# TRACE: A Generalizable Drift Detector for Streaming Data-Driven Optimization

**Yuan-Ting Zhong[1], Ting Huang[2], Xiaolin Xiao[3], Yue-Jiao Gong[1]***

[1]South China University of Technology
[2]Xidian University
[3]South China Normal University
{ytalienzhong, gnauhgnith, shellyxiaolin, gongyuejiao}@gmail.com

## Abstract

Many optimization tasks involve streaming data with unknown concept drifts, posing a significant challenge as Streaming Data-Driven Optimization (SDDO). Existing methods, while leveraging surrogate model approximation and historical knowledge transfer, are often under restrictive assumptions such as fixed drift intervals and fully environmental observability, limiting their adaptability to diverse dynamic environments. We propose **TRACE**, a TRAnsferable Concept-drift Estimator that effectively detects distributional changes in streaming data with varying time scales. TRACE leverages a principled tokenization strategy to extract statistical features from data streams and models drift patterns using attention-based sequence learning, enabling accurate detection on unseen datasets and highlighting the transferability of learned drift patterns. Further, we showcase TRACE's plug-and-play nature by integrating it into a streaming optimizer, facilitating adaptive optimization under unknown drifts. Comprehensive experimental results on diverse benchmarks demonstrate the superior generalization, robustness, and effectiveness of our approach in SDDO scenarios.

## Introduction

Many real-world optimization applications are driven by massive volumes of continuously arriving data. For instance, traffic optimization in smart cities relies on real-time data streams from sensors and monitoring systems (Styler and Nourbakhsh 2015; Kang et al. 2019; Ji et al. 2022). In streaming environments, the underlying data distribution may change unpredictably over time due to external factors such as traffic accidents or weather fluctuations, which is a phenomenon known as concept drift (Gower-Winter et al. 2025). These drift occurrences, often unknown and associated with limited data at each time step, give rise to Streaming Data-Driven Optimization (SDDO) problems (Zhong et al. 2024), where optimization strategies must adapt dynamically to maintain performance.

Recent efforts to address SDDO problems have led to the development of Streaming Data-Driven Evolutionary Algorithms (SDDEAs) (Richter et al. 2020; Yang et al. 2023; Zhang et al. 2024). These algorithms offer a promising approach by integrating evolutionary optimization techniques with data-driven modeling. Typically, SDDEAs build surrogate models from streaming data and transfer knowledge from past environments to accelerate optimization of the current environment (Luo et al. 2019). Although these approaches have demonstrated encouraging results, their performance often hinges on strong, and often unrealistic assumptions. For example, some methods assume fixed and known drift intervals, which allows them to explicitly adapt the optimization strategy at the beginning of each interval (Li, Chen, and Yao 2024; Zhang et al. 2024). Others assume immediate access to complete data from each environment before making optimization decisions (Yang et al. 2023; Liu et al. 2025). In more complex scenarios with unpredictable drifts and continuous data streams, the absence of reliable drift detection can lead SDDEAs to overfit outdated distributions or overlook recurring patterns, ultimately degrading optimization performance.

Despite increasing interest in SDDO, effective and generalizable drift detection methods tailored for streaming optimization remain scarce. While drift detection has been explored in stream data mining (Alsaedi et al. 2023; Wan, Liang, and Yoon 2024), existing methods are primarily designed for classification tasks, which struggle with the unique characteristics of SDDO. Specifically, they often assume discrete labels or bounded outputs, which are incompatible with the unbounded, real-valued domains common in SDDO. Furthermore, these methods typically focus on detecting abrupt changes in prediction accuracy, overlooking the subtle performance degradation that often signifies drift in optimization landscapes. To bridge this gap, we propose **TRACE**, a TRAnsferable Concept-drift Estimator that flexibly solves streaming data with varying time scales and distribution shifts, offering a tailored solution for SDDO environments. As illustrated in Figure 1, TRACE demonstrates a strong generalization ability, enabling it to detect drift in previously unseen datasets by learning and leveraging transferable drift patterns. Moreover, TRACE can be seamlessly integrated in many SDDEAs as a general-purpose detector, enabling adaptive optimization under unknown concept drift. Our main contributions are summarized as follows:
**1) Principled Stream Tokenization for Drift Modeling:** We introduce a tokenization strategy to transform streaming data into a sequence of statistical representations, capturing temporal distributional characteristics that are indica-

*Corresponding author
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

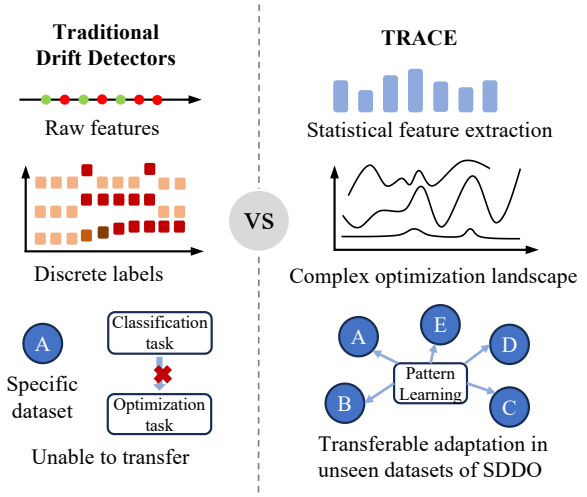



Figure 1: TRACE offers a learnable and generalizable approach to drift detection in the SDDO landscape.

tive of concept drift. The statistical sequences serve as informative inputs for drift detection models, enabling pattern understanding and facilitating scalable, label-efficient training of supervised detection models. **2) Unified Framework for Learning Transferable Drift Patterns:** Building upon the statistical representation mentioned above, we design a generalizable detection framework based on attention-driven sequence modeling. Our attention mechanism captures local and global temporal dependencies, which, together with training on diverse changing landscapes, prevents overfitting to specific data streams. Our framework allows the model to learn transferable drift patterns, ensuring robust and accurate detection across diverse tasks and previously unseen datasets. **3) Plug-and-Play Detection Module for Streaming Optimizers:** TRACE can be seamlessly integrated in many SDDEAs as a general-purpose detector, without requiring modifications to the optimization logic. To demonstrate this, we design TRACE-EA, a SDDO algorithm that embeds TRACE as a drift-awareness module. This flexible design demonstrates a pathway for enabling SDDEAs to effectively operate under streaming environments with unknown concept drift, proactively responding to distributional changes. **4) Comprehensive Validation across Tasks and Domains:** Extensive experiments demonstrate the effectiveness, generalization ability, and practical utility of our method in diverse SDDO scenarios. Our evaluation covers unseen datasets, cross-task optimization and real-world applications confirms its broad applicability.

