

Evolutionary hypergraph dismantling via deep reinforcement learning

Junjie Qian(钱俊杰), Wenlan Wang(王文蓝), Hanyun Wang(王涵韵), Qiqi Wang(王萁淇), Yao Zhang(张瑶), and Huijia Li(李慧嘉)

Citation: Chin. Phys. B, 2026, 35 (**6**): 068901. DOI: 10.1088/1674-1056/ae5a12
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

A node importance prediction algorithm based on graph attention and contrastive learning

Jun Ai(艾均), Yuming Zhang(张玉明), Zhan Su(苏湛), Chenye Guo(郭晨晔), and Mingsong Li(李铭松)
Chin. Phys. B, 2026, 35 (**5**): 058901. DOI: 10.1088/1674-1056/ae0680

Identification of vital nodes based on global and local features in hypergraphs

Li Liang(梁丽), Li-Yao Qi(齐丽瑶), and Shi-Cai Gong(龚世才)
Chin. Phys. B, 2025, 34 (**10**): 108904. DOI: 10.1088/1674-1056/adfefc

Six-degree gravity centrality for detecting influential nodes in networks

Jianbo Wang(王建波), Bohang Lin(林渤杭), Zhanwei Du(杜占玮), Ping Li(李平), and Xiao-Ke Xu(许小可)
Chin. Phys. B, 2025, 34 (**8**): 088902. DOI: 10.1088/1674-1056/adec62

Identifying important nodes of hypergraph: An improved PageRank algorithm

Yu-Hao Piao(朴宇豪), Jun-Yi Wang(王俊义), and Ke-Zan Li(李科赞)
Chin. Phys. B, 2025, 34 (**4**): 048902. DOI: 10.1088/1674-1056/adb269

Critical station identification of metro networks based on the integrated topological-functional algorithm: A case study of Chengdu

Zi-Qiang Zeng(曾自强), Sheng-Jie He(何圣洁), and Wang Tian(田旺)
Chin. Phys. B, 2025, 34 (**2**): 028902. DOI: 10.1088/1674-1056/ad9734

Evolutionary hypergraph dismantling via deep reinforcement learning

Junjie Qian(钱俊杰)[†], Wenlan Wang(王文蓝)[†], Hanyun Wang(王涵韵)[†], Qiqi Wang(王其淇)[‡], Yao Zhang(张瑶)[§], and Huijia Li(李慧嘉)[¶]

School of Statistics and Data Science, AAIS, LPMC and KLMDASR, Nankai University, Tianjin 300074, *China*

(Received 10 February 2026; revised manuscript received 25 March 2026; accepted manuscript online 1 April 2026)

Assessing the vulnerability of complex systems requires effective hypergraph dismantling strategies, yet existing methods struggle with the dynamic nature of cascading failures and the rugged optimization landscapes of high-order networks. In this paper, we propose a novel framework: hypergraph dismantling via evolutionary deep reinforcement learning (HD-EDR). First, we model a realistic dismantling environment incorporating hyperdegree-based and residual-capacity-based load redistribution mechanisms. Second, we introduce a hybrid learning architecture that synergizes the global exploration of evolutionary strategies with the gradient-based exploitation of deep reinforcement learning. A bidirectional parameter synchronization mechanism is designed to prevent the agent from being trapped in local optima. Furthermore, we integrate an inductive encoder to capture the evolving high-order dependencies of the residual network in real time. Extensive experiments across nine real-world datasets demonstrate that our framework significantly outperforms state-of-the-art baselines, providing a highly effective and robust strategy for maximizing structural damage in high-order networks.

Keywords: network dismantling, hypergraph, deep reinforcement learning, evolutionary strategies, combinatorial optimization

PACS: 89.75.Hc, 89.20.Ff, 07.05.Mh, 02.60.Pn

DOI: 10.1088/1674-1056/ae5a12 **CSTR:** 32038.14.CPB.ae5a12

1. Introduction

Analyzing network robustness is a critical issue in network science. Specifically, the network dismantling (ND) problem, which is defined as fragmenting a complex system into disconnected sub-networks by removing key nodes, has attracted significant attention. This problem is not only vital for designing robust infrastructure but also provides strategic insights for disrupting malicious networks (e.g., terrorist organizations or botnets[1–4]). Early research on ND primarily focused on simple graphs with pairwise interactions. Classic heuristic metrics, such as degree centrality, betweenness centrality, and eigenvector centrality, were widely used to identify critical nodes whose removal causes significant structural damage.[5–7] However, real-world systems (e.g., collaboration networks, biological pathways, and power grids) often involve high-order interactions that cannot be fully encoded by simple graphs. This has led to a surge of interest in hypergraphs. Initial approaches to hypergraph vulnerability extended traditional metrics to the high-order domain. For instance, hyperdegree and vector centrality were proposed to measure node importance based on hyperedge incidence.[8–10] More recently, researchers have applied tensor decomposition and spectral analysis to quantify hypergraph robustness.[11,12]

