OPEN ACCESS
IOP Publishing | ISECAEP
Journal of Reliability Science and Engineering
J. Reliab. Sci. Eng. **2** (2026) 015301 (16pp)
https://doi.org/10.1088/3050-2454/ae449c

Uncertainty-aware hierarchical graph neural network for reliable online drifting data stream prediction

Peng-Cheng Yan[1], Lin Qi[1], Enrico Zio[2,3] and Yan-Hui Lin[1,*]

[1] School of Reliability and Systems Engineering, Beihang University, Beijing, People's Republic of China
[2] Centre for Research on Risk and Crises (CRC), Mines Paris-PSL University, Sophia-Antipolis, France
[3] Energy Department, Politecnico di Milano, Milan, Italy

E-mail: linyanhui@buaa.edu.cn

Received 15 December 2025, revised 23 January 2026
Accepted for publication 11 February 2026
Published 26 February 2026

Abstract

Data streams generated by complex industrial systems commonly exhibit concept drift together with evolving spatial-temporal dependencies. These factors jointly impair the performance and reliability of conventional prediction models that assume stationary data distributions. To address this challenge, this paper proposes an uncertainty-aware hierarchical graph neural network (UHGNN) tailored for drifting data streams. The method explicitly extracts temporal and spatial features in data streams through a hierarchical graph architecture that integrates physical topology with high-order interactions across sensors. In addition, an epistemic uncertainty-driven drift detection mechanism is incorporated to identify concept drift in real time and to initiate fine-tuning for rapid adaptation. Experiments conducted on a real-world online industrial system, the diesel hydrofining process dataset, show that UHGNN effectively alleviates the adverse influence of concept drift and achieves higher predictive accuracy compared with baselines for drifting data streams. The method further demonstrates strong robustness with respect to key hyperparameters and maintains computational efficiency that supports industrial deployment.

Keywords: concept drift, data stream, hierarchical graph neural network, epistemic uncertainty

1. Introduction

The advancement of industrial sensing and information technologies has led to increasingly complex modern industrial systems. The volume, dimensionality, and velocity of the collected measurements continue to grow, with data often arriving continuously in the form of streams [1, 2]. As operating conditions evolve over time due to various factors such as equipment degradation, control adjustments, seasonal variations, or environmental disturbances, the underlying probability distribution of these streaming data also shifts. This phenomenon is commonly referred to as concept drift [3]. Such distributional shift can significantly degrade the performance of predictive models that were trained under earlier conditions. This challenge is prevalent across many industrial applications. For example, in the diesel hydrofining process (DHP), catalyst activity decline leads to gradual changes in monitoring data distributions [4]. For lithium-ion batteries, the behavior can evolve due to unexpected operational conditions, natural aging of batteries, and unpredictable usage patterns [5]. In wind turbines, seasonal variations in wind speed and ambient temperature, combined with cumulative mechanical wear, continuously reshape load spectra and structural stress patterns over time [6].

* Author to whom any correspondence should be addressed.

Data streams affected by concept drift pose challenges for traditional data-driven methods, which typically rely on the assumption of stationary distributions and lack mechanisms for retaining previously acquired knowledge while adapting to new operational conditions [7]. To address this issue, various strategies known collectively as drift detection and adaptation have been developed to update learning models in response to concept drift [8, 9]. For example, the dynamic extreme learning machine introduced in [10] extends the traditional extreme learning machine framework by adaptively adjusting hidden layer nodes to manage concept drift. The adaptive random forest [11] incorporates the random forests algorithm with a concept drift detection method to decide when obsolete trees should be replaced with new ones. Similarly, the self-adjusting memory k-nearest neighbor method [12] is designed to address heterogeneous forms of concept drift. Despite their practical utility, conventional machine learning-based approaches typically do not exploit the spatial dependencies inherent in the data [13]. Consequently, they are unable to leverage structural relationships in multi-sensor systems that often contain crucial information about system behavior under drifting conditions.

To explicitly characterize the interactions among multi-sensor data, graph neural networks (GNNs) can be employed. GNNs provide a natural and powerful framework for modeling multi-sensor systems by representing sensors as nodes and their spatial or functional dependencies as edges [13]. Through this representation, GNN-based models integrate node-wise temporal features with spatial relationships among sensors, enabling the extraction of spatial-temporal representations that reflect underlying system interactions. There are various GNN variants, and different types of GNNs primarily differ in how information aggregation and propagation are implemented [14]. For example, the graph convolutional network [15] performs neighborhood aggregation using graph convolution based on the adjacency structure; the graph attention network (GAT) [16] introduces a learnable attention mechanism that adaptively weights the contribution of each neighbor; and the graph sample and aggregation (GraphSAGE) [17] utilizes sampling strategies to obtain several neighbors for each node to support scalable training.

Building on the capabilities of GNN, many studies have applied them to data stream applications by transforming streaming data into spatial-temporal graphs. For instance, Wang *et al* [18] presented a streaming GNN model based on continual learning, which is trained incrementally and updates node representations at each time step. A streaming GNN system designed in [19] abstracts the input dynamic graph as a chronologically ordered stream of events and processes it using an optimized sliding window. Yu *et al* [20] introduced a GNN-based approach for streaming data that utilizes a competence model to explore latent semantic correlations in the stream. Collectively, these works demonstrate the effectiveness and adaptability of GNNs in handling data streams.

Capitalizing on the strong performance of GNNs in spatial-temporal structure modeling, recent work has extended GNNs to drifting data streams. For example, Zhou *et al* [21] proposed a self-adaptation framework based on GNN, which reconstructs the GNN learning process from a concept drift adaptation perspective. Building on this idea, the authors further developed a multi-stream self-adaptation method based on graph regularization [22], eliminates the need for a predefined graph by modeling deep spatial-temporal dependencies across multiple streams using Gumbel sampling and adaptive adjacency matrix. In [23], a continuous graph learning-based self-adaptation framework was introduced for multi-stream concept drift, incorporating an adaptive graph generator and an adaptive diffusion graph attention module that updates edge weights online to effectively handle structural drift. These advances demonstrate that graph-based modeling achieves significant effectiveness under concept drift, yet notable limitations still remain [24].

First, some existing methods rely on time-invariant information transmission, which limit their capacity to capture temporal dependencies in graph data. This is particularly vulnerable to long-term distributional shifts: as correlations between nodes change over time, time-invariant edge weights fail to reflect temporal relationship dynamics, leading to significant performance degradation. Second, most current GNN-based methods for drifting data stream are based on flat message-passing mechanisms, meaning they aggregate information only across observed edges in the original graph. This design restricts their ability to encode high-order spatial dependencies [25]. In industrial systems, such high-order dependencies often correspond to physically meaningful interactions among subsystems composed of multiple components, information that is critical for effective drift detection. Third, prior research has largely focused on learning spatial-temporal dependencies for point prediction while paying insufficient attention to modeling uncertainty. Nevertheless, real-world prediction tasks inevitably involve uncertainties [26], and quantifying uncertainty, especially the epistemic uncertainty arising from limited knowledge, plays a vital role in robust online prediction [27]. Epistemic uncertainty increases sharply when the model encounters inputs that differ from the training distribution, offering a valuable indicator for drift detection [28]. Moreover, in online prediction contexts where true labels may be delayed or unavailable, epistemic uncertainty can serve as an effective label-free indicator of distributional change. Consequently, explicitly modeling epistemic uncertainty is essential for timely detection of distribution shifts, supporting adaptive updates or intervention in evolving streaming environments.

Building on these insights, this paper introduces an uncertainty-aware hierarchical GNN (UHGNN) for reliable online prediction in drifting data streams. The proposed UHGNN first extracts temporal features from a dynamic perspective to derive graph structures, then organizes node representations into a hierarchical graph that captures both local and global spatial dependencies, thereby enabling high-order interaction modeling. Unlike flattened graph models, UHGNN employs a hierarchical architecture to capture evolving dependencies at the subsystem level, enabling the model to distinguish systemic concept drifts from localized sensor noise and providing a more robust foundation

for drift detection. During inference, epistemic uncertainty is explicitly quantified and leveraged as a drift indicator to trigger adaptation without requiring immediate ground-truth labels. By jointly modeling dynamic graph structures, hierarchical dependencies, and epistemic uncertainty, UHGNN is designed to deliver robust long-term predictions in streaming environments. The effectiveness of UHGNN is validated through experiments on a real-world industrial streaming dataset. Results show that hierarchical graph modeling enhances the capture of temporal and spatial dependencies, while epistemic uncertainty-driven drift detection (EUDD) enables fast and reliable adaptation. The UHGNN serves as a general architecture that can be readily adapted to diverse industrial settings by aligning its hierarchical nodes with the specific physical topology or functional regions of the target system.

The rest of this article is organized as follows. Section 2 presents the problem formulation. Section 3 describes the methodology of the proposed UHGNN. Section 4 reports the experimental results and provides discussions demonstrating the effectiveness of the method. Finally, section 5 concludes the article.

