Expert Systems With Applications 307 (2026) 130992

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Learning-based network diagnostics: Handling high fault densities with PMC/MM* model

Wenfei Liu [a,b], Jiafei Liu [a,b,c,*], Jingli Wu [a,b,c], Chia-Wei Lee [d], Dajin Wang [e], Gaoshi Li [a,b,c]

[a] Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education, Guangxi Normal University, GuiLin, 541004, Guangxi, China
[b] Guangxi Key Lab of Multi-Source Information Mining and Security, Guangxi Normal University, GuiLin, 541004, Guangxi, China
[c] Center for Applied Mathematics of Guangxi, Guangxi Normal University, GuiLin, 541004, Guangxi, China
[d] Department of Computer Science, University of Taipei, 100234, Taiwan, China
[e] Department of Computing, Montclair State University, Upper Montclair, 07043, USA

ARTICLE INFO

Keywords:
Interconnection network
Machine learning
Logistic regression
Fault diagnosis
PMC
MM*

ABSTRACT

With the rapid advancement of big data and cloud computing technologies, higher demands are placed on the stability and reliability of underlying interconnection networks. To address the diagnostic challenges in complex network topologies and high-fault-density scenarios, this paper proposes an intelligent diagnostic framework leveraging logistic regression. Building upon complex network theory, we first develop generation algorithms for hypercube, star graph, and BCube network topologies. A comprehensive multidimensional feature dataset comprising 4.6 million samples is constructed based on the PMC and MM* models. Through feature engineering, three key categories of indicators–structural features, proportional features, and differential features–are extracted to train a logistic regression classifier for the binary classification of faulty nodes. Experimental results demonstrate that the model maintains a validation accuracy exceeding 97.5% and a false positive rate below 3.3% across networks of varying scales, including star graphs S_n ($n = 6, 7, 8, 9$), hypercubes Q_n ($n = 14, 15, 16, 17$), and $BCube(n, k)$ networks where ($n = 7, 8, 9, 10$, $k = 3, 4$), even when faulty node proportions reach 50%. The model exhibits particularly strong adaptability to BCube networks commonly used in data centers. Feature importance analysis further reveals that PMC-related features significantly contribute to classification performance. This study provides an efficient method for real-time fault localization in large-scale networks and validates the potential of machine learning models in complex network diagnostics.

1. Introduction

Modern large-scale interconnection networks require ultra-high reliability. These systems operate in critical applications spanning cloud computing and aerospace control domains. They maintain continuous operation without service interruptions. Typical implementations include data center and parallel network architectures. Traditional fault diagnosis models, including the well-established PMC (Preparata-Metze-Chien) and MM* (Maeng-Malek) frameworks, rely on deterministic threshold-based rules derived from graph-theoretic principles. While these methods excel in small-scale static topologies, they suffer from two fundamental limitations: (1) inflexibility in dynamic networks, where node/link failures and topology reconfigurations occur frequently, and (2) in dense fault scenarios (e.g., $\geq 10\%$ faulty nodes), the fixed threshold fails to adapt to varying fault distributions, resulting in a high misdiagnosis rate.

The latest advancements in fault diagnosis have introduced data-driven machine learning and deep learning methods, but most studies prioritize complex models such as deep neural networks (DNNS) rather than interpretability and computational efficiency. For instance, Ding et al. (2010) discussed the application of principal component analysis techniques in fault detection and diagnosis, but noted that the computational complexity is relatively high, making it potentially more suitable for non-real-time fault diagnosis. Lahiri et al. (2022) proposed a fault diagnosis method based on decision trees and random forest classifiers, but it is only applicable to the power network based on specific transmission linear models and fault types. Wang et al. (2023) proposed a mechanical fault diagnosis method based on multi-layer wavelet attention convolutional neural Network (MWA-CNN). However, due to the complexity of the model structure, the training process of MWA-CNN requires more computing resources and time , and has higher requirements for hardware. Lin et al. (2023) employed neural networks

* Corresponding author.
E-mail address: liujiafei@gxnu.edu.cn (J. Liu).

https://doi.org/10.1016/j.eswa.2025.130992
Received 11 September 2025; Received in revised form 29 November 2025; Accepted 23 December 2025
Available online 30 December 2025

to achieve 91% accuracy but specifically focus on sparse fault scenarios within small-scale networks. Furthermore, the lack of standardized datasets for interconnection network fault diagnosis forces researchers to rely on conventional methods. This impedes sustainable progress, fair comparison, and reproducibility within the field. Wu et al. (2025) investigated the challenges faced by traditional system-level fault diagnosis models due to the regularity requirements of graph structures and the limitations on the number of diagnosable nodes. They proposed a novel fault diagnosis algorithm called TGSEGNN, which combines the test graph with symptom-enhanced graph neural networks under the PMC model. Many methods have also emerged in other areas of fault diagnosis. Huang et al. (2025) investigated the issue of intermittent fault diagnosis in symbiotic IoT caused by limited device resources and environmental heterogeneity. They proposed an IFDGAT-LSTM diagnostic framework that combines long short-term memory networks with graph attention networks. In other fields of fault diagnosis, Chen et al. (2025) investigated the challenges of dynamic distribution adaptation in bearing fault diagnosis under time-varying speeds and proposed a novel sliding mixed graph contrastive domain adaptation method (Mix-GraphCDA). Ngo et al. (2025) proposed a method based on a deep graph attention network to overcome the low accuracy in fault detection of distribution networks, insufffcient coverage of fault types, and the challenges posed by the integration of renewable energy sources.

To bridge these gaps, this paper proposes a lightweight and interpretable probabilistic diagnosis framework that synergizes classical PMC and MM* diagnostic models with logistic regression (LR). In contrast to existing methods that rely on computationally intensive deep learning architectures–such as DNNs, MWA-CNNs, or graph neural networks–our approach emphasizes model simplicity, real-time capability, and interpretability, without sacrificing diagnostic accuracy. Specifically, we extract two core diagnostic features–sum_PMC (aggregated neighbor test results under PMC) and sum_MM* (global test counts under MM*)–complemented by graph-theoretic metrics (e.g., node degree) to capture topological context. A key contribution is the construction of a publicly available, large-scale synthetic dataset spanning diverse topologies (hypercubes, BCube, star graphs) and a wide range of fault densities (10%-50%), comprising 4.6 million samples. This dataset enables reproducible and comparable research, addressing a critical gap in the field. By integrating classical diagnostic results as input features, we train a logistic regression classifier to perform efficient and accurate binary fault identification, even under high fault density scenarios where traditional and complex data-driven methods often struggle.

The key contributions of this work can be summarized as below:

(1) **Methodological Innovation**: An optimized fault diagnosis strategy is proposed to construct an intelligent diagnostic framework that integrates simulation data-driven methods, feature engineering enhancement, and logistic regression modeling. This framework can effectively identify and classify intermittent, local, and global fault types in processors, thereby providing technical support for fault management in data centers.
(2) **Dataset Contribution**: A public open dataset that integrates the architectures of hypercube, star network, and BCube network has been constructed. This dataset effectively combines PMC/MM* test results with graph-theoretic metrics, with a dataset size reaching 4,600,000 entries. The establishment of this dataset will provide convenience for subsequent research in the field of network fault diagnosis.
(3) **Practical Validation**: Finally, we systematically verify the fault diagnosis model under 16 topological structures and fault densities ranging from 10% to 50% respectively, and conduct a detailed analysis of the performance of the model. The experimental results show that this model can maintain a high validation accuracy rate and a low misdiagnosis rate under different topological structures and fault densities, and has strong adaptability on the BCube network.

The remainder of this paper is organized as follows. Section 2 reviews related work and preliminaries. Section 3 details the dataset generation by feature engineering. Section 4 elaborates on the methodology and presents comparative analyses to validate the approach. Section 5 concludes with future directions.

2. Related works and preliminaries

2.1. Related works

In recent years, the emerging technology industries such as big data and cloud computing have developed rapidly. With the fast growth of these industries, more stringent requirements for the stability and reliability of the underlying network infrastructure have been put forward. In this context, fault diagnosis of interconnection network has become crucial. It is a key technical link to ensure the efficient operation of the network. As a result, it has attracted extensive attention. It has attracted interest from both academia and industry. Many scholars have carried out research in this field and successively proposed a series of innovative and practical fault diagnosis strategies.

The PMC and MM* models established the cornerstone of interconnection network diagnosis through node-test rules (Dahbura & Masson, 1984; Sengupta & Dahbura, 1989). Wang et al. (2024) conducted research on the g-good neighbor diagnosability and g-extra diagnosability of networks under both the PMC model and the MM model, and applied these findings to enhanced hypercube networks. Wang et al. (2025a) established the g-good-neighbor diagnosability of multiprocessor systems, introducing an enhanced Comparative Model (MC Model) and establishing its applicability in hypercube networks. Lin et al. (2025) studied local fault diagnosis analysis for block-pattern-based regularly diagnosable networks, proposing a novel x-block local fault diagnosis strategy and applying it to 16 well-known network topologies. Chang and Hsieh (2025) investigated the conditional diagnosability of enhanced hypercubes under the PMC model. They demonstrated that the conditional diagnosability of an (n, k)-enhanced hypercube, is $4n-7$ (when $n \geq 5$ and $k = 3$) and $4n-3$ (when $n \geq 6$ and $5 \leq k \leq n$). Zhuo et al. (2025) investigated the non-inclusive diagnosability of folded hypercube-like networks (FH_n) and determined their diagnosability under the PMC and MM* models as $2n$, $2n-1$, and $2n-2$, respectively. Wan et al. (2024) explored component-level diagnostic strategies for star graph, deriving their 2-component and 3-component diagnosability under the PMC model. Yang et al. (2025) investigated the (t,k)-diagnosability of Cayley graphs generated by 2-trees under the PMC model and MM* model, and established their diagnosability in both frameworks. Qin et al. (2024) analyzed the diagnosability of multi-graph composite networks (MGCNs) and determined their diagnosability under the PMC model and MM* model for two distinct MGCN architectures. Huang et al. (2024b) analyzed the connectivity and h-extra diagnosability of complete josephus cubes under the h-extra fault-tolerant framework. Sun et al. (2023) established the relationship models between the component connectivity and component diagnosability of several types of rule networks.

Zhang et al. (2024) investigated probabilistic method-based local fault diagnosis in large-scale multiprocessor systems and proposed two diagnostic algorithms applicable to hybrid architectures. Huang et al. (2024a) designed an adaptive t/s diagnostic algorithm, named $APDMM*t/s$, tailored for system-level fault diagnosis in bijection-connected networks. Jiang et al. (2024) developed a unified data processing framework to rigorously assess the efficacy of deep learning (DL) models in fault diagnosis and reproduced eight cutting-edge DL architectures. Lin et al. (2024) proposed Ftk-DIAG-MM*, an adaptive system-level fault self-diagnosis strategy tailored for folded hypercubes and rigorously analyzed its t/k-diagnosability under the MM* model. Lv et al.