Beyond static structural research for ND, the dynamics of cascading failures[13] represent a more critical threat to system stability. The seminal Motter–Lai model[14] introduced the concept of load-capacity models, demonstrating how the redistribution of load following a node failure can trigger a global collapse. While extensive research has modeled these dynamics on simple graphs and interdependent networks, recent studies have begun to explore how high-order hyperedge connections amplify or mitigate these propagation effects.[15,16] To address complex combinatorial problems on graphs, deep reinforcement learning (DRL) has emerged as a powerful paradigm. Frameworks such as S2V-DQN[17] have successfully combined graph embeddings with Q-learning to solve classic problems such as minimum vertex cover, while subsequent works have applied GCNs to influence maximization tasks.[18,19]

However, despite its theoretical importance, developing an optimal dismantling strategy for hypergraphs under cascading failures proves to be computationally intractable (NP-hard), primarily due to the enormous state space of potential attack sequences and the highly unpredictable, non-linear dynamics of load redistribution. Although DRL has emerged as a promising paradigm for solving such combinatorial optimization problems by learning heuristics from environmental

[†]These authors contributed equally to this work.
[‡]Corresponding author. E-mail: qiqiwang@nankai.edu.cn
[§]Corresponding author. E-mail: yaozhang@nankai.edu.cn
[¶]Corresponding author. E-mail: hjli@nankai.edu.cn

© 2026 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

interactions,[18,20] applying standard DRL frameworks (e.g., Deep Q-Network, DQN) to this specific domain encounters substantial impediments. On one hand, the complex topological structure of hypergraphs creates a rugged optimization landscape in high-dimensional spaces, causing gradient-based agents to struggle with the exploration-exploitation dilemma and frequently become trapped in local optima, thereby missing the "butterfly effect" nodes essential for maximizing cascading damage.[21] On the other hand, this challenge is exacerbated by the need for dynamic structure adaptation; the network topology undergoes drastic transformations during the dismantling process due to node removals and subsequent failures, rendering traditional transductive embedding methods (e.g., HGNN[22] and HyperGCN[23]) ill-suited because they lack the flexibility to adapt to continuous topological changes without computationally prohibitive retraining.[24–26]

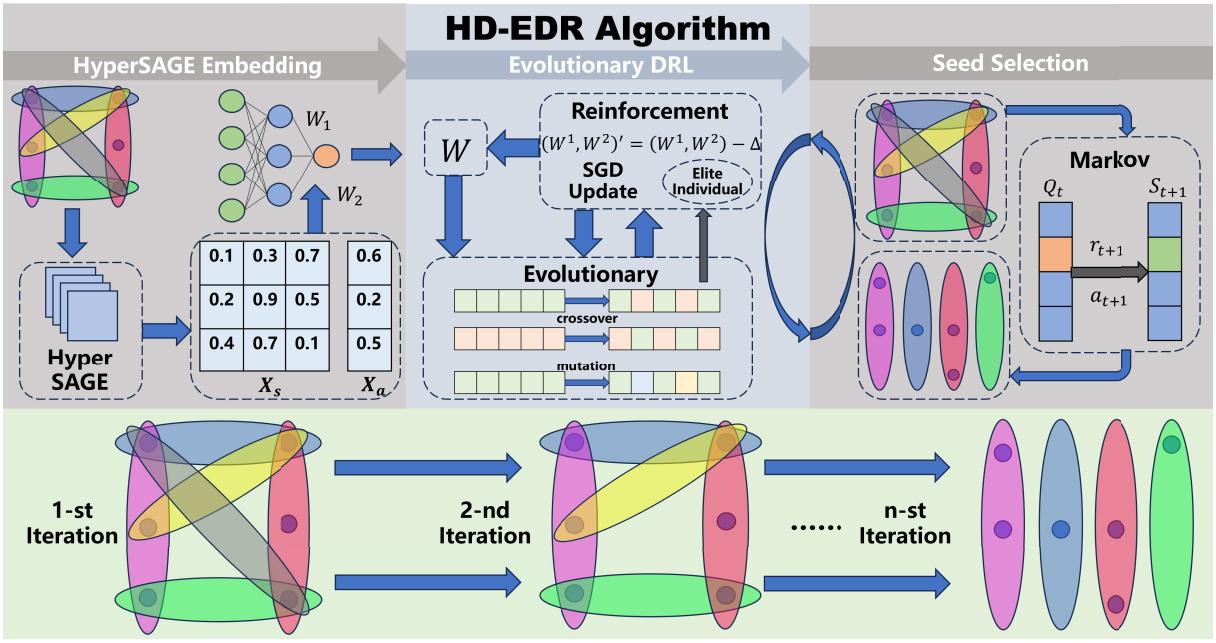


Fig. 1. First, the HyperSAGE embedding module inductively learns dynamic node and hyperedge to capture evolving topological features. Second, the evolutionary DRL engine optimizes the dismantling policy by synergizing global evolutionary search with local gradient-based updates. Finally, the seed selection process executes the learned policy as a Markov decision process, iteratively selecting critical nodes to maximize cascading failures over n steps.

To address these limitations, a novel hypergraph dismantling via evolutionary deep reinforcement learning (HD-EDR) framework is proposed in this paper. We illustrate the overall framework in Fig. 1. The main contributions of this paper are summarized as follows:

(i) Modeling cascading failures in hypergraphs We construct a realistic hypergraph dismantling environment by incorporating load redistribution models based on both hyperdegree and residual capacity. This shifts the focus from static topological removal to triggering dynamic avalanches, providing a more accurate evaluation of high-order network vulnerability under real-world cascading threats.