## Related Work

**SDDEAs.** SDDEAs address SDDO challenges by reusing knowledge from past environments to accelerate the current environmental optimization. *Solution-level reuse* is common: SAEF-1GP (Luo et al. 2019) augments training data with the optimal solution from most recent environment. SAEA-TL (Wu et al. 2024) reuses historical data whose surrogate yields the lowest RMSE in the current environment. DETO (Li, Chen, and Yao 2024) clusters past surrogate parameters and injects solutions from the closest cluster. *Model-level reuse* methods include DSEMFS (Yang et al. 2023), which aggregates surrogates from a maintained surrogates' pool, and MLDE (Zhang et al. 2024) employs meta-learning to initialize new models with parameters from previous ones. While offering promising directions for SDDO, these SDDEAs commonly assume fixed drift intervals and full data observation. This leads to treating each data batch as a distinct environment, which fails to capture unpredictable nature of complex data streams. DASE (Zhong and Gong 2025) takes a step forward by incorporating statistical drift detection (HCDD) into streaming optimization, but its fixed-threshold tests limit adaptability. This fundamental mismatch between method assumptions and the characteristics of complex streaming data limits the effectiveness of current SDDEAs. As a result, their practical utility is restricted in applications that demand continuous adaptation.

**Drift Detection in Stream Data Mining.** Stream data mining (Alsaedi et al. 2023; Wan, Liang, and Yoon 2024) has seen the development of numerous drift detection methods, broadly categorized as statistical or learning-based. Among *statistical methods*, DDM (Gama et al. 2004) monitors increases in prediction error means, while HDDM_A/W (Frias-Blanco et al. 2015) improve adaptivity via Hoeffding's and McDiarmid's inequalities. Two-window methods compare statistics between recent and historical data: ADWIN (Bifet and Gavaldà 2007) detects significant mean differences using Hoeffding's bound; KSWIN (Raab, Heusinger, and Schleif 2020) applies the Kolmogorov–Smirnov test, and PUDD (Lu et al. 2025) introduces a PU-index using the Chi-Square test for theoretically grounded detection. *Learning-based methods* have gained increasing interest recently: RADAR (Alsaedi et al. 2023) uses a recurrent variational embedding to learn latent dynamics and detects drift in the representation space. MCD-DD (Wan, Liang, and Yoon 2024) leverages contrastive learning to estimate maximum concept discrepancy between sample pairs, enabling robust detection in dynamic environments. However, these methods face key limitations in SDDO with regression-type objectives and surrogate models: 1) Designed for classification, they assume discrete labels or bounded outputs, hence unsuitable for SDDO's unbounded, real-valued domains. 2) In optimization, drift may manifest not as prediction error spikes, but as performance degradation stemming from evolving objective landscapes or shifting surrogate model behavior. Those methods struggle to distinguish such phenomena. 3) Threshold-based methods relying on hand-crafted rules often lack generalizability across different scenarios. The challenges highlight the pressing need for more flexible, generalizable, and adaptive drift detection mechanisms tailored to SDDO.

## Methodology

We begin by introducing a tokenization strategy to transform streaming data into sequences suitable for drift modeling. We then detail the architecture and training procedure of TRACE, our proposed transferable concept-drift estimator. Finally, we present TRACE-EA, an instantiation of SDDEA integrates TRACE as a plug-and-play module for adaptive

SDDO problems with unknown concept drift.

## Stream Tokenization for Drift Modeling

To effectively model concept drift in streaming data, we first transform the raw stream via stream tokenization, converting the continuous stream into a labeled sequence of discrete tokens that capture underlying dynamics. We detail this statistical approach below.

Given a stream $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_i, y_i), \cdots\}$, where $\mathbf{x}_i$ represents the sample $i$ and $y_i$ is the corresponding objective value, we train a surrogate model for each environment to predict the objective value given the input. Then, we compute the prediction error of the surrogate for sample $i$ as:

$$e_i = \begin{cases} |(y_i - \hat{y}_i)/y_i|, & y_i \neq 0 \\ |y_i - \hat{y}_i|, & \text{otherwise} \end{cases} \quad (1)$$

where $\hat{y}_i$ is the surrogate-predicted objective value.

In static environments, prediction errors fluctuate around a stable mean. However, when drift occurs, changes in the data distribution lead to a degradation in the surrogate model's predictive performance, often manifesting as noticeable change in prediction error. These patterns are what we aim to capture. Our goal is to create labeled sequences that can be used to train a supervised drift detection model. We achieve this through a two-step process: 1) extracting statistical features from the prediction errors within sliding windows to create tokens, and 2) combining these tokens into labeled sequences that incorporate environmental context and temporal coherence.

First, we apply a sliding window of length $n$ to capture the current statistical properties of prediction errors. At time $t$, window $t$ contains $n$ most recent data samples: $\{(\mathbf{x}_{t-n+1}, y_{t-n+1}), \ldots, (\mathbf{x}_t, y_t)\}$, by which we compute the prediction errors $\{e_{t-n+1}, \ldots, e_t\}$ using Eq.(1). From these, we extract a statistical feature vector: $fv_t = (\mu_t, \sigma_t, \min_t, \max_t, Q1_t, Q2_t, Q3_t)$, where $\mu_t$ and $\sigma_t$ denote the mean and standard deviation, and $Q1_t$, $Q2_t$, and $Q3_t$ are the first, second (median), and third quartiles of the error distribution within window $t$. Applying a sliding window over the stream yields a sequence of feature vectors.

