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2774–2782.

[30] J. Mou, K. Gao, P. Duan, J. Li, A. Garg, and R. Sharma, "A machine learning approach for energy-efficient intelligent transportation scheduling problem in a real-world dynamic circumstances," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15 527–15 539, 2023.

[31] S. Liu, H. Wang, W. Peng, and W. Yao, "A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 1087–1101, 2022.

[32] B. H. Nguyen, B. Xue, and M. Zhang, "A constrained competitive swarm optimizer with an svm-based surrogate model for feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 2–16, 2022.

[33] T. Sonoda and M. Nakata, "Multiple classifiers-assisted evolutionary algorithm based on decomposition for high-dimensional multiobjective problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1581–1595, 2022.

[34] L. Xie, G. Li, Z. Wang, L. Cui, and M. Gong, "Surrogate-assisted evolutionary algorithm with model and infill criterion auto-configuration," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 4, pp. 1114–1126, Aug. 2024.

[35] K. Zhao, X. Wang, C. Sun, Y. Jin, and A. Hayat, "Efficient large-scale expensive optimization via surrogate-assisted sub-problem selection," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

[36] X. Wu, Q. Lin, J. Li, K. C. Tan, and V. C. Leung, "An ensemble surrogate-based coevolutionary algorithm for solving large-scale expensive optimization problems," *IEEE Transactions on Cybernetics*, vol. 53, no. 9, pp. 5854–5866, 2022.

[37] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2018.

[38] L. Yu, Z. Meng, and H. Zhu, "A hierarchical surrogate-assisted differential evolution with core space localization," *IEEE Transactions on Cybernetics*, 2024.

[39] Y. Yuan and W. Banzhaf, "Expensive multiobjective evolutionary optimization assisted by dominance prediction," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 1, pp. 159–173, 2021.

[40] L. Zhao, Y. Hu, B. Wang, X. Jiang, C. Liu, and C. Zheng, "A surrogate-assisted evolutionary algorithm based on multi-population clustering and prediction for solving computationally expensive dynamic optimization problems," *Expert Systems with Applications*, vol. 223, p. 119815, Aug. 2023.

[41] X. Wu, S. Liu, J. Ji, L. Ma, and V. C. M. Leung, "A surrogate-assisted evolutionary algorithm for expensive dynamic multimodal optimzation," in *2024 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2024, pp. 1–8.

[42] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[43] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artificial Intelligence*, vol. 209, pp. 11–28, 2014.

[44] R. F. de Mello, Y. Vaz, C. H. Grossi, and A. Bifet, "On learning guarantees to unsupervised concept drift detection on data streams," *Expert Systems with Applications*, vol. 117, pp. 90–102, 2019.

[45] Z. Cai, R. Jiang, X. Yang, Z. Wang, D. Guo, H. Kobayashi, X. Song, and R. Shibasaki, "Memda: forecasting urban time series with memory-based drift adaptation," *arXiv preprint arXiv:2309.14216*, 2023.

[46] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift," *Frontiers in Artificial Intelligence*, vol. 7, p. 1330257, 2024.

[47] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2018.

[48] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *17th Brazilian Symposium on Artificial Intelligence*. Sao Luis, Maranhao, Brazil: Springer, September 29-Ocotber 1 2004, pp. 286–295.

[49] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *14th international workshop on knowledge discovery from data streams*, vol. 6. Citeseer, 2006, pp. 77–86.

[50] I. Frias-Blanco, J. D. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

[51] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2017, pp. 1–6.

[52] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, vol. 213, p. 118934, 2023.

[53] R. C. Cavalcante and A. L. Oliveira, "An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection," in *2015 international joint conference on neural networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[54] H. Wang, Y. Jin, C. Sun, and J. Doherty, "Offline data-driven evolutionary optimization using selective surrogate ensembles," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 203–216, Apr. 2019.

[55] J.-Y. Li, Z.-H. Zhan, C. Wang, H. Jin, and J. Zhang, "Boosting data-driven evolutionary algorithm with localized data generation," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 923–937, 2020.

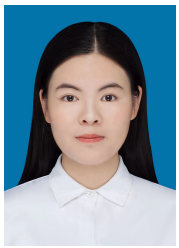
[56] P. Huang, H. Wang, and Y. Jin, "Offline data-driven evolutionary optimization based on tri-training," *Swarm and Evolutionary Computation*, vol. 60, p. 100800, Feb. 2021.

[57] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

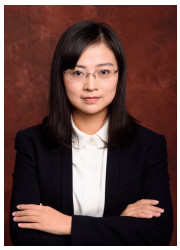
[58] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[59] C. W. Dunnett, "A multiple comparison procedure for comparing several treatments with a control," *Journal of the American Statistical Association*, vol. 50, no. 272, pp. 1096–1121, 1955.

[60] J. M. Bland and D. G. Altman, "Multiple significance tests: the bonferroni method," *Bmj*, vol. 310, no. 6973, p. 170, 1995.



Yuan-Ting Zhong received her bachelor's degree in Information Security from the South China University of Technology, Guangzhou, China, in 2023, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her current research interests include evolutionary computation and data-driven optimization.



Yue-Jiao Gong (S'10-M'15-SM'19) received the B.S. and Ph.D. degrees in Computer Science from Sun Yat-sen University, China, in 2010 and 2014, respectively. She is currently a Full Professor at the School of Computer Science and Engineering, South China University of Technology, China. Her research interests include optimization methods based on swarm intelligence, deep learning, reinforcement learning, and their applications in smart cities and intelligent transportation. She has published over 100 papers, including more than 50 in ACMIEEE TRANSACTIONS and over 50 at renowned conferences such as NeurIPS, ICLR, and GECCO. Dr. Gong was awarded the Pearl River Young Scholar by the Guangdong Education Department in 2017 and the Guangdong Natural Science Funds for Distinguished Young Scholars in 2022. She currently serves as Associate Editor of IEEE Transactions on Evolutionary Computation.