2. Problem formulation

2.1. Hierarchical graph-based regression

Consider a system equipped with N sensors. The system state at time t can be represented as a graph $G_t = (\boldsymbol{X}_t, A_t)$, where $\boldsymbol{X}_t = \left[x_t^1, \ldots, x_t^N\right]$ denotes the node feature matrix and x_t^i represents the measurement from the ith sensor at time t. The adjacency matrix $A_t \epsilon\, \mathbb{R}^{N\times N}$ characterizes the spatial relationships among sensors. To capture high-order dependencies and aggregate information hierarchically, this formulation can be extended to a hierarchical graph representation through clustering nodes by introducing a hidden layer [29]. The hierarchical graph can be expressed as $G_t = (\boldsymbol{X}_t, A_t, \boldsymbol{\Phi}_t, \Theta_t)$, where $\boldsymbol{\Phi}_t = \left[\phi_t^1, \ldots, \phi_t^M\right]$ with $M < N$ denotes the feature matrix of hidden nodes, and each hidden node feature ϕ_t^i is obtained by aggregating information from a subset of original node features in $\boldsymbol{X}_t$. The matrix $\Theta_t \epsilon\, \mathbb{R}^{M\times M}$ specifies the adjacency structure among hidden nodes.

Building on this hierarchical graph-based representation, the system's temporal evolution can be naturally represented by a sequence of graphs $\boldsymbol{\mathcal{G}}_t = [G_{t-W+1}, \ldots, G_{t-1}, G_t]$, where W denotes the length of observation window. This sequence provides a rich structure for capturing both spatial correlations among sensors and temporal dependencies associated with system dynamics. For a regression task, the objective is to establish a predictive model f that maps $\boldsymbol{\mathcal{G}}_t$ to the target y_t at time t, i.e., f: $\boldsymbol{\mathcal{G}}_t \rightarrow y_t$.

2.2. Concept drift detection and adaptation

In an online industrial scenario, data typically arrive in the form of a stream, which can be represented as a sequence $[S_1, S_2, \ldots, S_i, \ldots]$, where $S_i = \{\boldsymbol{\mathcal{G}}_t, y_t\}_{t=t_{0,i}}^{t_{\mathrm{end},i}}$ represents either a chunk of samples ($t_{0,i} < t_{\mathrm{end},i}$) or a single sample ($t_{0,i} = t_{\mathrm{end},i}$), $t_{0,i}$ and $t_{\mathrm{end},i}$ represent the start and end times of S_i, respectively. Each sample is independently generated from an underlying data distribution, i.e., $(\boldsymbol{\mathcal{G}}_t, y_t) \sim p_t(\boldsymbol{\mathcal{G}}, y)$. Concept drift arises when this distribution changes over time, such that $p_{t+1}(\boldsymbol{\mathcal{G}}, y) \neq p_t(\boldsymbol{\mathcal{G}}, y)$; in practice, such changes may also be recurring, meaning that a previously observed concept reappears. As the model receives input data sequentially, an essential task is to determine whether newly arriving data indicate a distributional change. Drift detection mechanisms therefore monitor the discrepancy between current behavior and historical patterns to decide whether a drift has taken place. Once a drift is detected, the model proceeds to adapt its parameters. Specifically, when the true label y_t becomes available, the objective is to align the predictive model f with respect to the current distribution $p_t(\boldsymbol{\mathcal{G}}, y)$, formulated as $\min l(f(\boldsymbol{\mathcal{G}}), y | (\boldsymbol{\mathcal{G}}, y) \sim p_t(\boldsymbol{\mathcal{G}}, y))$, where $l(\cdot)$ represents the loss function. The detection and adaptation process is continuously repeated as the stream progresses to S_{i+1}.

3. Methodology

This section elaborates on the methodology of the UHGNN. First, an overview of the overall framework is presented in section 3.1. Building upon this framework, section 3.2 details the temporal feature extraction process. Subsequently, section 3.3 introduces the spatial feature extraction process. Finally, section 3.4 describes the EUDD mechanism.

3.1. The proposed UHGNN framework

The architecture of UHGNN is illustrated in figure 1. Specifically, UHGNN first constructs graphs with spatial structure among sensors based on their physical connections or spatial proximity. It then applies an architecture consisting of GAT and long short-term memory network (LSTM) to extract temporal features from the nodes. This design enables the model to capture nonlinear temporal dependencies and derive dynamic edge weights. Subsequently, UHGNN performs clustering based on domain knowledge and groups nodes within the same functional region to form the hidden node layer. In the hidden node layer, GraphSAGE is employed to aggregate and update information across hidden nodes, producing node embeddings that encode both temporal features and spatial structure. Finally, UHGNN incorporates an EUDD concept drift detection method. This mechanism leverages epistemic uncertainty signals to identify changes in the data distribution within the data stream. Once drift is detected, the model performs fine-tuning to dynamically update its parameters. Through these integrated components, UHGNN achieves accurate and dynamic prediction by fusing temporal and spatial information from multiple sensors within evolving data streams.

3.2. Temporal feature extraction

Before extracting temporal features from the graph sequence $\boldsymbol{\mathcal{G}}_t$, it is necessary to preprocess the original node features to

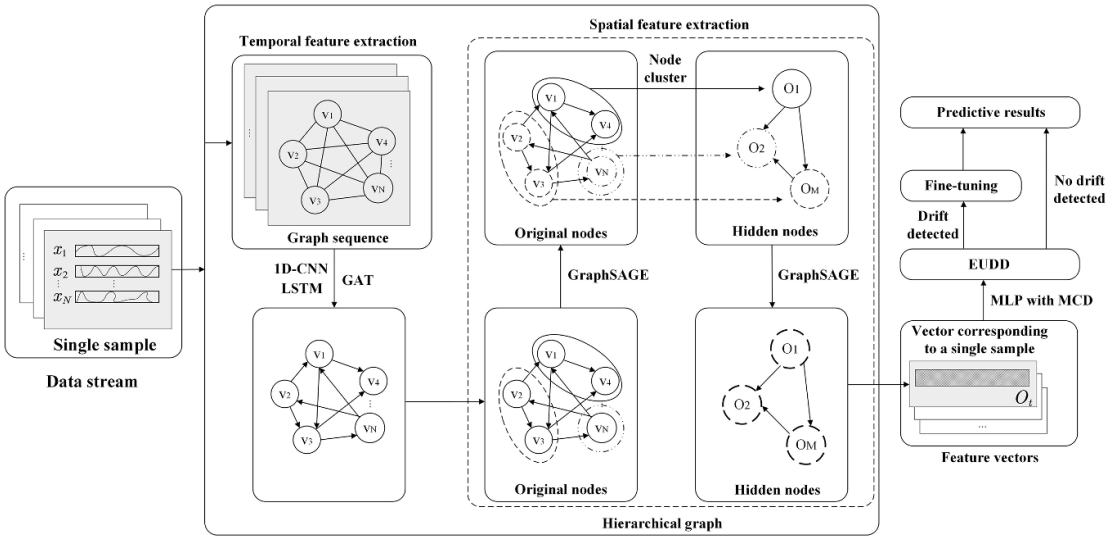


Figure 1. Overall framework of UHGNN. The architecture comprises three main stages: (1) temporal feature extraction; (2) spatial feature extraction; (3) epistemic uncertainty-driven drift detection.

reduce noise and improve the stability of subsequent spatial relationship modeling. This is essential because industrial data are often influenced by sensor errors and external disturbances, which can lead to instability in prediction. In this study, we apply a one-dimensional convolutional neural network (1D-CNN) to filter original features of the sensors, enabling effective denoising and smoothing. Let $\boldsymbol{x}_t^i = \left[x_{t-W+1}^i, \ldots, x_{t-1}^i, x_t^i\right]$ denote the input feature vector of the ith sensor in the time window, where W represents the length of time window. For simplicity, let $t_0 = t - W + 1$, the output of 1D-CNN layer at time $\tau \epsilon [t_0, t]$ is then given by:

$$\begin{aligned} h_\tau^i &= \left[h_\tau^{i,1}, h_\tau^{i,2}, \ldots, h_\tau^{i,S}\right] \\ h_\tau^{i,s} &= \sigma\left(\sum_{p=1}^{r} K_{s,p} x_\tau^i + b_s\right) \end{aligned} \tag{1}$$

where S represents the number of kernels, $\sigma(\cdot)$ represents the activation function, r represents kernel size, $K_{s,p}$ represents the convolution kernel weights, and b_s represents bias term. In this work, the 1D-CNN layer is employed as a preliminary step for denoising and local feature smoothing. The use of a fixed kernel size is a strategic trade-off to maintain computational efficiency and real-time responsiveness in large-scale industrial systems. By effectively suppressing high-frequency noise while preserving feature dimensionality, this layer facilitates the subsequent hierarchical graph processing. Potential variability in noise characteristics across heterogeneous sensors is then addressed by the hierarchical graph layers through multi-scale spatial aggregation, ensuring robust feature extraction.

After completing the feature preprocessing, to extract temporal features, UHGNN constructs time-varying edge weights based on the correlations between nodes. To achieve this, GAT is employed to generate a set of dynamic attention coefficients $\{\alpha_{t_0}^{ij}, \ldots, \alpha_{t-1}^{ij}, \alpha_t^{ij}\}$ for each node pair within the time window. Specifically, to compute the attention score α_τ^{ij} at time step τ, a scoring function between the ith and jth nodes is first introduced as follows:

$$e_\tau^{ij} = \boldsymbol{\alpha} \cdot \text{LeakyReLU}\left(\mathbf{W}_{\text{att}} \cdot \left[h_\tau^i \,||\, h_\tau^j \,||\, \text{Tembedding}(\tau)\right]\right) \tag{2}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{1\times d'}$ represents the attention vector, and $\mathbf{W}_{\text{att}} \in \mathbb{R}^{d'\times(2S+d)}$ represents the feature transformation, both are learnable parameters, d' and d represent the dimensionality of the parameter vector and the time embedding vector, respectively. This formulation defines a time-varying similarity function between nodes. It captures not only the correlation between node features but also incorporates temporal information through the term $\text{Tembedding}(\tau) \in \mathbb{R}^d$, enabling the modeling of interactions at different time steps. To make the attention mechanism capture periodic temporal dependencies, $\text{Tembedding}(\tau)$ maps real timestamp into a representation that reflects the actual temporal information, and is defined as follows:

$$\text{Tembedding}(\tau) = \left[\cos(\omega_1\tau), \cos(\omega_2\tau), \ldots, \cos(\omega_d\tau)\right] \tag{3}$$

where $\omega_i = 1/10^{i/d}$ for $i = 1, 2, \ldots, d$, and the value of d is determined based on the complexity of the data and the requirements of the model. The representation in (3) indicates that the time embedding vector is generated using a set of continuously differentiable cosine functions, which can be regarded as an expansion of temporal basis functions at different frequencies. By incorporating periodic components into the feature space, the model is able to learn temporal variations and phase differences across different time scales, making temporal dependencies more readily represented within the continuous latent space of the neural network.