To create training samples, we select $T$ consecutive feature vectors: $\{fv_1, fv_2, \cdots, fv_T\}$. To incorporate environmental context, we prepend a special context token $fv_0$, computed over all data observed in the current environment up to the start of the $T$ windowed vectors, summarizing the environment's state. Each training sample is then formed as a sequence of $(T+1)$ tokens: $\langle fv_0, fv_1, fv_2, \cdots, fv_T \rangle$, with an associated drift labeled $dl$. If no drift occurs within the subsequence, $dl = 0$; otherwise, $dl = l$ indicates drift occurs at the $l$-th step ($1 \leq l \leq T$). Thus, each sample is a labeled token sequence $X \in \mathbb{R}^{(T+1)\times d_f}$, where $d_f$ is the dimensionality of each feature vector.

## TRACE for Learning Transferable Drift Patterns

Leveraging labeled sequences derived from prediction errors, we unlock rich representations that implicitly capture evolving drift patterns in data streams – a feat often unattainable with raw data alone. Conventional statistical detectors struggle with complex temporal and structural changes, especially under sudden, incremental, or mixed drifts. To overcome these challenges, we propose TRACE, a learning-based drift estimator that extracts discriminative, transferable representations directly from sequences of window-level features. As illustrated in Figure 2, TRACE comprises three key components: 1) an embedding module that maps input features into a high-dimensional latent space, 2) an encoder that models temporal dependencies and structural patterns across the sequence, and 3) a classification head that predicts the drift label from encoded representations. The following sections detail each component of TRACE.

**Sequence Embedding.** Streaming data results in input sequences of variable lengths. To handle this, sequences are padded to a fixed maximum length using special `<PAD>` tokens that the model learns to ignore. Embedding and positional encoding then transform these statistical feature sequences into temporally aware representations, enabling the model to capture semantic relationships and sequential dependencies. Further details are provided in Appendix A[1].

**The TRACE Encoder.** TRACE's encoder employs two complementary self-attention mechanisms to capture diverse temporal dependencies critical for drift detection:

*1) Global Multi-Head Self-Attention (G-MSA)*: G-MSA models global interactions among all tokens, capturing long-range temporal and structural patterns indicative of concept drift. This allows the model to capture diverse interaction patterns across the sequence. The G-MSA follows the standard Transformer encoder (Vaswani et al. 2017), enabling each token to attend to every other token, capturing long-range dependencies indicative of drift, which often signal the emergence of concept drift. With the integration of positional encoding, G-MSA is sensitive to relative order of tokens, thus modeling both temporal and structural patterns. Formally, given embedded inputs with positional encoding $\mathbf{H}^{\text{pos}}$, G-MSA first calculates queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$ using learned projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$. For each attention head $i$, the attention output is $\mathrm{h}_i^G = \text{Softmax}\left(\mathbf{Q}_i\mathbf{K}_i^\top/\sqrt{d_k}\right)\mathbf{V}_i$. Finally, the outputs from $m$ heads are concatenated and projected via $\mathbf{W}_{\mathrm{G}}$: $\text{G-MSA}(\mathbf{H}^{\text{pos}}) = \text{Concat}(\mathrm{h}_1^G, \cdots, \mathrm{h}_m^G)\mathbf{W}_{\mathrm{G}}$.

*2) Context-Guided MSA (C-MSA)*: A crucial insight for drift detection is measuring how recent data deviates from the current environment. C-MSA directly addresses this by modeling the relationship between each token and a dedicated context token that encodes the current environment. Recall that the first token in the input sequence represents the most recent data distribution and serves as a reference for detecting distributional changes. To highlight deviations relative to this context, C-MSA uses the context token as the sole query: $\mathbf{q} = \mathbf{h}_0\mathbf{W}_q$, while treating all other tokens as keys and values, following the same formulation as G-MSA. This design explicitly models the relationship between the current environment context and recent tokens, effectively capturing incremental drift patterns.

[1] "Appendix.pdf" in our codebase.

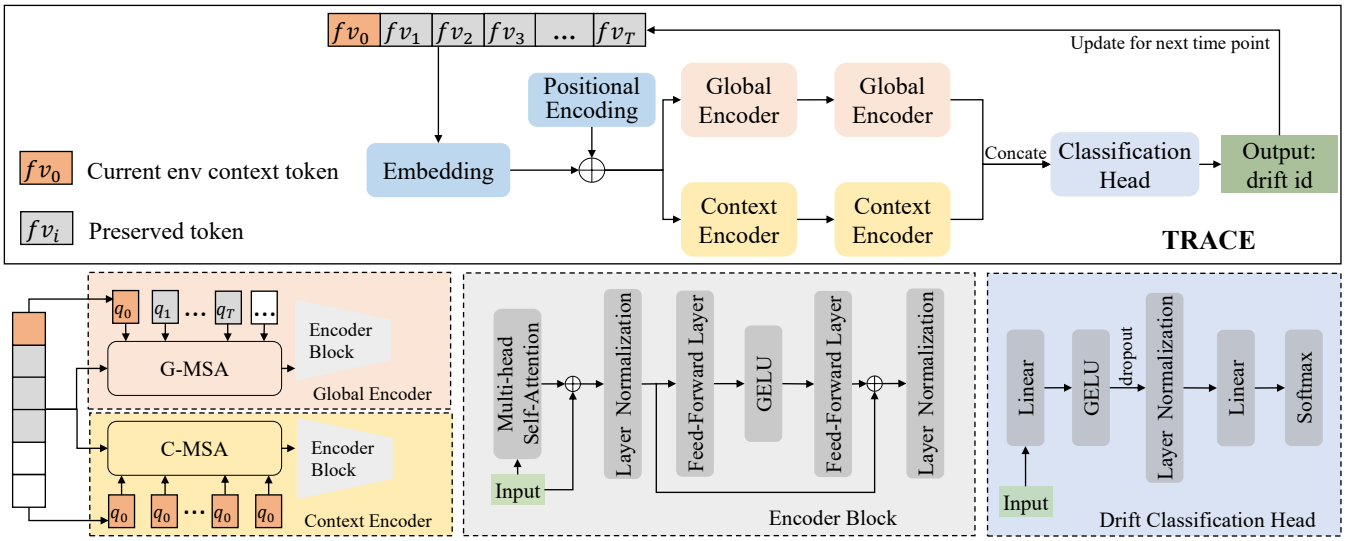



Figure 2: TRACE Architecture. The upper part illustrates the overall framework, while the lower part details its key components: the Sequence Embedding Module, the Dual-Attention Encoder, and the Drift Classification Head.