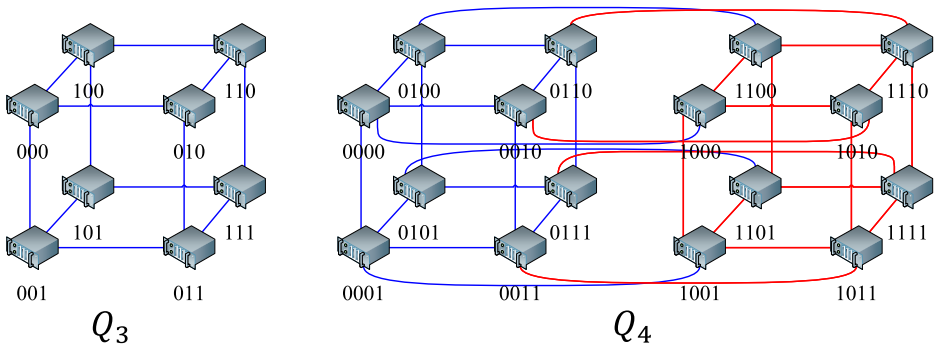

Fig. 1. The 3-dimensional hypercube Q_3 and 4-dimensional hypercube Q_4.

(2025) proposed a novel Partial Block Fault model and designing a fault diagnosis algorithm tailored specifically for the k-ary n-cube network.

2.2. Preliminaries

The reliability of modern computing systems–from cloud data centers to embedded multi-processor architectures–hinges on the robustness of their topologies of underlying interconnection networks. Next, we first introduce three representative topologies–hypercubes, star graphs, and BCube–whose structural properties profoundly influence diagnostic feasibility.

Definition 1. (Saad and Schultz, 1988) An n-dimensional hypercube Q_n is a binary interconnection network defined as:

(1) **Nodes**: The complete set of all possible n-bit binary strings comprises a total of 2^n distinct nodes.
(2) **Edges**: Two nodes are connected if their binary representations differ in exactly one bit position.

This gives each node degree n and total edge count $n \cdot 2^{n-1}$. See Fig. 1 for Q_3 (8 nodes) and Q_4 (16 nodes) examples.

Definition 2. (Akers and Krishnamurthy, 1989) An n-dimensional star graph S_n is a permutation-based interconnection network defined as follows:

(1) **Nodes**: All permutations of the symbol sequence $\langle n \rangle = \{1, 2, \ldots, n\}$, form $n!$ distinct nodes.
(2) **Edges**: Two permutations are connected if they differ by swapping, the first symbol with exactly one other symbol at position i $(2 \le i \le n)$

Thus each node has degree $n-1$, and the total edge count is $\frac{n!\cdot(n-1)}{2}$. See Fig. 2 for S_3 (6 nodes) and S_4 (24 nodes).

Definition 3. (Guo et al., 2009) The $BCube_{n,k}$ graph is defined as follows:

(1) **Nodes**: Each node is represented by a numeric string of length $k+1$, where each digit x_i (for $0 \le i \le k$) ranges from 0 to $n-1$. This structure yields n^{k+1} distinct nodes.
(2) **Edges**: Two nodes are connected by an edge if and only if their corresponding numeric strings differ in exactly one digit position. Formally, nodes $x = x_k x_{k-1} \ldots x_0$ and $y = y_k y_{k-1} \ldots y_0$ are adjacent if: (1) There exists a unique position j $(0 \le j \le k)$ where $x_j \ne y_j$; (2) All other positions $i \ne j$ satisfy $x_i = y_i$.

Each node has degree of $(k+1)(n-1)$ and a total edge count of $\frac{n^{k+1}\cdot(k+1)(n-1)}{2}$. The $BCube_{n,k}$ structure is recursively constructed by combining n copies of $BCube_{n,k-1}$ with additional switches, forming a hierarchical topology (Guo et al., 2009). See Fig. 3 for $BCube_{4,1}$.

The selection of network topologies directly impacts the design and performance of diagnostic models. To establish a foundation, we next introduce the two classical diagnostic frameworks–PMC and MM*–the underpin of traditional fault localization in these topologies.

Definition 4. (Preparata et al., 1967) (PMC Model) Given a processor network $G = (V, E)$, the PMC model is defined as:

(1) Each processor $u \in V$ can test its neighbors $v \in N(u)$;
(2) Test outcome $\sigma(u, v)$ satisfies:

$$\sigma(u,v) = \begin{cases} 0 & \text{if } u \text{ is FF and } v \text{ is FF} \\ 1 & \text{if } u \text{ is FF and } v \text{ is F} \\ 0/1 & \text{if } u \text{ is F (regardless of } u, v\text{'s status)} \end{cases}$$

where FF is fault-free node, F is faulty node. The working mechanism of the PMC model is shown in Fig. 4.

Definition 5. (Ahlswede and Aydinian, 2008; Malek, 1980) (MM* Model) Given a processor network $G = (V, E)$, the MM* model is defined as:

(1) Each processor $w \in V$ can test each pair of its neighbors $\{u, v\} \subseteq N(w)$
(2) Test outcome $\sigma_w(u, v)$ satisfies:

$$\sigma_w(u,v) = \begin{cases} 0 & \text{if } w \text{ is FF and } u, v \text{ bot h FF} \\ 1 & \text{if } w \text{ is FF and } (u \text{ is F or } v \text{ is F}) \\ 0/1 & \text{if } w \text{ is F (regardless of } u, v\text{'s status)} \end{cases}$$

The working mechanism of the MM* model is shown in Fig. 5.

Within the traditional framework of fault diagnosis strategies, the PMC and MM* models typically serve as the core foundational rules for deterministic fault identification. However, in our research, the significance of these two models extends far beyond this role–they not only constitute the fundamental guidelines for deterministic fault detection but also serve as pivotal input elements for data-driven diagnostic methodologies. We now transition to logistic regression–a probabilistic ML model that leverages these metrics to achieve real-time fault localization.

Definition 6. (Cox, 1959) (Logistic Regression)

(1) Model Principle: A generalized linear model that maps linear combinations to probability space (0–1) via the sigmoid function, suitable for binary classification.
(2) Loss Function: Cross-entropy loss quantifies the discrepancy between predicted probabilities and true labels.
(3) Optimization: Gradient descent or its variants (e.g., stochastic gradient descent) minimize the loss function.

Definition 7 (Model Evaluation Metrics)**.** Let TP denote the count of genuinely faulty nodes accurately identified as faulty, while TN represents the count of truly fault-free nodes correctly diagnosed as fault-free. Similarly, FP stands for the number of actually fault-free nodes mistakenly diagnosed as faulty, and FN signifies the number of genuinely faulty nodes incorrectly identified as fault-free.

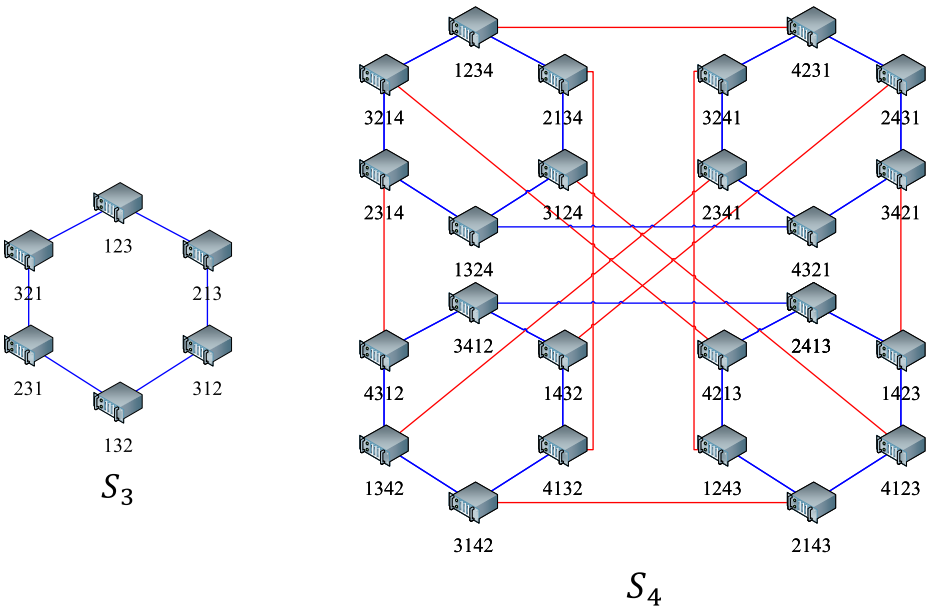

Fig. 2. The 3-dimensional star graph S_3 and 4-dimensional star graph S_4.

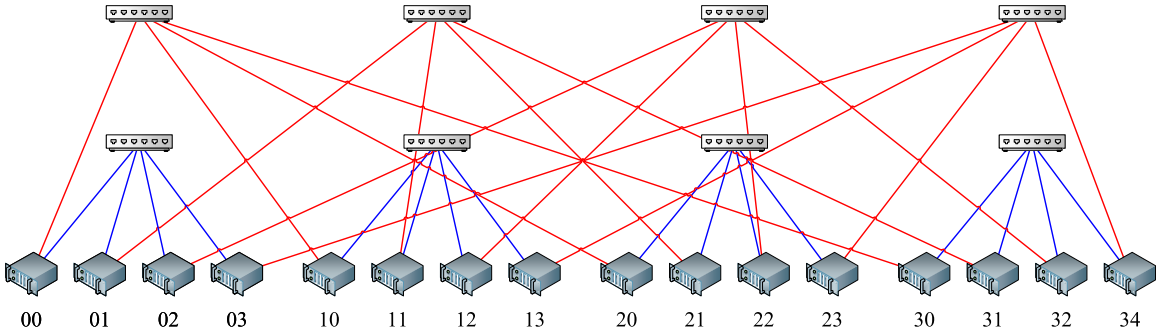

Fig. 3. The BCube with $n = 4, k = 1$.

(1) Accuracy: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
(2) Precision: Precision $= \frac{TP}{TP+FP}$
(3) Recall: Recall $= \frac{TP}{TP+FN}$
(4) F1-Score: F1-Score $= \frac{2*Precision*Recall}{Precision+Recall}$
(5) AUC: AUC $= \int_0^1 TPR(\tau) \cdot \frac{\partial FPR^{-1}(\tau)}{\partial \tau} d\tau$
(6) G-Mean: G-Mean $= \sqrt{Precision * Recall}$

3. Feature engineering construction

This section aims to build a systematic and comprehensive network fault diagnosis framework, which includes the following steps. First, based on complex network theory, different network topology generation algorithms are proposed to provide a foundation for network structure in subsequent research. Second, the simulation algorithms for the PMC model and MM* model under random fault propagation are explained in detail and a dataset is constructed based on topology features and simulation test results. Finally, a network fault diagnosis method based on logistic regression is proposed by combining feature engineering methods with fault diagnosis techniques.

3.1. Network topology

In this subsection, we design several network structure generation algorithms based on the definition of network structure, enabling its representation in a computer.

Algorithm 1: Generate hypercube graph Q_n.

Input: n: dimension
Output: Q_n: hypercube with 2^n nodes

```
nodes ← All binary strings of length n
G ← An empty undirected graph
G.addNodes(nodes)
for i ← 0 to |nodes| − 1 do
    for j ← i + 1 to |nodes| − 1 do
        u ← nodes[i]
        v ← nodes[j]
        if HammingDistance(u, v) == 1 then
            G.addEdge(u, v)
Q_n ← G
return Q_n
```

Algorithm 1 generates a hypercube graph Q_n with 2^n nodes. The steps are as follows:

(1) Node Generation(Step 1): Generate all n-length binary strings V, e.g., $\{000, 001, 010, \ldots, 111\}$ for $n = 3$.
(2) Graph Initialization(Steps 2–3): Create an empty undirected graph G and add all binary strings as vertices.