To ensure that the scales of edges across different node pairs are consistent and suitable for subsequent computation, a SoftMax normalization is applied over all neighbors

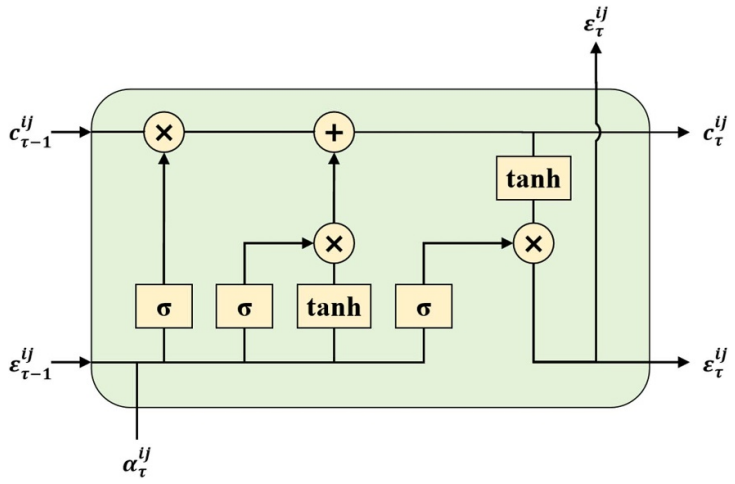


Figure 2. The architecture of LSTM.

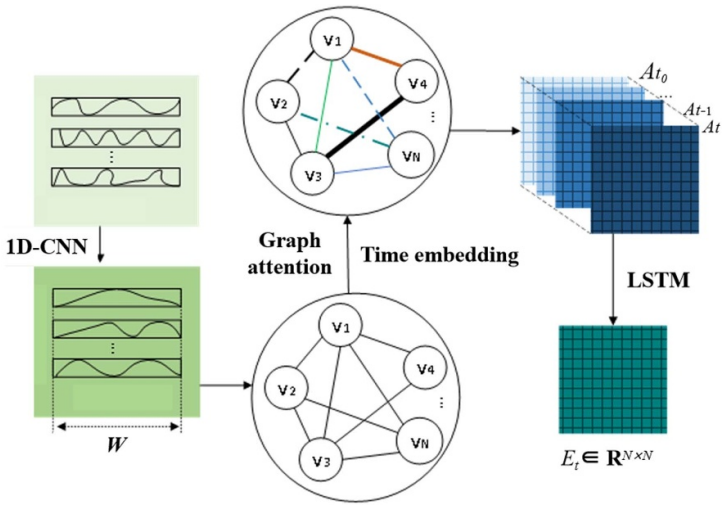


Figure 3. Flowchart of temporal feature extraction. The pipeline includes: (1) signal denoising via 1D-CNN; (2) graph structure mapping; (3) time embedding; (4) correlation capture via graph attention; (5) sequential feature extraction via LSTM.

of the *i*th node, yielding the final attention coefficients α_{τ}^{ij} expressed as

$$\alpha_{\tau}^{ij} = \frac{\exp\left(\mathrm{e}_{\tau}^{ij}\right)}{\sum_{k\in \mathrm{Neighbor}(i)} \exp\left(\mathrm{e}_{\tau}^{ik}\right)} \tag{4}$$

where $\mathrm{Neighbor}(i)$ denotes the set of all neighbor nodes of the *i*th node, which is determined by the adjacency structure. The attention coefficient α_{τ}^{ij} characterizes the strength of the dependency of the *i*th node on the *j*th node at time step τ. It reflects the importance of the connections between nodes in the graph and can dynamically capture how the spatial relationships among nodes evolve over time.

Although the GAT mechanism can represent the instantaneous relationships among nodes, it cannot explicitly capture the dynamic evolution patterns of time series and therefore fails to extract deeper temporal information. To address this limitation, an LSTM module is introduced after the GAT to model the temporal variations of the attention coefficients. For edge between the *i*th and *j*th nodes, its hidden feature at time step τ is updated as:

$$\varepsilon_{\tau}^{ij} = \mathrm{ReLU}\left(\mathrm{LSTM}\left(\alpha_{\tau}^{ij}, \varepsilon_{\tau-1}^{ij}\right)\right) \tag{5}$$

where ε_{τ}^{ij} represents the hidden state of the edge between the *i*th and *j*th nodes at time step τ, and $\varepsilon_{t_0}^{ij} = 0$. The architecture of LSTM is illustrated in figure 2, where c_{τ}^{ij} represents historical information and $c_{t_0}^{ij} = 0$.

The incorporation of LSTM enables the model to explicitly preserve temporal information, capturing the non-stationary evolution of attention over time. At the termination of each time window $[t_0, t]$, ε_t^{ij} represents the steady weight of the edge between the *i*th and *j*th nodes at the final time step *t* and is assigned as the corresponding element in the weight matrix E_t. By encapsulating temporal dynamics across the graph sequence $\mathcal{G}_t$, E_t provides structured input essential for subsequent stages of processing. Overall, the process of temporal feature extraction can be summarized as shown in figure 3, with detailed steps outlined in Algorithm 1.

Algorithm 1. Temporal feature extraction.

Input: Graph sequence $\mathcal{G}_t$
Output: Weight matrix of edges E_t

For $\tau \leftarrow t_0$ **to** t **do:**
  For $i \leftarrow 1$ **to** N **do:**
    Compute h_τ^i via (1).
  Obtain Tembedding (τ) via (3).
  End for
End for
For $i \leftarrow 1$ **to** N**do:**
  For $j \leftarrow 1$ **to** N**do:**
    Compute α_τ^{ij} via (2) and (4).
  End for
End for
For $i \leftarrow 1$ **to** N**do:**
  For $j \leftarrow 1$ **to** N**do:**
    Compute ε_τ^{ij} via (5).
  End for
End for
Obtain E_t that integrates temporal information.

3.3. Spatial feature extraction

After obtaining the edge weight matrix E_t that contains temporal information, to further extract spatial information, this study introduces a hierarchical aggregation mechanism, in which a hidden node layer is constructed to extract region-level spatial information across different functional regions. To establish the hidden node layer, the original nodes are categorized based on domain knowledge and engineering experience [30], ensuring that original nodes within the same cluster exhibit strong spatial or functional correlations, while correlations across different clusters remain weak. Formally, the original nodes are partitioned into M clusters, denoted as $\{c_1, c_2, \ldots, c_M\}$, where each cluster represents a group of nodes that are either physically adjacent or functionally correlated. For a given cluster c_i at time step t, a corresponding subgraph $G_t^{(i)}$ can be constructed using the node features and the local edge weight associated with that cluster, expressed as

$$G_t^{(i)} = \left(\boldsymbol{X}_t^{(i)}, E_t^{(i)}\right) \tag{6}$$

where $\boldsymbol{X}_t^{(i)}$ denotes the sensor feature of the nodes belonging to cluster c_i, and $E_t^{(i)}$ represents the edge weight matrix extracted from the E_t. This presentation not only preserves the original physical connectivity but also provides a structured foundation for subsequent region-level aggregation. For each subgraph $G_t^{(i)}$, a GraphSAGE algorithm is applied to perform feature aggregation and updating among its internal nodes, enabling the capture of spatial interaction patterns within each cluster. Specifically, the recursive aggregation process of GraphSAGE can be expressed as follows:

$$Z_t^{(i)} = \text{GraphSAGE}^L\left(X_t^{(i)}, E_t^{(i)}\right). \tag{7}$$

Where $Z_t^{(i)} \in \mathbb{R}^{N^{(i)} \times d_z}$ denotes the resulting feature matrix after processing with L GraphSAGE layers, $N^{(i)}$ denotes the number of nodes in the subgraph $G_t^{(i)}$ and d_z denotes the output feature dimension. At the lth layer, the aggregation and update steps are given as follows:

$$\begin{aligned} z_t^{j,(l)} &= \text{ReLU}\left(\mathbf{W}^{(l)}\left[z_t^{j,(l-1)} || m_t^{j,(l)}\right]\right) \\ m_t^{j,(l)} &= \frac{\sum_{k\in\text{Neighbor}(j)} E_t^{jk} z_t^{k,(l-1)}}{\sum_{k\in\text{Neighbor}(j)} E_t^{jk}} \end{aligned} \tag{8}$$

where represents the feature information of the jth node at the lth GraphSAGE layer, $m_t^{j,(l)}$ denotes the aggregated information at the lth layer, and E_t^{jk} denotes the edge weight between the jth and kth nodes. $\mathbf{W}^{(l)} \in \mathbb{R}^{d_z \times 2d_z}$ is the learnable parameter matrix. To ensure a unified feature dimensionality across different clusters in the hidden node layer, the node embeddings within each clustered subgraph are compressed into a region-level feature vector. Following the approach in [31], this study applies a concatenation of max pooling and mean pooling so as to retain both global trends and extreme-value characteristics:

$$Z_t^{(i)} = \text{CONCAT}\left(Z_t^{(i),\text{mean}}, Z_t^{(i),\text{max}}\right) \tag{9}$$

where $Z_t^{(i),\text{mean}} = \frac{1}{N^{(i)}}\sum_{n=1}^{N^{(i)}} z_t^{n,(L)}$ and $Z_t^{(i),\text{max}} = \max_{1\leqslant n\leqslant N^{(i)}}\{z_t^{n,(L)}\}$. Through the pooling process, $Z_t^{(i)}$ aggregates the multi-dimensional features of nodes within each cluster into a cluster-level vector, preserving the overall statistical characteristics of the cluster while enhancing sensitivity to abnormal nodes.