(ii) Integrating evolutionary strategies We propose a novel hybrid learning architecture, HD-EDR, that synergizes the global exploration of evolutionary strategies with the gradient-based exploitation of DRL. The designed bidirectional parameter synchronization mechanism effectively prevents the agent from being trapped in local optima, which significantly enhances the robustness of the policy search and ensures highly stable dismantling effectiveness across diverse network topologies.

(iii) Dynamic state representation Unlike existing heuristic methods that rely on static initial snapshots, we integrate an inductive HyperSAGE encoder to adaptively learn node embeddings from the continuously evolving hypergraph structure. This forward-looking capability allows our agent to capture real-time high-order dependencies, consistently inducing the earliest and most abrupt network fragmentation compared to transductive baselines.

(iv) Significant superiority in dismantling performance Extensive experiments across nine real-world hypergraph datasets demonstrate that HD-EDR achieves state-of-the-art performance. Quantitatively, our framework consistently records the lowest average accumulated normalized connectivity (ANC) area under the curve (AUC) under both load redistribution scenarios. Notably, HD-EDR outperforms the highly competitive DRL baseline, FINDER, by up to 25.4% in maximizing structural damage (e.g., on the DAWN dataset), proving its exceptional empirical superiority and generalization ability.

The rest of this paper is organized as follows. The problem formulation and preliminary knowledge are introduced in

Section 2, and the proposed HD-EDR is introduced in Section 3. Finally, the effectiveness of the proposed framework is validated in Section 4, and the paper is summarized along with potential future works in Section 5.

2. Problem formulation

2.1. Hypergraph

A hypergraph is defined as $G = (V, \mathcal{E})$, where $V = \{v_1, \ldots, v_N\}$ is the node set ($N = |V|$), and $\mathcal{E} = \{e_1, \ldots, e_M\}$ is the hyperedge set ($M = |\mathcal{E}|$). The size of a hyperedge e, denoted as $|e|$, represents the number of nodes it contains. Each hyperedge is defined as $e \subseteq V$ and $e \neq \varnothing$. Let k_{v_i} and $k_{v_i}^{\mathrm{H}}$ denote the degree and hyperdegree of node v_i, respectively, which are defined as the number of neighbors of v_i and the number of hyperedges containing v_i. Similar to the adjacency matrix, a hypergraph uses an incidence matrix to represent the relationship between nodes and hyperedges. The incidence matrix of a hypergraph is $\boldsymbol{A} \in \boldsymbol{R}^{N\times M}$, where each element of the incidence matrix $\boldsymbol{A}$ is defined as follows:

$$\boldsymbol{A}(i,j) = \begin{cases} 1, & v_i \in e_j, \\ 0, & \text{otherwise}. \end{cases} \tag{1}$$

The connectivity of a hypergraph is defined as the ratio of the number of nodes in the giant connected component (GCC) to the total number of nodes in the hypergraph

$$\sigma(G) = \frac{|V_{\mathrm{GCC}}|}{|V_{\mathrm{G}}|}, \tag{2}$$

where GCC is the connected component with the largest number of hyperedges in the hypergraph, and $|V_{\mathrm{GCC}}|$ and $|V_{\mathrm{G}}|$ represent the numbers of nodes in the GCC and the hypergraph, respectively. When the connectivity score is lower than 0.01, the hypergraph has fragmented into many tiny pieces, at which point the hypergraph is considered completely disintegrated.

2.2. Network dismantling and load redistribution model

When we dismantle a hypergraph, some nodes fail to work and their loads are redistributed to their functioning neighbor nodes. If the redistributed load causes a neighbor node to exceed its own capacity, the load will propagate through the network like an "avalanche" and ultimately result in a significant portion of nodes or even the entire network collapsing. This phenomenon is known as cascading failure.

The initial load $L^{(0)}$ of a node in a hypergraph is related to its hyperdegree, as shown below:

$$L_{v_i}^{(0)} = \alpha (k_{v_i}^{\mathrm{H}})^{\beta}, \quad i = 1, 2, \ldots, N, \quad \alpha \geqslant 1, \ \beta \geqslant 1. \tag{3}$$

Here, α and β are adjustable load parameters. As shown in Eq. (4), the node capacity C is defined as the limit of the load that a node can endure, which is proportional to its initial load

$$C_{v_i} = (1+T) L_{v_i}^{(0)}, \quad T \geqslant 0, \tag{4}$$

where T is the node capacity parameter.

This recursive redistribution triggers chain reactions, causing cumulative structural damage beyond the initial node removal. Therefore, an effective dismantling strategy must explicitly account for these cascading dynamics to maximize network fragmentation. To simulate cascading failures in hypergraphs, we introduce two specific load redistribution mechanisms based on the capacity-load model from Ref. [27].