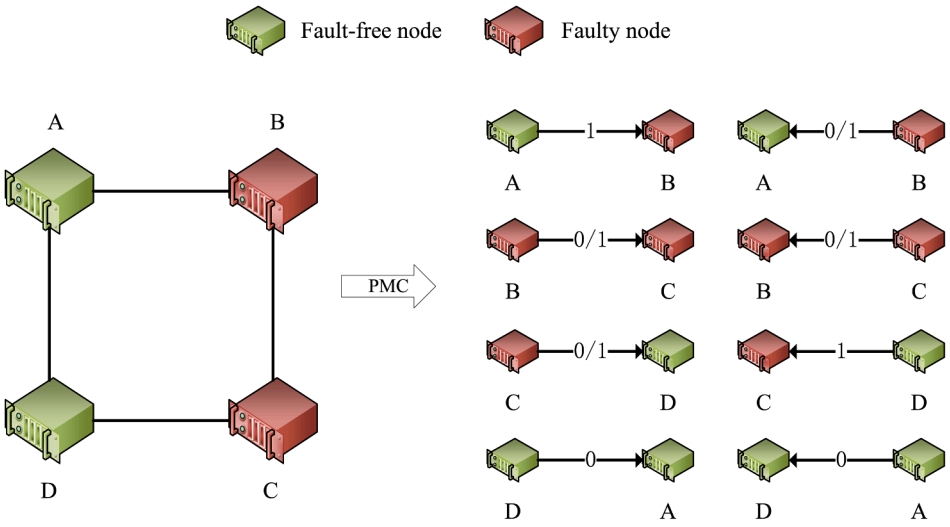

Fig. 4. The working mechanism of the PMC model.

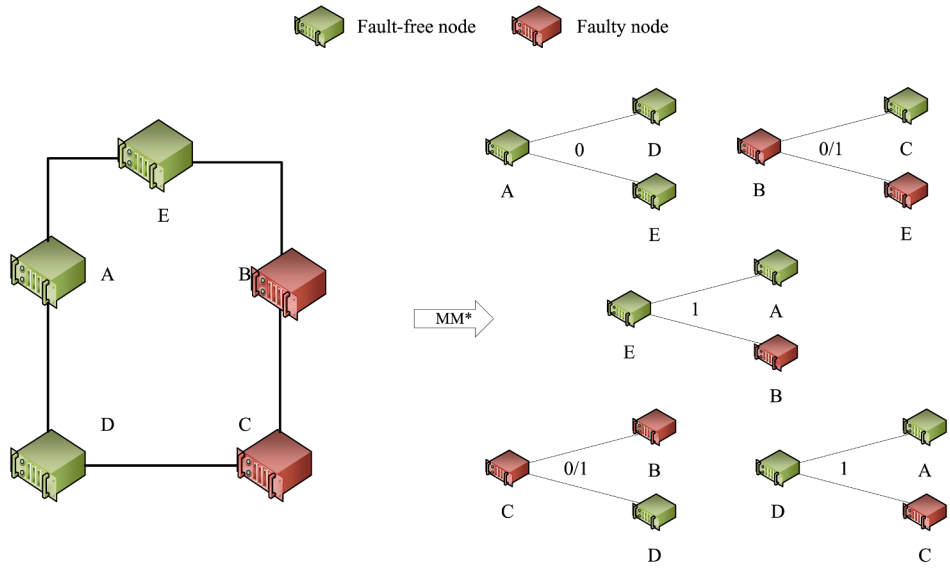

Fig. 5. The working mechanism of the MM* model.

Algorithm 2: Generate star graph S_n.

Input: n: dimension
Output: S_n: star graph

```
numbers ← [1, 2, ..., n]
perms ← All permutations of numbers
nodes ← [Each permutation in perms]
G ← An empty undirected graph
G.addNodesFrom(nodes)
for node ∈ nodes do
    for i ∈ [1, length(perm)] do
        str1 ← List representation of node
        Swap first character with i^th character in str1
        G.addEdge(node, str1)
S_n ← G
return S_n
```

(3) Edge Connection(Steps 4–9): Iterate over all node pairs (u, v) and compute their Hamming distance. If the Hamming distance equals 1 (i.e., only one bit differs), add an edge (u, v) to the graph.
(4) Return Graph(Steps 10–11): The final graph Q_n.

Algorithm 2 constructs a star graph S_n where nodes represent permutations of n elements, and edges are defined by single-element swaps. The steps are as follows:

(1) Node Generation(Steps 1–3): Generate all permutations of the list $[1, 2, \ldots, n]$.
(2) Graph Initialization(Steps 4–5): Create an empty undirected graph and add all permutations as nodes.
(3) Edge Connection(Steps 6–8): For each node, iterate through all character positions. Swap the first character with the i-th character ($i \in [1, n]$) to generate a new permutation. If the new permutation differs from the original, add an undirected edge between them.
(4) Return Graph(Steps 9–10): The resulting graph S_n contains nodes and edges.

Algorithm 3: Generate BCube graph $BCube(n, k)$.

Input: n, k
Output: $BCube(n, k)$: BCube graph

1 $G \leftarrow$ Empty undirected graph
2 $N \leftarrow n^{k+1}$ /*Total number of nodes*/
3 **for** $i \leftarrow 0$ ***to*** $N - 1$ **do**
4 $\quad d \leftarrow \lfloor i/n^k \rfloor$ /*Compute node dimension*/
5 $\quad G$.addNode(S_i, dimension $= d$)
6 **for** $level \leftarrow 0$ ***to*** k **do**
7 $\quad step \leftarrow n^{level}$
8 $\quad$ **for** $i \leftarrow 0$ ***to*** $N - 1$ **do**
9 $\quad\quad$ **for** $j \leftarrow 0$ ***to*** $step - 1$ **do**
10 $\quad\quad\quad$ **forall** *unordered pairs* $(a, b) \in \{0, 1, \ldots, n-1\}^2, a \neq b$ **do**
11 $\quad\quad\quad\quad u \leftarrow i + j + a \cdot step$
12 $\quad\quad\quad\quad v \leftarrow i + j + b \cdot step$
13 $\quad\quad\quad\quad G$.addEdge(S_u, S_v)
14 **return** G

Algorithm 4: Simulation algorithm of PMC model.

Input: G: graph, F: set of faulty nodes
Output: Test results dictionary with reliable reports

1 $node\_index \leftarrow$ Create mapping $\{node : index\}$ for all nodes in G
2 $results \leftarrow$ Empty dictionary
3 **for** *each node* $u \in G$ **do**
4 $\quad results[u] \leftarrow$ Empty list /*Initialize test entries*/
5 $\quad$ **for** *each neighbor* $v \in G[u]$ **do**
6 $\quad\quad$ **if** $u \in F$ **then**
7 $\quad\quad\quad$ **if** `random_choice`$(0, 1) = 1$ **then**
8 $\quad\quad\quad\quad results[u]$.append($v$) /*False positive/negative*/
9 $\quad\quad$ **else**
10 $\quad\quad\quad$ **if** $v \in F$ **then**
11 $\quad\quad\quad\quad results[u]$.append($v$) /*True positive*/
12 $results \leftarrow \{u : \text{list} \in results \mid \text{list is non-empty}\}$
13 **return** $results$

Algorithm 3 generates $BCube(n, k)$. The steps are as follows:

(1) Graph Initialization(Steps 1–2): Create an empty undirected graph G and compute the total node count $N = n^{k+1}$, corresponding to the capacity of the $k + 1$-dimensional topology.
(2) Node Addition with Dimension Labels(Steps 3–5): Create an empty undirected graph G and add all binary strings as vertices. This step establishes the foundational structure for edge connections.
(3) Hierarchical Edge Connection(Steps 6–13): Iterate over each level ($level \in [0, k]$) and connect nodes as follows:
   - Step Calculation: Define the current level's step size as $step = n^{level}$.
   - Subcube Partition: Partition nodes into subcubes of size $n \cdot step$ (e.g., n nodes per subcube when $step = 1$).
   - Intra-group Full Connection: Split each subcube into groups of length $step$, then add edges between distinct index pairs (a, b) $(a \neq b)$ within each group (e.g., connect $S_{i+j+a\cdot step}$ and $S_{i+j+b\cdot step}$).
(4) Return Graph(Step 14): The final graph $BCube(n, k)$.

3.2. Simulation algorithm of PMC and MM*

In this subsection, we develop simulation algorithms based on the diagnostic rules of the PMC and MM* models. The algorithm design adheres to the following specifications: In the simulation scenario, the input parameters comprise the system topology graph G and a randomly generated faulty node set F (Note: The random generation of F is solely for simulation verification). In practical applications, the set of faulty nodes is unknown and must be identified, rather than provided as input. The algorithm outputs a symptom dictionary structured as a $key : value$ pair, defined as $D = \{u : N^+(u) | \sigma(u, v) = 1\}$. Here, u represents the tester node, v the testee node, $N^+(u)$ denotes the test results indicating abnormal neighbor nodes of u, and $\sigma(u, v) = 1$ signifies a positive test result from u indicating a failure at node v. Notably, there is a distinction between simulation experiments and engineering practice. In simulations, we preset the faulty node set F to simulate system abnormalities and validate the algorithm's effectiveness. In actual deployment, F is not an input parameter. Instead, the algorithm serves as the core testing module of the diagnosis system, constructing a symptom dictionary in real time by analyzing test results.

Algorithm 4 simulates the PMC model for detection in a network. The steps are as follows:

(1) Initialization Parameters (Steps 1–2):
   - Create a mapping of nodes to their respective indices within the graph G, which will help in efficient access of nodes.
   - Initialize an empty dictionary $results$ to store the test results for each node in the network.
(2) Test All Neighbors of Each Node u (Steps 3–11):
   - For each node u in the graph G, initialize an empty list in $results[u]$ to store the test results corresponding to that node.
   - For each neighbor v of node u in the graph G, perform the following:
   - If node u is faulty (i.e., $u \in F$):
     – Generate a random choice between 0 and 1. If the result is 1, append node v to the values of u, indicating a false positive or false negative.
   - If node u is reliable (i.e., $u \notin F$):
     – If node v is faulty (i.e., $v \in F$), append v to the test results of u, indicating a true positive (i.e., node u correctly detects node v as faulty).
(3) Filter Out Empty Test Entries (Step 12):
   - After testing all neighbors, filter the $results$ dictionary to remove any entries where the list of test results is empty. This ensures that only nodes with non-empty test results are keep in the final output.
(4) Return Test Results (Step 13):
   - Return the filtered $results$ dictionary, which contains the reliable test results for each node in the network.

Theorem 1. *Given a graph $G = (V, E)$, where $|V| = N$, $|E| = M$, and a set of faulty nodes $F \subseteq V$, the time complexity of Algorithm 4 is $O(N + M)$.*

Proof. We analyze time expenditure according to algorithmic steps:

Steps 1–2: Initialization phase
- Creating a node index map requires traversing all nodes: $O(N)$;
- Initialize an empty dictionary: $O(1)$;
- Total time: $O(N)$.

Steps 3–13: Main test loop
- The outer loop iterates through each node $u \in V$: executed N times;
- For each node u, the inner loop iterates over all its neighbors $v \in N(u)$;
- The sum of the degrees of all nodes satisfies: $\sum_{u \in V} deg(u) = 2M$;
- Operations within the loop (set query, random selection, list append) are all $O(1)$;
- Total time: $O(N + M)$.

Step 14: Filter empty results

- Iterate through all N entries in the result dictionary: $O(N)$;
- Check if the list is empty: $O(1)$ per operation
- Total time: $O(N)$.