Building on this foundation, we hierarchically construct a hidden graph containing M nodes, where $Z_t^{(i)}$ serves as the feature vector of the ith node in this hidden graph. The rationale for the hierarchical graph design is to enable the model to perceive spatial features across different scales. While the base layer focuses on local sensor interactions, the hidden layer utilizes a hierarchical mechanism to synthesize these interactions into meso-scale representations. This design is particularly effective for drifting data streams, as it allows the model to monitor the collective behavior of sensor groups. By capturing these subsystem-level features, UHGNN can more effectively distinguish between transient local fluctuations and genuine concept drifts that affect entire functional units, thereby significantly enhancing the discriminative power of the subsequent drift detection process.

To process the hidden graph, a GAT module is first employed to generate attention-based inter-node weights, then a GraphSAGE module is applied to obtain the final output matrix $O_t \in \mathbb{R}^{M\times d_o}$ of the hidden graph, where d_o represents the output dimensionality. Finally, to obtain a global representation of the entire hierarchical graph, a max pooling operation is applied to O_t, transforming it into a 1D feature vector that serves as the final output of the graph-based model. This output not only integrates the local temporal features extracted from the base-node layer but also consolidates the global spatial dependencies captured by the hidden-node layer, thereby

providing structured inputs for subsequent uncertainty quantification and drift detection tasks.

3.4. Epistemic uncertainty-driven concept drift detection

The temporal and spatial feature extraction process yields a corresponding 1D representation vector. This vector jointly encodes multilayer information across both temporal and spatial dimensions, effectively capturing the sample's dynamic characteristics and structural dependencies. Subsequently, UHGNN feeds this vector into a fully connected (FC) layer for final mapping, producing the prediction output.

To quantify the epistemic uncertainty inherent in the model's predictions and thereby enable concept drift detection in data streams, UHGNN incorporates Monte Carlo dropout (MCD) [32] during both the training and inference stages of the FC layer. This method is theoretically equivalent to performing a Bayesian approximation of the model parameters, where repeated stochastic dropout operations serve as sampling from the parameter space [33]. Compared to other representative uncertainty estimation approaches like deep ensemble or Bayesian neural networks, MCD facilitates real-time uncertainty estimation in industrial streams without excessive computational costs or complex architectural changes. Its practical balance of efficiency and ease of implementation accounts for its widespread adoption in industrial applications [34]. Specifically, during training, randomly dropping neuron connections approximates sampling over multiple network substructures, enabling the model to account for parameter uncertainty. During inference, dropout remains active, and multiple stochastic forward passes are executed, with each pass corresponding to a distinct sample from the parameter subspace. Let T denote the number of stochastic forward passes, the prediction result at time t $\hat{y}_t$ is obtained by averaging the T outputs as follows:

$$\hat{y}_t = \frac{1}{T}\sum_{i=1}^{T}\hat{y}_t^{(i)} \tag{10}$$

where $\hat{y}_t^{(i)}$ represents the prediction result in the ith stochastic forward pass.

This procedure enables the model to capture output fluctuations induced by parameter uncertainty, thereby providing the foundation for subsequent uncertainty quantification. In this study, we utilize variance as the uncertainty measure. Consequently, the epistemic uncertainty arising from the variability of parameters can be obtained by computing the variance of the outputs generated from the T stochastic forward passes, namely [35]:

$$\hat{\sigma}_t^2 = \frac{1}{T}\sum_{i=1}^{T}\left(\hat{y}_t^{(i)} - \hat{y}_t\right)^2 \tag{11}$$

where $\hat{\sigma}_t^2$ represents the degree of predictive fluctuation of a sample in the input feature space and characterizes the epistemic uncertainty of the model. Notably, the estimated epistemic uncertainty in this framework is specifically utilized as a sensitive indicator for detecting distribution shifts, rather than as a direct component for constructing prediction intervals. While the epistemic uncertainty estimates in this work are not formally calibrated, they serve as a robust drift trigger. For identifying distribution shifts, the relative sensitivity of the uncertainty value is more vital than its absolute statistical calibration. As demonstrated in our experiments, the significant fluctuations in epistemic uncertainty during concept drift provide a reliable signal for timely model adaptation.

After obtaining the epistemic uncertainty of each sample, UHGNN feeds it into the adaptive window (ADWIN) [36] algorithm to detect significant changes in uncertainty over time. ADWIN dynamically adjusts the size of its sliding window and statistically evaluates the difference in the mean uncertainty between the current window and a historical window. When this difference reaches a statistically significant level, a concept drift is declared. Once a drift is detected, the model updates its parameters using new data as incremental samples for fine-tuning. This process involves iterating the model's parameter optimization several times to adapt to the updated distribution characteristics, thereby maintaining stable predictive performance. The updated model can rapidly respond to distributional changes in dynamic environments, ensuring reliable and accurate predictions.

4. Case study

In this study, we adopt the real-world DHP dataset as the experimental benchmark. The dataset information and corresponding predictive results are presented to systematically validate the effectiveness and superior performance of the proposed UHGNN for drifting data streams.

4.1. Dataset description

The DHP dataset was collected from a real petrochemical plant. A simplified flow diagram of the process, with several key sensors, is shown in figure 4. Diesel hydrofining is a catalytic conversion process in which feed oil and hydrogen undergo a series of reactions as they pass through catalyst beds inside the reactor under high temperature and high pressure. Under the action of the hydrotreating catalyst, nonhydrocarbon compounds such as sulfur, nitrogen, and oxygen are converted into hydrocarbons, hydrogen sulfide, ammonia, water, and other products, thereby effectively reducing the sulfur content in diesel. The sulfur content is influenced by multiple factors, including the physical properties of the feed oil, the activity level of the catalyst, and key operating parameters such as reaction temperature, pressure, and flow rate. For this dataset, the prediction task is to estimate the sulfur content of diesel based on online monitoring data provided by multiple sensors. These sensors continuously record the physical properties of the feed oil and the operating conditions of the process, supplying the necessary data for the model to achieve accurate sulfur-content prediction.

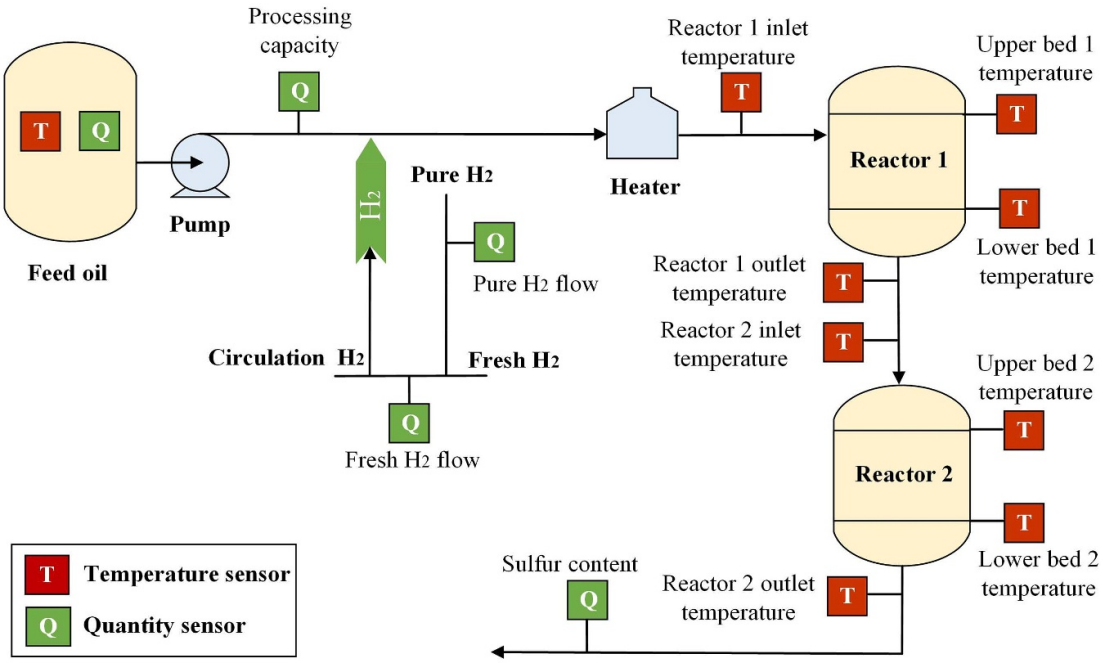


Figure 4. Illustration of the DHP.