Load redistribution model based on hyperdegree The initial approach[27] relies on a hyperdegree-dependent strategy. In this framework, the burden from a disabled node is reallocated to its operational neighbors through a two-phase mechanism. Initially, the load is partitioned equally among the hyperedges associated with the failed node. Subsequently, within each hyperedge, the load is uniformly disseminated to its surviving constituent nodes. This redistribution cycle propagates until the system reaches a steady state or total fragmentation occurs. The specific formulation is

$$\Delta L_{v_i \to v_j}^{(t)} = L_{v_i}^{(t)} \times \frac{1}{k_{v_i}^{\mathrm{H}}} \times \frac{1}{|e_{v_i,v_j}| - 1}, \tag{5}$$

where $\Delta L_{v_i \to v_j}^{(t)}$ represents the load distributed from node v_i to node v_j at time t, $L_{v_i}^{(t)}$ represents the load of node v_i at time t, and $|e_{v_i,v_j}|$ represents the size of the hyperedge containing both node v_i and node v_j. If there are multiple common hyperedges between node v_i and node v_j, the load distributed to node v_j is the sum of the loads distributed to it by each hyperedge. As the hypergraph dismantling process proceeds, the hyperdegree of nodes changes accordingly; therefore, $k_{v_i}^{\mathrm{H}}$ represents the hyperdegree of node v_i at that moment.

Load redistribution model based on node residual capacity To evaluate the robustness of our cascading failure algorithm under different scenarios, a second redistribution model from Ref. [27] is also introduced, which considers node residual capacity. When a node in the hypergraph fails, its load is distributed according to the proportion of the residual capacities of its neighbor nodes; the more residual capacity a neighbor node has, the more load it receives. Specifically,

$$\Delta L_{v_j}^{(t)} = C_{v_j} - L_{v_j}^{(t)}, \tag{6}$$

$$\Delta L_{v_i \to v_j}^{(t)} = L_{v_i}^{(t)} \times \frac{\Delta L_{v_j}^{(t)}}{\sum_{v_m \in \mathcal{N}(v_i)} \Delta L_{v_m}^{(t)}}, \tag{7}$$

where $\Delta L_{v_j}^{(t)}$ represents the residual capacity of node v_j at time t, and $\mathcal{N}(v_i)$ represents the set of neighbor nodes of node v_i.

If the sum of the load distributed to a node after a failure occurs in the hypergraph and its own load exceeds its capacity, the node fails, as shown below:

$$L_{v_j}^{(t+1)} = L_{v_j}^{(t)} + \Delta L_{v_i \to v_j}^{(t)} > C_{v_j}. \tag{8}$$

2.3. The objective function

The primary goal of hypergraph dismantling is to identify an optimal node removal sequence that maximizes structural degradation. Unlike static network attacks, the dismantling process under cascading failures is inherently dynamic and sequential. We define the attack strategy as a node removal sequence $A = (v_1, v_2, \ldots, v_N)$, where v_t represents the node targeted at step t. Following each removal, the hypergraph undergoes load redistribution and triggers potential cascading failures. Let G_0 denote the initial hypergraph and G_t represent the residual hypergraph after the t-th attack step and the subsequent stabilization. Mathematically, the problem of finding the most effective dismantling strategy can be formulated as minimizing the overall robustness of the hypergraph throughout the attack process. We define the objective function as the average connectivity of the residual network over time:

$$A^* = \arg\min_{A} \left(\frac{1}{N} \sum_{t=1}^{N} \frac{\sigma(G_t)}{\sigma(G_0)} \right), \tag{9}$$

where $\sigma(G_t)$ is the connectivity metric (e.g., the size of the giant component) of the residual hypergraph at step t.

Minimizing Eq. (9) is equivalent to finding a sequence that triggers the most rapid collapse of the network's giant component. This optimization objective necessitates that the proposed algorithm go beyond simple topological metrics to deeply mine the underlying dependency relationships between nodes, thereby maximizing the "damage efficiency" of the attack.

3. The HD-EDR algorithm

We propose HD-EDR (Fig. 1) to tackle the ND problem in hypergraphs. The essence of this method lies in transforming the original ND task into a continuous weight optimization problem for DQNs. To solve this, we devise an EDRL scheme. The HD-EDR framework is composed of two primary elements: (1) DQN weight optimization, which serves as the bridge transforming the hypergraph ND challenge into a parameter optimization task; and (2) EDRL, a hybrid engine that merges evolutionary algorithms (EAs) with DRL. In EDRL, the weights of each DQN are viewed as distinct policies, allowing for the simultaneous optimization of an entire population. The subsequent subsections will detail the implementation of these components.

3.1. Modeling the hypergraph dismantling with DQN

Here, we present the conversion of the hypergraph ND task into a continuous optimization problem regarding DQN weights. This procedure is executed through a three-step pipeline: (1) hypergraph feature embedding, which is designed to encode the potential and high-order characteristics of the graph structure; (2) DQN-based node removal, where the model evaluates Q-values based on the learned features to select optimal nodes via a Markov decision process (MDP); and (3) simulation optimization, which focuses on pruning unnecessary calculations related to network connectivity to enhance efficiency.

3.1.1. Hypergraph feature embedding

Let $G(V \setminus S)$ denote the size of the GCC of the hypergraph after removing the node set S. The ND problem aims to identify an optimal node set S^* with a cardinality of $|S| = K$, so as to minimize the size of the GCC. This can be equivalently formulated as maximizing the dismantling effect $\Phi(S)$, expressed as follows:

$$S^* = \text{argmax}\{\Phi(S) \mid S \in V, |S| = K\}, \tag{10}$$

where $\Phi(S) = |V| - |G(V \setminus S)|$ represents the number of nodes that are either removed or disconnected from the GCC.