In summary, the overall time complexity of the algorithm is: $O(N) + O(N+M) + O(N) = O(N+M)$.

□

Next, Algorithm 5 simulates the MM* model for detection in a network. The steps are as follows:

Algorithm 5: Simulation algorithm of MM* model.

Input: G: graph, F: set of faulty nodes
Output: Test results dictionary with pairwise fault reports

```
node_index ← {node : index | ∀node ∈ G} /*Node-to-index mapping*/
results ← ∅ /*Initialize empty dictionary*/
for each node u ∈ G do
    neighbors ← List of u's neighbors
    if |neighbors| is odd then
        v ← random_choice(neighbors)
        neighbors.append(v) /*Duplicate a random neighbor*/
    pairs ← Group neighbors into consecutive pairs /*e.g., [n1, n2], [n3, n4]*/
    for each pair (v1, v2) ∈ pairs do
        if u ∈ F then
            if random_choice(0, 1) = 1 then
                results[u].add((v1, v2)) /*False pair report*/
        else
            if v1 ∈ F ∨ v2 ∈ F then
                results[u].add((v1, v2)) /*Valid fault pair*/
results ← {u : pairs ∈ results | pairs ≠ ∅}
return results
```

(1) Initialization Parameters (Steps 1–2):
- Create a node-to-index mapping for all nodes in the graph G, which facilitates quick access to nodes by their indices.
- Initialize an empty dictionary *results* to store the test results for each node in the network.

(2) Test All Neighbors of Each Node u (Steps 3–15):
- For each node u in the graph G, gather the list of neighbors of u.
- If the number of neighbors of node u is odd, randomly select one of the neighbors v and append v to the neighbors list.
- Group the neighbors of node u into consecutive pairs. For example, if the neighbors are $[n_1, n_2, n_3, n_4]$, the pairs would be $[n_1, n_2]$ and $[n_3, n_4]$.
- Generate Test Results for Each Pair:
  - For each pair of neighbors (v_1, v_2), perform the following:
  - If node u is faulty (i.e., $u \in F$):
    - Generate a random choice between 0 and 1. If the result is 1, append the pair (v_1, v_2) to the test results of u, indicating a false pair report.
  - If node u is reliable (i.e., $u \notin F$):
    - If either v_1 or v_2 is faulty (i.e., $v_1 \in F$ or $v_2 \in F$), append the pair (v_1, v_2) to the test results of u, indicating a valid fault detection.

(3) Filter Non-Empty Entries (Step 16):
- After processing all nodes, filter the *results* dictionary to remove any entries where the list of reported pairs is empty. This ensures that only nodes with valid test results are retained in the final output.

(4) Return Test Results (Step 17):
- Return the filtered *results* dictionary, which contains the pairwise fault detection results for each node in the network.

Theorem 2. *Given a graph $G = (V, E)$, where $|V| = N$, $|E| = M$, and a set of faulty nodes $F \subseteq V$, the time complexity of Algorithm 5 is $O(N + M)$.*

Proof. We analyze time expenditure according to algorithmic steps:

Steps 1–2: Initialize node index mapping and result dictionary, with a time complexity of $O(N)$
Step 3–15: The outer loop iterates over all nodes $u \in V$, and each node is processed as follows:
- Step 4: Get the list of neighbors in time $O(\deg(u))$;
- Step 5–7: Check the parity of the number of neighbors and possibly replicate neighbors, with a time complexity of $O(1)$.
- Step 8: Group neighbors into pairs in $O(\deg(u))$ time;
- Step 9–15: The inner loop processes each neighbor pair with a time complexity of $O(\deg(u))$;
- Therefore, the processing time for each node u is $O(\deg(u))$, and the total processing time for all nodes is: $\sum_{u \in V} O(\deg(u)) = O(M)$.

Step 16: Filter non-empty results with a time complexity of $O(N)$.

In summary, the overall time complexity of the algorithm is: $O(N) + O(M) = O(N + M)$. □

3.3. Feature engineering-dataset construction

In this subsection, we introduce the algorithm for generating the simulated dataset, whose inputs are the network graph G and the number of faulty nodes num_faulty. The output is a CSV file that contains node features and fault labels. These features are categorized into three groups: (1) structural features, including sum_PMC (the sum of the number of times each node is tested as 1 under the PMC model), sum_MM* (the sum of the number of times each node is tested as 1 under the MM* model), and $degree$ (the node degree); (2) ratio features, encompassing $ratio\_PMC$ (the proportion of sum_PMC and $degree$) and $ratio\_MM*$ (the proportion of sum_MM* and $degree$); and (3) differential features, namely $diff\_PMC$ (the difference between sum_PMC and $degree$) and $diff\_MM*$ (the difference between sum_MM* and $degree$). The Algorithm 6 generates the simulated dataset, and Fig. 6 is flowchart of dataset constraction.

(1) Randomly select the faulty node: Randomly select num_faulty nodes from the network graph indicates that faulty nodes.
(2) Execute to testing model: Execute the Algorithm 4 (PMC model) and Algorithm 5 (MM* model) respectively.
(3) Count positive nodes: Count the number of times each node is tested as positive in the PMC model and the MM* model.
(4) Construct the data matrix: the fault diagnosis result (0/1) of the faulty node and the fault indicator (PMC count /MM* count) are taken as the rows of the matrix, where the fault label is F/FF.
(5) Persistence result: Save the matrix as a CSV file.

According to the steps of Algorithm 6, the resulting data set style is shown in Table 1.

4. Methodology and experimental design

In this study, the experimental dataset is derived from the feature-engineered dataset generated as detailed in Section 3. The dataset comprises 4,600,820 samples, with each sample characterized by seven distinct features. Additionally, each sample is associated with a binary label, where F denotes a faulty node and FF signifies a non-faulty node. To facilitate compatibility with binary classification models, the categorical labels are numerically encoded as follows: FF is mapped to 0, and F to 1. Regarding the class distribution within the dataset, the proportion of faulty samples is 30%, while the proportion of non-faulty samples is 70%. Statistical tests have been conducted to confirm the absence of significant class imbalance in the dataset, thereby ensuring the robustness and reliability of subsequent analyses.

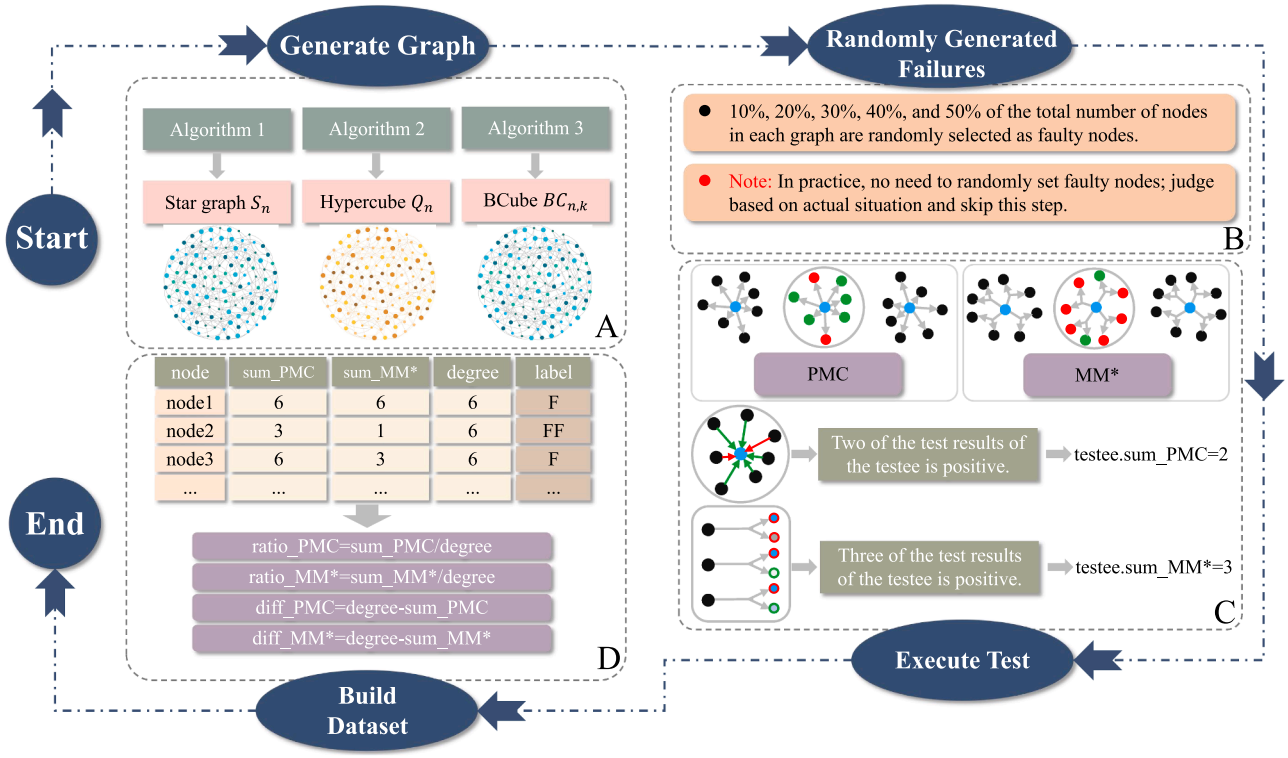

Fig. 6. The flowchart of dataset construction.

Algorithm 6: Data generation algorithm.

Input: Network graph G, number of faulty nodes num_faulty
Output: CSV file containing node statistics and fault labels

Procedure *Generate Faulty Nodes*
  Randomly select num_faulty nodes from G ;
  constraint $\nexists\, u \in G$ *where* $N(u) \subseteq F$;
Procedure *Execute Diagnostic Models*
  PMC test: Algorithm 4 ;
  MM* test: Algorithm 5 ;
Procedure *Count Fault Indicators*
  PMC_count = Count the number of times each node is diagnosed as 1 under the PMC model ;
  $MM*\_count$ = Count the number of times each node is diagnosed as 1 under the MM* model ;
Procedure *Construct Result Matrix*
  Create matrix with format: [$node\_id$, PMC_count, $MM*\_count$, $fault\_label$] ;
  where $fault\_label \in \{F, FF\}$;
Procedure *Calculation parameters*
  $ratio\_PMC = \frac{sum\_PMC}{degree}$;
  $ratio\_MM* = \frac{sum\_MM*}{degree}$;
  $diff\_PMC = degree - sum\_PMC$;
  $diff\_MM* = degree - sum\_MM*$;
Procedure *Construct Result Matrix*
  Create matrix with format: [$node\_id$, sum_PMC, sum_MM*, $ratio\_PMC$, $ratio\_MM*$, $diff\_PMC$, $diff\_MM*$, $fault\_label$] ;
  where $fault\_label \in \{F, FF\}$;
Procedure *Persist Results*
  Save matrix to CSV format ;

4.1. Model construction and optimization

Logistic Regression (LR) is a classical generalized linear classification model. It demonstrates effective binary classification capabilities in fault diagnosis tasks within the industrial control domain. This section will comprehensively elucidate the implementation mechanism of a fault diagnosis method grounded in Logistic regression. The model's core principle lies in establishing a nonlinear relationship between input features and target variables. It transforms the linear combination values of these features into probability estimates via the sigmoid function. Mathematically, its essence involves determining the optimal parameter vector using the maximum likelihood estimation method. This approach aims to minimize the cross entropy loss function, which quantifies the disparity between the model's predicted outputs and the actual labels.