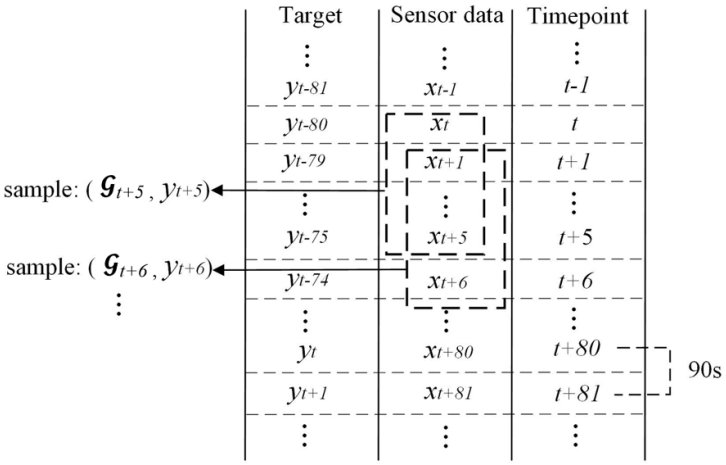


Figure 5. Illustration of data generation for the DHP dataset.

The data generation procedure of the DHP dataset is illustrated in figure 5. At each time point, measurements collected from 19 sensors are recorded. The input used for predicting sulfur content is a multivariate time series consisting of 6 consecutive monitoring points. During online monitoring of the DHP operation, the interval between two adjacent sampling times is 90 s, and the target sulfur content ranges between 3 and 7. As time progresses, the activity of the catalyst gradually degrades and may eventually impair the reaction. Since no sensor can directly quantify changes in catalyst activity, the distribution of monitoring samples drifts over time under the influence of catalyst degradation.

To simulate the practical scenario of online sulfur-content prediction, the DHP dataset is divided into an offline dataset and an online dataset, which are used for offline model training and online prediction, respectively. The number of samples contained in each dataset, as well as their actual sampling periods, are summarized in table 1. To evaluate the accuracy of sulfur-content prediction, this study employs the root mean squared error (RMSE) as the evaluation metric.

In the subsequent experiments, the online dataset is evenly divided into 18 data blocks, each containing 3 consecutive days of collected data, producing a total of 3890 samples. To examine the general impact of catalyst activity degradation on the distributional characteristics of the monitoring data, we first perform a simple LSTM on the online dataset without considering concept drift, with the results shown in figure 6. The fluctuations in RMSE observed across Data Blocks 2 through 6 indicate the presence of concept drift within the DHP dataset and reveal a repeated drift pattern.

Table 1. Sample size of DHP dataset.

Dataset	Sample size	Real time
Total	138 674	07/18 0:00–11/07 0:00
Offline	68 646	07/18 0:00–09/09 0:00
Online	70 028	09/09 0:00–11/07 0:00

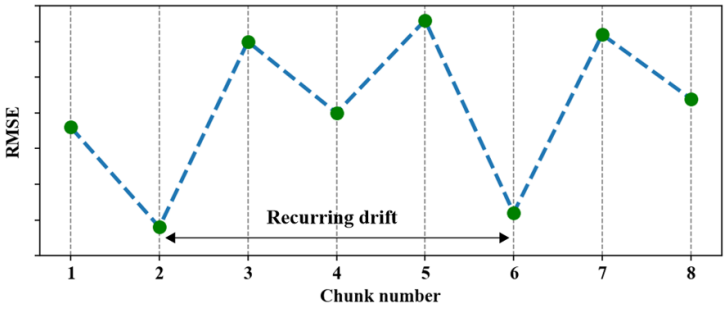


Figure 6. Performance of a LSTM on DHP's online dataset without drift detection.

Table 2. Functional regions of sensors in the DHP dataset.

Number	Functional regions	Description
1	Raw-material region	Diesel distillate yield
2		Diesel temperature
3		Diesel byproduct temperature
4		Fractionation column flow
5		Cold diesel make-up flow
6	Hydrofining region	Total feed flow
7		Pure H_2 flow
8		Circulation H_2 flow
9		Fresh H_2 flow
10	Reactor 1	Reactor 1 inlet temperature
11		Upper bed 1 temperature
12		Lower bed 1 temperature
13		Reactor 1 outlet temperature
14	Reactor 1	Reactor 2 inlet temperature
15		Upper bed 2 temperature
16		Lower bed 2 inlet temperature
17		Lower bed 2 temperature
18		Reactor 1 outlet temperature
19	Monitoring region	Sulfur content

4.2. Model setup

In the real-world industrial environment, explicit physical connections often exist among sensors. For the DHP dataset, the monitoring data exhibit not only typical temporal characteristics but also rich spatial structural information. These connections reflect underlying physical mechanisms such as material flow, energy transfer, and process coupling among different units. Consequently, the spatial dependencies among sensors play a critical role in modeling.

Based on the inherent physical structure, the entire sensor system can be formalized as a graph, where nodes represent the sensors and edges describe the topological connections among sensors. This graph representation provides the structural foundation for subsequent GNN learning. Specifically, the DHP dataset contains 19 sensors whose functional region and information are summarized in table 2. These sensors can be grouped into five functional regions corresponding to the raw-material region, the hydrotreating region, Reactor 1, Reactor 2, and the monitoring region. Each region reflects a distinct stage of the industrial process and corresponds to different monitoring objectives.

Integrating the aforementioned engineering knowledge, the graph construction of UHGNN is conducted as illustrated in figure 7. First, each sensor is modeled as a node, and its

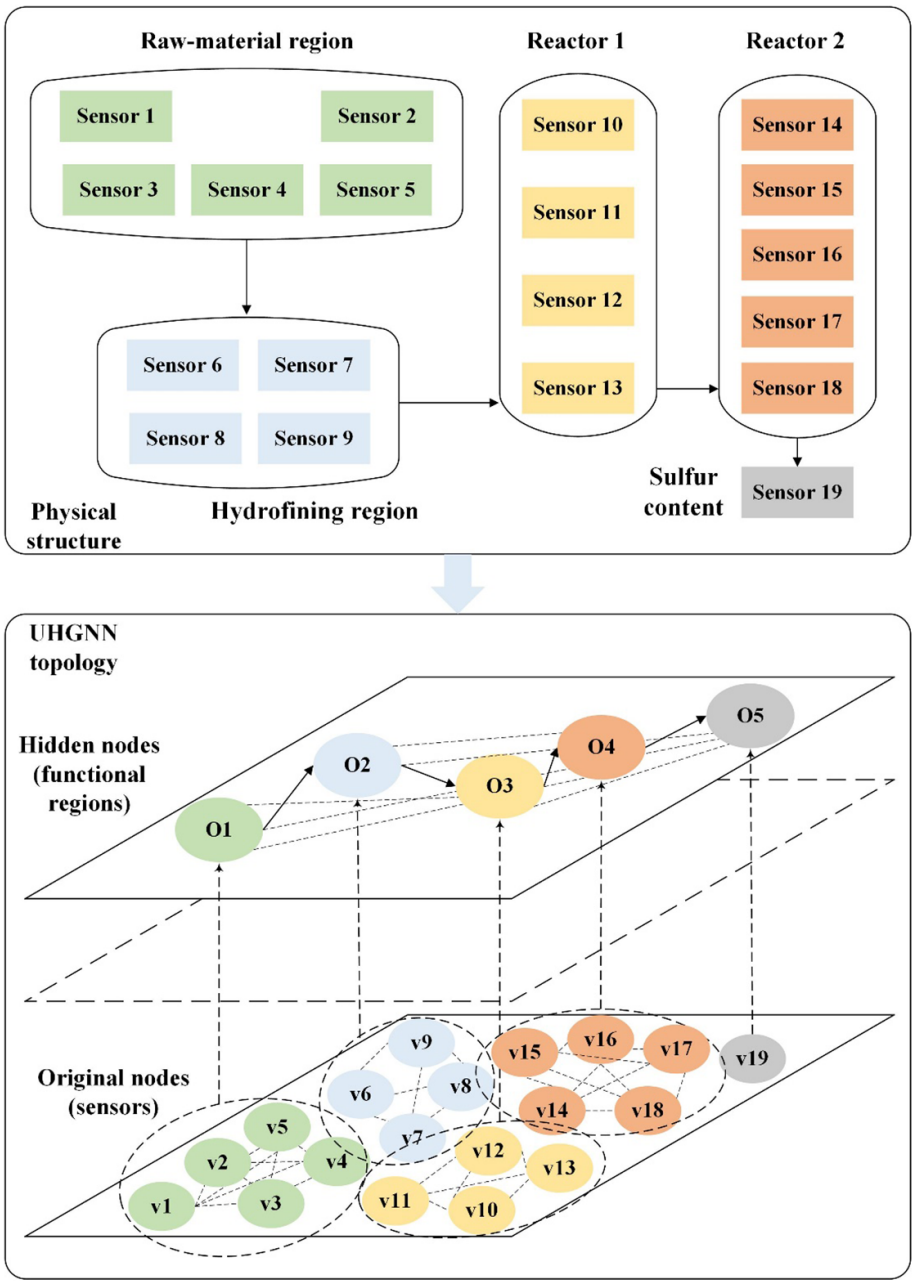


Figure 7. Illustration of transforming the physical structure of DHP into the topology of UHGNN. The sensors are partitioned into five functional regions based on the DHP system layout, which serve as the basis for constructing the hidden nodes in the hierarchical topology.

observed time series in time window is used as the node feature input. According to the physical structure of the system, the 19 nodes are divided into 5 clusters, each corresponding to one of the 5 functional regions. Nodes within the same cluster are connected by edges that reflect their local physical relationships, and the associated edge weights are learned during model training. For the hidden node layer, each cluster is represented by a hidden node that captures the overall state characteristics of its corresponding functional region. Hidden nodes are also connected to represent the function dependencies among different regions, and these edge weights are likewise determined through model training. Through this process, the raw time-series sensor data are transformed into hierarchical graph-based data, thus establishing a mapping from the physical process structure to the topological structure. This allows the model to effectively capture temporal dependencies and spatial relationships inherent in the industrial process.