The hypergraph feature embedding module is designed to extract low-dimensional representations for nodes and the global graph state. As shown in Eq. (11), previous works often utilize NHNE to learn static embeddings $\boldsymbol{H}$. However, the dismantling problem involves continuous node removal and structural fracture, meaning the topology changes at every step. Static methods such as NHNE are transductive and cannot handle such dynamic changes efficiently. Therefore, we adopt an inductive message-passing mechanism. Instead of retraining embeddings, we update node and hyperedge features iteratively based on the current remaining structure. This process consists of three simplified aggregation steps.

1. Node-to-hyperedge aggregation First, we generate the feature for each hyperedge e_j by aggregating the features of the nodes contained within it. This captures the local group information. Let $\boldsymbol{h}_{v_i}$ denote the feature of node v_i. The embedding $\boldsymbol{h}_{e_j}$ for hyperedge e_j is calculated as

$$\boldsymbol{h}_{e_j} = \phi \left(\boldsymbol{W}_1 \cdot \sum_{v_i \in e_j} \boldsymbol{h}_{v_i} \right), \tag{11}$$

where $\boldsymbol{W}_1$ is a trainable weight parameter and ϕ is a nonlinear activation function (e.g., ReLU). The summation aggregates information from all nodes v_i that belong to hyperedge e_j.

2. Hyperedge-to-node aggregation Next, we update the node representations by gathering information from the hyperedges they belong to. This allows nodes to sense the high-order structure. The updated feature $\boldsymbol{h}'_{v_i}$ for node v_i is defined as

$$\boldsymbol{h}'_{v_i} = \phi \left(\boldsymbol{W}_2 \cdot \sum_{e_j \ni v_i} \boldsymbol{h}_{e_j} + \boldsymbol{W}_{\text{self}} \cdot \boldsymbol{h}_{v_i} \right), \tag{12}$$

where $\{e_j \ni v_i\}$ represents the set of hyperedges containing node v_i. This step combines the structural context from hyperedges with the node's own previous feature.

3. Global state representation Finally, to allow the DQN agent to perceive the overall status of the residual network, we introduce a virtual global node. This node aggregates features from all existing hyperedges in the current graph to form a global state vector $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \sum_{\forall e_j \in \mathcal{E}'} \boldsymbol{h}_{e_j}, \tag{13}$$

where $\mathcal{E}'$ is the set of all remaining hyperedges in the current graph.

Through these steps, we obtain the updated node embeddings $\boldsymbol{H} = [\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_N]$ and the global state $\boldsymbol{Z}$, which serve as the dynamic inputs for the subsequent decision-making process.

3.1.2. Reinforcement learning based on the Markov decision process

The proposed module (Algorithm 1) is designed to approximate an optimal node removal sequence under an MDP framework spanning K discrete intervals. Here, the state vector $\boldsymbol{s}$ is composed of the hypergraph's topology, structural impact estimations, and the current node status. Action selection is driven by the state-action function $Q(\boldsymbol{s}, a)$, where the system greedily selects the node v_a that yields the highest Q-value to update the removal set S. Simultaneously, the reward function measures the dismantling effectiveness, defined as the decrease in the GCC size triggered by action a_t. These interactions are stored as 4-tuples $(\boldsymbol{s}_t, a_t, r_t, \boldsymbol{s}_{t+1})$ within the replay buffer $\mathcal{R}$ to train the DQN.

State In our inductive learning paradigm, the state at step t is defined by the current residual hypergraph topology and its corresponding embeddings, rather than a fixed-size matrix. We maintain a dynamic candidate set by tracking the node occupancy vector $\boldsymbol{O}$, where $o_i = 1$ indicates node v_i has been removed. Instead of feeding the entire variable-sized graph state into the network, we construct a localized, fixed-dimension state representation for each valid candidate node to ensure compatibility with the fixed-structure MLP.

State-action function To evaluate the potential of candidate nodes, the DQN leverages a deep neural architecture to map the node-specific state to a dismantling score. Let $Q(\boldsymbol{s}_t, a_i)$ denote the state-action value function for targeting node v_i. To keep the input dimension strictly consistent regardless of the dynamically shrinking graph scale, the input feature for action a_i is constructed by concatenating the specific node's current embedding $\boldsymbol{h}_{v_i} \in \mathbb{R}^d$ and the global state representation $\boldsymbol{Z} \in \mathbb{R}^d$. Thus, the input vector $[\boldsymbol{h}_{v_i} \oplus \boldsymbol{Z}]$ always has a fixed dimension of $2d$. Our network structure is based on a three-layer multi-layer perceptron (MLP), which computes the Q-values as

$$Q(\boldsymbol{s}_t, a_i) = \mathrm{DQN}([\boldsymbol{h}_{v_i} \oplus \boldsymbol{Z}]) = MLP_{\Theta}([\boldsymbol{h}_{v_i} \oplus \boldsymbol{Z}]), \tag{14}$$

where Θ represents the learnable weights. During the decision-making process, nodes with $o_i = 1$ in the occupancy vector $\boldsymbol{O}$ are masked out to prevent invalid actions. The output $Q(\boldsymbol{s}_t, a_i)$ serves as a metric for the quality of choosing node v_i under the current graph state.