Based on the logistic regression model, this section proposes a logistic regression training algorithm for data center fault diagnosis (see Algorithm 7 for details). Given a dataset with D-dimensional feature vectors $\mathbf{x} = [1, x_1, x_2, \ldots, x_D]^T$ (augmented with bias term) and binary labels $y \in \{0, 1\}$, the logistic regression model implements the following components:

Feature Representation and Linear Discrimination The model employs augmented feature vectors $\mathbf{x} = [1, x_1, \ldots, x_D]^T \in \mathbb{R}^{D+1}$, where the leading unit term serves as a bias absorption component. Subsequent dimensions contain Z-score normalized features ($x_i' = \frac{x_i - \mu_i}{\sigma_i}$) to ensure scale invariance. This unified parameter space formulation enables the linear discriminant function:

$$z = \mathbf{w}^T \mathbf{x} = w_0 + \sum_{i=1}^{D} w_i x_i$$

where the weight vector $\mathbf{w}$ integrates both decision boundary parameters (w_0) and feature significance coefficients ($\{w_i\}_{i=1}^{D}$).

Probabilistic Mapping The sigmoid activation transforms discriminant values into posterior probabilities:

$$P(y = 1|\mathbf{x}) = \sigma(z) = \frac{1}{1 + \exp(-z)} \in (0, 1)$$

Table 1
Dataset.

node	sum_PMC	sum_MM*	degree	ratio_PMC	ratio_MM*	diff_PMC	diff_MM*	label
node1	2	4	8	0.25	0.5	6	4	FF
node2	3	3	16	0.1875	0.1875	13	13	FF
node3	23	29	35	0.6571	0.8286	12	6	F
node4	2	0	8	0.25	0	6	8	FF
node5	0	0	7	0	0	7	7	FF
node6	13	13	16	0.8125	0.8125	3	3	F
node7	34	33	40	0.85	0.825	6	7	F
node8	1	3	7	0.1429	0.4286	6	4	FF
node9	2	5	8	0.25	0.625	6	3	FF
node10	5	4	8	0.625	0.5	3	4	F
...	...	...	...	...	...	...	...	F/FF

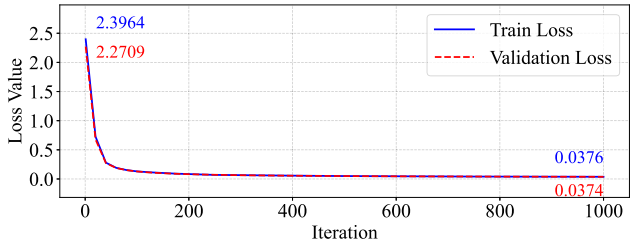

Fig. 7. Training and validation loss curves over 1000 iterations.

This function exhibits essential differentiable properties: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ enables simultaneous probability interpretation and efficient gradient computation, crucial for optimization convergence.

Optimization Framework The cross-entropy loss function quantifies prediction divergence:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \ln \sigma(z_i) + (1 - y_i)\ln(1 - \sigma(z_i))\right]$$

with matrix-form gradient:

$$\nabla\mathcal{L} = \frac{1}{N}\mathbf{X}^T(\sigma(\mathbf{X}\mathbf{w}) - \mathbf{y})$$

where the design matrix $\mathbf{X} \in \mathbb{R}^{N\times(D+1)}$ contains standardized samples and $\mathbf{y}$ denotes label vectors. This formulation enables batch processing while preserving numerical stability.

Parameter Optimization Batch gradient descent updates parameters iteratively:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\nabla\mathcal{L} \quad (t = 1, \ldots, 1000)$$

with learning rate $\eta = 0.1$ empirically determined to balance convergence speed and precision in the convex loss landscape.

Performance Metrics A quadrilateral evaluation system derives from confusion matrices: Accuracy, Precision, Recall, F1-score.

This framework comprehensively assesses classification accuracy, prediction reliability, defect detection completeness, and class-imbalance robustness. The following further explores the applicability of the proposed diagnostic algorithm under various fault modes. Sequential faults manifest as continuous failures in local topologies, but the probability of all neighbors of a single node failing simultaneously is extremely low, making the diagnostic process indistinguishable from random independent faults. Intermittent faults can be treated as dynamic random independent faults, and effective detection can be achieved by periodically executing the diagnostic algorithm. Although collaborative faults may affect diagnostic accuracy (especially when all neighbors fail), the probability of such occurrences is minimal, and most nodes can still be correctly identified with the aid of intelligent diagnostic algorithms.

Algorithm 7: Logistic regression training for data center fault diagnosis.

Input: Standardized features $X \in \mathbb{R}^{N\times D}$, labels $\mathbf{y} \in \{0, 1\}^N$, learning rate η, iterations T
Output: Trained parameters $\mathbf{w} \in \mathbb{R}^D$, $w_0 \in \mathbb{R}$

Initialize $\mathbf{w} \sim \mathcal{N}(0, 1)$ and $w_0 \sim \mathcal{N}(0, 1)$; `// Random parameter initialization`
for $t \leftarrow 1$ ***to*** T **do**
  Compute combined output $\mathbf{z} = X\mathbf{w} + w_0$; `// Linear combination with bias`
  Apply sigmoid activation $\mathbf{p} = \frac{1}{1+\exp(-\mathbf{z})}$ Calculate cross-entropy: $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \ln p_i + (1 - y_i)\ln(1 - p_i)]$
  Compute parameter gradients:
  $\nabla_{\mathbf{w}}\mathcal{L} = \frac{1}{N}X^\top(\mathbf{p} - \mathbf{y})$; `// Weight gradient`
  $\nabla_{w_0}\mathcal{L} = \frac{1}{N}\mathbf{1}^\top(\mathbf{p} - \mathbf{y})$; `// Bias gradient`
  Update parameters with learning rate:
  $\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla_{\mathbf{w}}\mathcal{L}$
  $w_0 \leftarrow w_0 - \eta\nabla_{w_0}\mathcal{L}$
return $\mathbf{w}, w_0$

Table 2
Validation set performance metrics.

Metric	Value	95% Confidence Interval
Accuracy	99.13%	99.08%–99.18%
Precision	98.27%	98.18%–98.36%
Recall	98.85%	98.77%–98.93%
F1 Score	98.56%	98.50%–98.62%

Table 3
Feature importance ranking.

Feature	Absolute Weight	Directionality
ratio_PMC	2.7668	Positive (+)
sum_PMC	1.0914	Positive (+)
ratio_MM*	1.0656	Positive (+)
diff_PMC	0.8546	Negative (-)
diff_MM*	0.7451	Negative (-)
sum_MM*	0.3322	Positive (+)
degree	0.2222	Positive (+)

4.2. Simulation experiment and evaluation

This study employs a systematic experimental design as follows. First, the dataset is partitioned into a training set and a validation set at an 8:2 ratio, with a fixed random seed (its size is 42) to maintain the original class distribution in both subsets. Second, a grid search is conducted to optimize the learning rate parameter, identifying the optimal value as $\alpha = 0.1$. The number of iterations T is set to 1000 based on the

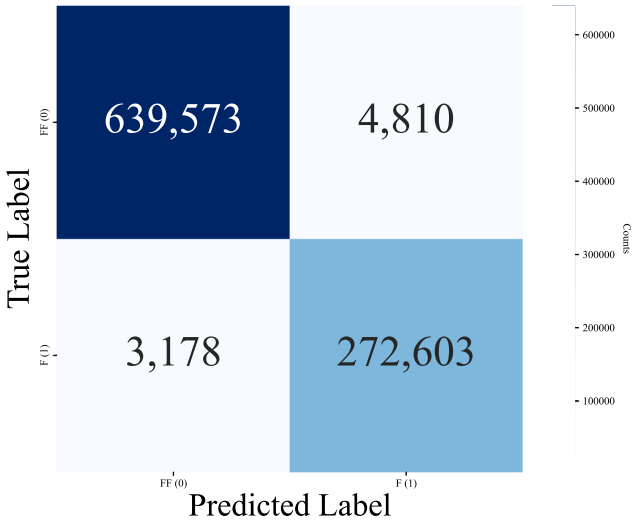

Fig. 8. Confusion matrix statistics of the validation set under 30% fault density in 920,164 nodes.

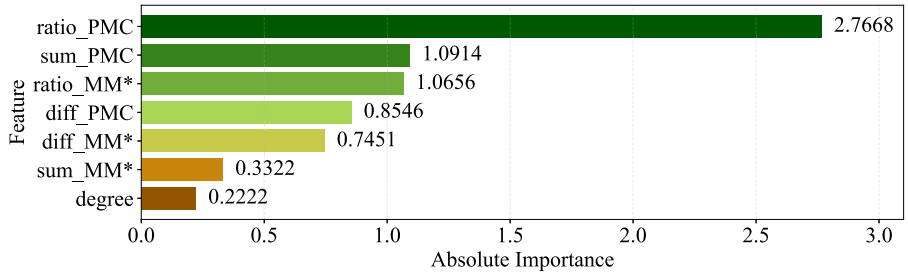

Fig. 9. Importance analysis of different characteristic parameters.

convergence characteristics of the loss curve. During training, the training/validation loss and accuracy metrics are recorded synchronously every 20 iterations. Intermediate model weights are saved at key iteration milestones (iterations 0, 100, and 500) for dynamic analysis. For performance evaluation, a multi-dimensional assessment framework is constructed, incorporating four core metrics: accuracy, precision, recall, and F1-score. Additionally, confusion matrices are utilized to visualize the distribution of classification errors, and loss curves are monitored to observe the model's convergence behavior.

The complete training process over 1000 iterations (as shown in Fig. 7) demonstrates a stable convergence trend of the model's loss function: the training loss decreases steadily from an initial value of 2.3964 to a final value of 0.0378, and the validation loss synchronously converges from 2.2709 to 0.0376. As shown in Fig. 8, the confusion matrix analysis results on the validation set (20% of the total dataset) obtained from demonstrate that the model maintains low false positive and false negative rates, controlled at 0.7% and 1.1% respectively. The validation accuracy reaches 97.5% at iteration 100 and ultimately improves to 99.13%. Final evaluation metrics (as presented in Table 2) indicate that the model achieves excellent performance on a validation set comprising 1,020,164 samples: it attains an accuracy of 99.13% (95% confidence interval: ±0.05%), a precision of 98.27%, a recall of 98.85%, and an F1-score of 98.56%. Feature importance analysis (as illustrated in Fig. 9) reveals key discriminative patterns: PMC-related features dominate the influence, with the ratio feature $ratio\_PMC$ emerging as the strongest predictor with an absolute weight of 2.7668, followed by sum_PMC (1.0914) and $ratio\_MM*$ (1.0656). The differential features $diff\_PMC$ (-0.8546) and $diff\_MM*$ (-0.7451) exhibit negative inhibitory effects, while the structural feature $degree$ (0.2222) has a relatively weak impact. The observed dominance of PMC-related features over MM*-based ones in the importance analysis can be attributed to the fundamental trade-off between accuracy and speed inherent in their respective diagnostic models. The PMC model prioritizes diagnostic accuracy by employing more exhaustive testing, which, although computationally more intensive, yields richer and more reliable system state information. In contrast, the MM* model emphasizes faster execution at the potential cost of some precision. Given that the primary objective of our fault diagnosis task is to maximize accuracy, the features derived from the more precise PMC model naturally assume greater importance, as they are more directly correlated with achieving the primary goal of reliable fault identification. Detailed data from the confusion matrix (as displayed in Fig. 8) show that: among positive samples, the model correctly detects 272,603 fault cases (true positives, TP) and misses 3178 cases (false negatives, FN, with a missed detection rate of 1.15%); among negative samples, it accurately identifies 639,573 normal states (true negatives, TN) and misclassifies 4810 cases as faults (false positives, FP, with a false alarm rate of 0.75%). This error distribution confirms that the model achieves a low false alarm rate while maintaining a high recall rate (98.85%).