TRACE combines G-MSA and C-MSA to model both global temporal dependencies and localized context-aware deviations. The concatenated outputs form a joint representation that encodes structural patterns and environment-specific changes. This dual-attention encoder enables TRACE to generalize across diverse optimization tasks, with minimal assumptions about drift characteristics.

**Drift Classification Head.** The output of the dual-attention encoder is fed into a pointer-like classification head. This head predicts the drift index within the input sequence or indicates the absence of drift. This formulation enables both explicit localization of drift events and confirmation of distributional stability.

Specifically, the classification head consists of two linear layers with GELU activation, followed by dropout and layer normalization to enhance regularization and training stability. It outputs a probability distribution over $T+1$ classes: $T$ classes correspond to the time windows in the sequence, and the extra class indicates the no-drift case. Formally, given the concatenated embedding $\mathbf{z}$ from G-MSA and C-MSA, the drift classification head is:

$$\mathbf{y} = \text{Softmax}(\mathbf{W_2} \cdot \text{LayerNorm}(\phi(\mathbf{W}_1 \cdot \mathbf{z} + \mathbf{b}_1)) + \mathbf{b}_2) \quad (2)$$

where $\mathbf{y} \in \mathbb{R}^{T+1}$ represents the probabilities over all windows $1, \ldots, T$ and the no-drift class (label 0).

**Training for Transferability.** To provide a controlled environment for learning, TRACE is trained using synthetic streams from SDDObench (Zhong et al. 2024), a dedicated benchmark for SDDO, building on prior work in benchmarking dynamic optimization problems (Li et al. 2008; Yazdani et al. 2022). SDDObench simulates diverse types of concept drifts, allowing TRACE to learn from varied drift and non-drift scenarios in a controlled yet rich setting. Training mimics a realistic streaming process: starting with a surrogate model built for the initial environment. As the stream progresses, labeled sequences are generated via a sliding window to represent recent distributional characteristics. When new data signals a shift, the surrogate model is updated for the current optimization landscape. To improve generalization, training samples are randomly truncated after the true drift index, exposing TRACE to diverse temporal patterns. The model is optimized using cross-entropy loss over the predicted drift index, promoting both detection and accurate localization. This setup enables TRACE to generalize well to new datasets and unseen drift conditions.

## Integration into Streaming Optimizers

**The TRACE-EA Framework.** TRACE is designed as a plug-and-play drift detection module compatible with many SDDEAs. To demonstrate its effectiveness, we develop TRACE-EA by integrating TRACE into DASE (Zhong and Gong 2025) as the drift detector, replacing DASE's original hand-crafted drift detection mechanism, namely the HCDD. TRACE-EA addresses three challenges in SDDO problems: accurately detecting distribution changes, rapidly adapting optimization without restarting, and enabling transferable drift detection across unseen tasks. The following part details the detection-adaptation loop that governs TRACE-EA's continuous operation. A more detailed description of TRACE-EA is provided in the Appendix B.

**The Detection-Adaptation Loop.** TRACE-EA operates continuously via a detection-adaptation loop integrating streaming data processing, drift detection, and adaptive optimization. At each time $t$, the algorithm receives a small data batch $\mathbb{D}_t$, updates the sliding window, and constructs token sequences as input to TRACE. If no drift is detected, optimization proceeds normally. Upon drift detection, a new environment is instantiated, and the archive is queried to identify similar past environments based on feature similarity. A knowledge transfer module then reuses relevant surrogate

models and population knowledge to warm-start the optimization in the new environment. Here, a core component is an archive-based knowledge transfer mechanism. Instead of discarding prior optimization information when environment changes, TRACE-EA maintains an archive of past environments and learned optimization knowledge. When a drift is detected, it identifies the most relevant prior environment using similarity metrics and selectively transfers surrogate models and population states. After adaptation, the archive is updated with new knowledge. This continuous detection-adaptation loop allows TRACE-EA to maintain high performance in dynamic environments by rapidly responding to changes and avoiding redundant re-optimization. By reusing experience to guide adaptation, TRACE-EA achieves efficient and robust performance under streaming data environments.

## Experiments

Our evaluation addresses the following research questions: **RQ1**: How well does TRACE generalize to unseen datasets for accurate drift detection? **RQ2**: Can TRACE-EA effectively solve optimization problems unseen during training? **RQ3**: What is the contribution of each core component of TRACE to the drift detection performance? **RQ4**: How does TRACE-EA perform on real-world stream clustering tasks?

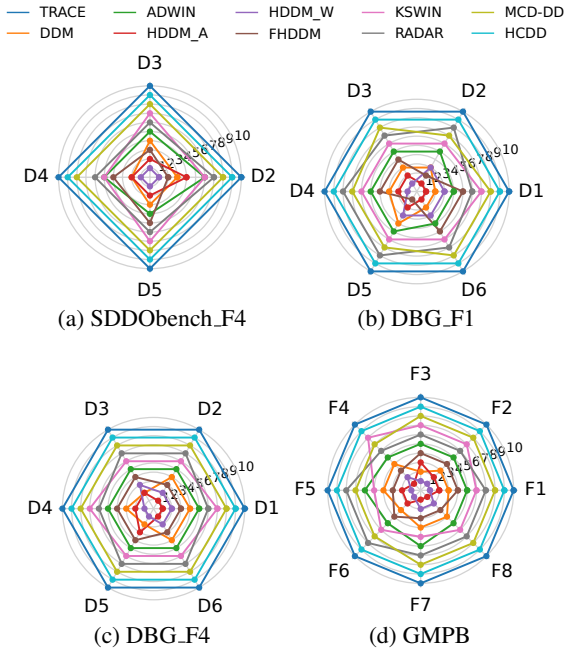
### Experimental Setup

**Training setup.** SDDObench (Zhong et al. 2024) generates training data. For each instance, we create 60 environments, each containing a randomly chosen number of samples from $\{600, 750, 900\}$. We set sliding window size $n = 30$ and max sequence length 20, using a Radial Basis Function Network (RBFN) as surrogate for error computation (Zhong and Gong 2025). Training runs for 50 epochs with batch size 32 and learning rate $5 \times 10^{-4}$ on an AMD EPYC 9745 CPU and NVIDIA RTX 4080 Super GPU (Python 3.10.12 and PyTorch 2.0.1).