Driven by the goal of obtaining high cumulative rewards, the system adopts a deterministic greedy strategy based on the learned Q-values. Specifically, the agent consistently picks the node associated with the highest expected return. This decision-making process is formulated as

$$Q(\boldsymbol{s}_t, a_t) = \max_a Q(\boldsymbol{s}_t, a), \tag{15}$$

$$a_t = \arg\max_a Q(\boldsymbol{s}_t, a). \tag{16}$$

Formally, for any given time t, the system first computes the peak Q-value via Eq. (15) and subsequently identifies the optimal action a_t via Eq. (16).

Transition Upon the execution of action a_t, the system undergoes a state transition from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$. This evolution is mathematically formalized through the following steps:

$$\boldsymbol{O}_{t+1} = \boldsymbol{O}_t \cup \{o_{a_t} \leftarrow 1\}, \tag{17}$$

$$S_{t+1} = \{v_i : o_i = 1,\ i = 1, \ldots, N\}, \tag{18}$$

$$\boldsymbol{Z}_{t+1} = \sum_{\forall e_j \in \mathcal{E}'_{t+1}} \boldsymbol{h}_{e_j}, \tag{19}$$

$$\boldsymbol{s}_{t+1} = [\boldsymbol{H} \oplus \boldsymbol{Z}_{t+1} \oplus \boldsymbol{O}_{t+1}]. \tag{20}$$

Specifically, Eq. (17) updates the occupancy vector by setting the index corresponding to the selected action a_t to 1, thereby generating the new state vector $\boldsymbol{O}_{t+1}$. Consequently, S_{t+1} in Eq. (18) defines the set of removed nodes, encompassing all elements where $o_i = 1$ within the sequence $\boldsymbol{O}_{t+1}$. Regarding the reward estimation, Eq. (19) updates the global state $\boldsymbol{Z}_{t+1}$ based on the remaining hyperedges $\mathcal{E}'_{t+1}$ in the residual graph after the removal action. Finally, the new state representation $\boldsymbol{s}_{t+1}$ is synthesized in Eq. (20) by concatenating the static features $\boldsymbol{H}$ with the updated dynamic components $\boldsymbol{Z}_{t+1}$ and $\boldsymbol{O}_{t+1}$.

Reward To evaluate the quality of a decision in the current state, we define the reward based on the incremental gain in the dismantling performance. The calculation is given by

$$r_t = \Phi(S_{t+1}) - \Phi(S_t), \tag{21}$$

where $\Phi(S_t)$ represents the structural impact of the removed node set S_t on the hypergraph, specifically quantified by the reduction in the size of the GCC. Through this formulation, the hypergraph ND task is effectively recast as a parameter optimization problem within the DQN framework.

3.1.3. Optimizing the dismantling simulation process

As outlined in Algorithm 1 (line 7), the DQN process requires computing the dismantling reward for each candidate node, a task that demands simulating the connectivity function $\Phi(S)$ multiple times (K runs). Since a full calculation of the GCC size is computationally expensive, we propose a method to optimize the dismantling simulation process. Specifically, we adopt a recursive approach: upon the removal of node v at time $t+1$, the network status inherits the topology from the previous timestamp t. This allows $\Phi(S)$ to be computed incrementally based on the existing component structure rather than initiating a fresh calculation from the initial graph state.

Algorithm 1 DQN-based Seed Node Selection: SNS()

Input: Weight parameter Θ of DQN, size of removal set K, feature $\boldsymbol{H}$, and replay buffer $\mathcal{R}$.
Output: Removal set S, dismantling score $\Phi(S)$, and replay buffer $\mathcal{R}$.

1: Initialize $\boldsymbol{O}$ as $[0,0,\ldots,0]_N$.
2: Calculate global state $\boldsymbol{Z}_1$ by Eq. (13).
3: **for** $t=1$ to K **do**
4: Calculate $Q(\boldsymbol{s}_t,a)$ by Eq. (14).
5: Select the action a_t with the highest Q value by Eqs. (15) and (16).
6: Execute action a_t and update $\boldsymbol{O}_t$ to $\boldsymbol{O}_{t+1}$ by Eq. (17).
7: Simulate the process of node removal and calculate reward r_t by Eq. (21).
8: Calculate $\boldsymbol{Z}_{t+1}$ by Eq. (13) based on the residual graph.
9: Update state $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ by Eq. (20).
10: Save $(\boldsymbol{s}_t,a_t,r_t,\boldsymbol{s}_{t+1})$ to replay buffer $\mathcal{R}$.
11: **return** removal set S, $\Phi(S)$, $\mathcal{R}$.

By significantly reducing the computational latency of fitness evaluation, this efficient simulation approach lays a solid groundwork for the population-based evolutionary optimization framework presented in the following section.

3.2. Evolutionary deep reinforcement learning

We propose HD-EDR to refine DQN weights by leveraging the complementary strengths of EA and DRL. This hybrid approach aims to harmonize exploration with exploitation through a dual-process guidance mechanism. Starting with an initialized DQN population, the method iteratively evolves candidates using standard EA operators such as mutation and crossover. Concurrently, DRL is tasked with the fine-tuning of elite individuals, which are subsequently merged back into the population. The process concludes with a selection phase that guarantees the retention of diverse and effective agents for future generations.