The experimental results (see Table 4 and Fig. 11) show that the proposed model demonstrates excellent fault detection performance across datasets with different topologies and scales. For star networks $S_n(n = 6, 7, 8, 9)$, the model achieves perfect detection in low-fault-rate scenarios (e.g., 72 faulty nodes in S_6) with an accuracy, F1 score, and AUC of 1.0000. Even in the highest fault rate scenario (50% faults in S_9, with 181,440 faulty nodes out of 362,880 total nodes), the accuracy remains 0.9472, and the AUC is 0.9883. Notably, as the network scale increases exponentially (from 720 nodes in S_6 to 362,880 nodes in S_9), the model shows strong scalability. For example, in S_9, with 145,152 faults, the G-Mean score reaches 0.9792, and the false positive rate is controlled at 3.3% (4,797 FP vs. 212,931 TN). This indicates that the

Table 4
Model performance evaluation results of $S_n(n = 6, 7, 8, 9)$ and $Q_n(n = 14, 15, 16, 17)$.

Graph	Nodes	Faults	Accuracy	Precision	Recall	F1-Score	G-Mean	AUC	TN	FP	FN	TP
		72	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	648	0	0	72
		144	0.9944	0.9930	0.9792	0.9860	0.9887	0.9998	575	1	3	141
S_6	720	216	0.9819	0.9593	0.9815	0.9703	0.9818	0.9977	495	9	4	212
		288	0.9569	0.9446	0.9479	0.9463	0.9554	0.9907	416	16	15	273
		360	0.8931	0.8920	0.8944	0.8932	0.8931	0.9602	321	39	38	322
		504	0.9992	0.9941	0.9980	0.9960	0.9987	1.0000	4533	3	1	503
		1008	0.9976	0.9911	0.9970	0.9941	0.9974	1.0000	4023	9	3	1005
S_7	5040	1512	0.9841	0.9655	0.9821	0.9738	0.9836	0.9985	3475	53	27	1485
		2016	0.9621	0.9475	0.9583	0.9529	0.9615	0.9933	2917	107	84	1932
		2520	0.9236	0.9268	0.9198	0.9233	0.9236	0.9755	2337	183	202	2318
		4032	0.9999	0.9988	1.0000	0.9994	0.9999	1.0000	36283	5	0	4032
		8064	0.9987	0.9948	0.9988	0.9968	0.9987	1.0000	32214	42	10	8054
S_8	40320	12096	0.9894	0.9753	0.9897	0.9825	0.9895	0.9995	27921	303	124	11972
		16128	0.9690	0.9505	0.9732	0.9617	0.9697	0.9961	23374	818	432	15696
		20160	0.9306	0.9223	0.9404	0.9313	0.9306	0.9827	18563	1597	1201	18959
		36288	1.0000	0.9998	0.9999	0.9999	1.0000	1.0000	326584	8	2	36286
		72576	0.9991	0.9966	0.9987	0.9976	0.9989	1.0000	290059	245	97	72479
S_9	362880	108864	0.9939	0.9854	0.9944	0.9898	0.9940	0.9998	252408	1608	614	108250
		145152	0.9789	0.9674	0.9804	0.9739	0.9792	0.9978	212931	4797	2842	142310
		181440	0.9472	0.9439	0.9510	0.9474	0.9472	0.9883	171187	10253	8891	172549
		1638	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	14746	0	0	1638
		3276	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	13108	0	0	3276
Q_{14}	16384	4915	0.9996	0.9994	0.9992	0.9993	0.9995	1.0000	11466	3	4	4911
		6553	0.9968	0.9971	0.9950	0.9960	0.9965	0.9999	9812	19	33	6520
		8192	0.9836	0.9849	0.9822	0.9836	0.9836	0.9987	8069	123	146	8046
		3276	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	29492	0	0	3276
		6553	1.0000	0.9998	1.0000	0.9999	1.0000	1.0000	26214	1	0	6553
Q_{15}	32768	9830	0.9996	0.9988	1.0000	0.9994	0.9997	1.0000	22926	12	0	9830
		13107	0.9962	0.9921	0.9983	0.9952	0.9965	0.9999	19557	104	22	13085
		16384	0.9826	0.9751	0.9905	0.9827	0.9825	0.9990	15969	415	156	16228
		6553	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	58983	0	0	6553
		13107	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	52429	0	0	13107
Q_{16}	65536	19660	0.9998	0.9995	0.9998	0.9997	0.9998	1.0000	45867	9	3	19657
		26214	0.9977	0.9968	0.9974	0.9971	0.9976	1.0000	39238	84	68	26146
		32768	0.9875	0.9868	0.9882	0.9875	0.9875	0.9992	32336	432	387	32381
		13107	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	117965	0	0	13107
		26214	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	104857	1	0	26214
Q_{17}	131072	39321	0.9998	0.9993	0.9999	0.9996	0.9998	1.0000	91724	27	3	39318
		52428	0.9974	0.9946	0.9990	0.9968	0.9977	1.0000	78359	285	50	52378
		65536	0.9873	0.9811	0.9938	0.9874	0.9873	0.9995	64280	1256	407	65129

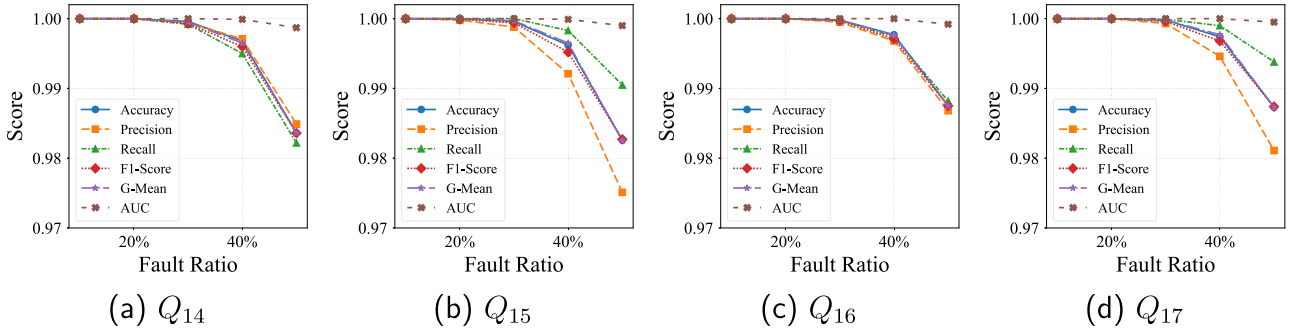

Fig. 10. Comparison of performance indicators for $Q_n(n = 14, 15, 16, 17)$.

model has strong capabilities to capture the topological characteristics of large-scale networks.

In the tests on hypercube networks $Q_n(n = 14, 15, 16, 17)$ (the experimental results see Table 4 and Fig. 10), the scheme exhibits even greater robustness. When the faulty node ratio reaches 50% (e.g., in Q_{17}, 65,536 faults out of 131,072 nodes), the accuracy remains at 0.9873, the true positive rate (TPR) is as high as 0.9938, and the false positive rate (FPR) is only 1.92% (1,256 FP vs. 64,280 TN). Of particular note is that in the medium-scale Q_{16} network (65,536 nodes), with 26,214 faults (40% fault ratio), the model still achieves an accuracy of 0.9977 and an F1 score of 0.9971, with a false negative rate of only 0.26% (68 FN vs. 26,146 TP). This result is highly relevant for fault-sensitive systems. The data also reveal that when the number of faulty nodes exceeds 30% of the total network nodes, performance curves for all models show an inflection point. However, the performance degradation in hypercube networks (from 1.0000 to 0.9873 in Q_{17}) is significantly less than that in star networks (from 1.0000 to 0.9472 in S_9), likely due to differences in network connection redundancy.

Table 5
Model performance evaluation results of $BCube(n,k)$, where $n = 7,8,9,10, k = 3,4$.

Graph	Nodes	Faults	Accuracy	Precision	Recall	F1-Score	G-Mean	AUC	TN	FP	FN	TP
BCube(7,3)	2401	240	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2161	0	0	240
		480	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1921	0	0	480
		720	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1681	0	0	720
		960	0.9979	0.9948	1.0000	0.9974	0.9983	1.0000	1436	5	0	960
		1200	0.9913	0.9836	0.9992	0.9913	0.9912	1.0000	1181	20	1	1199
BCube(7,4)	16807	1680	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	15127	0	0	1680
		3361	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	13446	0	0	3361
		5042	0.9999	0.9996	1.0000	0.9998	0.9999	1.0000	11763	2	0	5042
		6722	0.9990	0.9976	1.0000	0.9988	0.9992	1.0000	10069	16	0	6722
		8403	0.9953	0.9910	0.9996	0.9953	0.9953	1.0000	8328	76	3	8400
BCube(8,3)	4096	409	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3687	0	0	409
		819	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3277	0	0	819
		1228	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2868	0	0	1228
		1638	0.9990	0.9976	1.0000	0.9988	0.9992	1.0000	2454	4	0	1638
		2048	0.9963	0.9927	1.0000	0.9964	0.9963	1.0000	2033	15	0	2048
BCube(8,4)	32768	3276	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	29492	0	0	3276
		6553	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	26215	0	0	6553
		9830	1.0000	0.9999	1.0000	0.9999	1.0000	1.0000	22937	1	0	9830
		13107	0.9994	0.9986	1.0000	0.9993	0.9995	1.0000	19642	19	0	13107
		16384	0.9968	0.9936	0.9999	0.9968	0.9968	1.0000	16279	105	1	16383
BCube(9,3)	6561	656	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	5905	0	0	656
		1312	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	5249	0	0	1312
		1968	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	4593	0	0	1968
		2624	0.9991	0.9977	1.0000	0.9989	0.9992	1.0000	3931	6	0	2624
		3280	0.9962	0.9924	1.0000	0.9962	0.9962	1.0000	3256	25	0	3280
BCube(9,4)	59049	5904	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	53145	0	0	5904
		11809	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	47240	0	0	11809
		17714	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	41335	0	0	17714
		23619	0.9998	0.9994	1.0000	0.9997	0.9998	1.0000	35416	14	0	23619
		29524	0.9974	0.9948	1.0000	0.9974	0.9974	1.0000	29372	153	1	29523
BCube(10,3)	10000	1000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	9000	0	0	1000
		2000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	8000	0	0	2000
		3000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	7000	0	0	3000
		4000	0.9996	0.9990	1.0000	0.9995	0.9997	1.0000	5996	4	0	4000
		5000	0.9950	0.9903	0.9998	0.9950	0.9950	1.0000	4951	49	1	4999
BCube(10,4)	100000	10000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	90000	0	0	10000
		20000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	80000	0	0	20000
		30000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	70000	0	0	30000
		40000	0.9997	0.9993	1.0000	0.9996	0.9998	1.0000	59971	29	0	40000
		50000	0.9975	0.9950	1.0000	0.9975	0.9975	1.0000	49748	252	1	49999

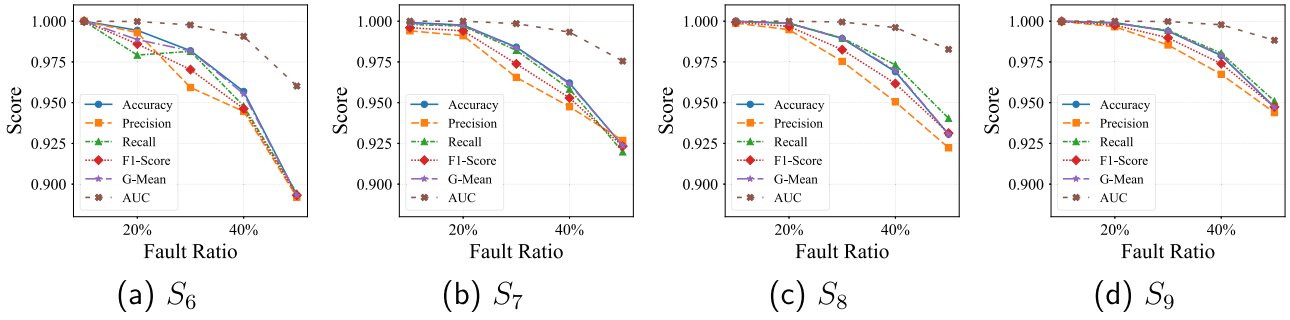

Fig. 11. Comparison of performance indicators for $S_n (n = 6,7,8,9)$.