**Competitors.** We consider two groups of competitors. First, for drift detection only, TRACE[2] is compared with established drift detectors from different fields, including DDM (2004), ADWIN (2007), HDDM_A/W (2015), FHDDM (2016), KSWIN (2020), RADAR (2023), MCD_DD (2024), and HCDD (2025). Second, for the higher-level SDDO tasks, our TRACE-EA is compared against several state-of-the-art SDDEAs, including SAEF-1GP (2019), BDDEA-LDG (2020), TT-DDEA (2021), DSEMFS (2023), DETO (2024), MLDE (2024), and DASE (2025). All baselines are obtained from publicly available source code or faithfully re-implemented based on their original papers.

**Generalization Tests.** To evaluate the generalization capabilities of TRACE, we conduct experiments using both In-Distribution (ID) and Out-Of-Distribution (OOD) datasets. For ID, we utilize *SDDObench* (Zhong et al. 2024) that employs a distinct set of problem instances generated with configurations explicitly disjoint from those used during training. For OOD, we conduct experiments on *DBG* (Li et al. 2008) and *GMPB* (Yazdani et al. 2022) to evaluate TRACE's performance on unseen distributions. In addition, we also evaluate TRACE on four real-world data stream clustering datasets to provide a realistic assessment.

[2]https://github.com/YTALIEN/TRACE



(a) SDDObench_F4 (b) DBG_F1

(c) DBG_F4 (d) GMPB

Figure 3: Detector performance comparison. See Appendix D-1 for additional results.

**Performance Metrics.** Standard metrics are used: *Precision* and *F1* for drift detection accuracy, while the *Dynamic Tracking Error* ($E_{DT}$) for the average deviation from the optimum over time. All results are averaged over 11 independent runs. Statistical significance is determined using the Kruskal–Wallis test (Kruskal and Wallis 1952) followed by Dunnett's post-hoc analysis (Dunnett 1955). More experimental details are in Appendix C.

### Detector Performance Comparison (RQ1)

Figure 3 presents the precision on a subset of benchmarks; complete results across all benchmarks are available in the Appendix D-1. TRACE consistently outperforms all baselines, achieving the highest precision on most tasks. It remains robust under both incremental and sudden drift scenarios (SDDObench_F1: D2, D4), where traditional methods often suffer from high false positive. Moreover, TRACE demonstrates strong performance on OOD datasets (GMPB and DBG), indicating effective transferability. These results highlight TRACE's ability to capture generalizable representations of concept drift, enabling accurate detection across varying drift types. Overall, TRACE not only surpasses statistical-based methods in accuracy but also generalizes reliably to diverse, unseen tasks.

| Instance | TRACE-EA | SAEF-1GP | BDDEA-LDG | TT-DDEA | DSEMFS | DETO | MLDE | DASE |
|---|---|---|---|---|---|---|---|---|
| (a) F4D1 | **9.9e-02±3.7e-02** | 4.9e+01±2.2e-01 | 4.4e+00±2.1e+00 | 6.3e+00±4.5e-01 | 6.4e+01±1.1e+00 | 1.6e+02±2.2e+00 | 1.1e+02±1.8e+01 | 2.7e-01±1.7e-01 |
| (a) F4D2 | **2.3e+00±2.0e-01** | 5.8e+01±1.0e+00 | 2.6e+01±3.2e+00 | 6.5e+01±1.4e+01 | 4.4e+01±1.1e+00 | 1.4e+02±9.0e-01 | 9.5e+01±6.4e+00 | 2.4e+01±1.6e-01 |
| (a) F4D4 | **1.4e+01±1.5e-01** | 6.8e+01±9.6e-01 | 4.9e+01±4.2e-01 | 1.2e+02±6.3e+01 | 6.0e+01±2.4e+00 | 1.0e+02±5.8e+00 | 6.4e+01±6.4e+00 | 1.6e+01±2.9e-01 |
| (b) F1D1 | **5.2e+01±1.7e+00** | 6.3e+01±2.0e+00 | 6.2e+01±9.6e-02 | 5.9e+01±2.2e-02 | 5.3e+01±6.3e-02 | 6.0e+01±2.9e+00 | 6.2e+01±1.1e+00 | 5.9e+01±2.0e+00 |
| (b) F1D2 | **3.9e+01±6.4e-01** | 6.3e+01±2.5e-02 | 6.2e+01±9.6e-02 | 5.9e+01±2.2e-02 | 6.3e+01±8.4e-02 | 6.2e+01±3.7e-02 | 5.9e+01±2.3e+00 | 5.9e+01±9.1e-01 |
| (b) F1D6 | **3.7e+01±1.4e-01** | 5.4e+01±3.9e-02 | 5.1e+01±1.5e-02 | 5.3e+01±1.4e+00 | 5.3e+01±1.1e-01 | 5.7e+01±5.7e-01 | 4.7e+01±9.7e-01 | 5.0e+01±6.1e-01 |
| (c) F1 | **8.2e+02±1.1e+01** | 1.5e+03±1.4e+01 | 1.0e+03±2.5e+00 | 1.2e+03±4.5e+01 | 1.1e+03±1.0e+01 | 1.1e+03±1.1e+01 | 1.2e+03±4.5e+01 | 1.3e+03±4.1e+01 |
| (a) F5 | **8.6e+02±7.9e+00** | 9.9e+03±4.8e+01 | 1.7e+03±2.7e+01 | 1.0e+03±3.3e+02 | 1.7e+03±4.1e+01 | 1.1e+03±4.2e+01 | 1.1e+03±2.7e+02 | 1.1e+03±4.1e+02 |
| (a) F8 | **8.6e+02±7.4e+00** | 1.5e+03±6.3e+01 | 2.5e+03±1.0e+02 | 1.1e+03±2.6e+02 | 2.2e+03±7.7e+01 | 1.0e+03±1.4e+01 | 1.1e+03±1.6e+02 | 9.7e+02±2.6e+01 |

Table 1: Comparison of $E_{DT}$ values among SDDEAs across benchmarks. See Appendix D-2 for all results.