It is worth noting that the hierarchical structure of UHGNN is based on the static physical topology and functional regions of the DHP system. This expert-defined clustering ensures that the model remains computationally efficient and physically interpretable. For scenarios where the system structure is unknown or dynamic, exploring adaptive hierarchical modeling remains a valuable direction for future research.

Table 3. Architecture and hyperparameters of UHGNN.

Component	Layer	Hyperparameter
Temporal feature extraction	1D-CNN	Kernel size = 3, Activation function: LeakyReLU
	Time embedding	Dimension $d = 4$
	GAT	Input dimension = 36, Hidden units = 16
	LSTM	Input dimension = 1, Hidden units = 16
Spatial feature extraction	GraphSAGE on original node layer	Input dimension = 12, Hidden units = 16, Dropout rate = 0.1
	GAT on original node layer	Input dimension = 64, Hidden units = 32
	GraphSAGE on hidden nodes	Input dimension = 32, Hidden units = 16, Dropout rate = 0.1
	GAT on hidden node layer	Input dimension = 64, Output dimension = 32
	Pooling layer	Output dimension = 64
	Activation function	LeakyReLU
FC	FC layer 1	Neurons = 64
	FC layer 2	Neurons = 32
	Activation function	ReLU

UHGNN comprises three main components: a temporal feature extraction module, a spatial feature extraction module, and a FC layer. The corresponding hyperparameter settings are summarized in table 3.

For data partitioning, the dataset is separated into an offline phase and an online phase. The offline dataset is used for initial training and validation, while the online dataset supports continual learning and predictive updating. For the offline data, we randomly select 80% of it as the training dataset, 10% as validation dataset, and the remaining as the test dataset. This split ensures a reliable assessment of the model's generalization capability.

4.3. Comparative results

To verify the effectiveness of UHGNN, we compare it with the following baselines for drifting data stream regression on the DHP dataset.

Chebyshev-based under sampling (ChebyUS) [37]: A method that leverages Chebyshev-based distance measures to identify samples that deviate from the current data distribution and update the model accordingly.

Chebyshev-based over sampling (ChebyOS) [37]: In contrast to ChebyUS, this method over-samples rare samples based on Chebyshev's inequality values during training process to create a more balanced dataset.

Multiple output regression for streaming time series (MORSTS) [38]: An online ensemble framework that introduces new base learners when a noticeable increase in prediction error is observed.

Histogram-based under sampling (HistUS) [39] A method that employs an online histogram to assess how well incoming samples conform to the current distribution.

Histogram-based over sampling (HistOS) [39]: Similar to HistUS, but this method reinforces rare samples by repeatedly retraining the model whenever the rarity-aware probability exceeds a random threshold.

Deep ensemble [40]: A neural network-based ensemble framework in which LSTM is adopted as the base learner in this study. To enable drift detection, several drift detection techniques are integrated into the framework, with the ensemble being fine-tuned upon drift detection to maintain adaptation to the current data stream. Specifically, two drift detection strategies are employed: **ADWIN** [36], which detects drift by monitoring statistical parameters such as the mean or variance of input samples, and **EUDD** (proposed in this study), which extends ADWIN by using epistemic uncertainty values instead of sample targets for drift detection.

Continuous graph learning-based self-adaptation for multi-stream concept drift (CGLM) [23]: A graph-based method that addresses multi-stream concept drift by dynamically generating correlation graphs and performing self-adaptation through subgraph updating based on real-time drift detection.

The online prediction performance of all methods is summarized in figure 8. The results indicate that UHGNN delivers consistently reliable predictions across most chunks. Notably, at locations where concept drift occurs, such as the 6-th and the 10-th chunks, UHGNN promptly detects drift and performs adaptive model adjustments. This illustrates the successful integration of EUDD mechanism with a hierarchical graph representation framework.

Table 4 presents the average RMSE values for all methods. It can be observed that UHGNN achieves the lowest average RMSE and significantly outperforms all baselines. ChebyOS and ChebyUS perform the worst due to their lack of strategies for handling recurring drifts. HistOS and HistUS yield better performance than Chebyshev-based methods, attributed to

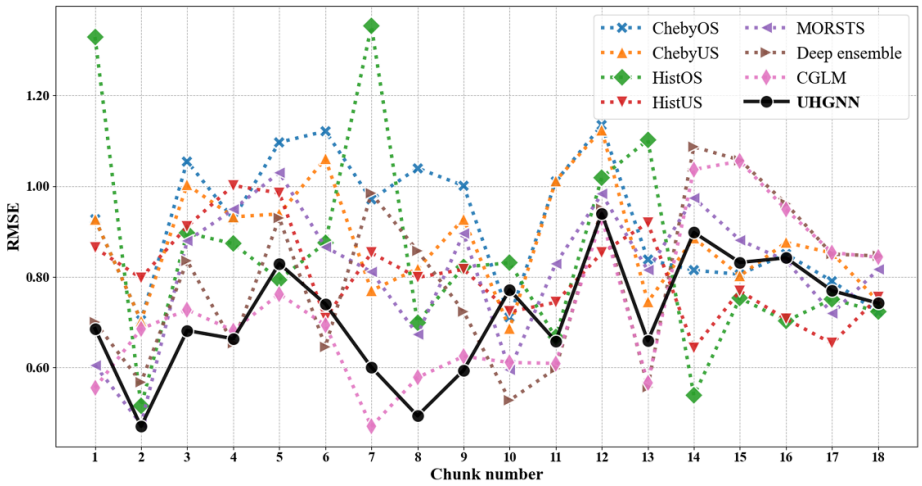


Figure 8. RMSE results of different methods across all chunks.

Table 4. Average RMSE of different methods for online prediction.

Method	Average RMSE
ChebyOS	0.9178 ± 0.0231
ChebyUS	0.8784 ± 0.0114
HistOS	0.8469 ± 0.0198
HistUS	0.8056 ± 0.0319
MORSTS	0.8129 ± 0.0221
Deep ensemble (no drift detection)	0.7963 ± 0.0156
Deep ensemble + ADWIN	0.7881 ± 0.0103
Deep ensemble + EUDD	0.7702 ± 0.0157
CGLM	0.7351 ± 0.0160
UHGNN	$\mathbf{0.7151} \pm 0.0121$

their capability of capturing data distributions via the online histogram approach. MORSTS and Deep ensemble have limited effectiveness because they do not incorporate drift detection. In contrast, deep ensembles combined with ADWIN and EUDD partially mitigate performance degradation, yet their inability to retain historical distributional information limits their effectiveness. CGLM delivers competitive results, as its graph learning and adaptation mechanisms effectively capture evolving correlations. However, CGLM's performance relies on the availability of real-time labels for drift detection and model updating, which may restrict its practical application in scenarios where ground-truth labels are delayed or costly to obtain.

In summary, the results indicate that UHGNN can effectively capture the spatial-temporal dependencies inherent in industrial processes and achieve effective drift detection through epistemic uncertainty, demonstrating its practical applicability in drifting data streams.

To further evaluate the predictive performance of UHGNN under drifting data streams, we compute the temporal evolution of its prediction errors. Figure 9 illustrates the point-wise error variation with respect to the arrival order of samples. It can be observed that when no fine-tuning is performed, the prediction errors of UHGNN accumulate significantly and exhibit multiple large fluctuations. This reflects the limited capability of the end-to-end model to adapt its structure in the presence of continuous distributional drift and operational disturbances, making it difficult to maintain stable long-term predictive performance. In contrast, once fine-tuning is triggered, UHGNN is able to substantially suppress error propagation and quickly return to a low-error region after distributional changes occur. This demonstrates that the EUDD mechanism, when combined with fine-tuning, can effectively enhance the drift-handling capability of model.

4.4. Ablation study

To evaluate the contributions of hierarchical architecture in UHGNN, we conducted an ablation study by replacing the hierarchical layers with a flat graph aggregation scheme, while all other network structures are kept identical. The results in figure 10 reveal a significant performance drop in the flat variant, validating the necessity of hierarchical modeling for complex industrial logic. Additionally, the superiority of the EUDD mechanism is evidenced by its consistent outperformance over the ADWIN-based trigger when integrated with a deep ensemble backbone (see section 4.3). This underscores that epistemic uncertainty provides a more sensitive and reliable signal for detecting concept drifts than traditional statistical methods in streaming environments.

4.5. Complexity analysis

To comprehensively evaluate the feasibility of UHGNN in real-world industrial scenarios, this section analyzes its computational complexity in both the offline training stage and the online prediction stage.