The experimental results in the data center network are shown in Table 5 and Fig. 12. In the large-scale test for $BCube(10,4)$ with 100,000 nodes, even with a 50% fault rate (50,000 faults), the model maintains an accuracy of 0.9975, a recall rate of 1.0000, and an AUC of 1.0000. In the high-fault-density test for $BCube(9,4)$ with 59,049 nodes, when 29,524 faults are injected (50% fault ratio), the model only produces 153 false positives (FP rate 0.26%) and 1 false negative (FN rate 0.0034%), with the G-Mean score remaining stable at 0.9974. This robustness may stem from the multi-path connection feature of the BCube network and the model's hierarchical feature extraction mechanism. Comparative experiments show that under the same 50% fault ratio, the detection performance of the BCube structure is significantly better than that of star networks. This conclusion is drawn from the experimental comparisons presented in Tables 4 and 5. Under comparable network scales and fault densities (e.g., S_7 vs $BCube(9,3)$ or Q_{16} vs $BCube(9,4)$), the diagnostic method consistently achieves superior performance in the BCube topology. We attribute this advantage primarily to the higher node degree inherent to the BCube structure. For networks of similar size, a higher

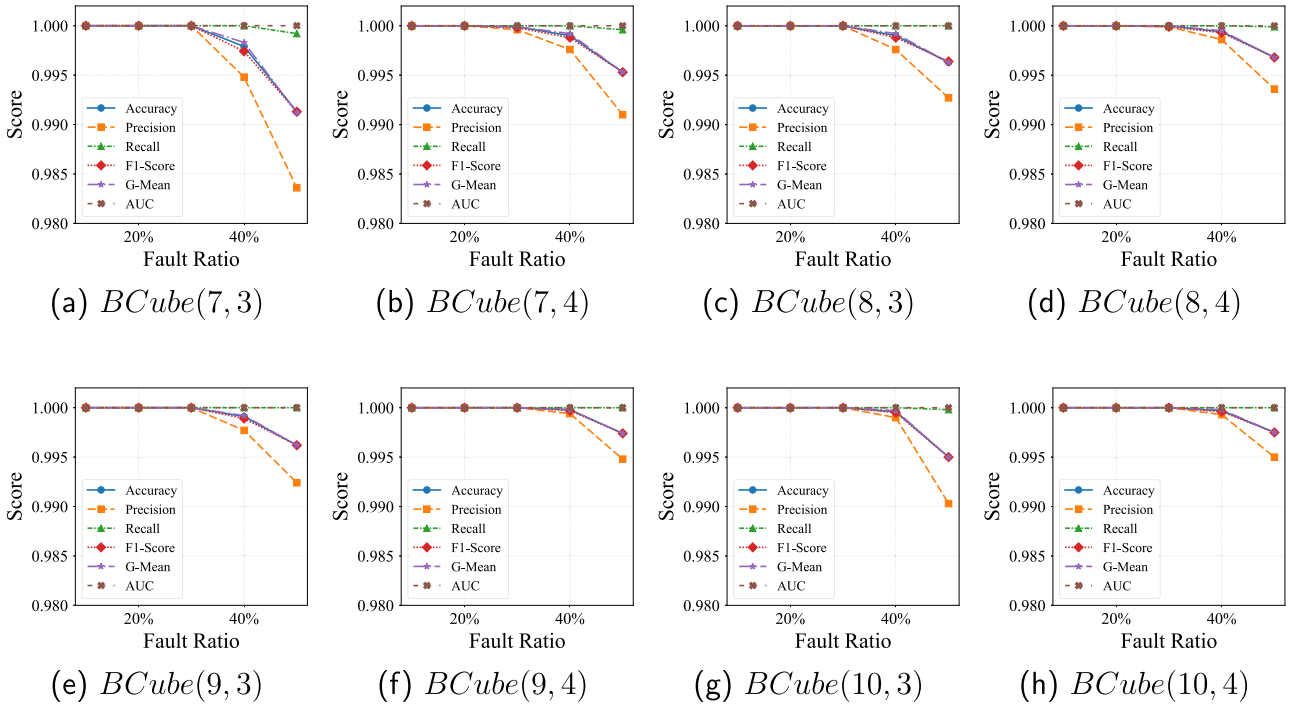

Fig. 12. Performance metrics comparison under different BCube configurations: (a)-(b) BCube(7,k), (c)-(d) BCube(8,k), (e)-(f) BCube(9,k), (g)-(h) BCube(10,k), where $k = 3, 4$.

node degree enhances connectivity and path diversity, which in turn provides more redundant information for the fault localization process, thereby improving diagnostic accuracy.

From the perspective of algorithmic efficiency, the AUC values for all test cases are above 0.96, with 23 out of 40 experiments on BCube networks reaching a perfect discrimination ability of 1.0000. It is also noteworthy that as fault density increases, precision and recall show asymmetric trends: in the case of 2520 faults in S_7, precision (0.9268) is slightly higher than recall (0.9198), while in Q_{17} with 65,536 faults, recall (0.9938) exceeds precision (0.9811). This may reflect how the distribution of fault patterns in different network topologies influences the model's decision boundaries. These findings provide valuable insights for parameter optimization in future models across different network environments.

Under high fault density ($\geq 50\%$), diagnostic performance exhibits a predictable decline. For instance, the accuracy of the hypercube Q_{17} drops from nearly 100% to 98.73%, and that of the star graph S_9 decreases to 94.72% (Tables 4 and 5). This is primarily due to the diminished reliability of PMC/MM* model testing (randomized test results from faulty nodes) and overlapping feature distributions (e.g., reduced discriminability of the *ratio_PMC* feature), leading to blurred decision boundaries in logistic regression. However, the moderating effect of topology is significant. The BCube network, with its high connection redundancy (Fig. 12), shows the smallest performance degradation, validating the resilience of network fault tolerance in extreme scenarios and pointing the way for future research (e.g., GNN enhancement).

4.3. Comparative experimental analysis

To comprehensively evaluate the performance of the proposed diagnostic method, we conducted comparative experiments with existing representative algorithms on multiple datasets. All experiments were performed under the datasets and settings used in the original papers of the compared algorithms to ensure fairness in the comparison. The results are shown in Tables 6 and 7.

Table 6
Comparison results with the ECD-PMC algorithm.

Fault type	Dataset	Method	Accuracy
1-extra 2-component fault set	Q_8	ECD-PMC	0.7970
		OURS	**0.9922**
	Q_9	ECD-PMC	0.8340
		OURS	**0.9980**
	Q_{10}	ECD-PMC	0.8960
		OURS	**0.9980**
	Q_{11}	ECD-PMC	0.9300
		OURS	**0.9985**
	Q_{12}	ECD-PMC	0.9590
		OURS	**0.9993**
	Q_{13}	ECD-PMC	0.9750
		OURS	**0.9999**
	Q_{14}	ECD-PMC	0.9860
		OURS	**0.9999**
1-extra 3-component fault set	Q_{10}	ECD-PMC	0.9090
		OURS	**0.9971**
	Q_{11}	ECD-PMC	0.9490
		OURS	**0.9985**
	Q_{12}	ECD-PMC	0.9700
		OURS	**0.9993**
	Q_{13}	ECD-PMC	0.9810
		OURS	**0.9998**
	Q_{14}	ECD-PMC	0.9890
		OURS	**0.9996**

Table 7
Comparison results with other learning-based algorithms.

Dataset	Methods	Accuracy
Power	OGGCN	0.8524
	OGGAT	0.8500
	OGSAGE	0.8350
	TGGCN	0.9187
	TGGAT	0.9054
	TGSAGE	0.8726
	TGSEGNN	0.9208
	OURS	**0.9844**

In comparison with the ECD-PMC algorithm (Wang et al., 2025b), our method achieved higher accuracy across multiple fault diagnosis tasks. Specifically, on the 1-extra 2-component fault set, for Q_8 to Q_{14}, our method achieved accuracies of 0.9922, 0.9980, 0.9980, 0.9985, 0.9993, 0.9999, and 0.9999, respectively, showing significant improvement over ECD-PMC's 0.7970, 0.8340, 0.8960, 0.9300, 0.9590, 0.9750, and 0.9860. On the 1-extra 3-component fault set, for Q_{10} to Q_{14}, our method also demonstrated comprehensive superiority, achieving 0.9971 and 0.9996 for Q10 and Q14, respectively, outperforming ECD-PMC's 0.9090 and 0.9890. This indicates that our method exhibits superior performance in handling faults of varying complexity (Algorithm 7).

In comparison with other learning-based methods (Wu et al., 2025), the proposed algorithm achieved an accuracy of 0.9844 on the Power dataset, outperforming all benchmark algorithms, including OGGCN (0.8524), OGGAT (0.8500), OGSAGE (0.8350), TGGCN (0.9187), TGGAT (0.9054), TGSAGE (0.8726), and TGSEGNN (0.9208). These results further validate the effectiveness and superior generalization capability of our algorithm in other fault diagnosis tasks.

In conclusion, the experimental results demonstrate that the proposed learning-based diagnostic algorithm achieves higher accuracy across various fault diagnosis tasks, exhibiting superior performance and broad applicability.

5. Concluding remarks

This study proposes an intelligent diagnostic solution integrating logistic regression with classical diagnostic models to address the need for efficient fault diagnosis in large-scale interconnection networks. By systematically designing generation algorithms for typical network topologies–including hypercubes (Q_n), star graphs (S_n), and BCube networks ($BC_{n,k}$)–and combining dynamic simulation testing rules from the PMC and MM* models, a multi-dimensional feature dataset with 4.6 million samples is constructed. During feature engineering, key features are extracted, including structural features (e.g., node degree, total PMC/MM* tests), proportional features (e.g., ratios of test outcomes), and differential features. Z-score standardization is applied to optimize model convergence.