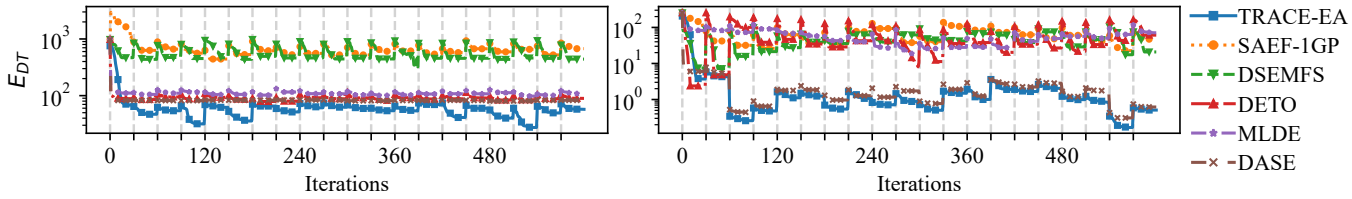



Figure 4: Convergence trajectories over the first 20 environments. Left: SDDObench_F4D4; Right: DBG_F1D2.

| Instance | D1 | | D2 | | D3 | | D4 | |
|---|---|---|---|---|---|---|---|---|
| Metric | Prec | F1 | Prec | F1 | Prec | F1 | Prec | F1 |
| **TRACE** | **0.77** | **0.73** | **0.75** | **0.71** | **0.73** | **0.70** | **0.69** | **0.65** |
| *w/o PE* | 0.65 | 0.45 | 0.64 | 0.46 | 0.65 | 0.45 | 0.63 | 0.50 |
| *w/o G-MSA* | 0.59 | 0.25 | 0.55 | 0.20 | 0.51 | 0.20 | 0.45 | 0.17 |
| *w/o C-MSA* | 0.50 | 0.20 | 0.48 | 0.10 | 0.52 | 0.15 | 0.40 | 0.20 |
| *vanilla class* | 0.60 | 0.51 | 0.65 | 0.55 | 0.66 | 0.50 | 0.65 | 0.60 |

Table 2: Ablation study results on DBG_F1 (D1-D4).

## SDDO Performance Comparison (RQ2)

Table 1 reports the average and standard deviation of $E_{DT}$ values (full results in Appendix D-2). TRACE-EA consistently outperforms all baselines across diverse benchmarks and drift types, highlighting its strong generalization. Its advantage stems from accurate drift detection and effective reuse of historical knowledge (archived populations and surrogate models) for initialization. Concretely, as shown in Figure 4, in the complex optimization landscape of SDDObench_F4 with incremental drift (D4), TRACE precisely detects the onset of drift, enabling the algorithm to adapt quickly and prevent performance degradation. This capability mitigates data scarcity and enhances efficiency under streaming environments. TRACE also demonstrates a fast detection response and low computational overhead. Further experimental results are presented in Appendix D-4.

## Ablation and In-Depth Analysis (RQ3)

**Ablation Study.** We examine component contributions via four variants: 1) *w/o PE*: Remove positional encodings; 2) *w/o G-MSA*: Removes G-MSA; 3) *w/o C-MSA*: Removes C-MSA; 4) *vanilla class*: Replaces the pointer-like classifier with a standard fully connected layer. Table 2 summarizes the results. The removal of G-MSA causes a sharp performance drop, underscoring its role in capturing long-range patterns. Eliminating C-MSA also degrades accuracy, indicating its importance in environment-aware token interpretation. Replacing the pointer-like classification head with a vanilla classifier reduces performance, highlighting the advantage of explicit drift localization.

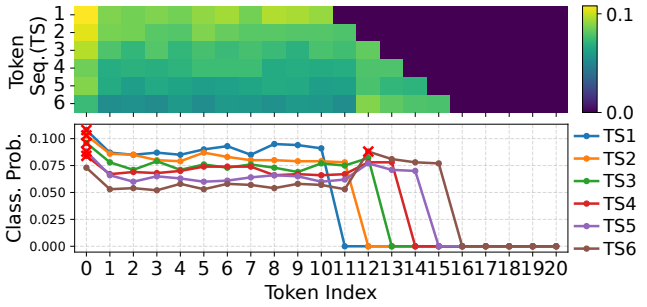



Figure 5: Attention distribution learned by C-MSA. Higher weights are assigned to tokens near drift time point and the first context token, with red '×' indicating TRACE's predicted drift and the ground truth at token 12.

**What has C-MSA learned?** To investigate TRACE's internal mechanism, we analyze the attention patterns learned by C-MSA. Concretely, we visualize its attention weights on token sequences from six consecutive steps in DBG_F1D1 (Figure5), along with the classification scores. The results reveals that attention is unevenly distributed, with C-MSA consistently focuses on tokens corresponding to time steps with notable distributional shifts. The first token, representing the current environment, consistently receives high attention, underscoring its role as an anchor of the current environment. This selective focus indicates that C-MSA effectively identifies drift relevant features, contributing to both the interpretability and generalization ability of TRACE.