The offline training stage consists of two major computational components, namely temporal feature extraction and spatial feature extraction. In the temporal feature extraction stage, UHGNN first applies a 1D-CNN to extract temporal features from each node and then generates pairwise interaction weights at each time step through temporal encoding and an attention mechanism. These interaction weights are

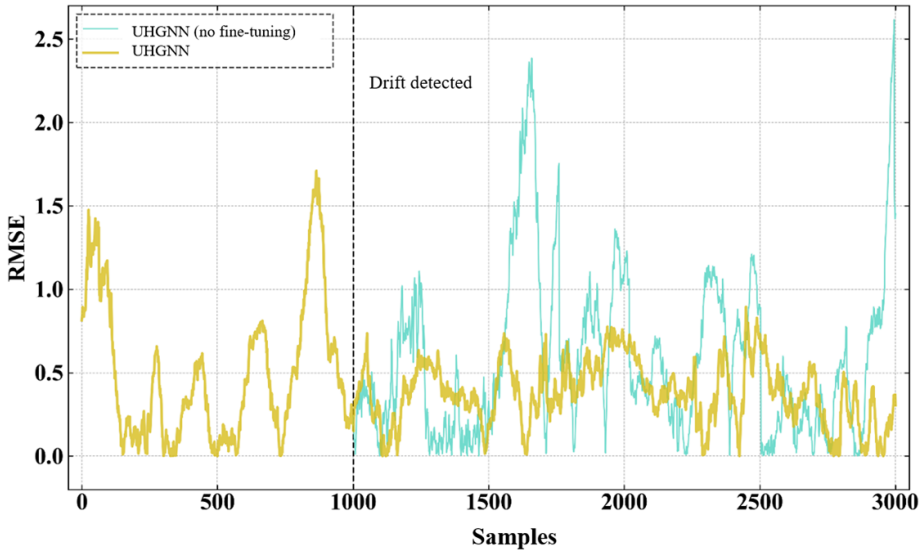


Figure 9. Performance of fine-tuned UHGNN on DHP dataset.

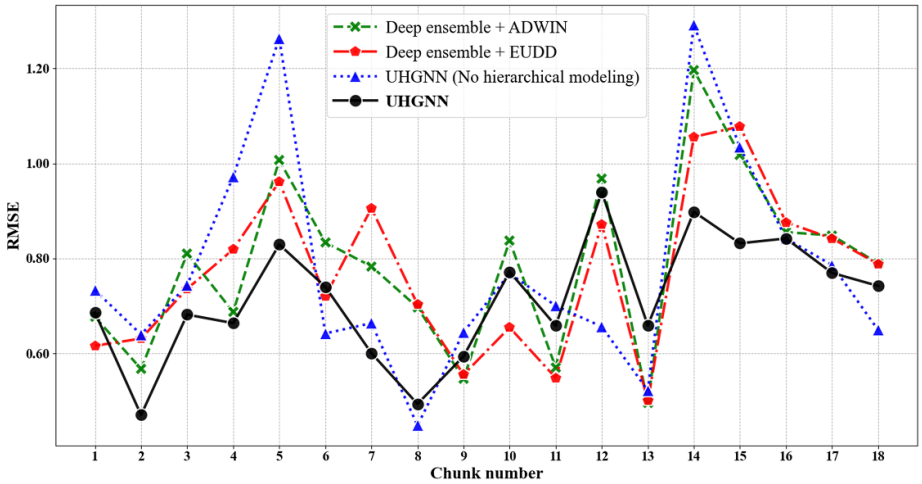


Figure 10. RMSE results under different ablation settings.

subsequently aggregated by an LSTM module to form the final weight matrix. The computational cost of this stage is dominated by the construction of pairwise attention among nodes and temporal aggregation, yielding a total time complexity of $O\left(B \cdot N^2 \cdot W\right)$, where B represents the training batch size, N represents the number of nodes, and W represents the length of time window. In the spatial feature extraction stage, UHGNN performs dense graph message passing through L layers of a GraphSAGE architecture, where each layer aggregates features from all node pairs followed by linear transformations. The time complexity of this component is $O\left(B \cdot L \cdot N^2 \cdot d_z\right)$, where d_z represents the dimensionality of node embedding vector. Then in the hidden graph aggregation stage, UHGNN applies max pooling and mean pooling within each cluster and performs cross-cluster information fusion through GAT and GraphSAGE, leading to a computational complexity of $O\left(B \cdot L \cdot M^2 \cdot d_z\right)$. By combining these components, the overall cost of UHGNN in spatial feature extraction is $O\left(B \cdot L \cdot \left(M^2 + N^2\right) \cdot d_z\right)$

In the online learning stage, the computation involves epistemic uncertainty quantification, concept drift detection, and model fine-tuning. In the epistemic uncertainty quantification stage, the model performs multiple forward passes with MCD, resulting in a complexity of $O\left(T \cdot B \cdot N^2 \cdot d_z\right)$, where T denotes the number of stochastic forward pass, implying that this cost scales linearly with the number of input samples. In the drift detection step, ADWIN processes the uncertainty associated with the data stream, and each update requires a mean-difference test over a dynamic window with computational complexity $O(\log W)$. Once a drift is detected, the model conducts fine-tuning based on samples arriving after the detection point, and obtains fine-tuning results in a complexity of $O\left(T_{\text{ft}} \cdot |D_{\text{ft}}| \cdot N^2 \cdot d_z\right)$, where $|D_{\text{ft}}|$ represents the number of fine-tuning samples, and T_{ft} represents the number of fine-tuning iterations.

In summary, during offline training phase, the overall computational complexity of UHGNN is dominated by the temporal feature extraction, message-passing in GraphSAGE,

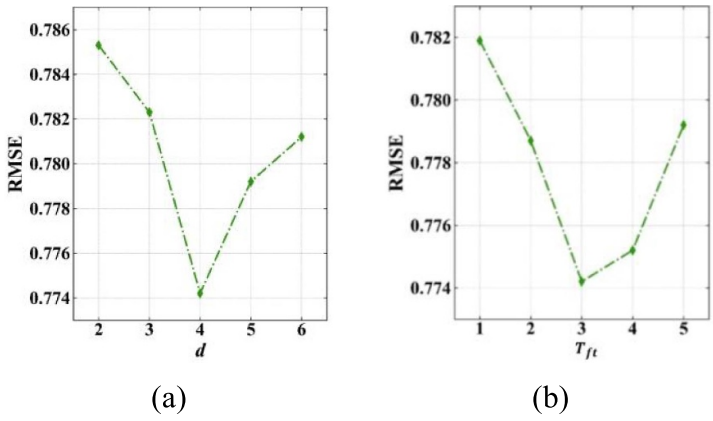


Figure 11. Performance of UHGNN under different hyperparameters. (a) d. (b) T_{ft}.

both are correlated with a complexity level of $O\left(N^2\right)$. In practice, since the value of N in industrial monitoring systems is typically small and fixed, this computational cost remains controllable in real engineering deployments. During the online phase, the overhead associated with epistemic uncertainty quantification and drift detection grows linearly with the number of evaluated streaming samples, while the fine-tuning strategy provides a flexible mechanism to balance predictive accuracy and computational latency. Overall, UHGNN achieves strong predictive performance and robust adaptability to concept drift, while keeping both computational and memory requirements within acceptable limits, making it suitable for long-term industrial process monitoring and prediction applications.

4.6. Sensitivity analysis

To further evaluate the robustness of UHGNN, this section conducts sensitivity experiments on two core parameters, namely the dimensionality of the time embedding vector d and the number of fine-tuning iterations T_{ft}. The results are presented in figure 11. It can be observed that the overall prediction performance of the model exhibits only slight fluctuations as these parameters vary, indicating that UHGNN possesses strong stability and generalization capability in terms of parameter configuration.

Specifically, when the value of d increases from 2 to 6, the variation in RMSE remains small, showing a trend of decreasing first and then slightly increasing again. The best performance is achieved when $d = 4$. This suggests that a moderate value of d effectively captures the temporal dynamics in industrial processes, whereas an excessively low dimension may lead to insufficient temporal feature representation, and an overly high dimension may introduce redundant representations and amplify training perturbations. Additionally, with respect to the value of T_{ft}, as the number increases from 1 to 5, RMSE similarly shows a pattern of decreasing first and then rising again, with the best performance obtained when $T_{\mathrm{ft}} = 3$. This result indicates that an appropriate number of update iterations helps the model quickly adapt to new data patterns after drift, whereas too few updates fail to sufficiently correct model parameters, and too many updates may introduce short-term noise, causing the model to deviate from its previously stable structure. Therefore, properly controlling the frequency and magnitude of fine-tuning is crucial for maintaining model stability in online learning scenarios.

Overall, the sensitivity results demonstrate that UHGNN is not highly sensitive to changes in these key parameters, and the performance variation around the optimal settings remains small. This confirms that the model features strong robustness and high engineering applicability in practical deployment.

5. Conclusion

This study proposes an uncertainty-aware hierarchical GNN for reliable online drifting data stream prediction. The method explicitly models both temporal dependencies and spatial structural relationships through the hierarchical GNN architecture and captures epistemic uncertainty to facilitate concept drift detection, enabling adaptive model fine-tuning in response to drifting data streams. Experiments on the real-world DHP dataset demonstrate that the proposed method effectively mitigates performance degradation caused by concept drift, thereby improving prediction accuracy and reliability in complex industrial data-stream environments. This provides methodological foundations and theoretical support for industrial process monitoring.

For future work, we plan to further investigate the following direction: the current graph partitioning and node-relationship construction rely on domain knowledge and predefined rules. Future research may explore learnable adjacency matrices, spatial-temporal adaptive graph generation, and strategies that integrate structural priors with data-driven mechanisms, enabling more flexible and self-consistent graph-structure construction.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 72471011.