3.2.1. Mutual guidance between EA and DRL

A fundamental limitation of DRL is its susceptibility to local optima, primarily driven by the fixed directionality of gradient descent.[25] To mitigate this, HD-EDR implements a shared experience replay mechanism, archiving interaction transitions from the entire population rather than a single agent. This allows the DRL component to leverage the collective historical data of the evolutionary process. Conversely, the EA component facilitates global exploration within the parameter space via stochastic operators (i.e., crossover, mutation, and selection), effectively searching for the optimal removal node set. However, in the context of large-scale hypergraphs, the solution space for the ND problem is expansive, making it computationally prohibitive to identify optimal solutions within a limited number of evolutionary iterations. To overcome this efficiency bottleneck, we integrate an n_t-step Q-learning technique rooted in DRL to expedite both the training convergence and the evolutionary progression of the DQN.

Algorithm 2 HD-EDR Algorithm Framework.

Input: Maximum number of generations n_{g}, population size n_{p}, crossover probability $P_{\mathrm{crossover}}$, and mutation probability P_{mutation}.
Output: Best removal set S^*.

1: Initialize n_{p} DQNs for population $pop(0)$ and an empty replay buffer $\mathcal{R}$.
2: $\tau,\mathcal{R} \leftarrow$ Evaluation$(pop(0),\mathcal{R})$.
3: **while** $g < n_{\mathrm{g}}$ **do**
4: Index $[1,n_{\mathrm{p}}/2]$ and $[n_{\mathrm{p}}/2+1,n_{\mathrm{p}}]$ randomly pair to form tuple $Pairs[(T_1,L_1),\ldots,(T_{n_{\mathrm{p}}/2},L_{n_{\mathrm{p}}/2})]$.
5: **for** each pair (T,L) in $Pairs$ **do**
6: **if** $U(0,1) < P_{\mathrm{crossover}}$ **then**
7: Perform crossover $(pop(g)[T],pop(g)[L])$, and the generated individuals are added to $pops(g)$.
8: **end if**
9: **end for**
10: **for** each individual $pop(g)$ in $pops(g)$ **do**
11: **if** $U(0,1) < P_{mutation}$ **then**
12: Perform mutate operation.
13: **end if**
14: **end for**
15: $\tau,\mathcal{R} \leftarrow$ Evaluation$(pops(g),\mathcal{R})$.
16: Find the best individual π in $pops(g)$.
17: Randomly sample n_{b} batches $\{(\boldsymbol{s}_t,a_t,r_t,\boldsymbol{s}_{t+1})\}_{n_{\mathrm{b}}}$ from $\mathcal{R}$, where each batch consists of $n-step$ transition tuples.
18: Calculate the loss value L_{loss} according to Eq. (22).
19: Update weight Θ of π using SGD according to total loss L_{loss}.
20: Select $n_{\mathrm{p}}/2$ individuals with high fitness from $\{pop(g)\cup pops(g)\cup\pi\}$ to $pop(g+1)$.
21: Randomly select $n_{\mathrm{p}}/2$ solution from $pop(g)\cup pops(g+1)$ to $pop(g+1)$.
22: $g \leftarrow g+1$.
23: **end while**

During the evaluation phase, the historical transition tuples, denoted as $(\boldsymbol{s}_t,a_t,r_t,\boldsymbol{s}_{t+1})$, are accumulated in the replay buffer $\mathcal{R}$. After performing n training iterations, HD-EDR

samples batches of these tuples from $\mathcal{R}$ utilizing the n_t-step Q-learning strategy. The objective is to minimize the DQN loss function L_{loss}, which is formulated as

$$L_{\text{loss}} = \mathbb{E}_{\varsigma\in\mathcal{R}}\left[\left(\sum_{k=0}^{n-1}\gamma^k r_{t+k}+\gamma^n \max_{a'}\hat{Q}(s_{t+n},a')-Q(s_t,a_t)\right)^2\right], \quad (22)$$

where γ serves as the discount factor governing the weight of future rewards and E signifies the expectation operator. The symbol $\mathcal{R}$ represents the experience replay repository storing the transition data. The loss function calculates the squared temporal difference (TD) error using an n-step return. The term $\varsigma = (s_t, a_t, r_t, s_{t+1})_{n_b}$ corresponds to a mini-batch of size n_b retrieved from $\mathcal{R}$ for the network update. By minimizing L_{loss}, the system computes the gradient $\Delta\Theta$ and employs stochastic gradient descent (SGD) to backpropagate errors and update the DQN weights Θ. The procedural details are outlined in Algorithm 2.

Following the optimization phase, the high-performing individual policy π is reinjected into the continuously evolving population. This mechanism allows the evolutionary framework to directly benefit from gradient-based search capabilities. Furthermore, the injection of optimized individuals plays a crucial role in stabilizing the overall learning trajectory.

3.2.2. EA design

The EA mainly consists of the solution representation, genetic operations (i.e., crossover and mutation), and a selection operation, which tries to evolve a population of solutions for the ND problem in hypergraphs.

Representation of the ND problem in hypergraphs We model each individual pop_i as a DQN, where the network's weight parameters Θ correspond to the individual's gene sequence. The fitness or quality of these sequences is assessed using Algorithm 1. In practice, once an individual is instantiated, the function SNS() is invoked to yield the removal node set S along with its corresponding dismantling score $\Phi(S)$.