**What has G-MSA learned?** To understand the representations learned by G-MSA, we extract its output embeddings from six consecutive time steps (TS1–TS6) in DBG_F1D1, consistent with the previous subsection. These high-dimensional vectors are projected into 2D using PCA (Shlens 2014) for visualization. Figure 6 shows the token distributions per time point. Blue points indicate in-distribution tokens (aligned with the first context token), orange points denote drifted tokens, and green points cor-

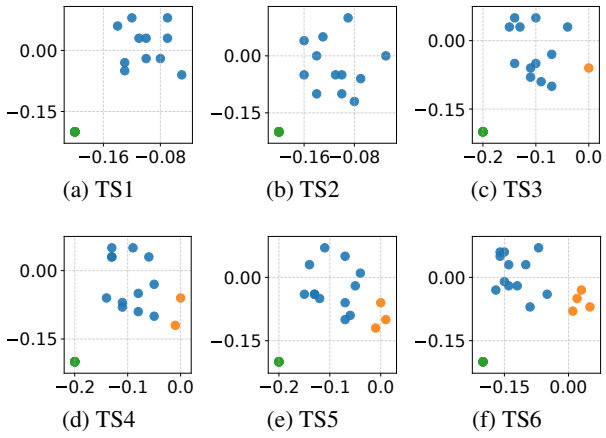



Figure 6: G-MSA token distribution via PCA. Blue Points: context-similar tokens; Orange Points: distributional changes; Green Points: padding tokens.

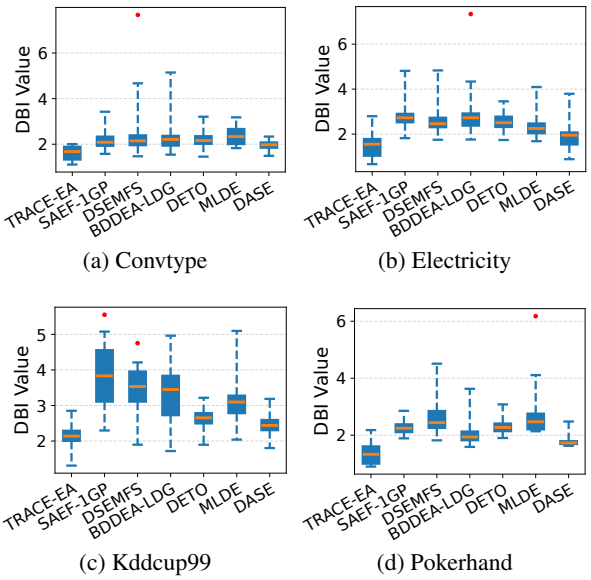



Figure 7: Boxplot of DBI values obtained by TRACE-EA and SDDEAs on real-world data stream clustering datasets.

respond to padding. As the stream arriving, token clusters shift, and drifted tokens become increasingly separated from the main cluster. This behavior demonstrates that G-MSA captures global sequence dependencies and distinguishes between coherent and distributionally distinct tokens. Such structured separation highlights G-MSA's critical role in enabling TRACE to identify drift across time.

### Application to Stream Clustering Tasks (RQ4)

To evaluate the practical utility and generalization capability of TRACE-EA in real-world scenarios, we apply it to streaming clustering tasks. These tasks are fundamental in streaming environments, where maintaining clustering quality under continuous distributional changes is critical for decision-making in domains such as network monitoring (Borghesi et al. 2019; Bulut and Singh 2005), energy systems (Yu et al. 2019), and user behavior analysis (Wang et al. 2016; Daza et al. 2023). For proof-of-principled evaluation, we use four widely-used and well-established datasets: *Convtype* (Blackard and Dean 1999), *Electricity* (Asuncion, Newman et al. 2007), *Kddcup99* (KddCup99 2007), and *Pokerhand* (Asuncion, Newman et al. 2007). These datasets feature diverse and well-documented drift patterns and are commonly used for evaluating stream clustering algorithms. More details of datasets are provided in the Appendix E.

All algorithms in the experiment are integrated into the ACDE (Das, Abraham, and Konar 2008) framework to encode individuals for optimization, using the Davies-Bouldin Index (DBI) (Davies and Bouldin 2009) as the optimization objective. Lower DBI values indicate better clustering performance. At each time point, when a new batch of data arrives, TRACE evaluates whether concept drift has occurred. If no drift is detected, optimization proceeds using the current state. If drift is detected, surrogate is updated to adapt to the new environment before resuming optimization. Figure 7 summarizes DBI values over 11 runs. TRACE-EA consistently achieves lower scores with reduced variance, demonstrating superior clustering quality and robustness. Improvements are especially pronounced on *Electricity* and *Kddcup99*, which involve varying drifts, highlighting TRACE-EA's effectiveness in dynamic real-world environments.

## Conclusion

This paper presents TRACE, a novel and transferable concept drift estimator designed for detecting distributional changes in streaming data for SDDO problems. TRACE leverages a principled stream tokenization method to extract statistical representations from data streams, facilitating learning-based drift modeling. The resulting labeled token sequences are processed by a dual-attention encoder to jointly captures the global context and localized drift patterns, enabling precise identification of distributional shifts. A pointer-like classification head then accurately pinpoints the most probable drift index within the sequence. Building upon TRACE, we further introduce TRACE-EA, an instantiation of SDDEA that integrates TRACE as a plug-and-play drift detector. TRACE-EA supports generalized, drift-aware optimization across previously unseen datasets and tasks. Comprehensive experiments on synthetic and real-world datasets confirm that TRACE achieves accurate and robust drift detection, while TRACE-EA consistently enhances the optimization performance of SDDO. However, this study also has limitations. First, the fixed sliding window causes detection delay, which could be mitigated by adaptive windowing techniques. Second, a tighter integration of the detector and optimizer, beyond the current plug-and-play design, could yield further performance gains. The field of automated algorithm design offers a promising avenue for discovering such synergistic frameworks automatically (Chen et al. 2024; Ma et al. 2025; Guo et al. 2025).

## Acknowledgments

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 (Grant Nos. 2025ZYGXZR027 and ZYTS25297).

## References

Alsaedi, A.; Sohrabi, N.; Mahmud, R.; and Tari, Z. 2023. RADAR: Reactive concept drift management with robust variational inference for evolving iot data streams. In *Proceedings of IEEE 39th International Conference on Data Engineering (ICDE)*, 1995–2007.

Asuncion, A.; Newman, D.; et al. 2007. UCI machine learning repository.

Bifet, A.; and Gavaldà, R. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448.

Blackard, J. A.; and Dean, D. J. 1999. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3): 131–151.