The proposed logistic regression framework demonstrates superior diagnostic capabilities across varied network topologies. It exhibits remarkable robustness in large-scale star graph and hypercube configurations under high fault density conditions; for instance, in star graph S_9 with 50% fault density (181,440 faulty nodes out of 362,880 total), the model maintains an accuracy of 0.9472 and an AUC of 0.9883, while in hypercube Q_{17} under 50% fault rate (65,536 faulty nodes out of 131,072 total), accuracy remains at 0.9873 with a recall of 0.9938 (Table 4). Concurrently, the framework achieves exceptional precision and complete detection coverage in extreme BCube architecture testing, as evidenced by BCube(10,4) with 100,000 nodes and 50% faults, where accuracy reaches 0.9975 and recall is 1.0000 (Table 5). Comparative evaluations confirm the framework's unique adaptability to modern data center network structures, achieving consistently reliable fault identification without compromising detection integrity across all tested topologies.

Feature importance analysis underscores the dominant role of PMC-related metrics in decision-making processes, with $ratio\_PMC$ emerging as the most influential classifier component (absolute weight of 2.7668, as shown in Table 3). These results, supported by comprehensive data from diverse network scales and fault scenarios, highlight the framework's outstanding generalization capacity and operational effectiveness in heterogeneous networked environments.

While the evaluation demonstrates the effectiveness of our approach on synthetic datasets, its practical deployment in real-world data centers may face challenges such as test result noise and dynamic topology changes. Regarding noise resilience, our method incorporates an aggregation mechanism that sums test results from multiple neighbors. This design inherently mitigates the impact of sporadic testing errors or transient faults, as the influence of a single noisy data point is diluted in the aggregated value. Concerning dynamic topologies where nodes may be added or removed, the primary adjustment lies in the test assignment phase at the beginning of each diagnostic cycle. The core diagnostic algorithm itself remains unaffected, as it only requires re-initializing the test protocols to accommodate the new topology.

Future research will focus on the following directions: (1) integrating graph neural networks (GNNs) to enhance hierarchical representation of topological features; (2) developing probabilistic models for dynamic fault propagation to simulate real-world environments; (3) designing lightweight diagnostic frameworks based on edge computing to reduce resource consumption through model pruning and quantization; and (4) exploring cross-topology transfer learning mechanisms to improve generalization across heterogeneous hybrid networks.

CRediT authorship contribution statement

Wenfei Liu: Data curation, Writing – Original draft preparation, Software; **Jiafei Liu:** Conceptualization, Methodology, Foundation; **Jingli Wu:** Visualization, Investigation, Foundation; **Chia-Wei Lee:** Supervision, Writing – Reviewing and Editing; **Dajin Wang:** Supervision, Writing – Reviewing and Editing; **Gaoshi Li:** Writing – Reviewing and Editing.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [62302107, 62366007], Guangxi Natural Science Foundation [2025GXNSFBA069563, 2025GXNSFAA069507], Research Fund of Guangxi Key Lab of Multi source Information Mining Security [24-A-03-01], Basic Ability Enhancement Program for Young and Middle-Aged Teachers of Guangxi, China [2023KY0063], Young Elite Scientists Sponsorship Program by GXAST [2025YESSGX010], and Innovation Project of Guangxi Graduate Education [XYCS2025121].

References

Ahlswede, R., Aydinian, H. (2008). On diagnosability of large multiprocessor networks, *Discrete Applied Mathematics 156*, 3464–3474, https://doi.org/10.1016/j.dam.2008.02.001

Akers, S., Krishnamurthy, B. (1989). A group-theoretic model for symmetric interconnection networks, *IEEE Transactions on Computers 38*, 555–566, https://doi.org/10.1109/12.21148.

Chang, N.W., & Hsieh, S.Y. (2025). Conditional diagnosability of enhanced hypercubes under the PMC model. *IEEE Transactions on Networking*, *33*(4), 1603–1613, https://doi.org/10.1109/TON.2025.3545178

Chen, K., Meng, Z., Sun, D., Cao, W., Ren, Y., Sun, W., & Wu, J. (2025). A novel sliding mixing graph contrastive domain adaptation method for fault diagnosis under time-varying speeds. *Expert Systems with Applications*, *270*, 126576. https://doi.org/10.1016/j.eswa.2025.126576

Cox, D.R. (1959). The regression analysis of binary sequences, *Journal of the Royal Statistical Society. Series B: Methodological 21*, 238, https://doi.org/10.1111/j.2517-6161.1959.tb00334.x

Dahbura, A.T., & Masson, G.M. (1984). An $O(n^{2.5})$ fault identification algorithm for diagnosable systems. *IEEE Transactions on Computers*, *C-33*(6), 486–492. https://doi.org/10.1109/TC.1984.1676472

Ding, S., Zhang, P., Ding, E., Naik, A., Deng, P., & Gui, W. (2010). On the application of PCA technique to fault diagnosis. *Tsinghua Science and Technology*, *15*(2), 138–144. https://doi.org/10.1016/S1007-0214(10)70043-2

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., & Lu, S. (2009). BCube: A high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, *39*(4), 63–74. https://doi.org/10.1145/1594977.1592577

Huang, Y., Lin, L., Wang, X., Garg, S., Moussa, S., & Alrashoud, M. (2025). Graph-neural-network-based intermittent fault diagnosis for reliability of symbiotic internet of things. *IEEE Internet of Things Journal*, *12*(20), 41276–41286. https://doi.org/10.1109/JIOT.2025.3577566

Huang, Y., Lin, L., Xu, L., & Hsieh, S.Y. (2024a). Adaptive system-level fault diagnosis of bijective connection networks. *IEEE Transactions on Reliability*, *74*(2), 2916–2926. https://doi.org/10.1109/TR.2024.3425759

Huang, Z., Zhang, M., & Lee, C.W. (2024b). Connectivity and diagnosability of the complete josephus cube networks under *h*-extra fault-tolerant model. *Theoretical Computer Science*, *1020*, 114925. https://doi.org/10.1016/j.tcs.2024.114925

Jiang, D., He, C., Chen, Z., & Zhao, J. (2024). Are novel deep learning methods effective for fault diagnosis? *IEEE Transactions on Reliability*, *74*(3), 4170-4184. https://doi.org/10.1109/TR.2024.3510387

Lahiri, S., Abhijnan, C., & De, M.A. (2022). Fault diagnosis in power transmission line using decision tree and random forest classifier. In *2022 IEEE 6th International conference on condition assessment techniques in electrical systems (CATCON)* (pp. 57–61). https://doi.org/10.1109/CATCON56237.2022.10077633

Lin, L., Guan, K., Huang, Y., Hsieh, S.Y., & Chen, G. (2025). Local fault diagnosis analysis based on block pattern of regular diagnosable networks. *IEEE Transactions on Networking*, *33*(2), 849–864. https://doi.org/10.1109/TNET.2024.3507152

Lin, L., Zhou, S., & Hsieh, S.Y. (2023). Neural network enabled intermittent fault diagnosis under comparison model. *IEEE Transactions on Reliability*, *72*(3), 1206–1219. https://doi.org/10.1109/TR.2022.3199504

Lin, Y., Lin, L., Huang, Y., Xu, L., & Hsieh, S.Y. (2024). Endogenous security of FQ_n networks: Adaptive system-level fault self-diagnosis. *IEEE Transactions on Reliability*, *73*(3), 1659–1668. https://doi.org/10.1109/TR.2024.3356200

Lv, M., Dong, H., & Fan, W. (2025). Efficient fault tolerance and diagnosis mechanism for network-on-chips. *Journal of Network and Computer Applications*, *237*, 104133. https://doi.org/10.1016/j.jnca.2025.104133

Malek, M., A comparison connection assignment for diagnosis of multiprocessor systems, in: *Proceedings of the 7th annual symposium on computer architecture, ISCA '80, association for computing machinery* New York, NY, USA, (1980) 31–36.

Ngo, Q.H., Nguyen, B.L.H., Zhang, J., Schoder, K., Ginn, H., & Vu, T. (2025). Deep graph neural network for fault detection and identification in distribution systems. *Electric Power Systems Research*, *247*, 111721. https://doi.org/10.1016/j.epsr.2025.111721

Preparata, F.P., Metze, G., Chien, R.T., On the connection assignment problem of diagnosable systems, *IEEE Transactions on Electronic Computers*, *EC-16*, (1967) 848–854, https://doi.org/10.1109/PGEC.1967.264748

Qin, X.W., Hao, R.X., & Peng, S.L. (2024). Diagnosability of multigraph composition networks. *Theoretical Computer Science*, *988*, 114375. https://doi.org/10.1016/j.tcs.2023.114375

Sengupta, A., & Dahbura, A.T. (1989). On self-diagnosable multiprocessor systems: Diagnosis by the comparison approach. In *[1989] The nineteenth international symposium on fault-tolerant computing. digest of papers* (pp. 54–61). https://doi.org/10.1109/FTCS.1989.105543

Saad, Y., Schultz, M., Topological properties of hypercubes, *IEEE Transactions on Computers 37*, (1988) 867–872.

Sun, X., Fan, J., Cheng, B., Zhou, J., & Wang, Y. (2023). Relationship between component connectivity and component diagnosability of some regular networks. *The Computer Journal*, *66*(8), 2033–2042. https://doi.org/10.1093/comjnl/bxac061

Wan, Z., Lin, L., Huang, Y., & Hsieh, S.Y. (2024). Component diagnosis strategy of star graphs interconnection networks. *IEEE Transactions on Reliability*, *73*(4), 1907–1917. https://doi.org/10.1109/TR.2024.3357700

Wang, H., Liu, Z., Peng, D., & Zuo, M.J. (2023). Interpretable convolutional neural network with multilayer wavelet for noise-robust machinery fault diagnosis. *Mechanical Systems and Signal Processing*, *195*, 110314. https://doi.org/10.1016/j.ymssp.2023.110314

Wang, M., Xiang, D., Qu, Y., & Li, G. (2024). The diagnosability of interconnection networks. *Discrete Applied Mathematics*, *357*, 413–428. https://doi.org/10.1016/j.dam.2024.07.030

Wang, M.J.S., Xiang, D., & Hsieh, S.Y. (2025a). *G*-Good-neighbor diagnosability under the modified comparison model for multiprocessor systems. *Theoretical Computer Science*, *1028*, 115027. https://doi.org/10.1016/j.tcs.2024.115027

Wang, Q., Liu, J., Wang, D., Liu, W., Wu, J., & Li, G. (2025b). A novel conditional diagnostic scheme for hypercube-based multiprocessor systems. *IEEE/ACM Transactions on Networking*. https://doi.org/10.1109/TON.2025.3638453

Wu, C., Wu, J., Huang, Y., Lin, L., & Wang, D. (2025). A novel fault node diagnosis of interconnected network based on PMC-syndrome-enhanced graph neural network. *Journal of the Chinese Institute of Engineers*, (pp. 1–11). https://doi.org/10.1080/02533839.2025.2517799

Yang, L., Zhou, S., & Cheng, E. (2025). The (t,k)-diagnosability of cayley graph generated by 2-tree. *Journal of Parallel and Distributed Computing*, *200*, 105068. https://doi.org/10.1016/j.jpdc.2025.105068

Zhang, Q., Zhou, S., & Hsieh, S.Y. (2024). A probabilistic approach for local diagnosis in large multiprocessor systems. *IEEE Transactions on Networking*, *33*(3), 1086–1096. https://doi.org/10.1109/TON.2024.3520303

Zhuo, N., Zhang, S., Chang, J.M., & Ye, C. (2025). Non-inclusive diagnosability of folded hypercube-like networks. *Discrete Applied Mathematics*, *364*, 237–246. https://doi.org/10.1016/j.dam.2025.01.006