ORCID iD

Yan-Hui Lin 0000-0001-6248-7537

References

[1] Yin S, Rodriguez-Andina J J and Jiang Y 2019 Real-time monitoring and control of industrial cyberphysical systems: with integrated plant-wide monitoring and control framework *IEEE Ind. Electron. Mag.* **13** 38–47

[2] Yang L and Shami A 2023 A multi-stage automated online network data stream analytics framework for IIoT systems *IEEE Trans. Ind. Inform.* **19** 2107–16

[3] Qiao H, Novikov B and Blech J O 2022 Concept drift analysis by dynamic residual projection for effectively detecting botnet cyber-attacks in IoT scenarios *IEEE Trans. Ind. Inform.* **18** 3692–701

[4] Lin Y-H and Chang L 2022 An online transfer learning framework for time-varying distribution data prediction *IEEE Trans. Ind. Electron.* **69** 6278–87

[5] Peng X, Duan S, Sankavaram C and Jin X 2024 Unsupervised adaptive fleet battery pack fault detection with concept drift under evolving environment *IEEE Trans. Autom. Sci. Eng.* **21** 2276–88

[6] Jastrzebska A, Hernández A M, Nápoles G, Salgueiro Y and Vanhoof K 2022 Measuring wind turbine health using fuzzy-concept-based drifting models *Renew. Energy* **190** 730–40

[7] Károly A I, Galambos P, Kuti J and Rudas I J 2021 Deep learning in robotics: survey on model structures and training strategies *IEEE Trans. Syst. Man Cybern. Syst.* **51** 266–79

[8] Lu J, Liu A, Dong F, Gu F, Gama J and Zhang G 2019 Learning under concept drift: a review *IEEE Trans. Knowl. Data Eng.* **31** 2346–63

[9] Arora S, Rani R and Saxena N 2024 A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **14** e1536

[10] Xu S and Wang J 2017 Dynamic extreme learning machine for data stream classification *Neurocomputing* **238** 433–49

[11] Gomes H M *et al* 2017 Adaptive random forests for evolving data stream classification *Mach. Learn.* **106** 1469–95

[12] Losing V, Hammer B and Wersing H 2016 KNN classifier with self adjusting memory for heterogeneous concept drift *Proc. 16th Int. Conf. Data Mining* pp 291–300

[13] Zhou L and Wang H 2024 MST-GAT: a multi-perspective spatial-temporal graph attention network for multi-sensor equipment remaining useful life prediction *Inf. Fusion* **110** 102462

[14] Wu Z, Pan S, Chen F, Long G, Zhang C and Yu P S 2021 A comprehensive survey on graph neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **32** 4–24

[15] Kipf T N and Welling M 2017 Semi-supervised classification with graph convolutional networks *Proc. Int. Conf. Learning Representations*

[16] Velickovic P *et al* 2017 Graph attention networks *Stat* **1050** 10–48550

[17] Hamilton W L, Ying R and Leskovec J 2017 Inductive representation learning on large graphs *Proc. 31st Int. Conf. Advances in Neural Information Processing Systems* pp 1025–35

[18] Wang J, Song G, Wu Y and Wang L 2020 Streaming graph neural networks via continual learning *Proc. 29th Int. Conf. Information and Knowledge Management* pp 1515–24

[19] Chen C *et al* 2023 NeutronStream: a dynamic GNN training framework with sliding window for graph streams *Proc. VLDB Endow.* **17** 455–68

[20] Yu H, Wen J, Sun Y, Wei X and Lu J 2025 CA-GNN: a competence-aware graph neural network for semi-supervised learning on streaming data *IEEE Trans. Cybern.* **55** 684–97

[21] Zhou M, Lu J, Song Y and Zhang G 2023 Multi-stream concept drift self-adaptation using graph neural network *IEEE Trans. Knowl. Data Eng.* **35** 12828–41

[22] Zhou M, Lu J, Lu P and Zhang G 2024 Dynamic graph regularization for multi-stream concept drift self-adaptation *IEEE Trans. Knowl. Data Eng.* **36** 6016–28

[23] Zhou M and Lu J 2025 Continuous graph learning-based self-adaptation for multi-stream concept drift *IEEE Trans. Cybern.* **55** 3760–73

[24] Zeghina A, Leborgne A, Ber F L and Vacavant A 2024 Deep learning on spatiotemporal graphs: a systematic review, methodological landscape, and research opportunities *Neurocomputing* **594** 127861

[25] Zhong Z, Li C-T and Pang J 2023 Hierarchical message-passing graph neural networks *Data Min. Knowl. Discov.* **37** 381–408

[26] Zhang X, Wang T, Ma L and Mahadevan S 2025 Reliability engineering, risk management, and trustworthiness assurance for AI systems *J. Reliab. Sci. Eng.* **1** 022001

[27] Lin Y-H and Li G-H 2022 A Bayesian deep learning framework for RUL prediction incorporating uncertainty quantification and calibration *IEEE Trans. Ind. Inform.* **18** 7274–84

[28] Lin Y-H and Qi L 2025 Uncertainty-driven online deep ensemble for imbalanced drifting data stream regression *IEEE Trans. Ind. Inform.* **22** 49–59

[29] Ying R, You J, Morris C, Ren X, Hamilton W L and Leskovec J 2018 Hierarchical graph representation learning with differentiable pooling *Proc. 32nd Int. Conf. on Neural Information Processing Systems* pp 4805–15

[30] Yang L, Tsung F, Wang K and Zhou J 2024 Wind power forecasting based on a spatial–temporal graph convolution network with limited engineering knowledge *IEEE Trans. Instrum. Meas.* **73** 1–13

[31] Chen D *et al* 2024 Bayesian hierarchical graph neural networks with uncertainty feedback for trustworthy fault diagnosis of industrial processes *IEEE Trans. Neural Netw. Learn. Syst.* **35** 18635–48

[32] Gal Y and Ghahramani Z 2016 Dropout as a bayesian approximation: representing model uncertainty in deep learning *Proc. Int. Conf. Machine Learning* pp 1050–9

[33] Ahmadian R, Ghatee M and Wahlström J 2024 Improved user identification through calibrated monte-carlo dropout *Knowl.-Based Syst.* **305** 112581

[34] Li D, Chen J, Huang R, Chen Z and Li W 2024 Sensor-aware CapsNet: towards trustworthy multisensory fusion for remaining useful life prediction *J. Manuf. Syst.* **72** 26–37

[35] Depeweg S, Hernández-Lobato J M, Doshi-Velez F and Udluft S 2017 Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning *Proc. Int. Conf. Machine Learning*

[36] Bifet A and Gavaldà R 2007 Learning from time-changing data with adaptive windowing *SDM*

[37] Aminian E, Ribeiro R P and Gama J 2021 Chebyshev approaches for imbalanced data streams regression models *Data Min. Knowl. Discov.* **35** 2389–466

[38] Peng T, Sellami S, Boucelma O and Chbeir R 2023 Multi-output regression for imbalanced data stream *Expert Syst.* **40** e13417

[39] Aminian E, Ribeiro R P and Gama J 2025 Histogram approaches for imbalanced data streams regression *Mach. Learn.* **114** 274
[40] Lakshminarayanan B, Pritzel A and Blundell C 2017 Simple and scalable predictive uncertainty estimation using deep ensembles *Proc. 31st Int. Conf. on Neural Information Processing Systems* pp 6405–16

Peng-Cheng Yan received the Bachelor's degree in quality and reliability engineering from Beihang University, Beijing, China, in 2022.

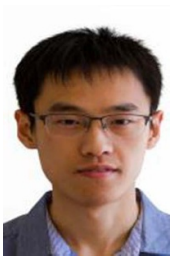
He is currently pursuing the PhD degree at the School of Reliability and Systems Engineering, Beihang University, Beijing, China. His research interests include uncertainty analysis and its applications in engineering systems.

Lin Qi received the bachelor's degree in 2023 from Northeast Forestry University, Harbin, China.

He is currently pursuing the master's degree with the School of Reliability and Systems Engineering, Beihang University, Beijing, China. His research interests include deep learning for industrial process monitoring.

Enrico Zio received the PhD degree from the Politecnico di Milano, Milan, Italy, in 1996, and the PhD degree in probabilistic risk assessment from MIT, MA, USA, in 1998.

He is currently a Full Professor with the Centre for Research on Risk and Crises (CRC), MINES Paris-PSL University, Paris, France; a Full Professor and the President of the Alumni Association at the Politecnico di Milano; a Distinguished Guest Professor at Tsinghua University, Beijing, China; an Adjunct Professor with the City University of Hong Kong, Hong Kong, Beihang University, Beijing, and Wuhan University, Wuhan, China; and the Co-Director of the Center for REliability and Safety of Critical Infrastructures (CRESCI) and the Sino-French Laboratory of Risk Science and Engineering (RISE), Beihang University. He is the author or coauthor of seven books and more than 300 articles in international journals. His research focuses on the modeling of the failure repair maintenance behavior of components and complex systems; the analysis of their reliability, maintainability, prognostics, safety, vulnerability, resilience, and security characteristics; and the development and use of Monte Carlo simulation methods, artificial techniques, and optimization heuristics.



Yan-Hui Lin received the PhD degree in industrial science and technology from the Université Paris-Saclay, Paris, France, in 2016.

He was with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, as a Senior Research Associate. He is currently an Associate Professor of System Engineering with the School of Reliability and Systems Engineering, Beihang University, Beijing, China. His research interests include degradation modeling, reliability assessment, and prognostic and health management (PHM).