Borghesi, A.; Bartolini, A.; Lombardi, M.; Milano, M.; and Benini, L. 2019. Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 33, 9428–9433.

Bulut, A.; and Singh, A. K. 2005. A unified framework for monitoring data streams in real time. In *Proceedings of 21st International Conference on Data Engineering (ICDE)*, 44–55. IEEE.

Chen, J.; Ma, Z.; Guo, H.; Ma, Y.; Zhang, J.; and Gong, Y.-J. 2024. SYMBOL: Generating flexible black-box optimizers through symbolic equation learning. In *The Twelfth International Conference on Learning Representations*.

Das, S.; Abraham, A.; and Konar, A. 2008. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1): 218–237.

Davies, D. L.; and Bouldin, D. W. 2009. A cluster separation measure. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, (2): 224–227.

Daza, R.; Morales, A.; Tolosana, R.; Gomez, L. F.; Fierrez, J.; and Ortega-Garcia, J. 2023. edBB-Demo: Biometrics and behavior analysis for online educational platforms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 16422–16424.

Dunnett, C. W. 1955. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 50(272): 1096–1121.

Frias-Blanco, I.; Campo-Avila, J. D.; Ramos-Jimenez, G.; Morales-Bueno, R.; Ortiz-Diaz, A.; and Caballero-Mota, Y. 2015. Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3): 810–823.

Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. 2004. Learning with drift detection. In *Proceeding of Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence*, 286–295.

Gower-Winter, B.; Krempl, G.; Dragomiretskiy, S.; Jelsma, T.; and Siebes, A. 2025. Identifying predictions that influence the future: Detecting performative concept drift in data streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 11726–11734.

Guo, H.; Ma, Z.; Ma, Y.; Zhang, X.; Chen, W.-N.; and Gong, Y.-J. 2025. DesignX: Human-Competitive Algorithm Designer for Black-Box Optimization. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*.

Huang, P.; Wang, H.; and Jin, Y. 2021. Offline data-driven evolutionary optimization based on tri-training. *Swarm and Evolutionary Computation*, 60: 100800.

Ji, J.; Wang, J.; Jiang, Z.; Jiang, J.; and Zhang, H. 2022. STDEN: Towards physics-guided neural networks for traffic flow prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 4048–4056.

Kang, Y.; Lyu, S.; Kim, J.; Park, B.; and Cho, S. 2019. Dynamic vehicle traffic control using deep reinforcement learning in automated material handling system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 9949–9950.

KddCup99. 2007. https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Kruskal, W. H.; and Wallis, W. A. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260): 583–621.

Li, C.; Yang, S.; Nguyen, T.-T.; Yu, E. L.; Yao, X.; Jin, Y.; Beyer, H.; and Suganthan, P. N. 2008. Benchmark generator for CEC 2009 competition on dynamic optimization. Technical report.

Li, J.-Y.; Zhan, Z.-H.; Wang, C.; Jin, H.; and Zhang, J. 2020. Boosting data-driven evolutionary algorithm with localized data generation. *IEEE Transactions on Evolutionary Computation*, 24(5): 923–937.

Li, K.; Chen, R.; and Yao, X. 2024. A data-driven evolutionary transfer optimization for expensive problems in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 28(5): 1396–1411.

Liu, Z.; Wang, H.; Gong, M.; and Jin, Y. 2025. Data stream driven dynamic multiobjective optimization using surrogate transfer. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Lu, P.; Lu, J.; Liu, A.; and Zhang, G. 2025. Early concept drift detection via prediction uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19124–19132.

Luo, W.; Yi, R.; Yang, B.; and Xu, P. 2019. Surrogate-assisted evolutionary framework for data-driven dynamic optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(2): 137–150.

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

Pesaranghader, A.; and Viktor, H. L. 2016. Fast hoeffding drift detection method for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, volume 9852, 96–111.

Raab, C.; Heusinger, M.; and Schleif, F.-M. 2020. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416: 340–351.

Richter, J.; Shi, J.; Chen, J.-J.; Rahnenführer, J.; and Lang, M. 2020. Model-based optimization with concept drifts. In *Proceedings of Genetic and Evolutionary Computation Conference*, 877–885.

Shlens, J. 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Styler, A.; and Nourbakhsh, I. 2015. Real-time predictive optimization for energy management in a hybrid electric vehicle. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Wan, K.; Liang, Y.; and Yoon, S. 2024. Online drift detection with maximum concept discrepancy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2924–2935.

Wang, G.; Zhang, X.; Tang, S.; Zheng, H.; and Zhao, B. Y. 2016. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, 225–236.

Wu, X.; Liu, S.; Ji, J.; Ma, L.; and Leung, V. C. 2024. A surrogate-assisted evolutionary algorithm for expensive dynamic multimodal optimzation. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 1–8.

Yang, C.; Ding, J.; Jin, Y.; and Chai, T. 2023. A data stream ensemble assisted multifactorial evolutionary algorithm for offline data-driven dynamic optimization. *Evolutionary Computation*, 1–25.

Yazdani, D.; Omidvar, M. N.; Cheng, R.; Branke, J.; Nguyen, T. T.; and Yao, X. 2022. benchmarking continuous dynamic optimization: Survey and generalized test suite. *IEEE Transactions on Cybernetics*, 52(5): 3380–3393.

Yu, H.; Da Xu, L.; Cai, H.; Li, S.; Xu, B.; and Jiang, L. 2019. A stream processing framework based on linked data for information collaborating of regional energy networks. *IEEE Transactions on Industrial Informatics*, 17(1): 179–188.

Zhang, H.; Ding, J.; Feng, L.; Tan, K. C.; and Li, K. 2024. solving expensive optimization problems in dynamic environments with meta-learning. *IEEE Transactions On Cybernetics*.

Zhong, Y.-T.; and Gong, Y.-J. 2025. Data-driven evolutionary computation under continuously streaming environments: A drift-aware approach. *IEEE Transactions on Evolutionary Computation*.

Zhong, Y.-T.; Wang, X.-C.; Sun, Y.-H.; and Gong, Y.-J. 2024. SDDObench: A benchmark for streaming data-driven optimization with concept drift. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 59–67.