Crossover To enhance population diversity while retaining high-quality traits, the crossover phase begins by ranking the current generation, $pop(g)$, in descending order of their fitness values, denoted as $|\boldsymbol{F}(S)|$. The population is then bisected: the top 50% are categorized as the elite group, while the remaining bottom 50% constitute the ordinary group. Mating pairs are formed by randomly selecting one candidate from the elite group and one from the ordinary group, represented as the pair $(pop(g)[T], pop(g)[L])$.

The crossover procedure is executed with a predefined probability $P_{\text{crossover}}$. Unlike traditional methods, this approach employs a probabilistic gene exchange mechanism. For every corresponding gene pair (x_k^t, x_k^l) within the selected parents, a random variable r is drawn from a uniform distribution $\boldsymbol{U}(0,1)$. If $r > 0.5$, the genetic information at position k is swapped between the two individuals. This process is visually illustrated in Fig. 2.

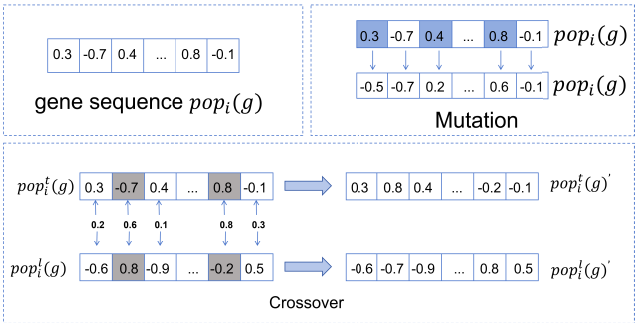


Fig. 2. Schematic illustration of crossover and mutation operations.

Mutation To prevent premature convergence and ensure genetic diversity, the mutation operator is applied to the gene set w of each individual $pop_i(g)$. This process involves introducing random perturbations using a function $\mathcal{N}$, which is governed by the standard normal probability density function $e^{-y^2/2}/\sqrt{2\pi}$. Specifically, this function is utilized to generate random values constrained within the interval $[-1,1]$. The outcome of this stochastic process forms the new population, denoted as $pops(g)$. A schematic representation of the mutation operator is illustrated in Fig. 2.

Selection To strike a balance between exploitation (maintaining high-fitness solutions) and exploration (preserving genetic diversity), the selection mechanism constructs the next generation from a composite pool. This pool includes the parent population $pop(g)$, the offspring $pops(g)$, and a specific individual π derived from DRL. The integration of π serves as a conditional heuristic: it accelerates the evolutionary process if it outperforms existing candidates, yet is naturally discarded if suboptimal, ensuring robust performance. The formation of the new population follows a hybrid approach: the top $n_p/2$ individuals are strictly selected based on fitness dominance to ensure effectiveness. Meanwhile, to mitigate premature convergence, the remaining $n_p/2$ slots are filled by individuals chosen from the residual pool. This procedure corresponds to lines 22 and 23 of Algorithm 2.

3.3. Complexity analysis

To demonstrate the scalability of the proposed HD-EDR framework, we analyze its time and space complexity. Let $N = |V|, M = |E|, N_{\text{nz}}$ be the number of non-zero elements in the incidence matrix $\boldsymbol{A}$, and d be the hidden embedding dimension.

Time complexity The time consumption per evolutionary generation primarily involves hypergraph feature embedding, DQN-based simulation, and parameter updates. The HyperSAGE message passing and linear transformations take $O(N_{\text{nz}} \cdot d + (N+M)d^2)$ time. For a population of size n_p,

selecting K nodes requires $O(n_p \cdot K \cdot N \cdot d^2)$ for DQN forward passes. Thanks to the incremental simulation optimization in Subsection 3.1.3, the structural update cost is reduced to a localized constant $O(C_{inc})$ rather than a full recalculation. Furthermore, the DRL and EA parameter updates take $O(n_p \cdot d^2 + n_b \cdot d^2)$, where n_b is the batch size. Overall, the time complexity is bounded by $O(n_p \cdot K \cdot N \cdot d^2 + N_{nz} \cdot d)$. Since d, K, and n_p are relatively small constants, the time complexity scales linearly with the hypergraph size, ensuring computational feasibility on large networks.

Space complexity The memory footprint is determined by the graph structure, model parameters, and the replay buffer $\mathcal{R}$. Storing the sparse incidence matrix $\boldsymbol{A}$ and the dynamic embeddings ($\boldsymbol{H}$ and $\boldsymbol{Z}$) requires $O(N_{nz} + (N+M)d)$ space. Maintaining n_p DQN models with fixed-dimension inputs takes $O(n_p \cdot d^2)$ memory. Finally, storing B transitions in the replay buffer requires $O(B \cdot d)$ space. Therefore, the total space complexity is $O(N_{nz} + (N+M)d + n_p \cdot d^2 + B \cdot d)$. By leveraging sparse representations and fixed-dimension state vectors, HD-EDR strictly avoids the $O(N^2)$ memory bottleneck typical of dense adjacency matrices, making it highly memory-efficient.

Algorithm 3 Evaluation().

Input: The population $pop(g)$ and replay buffer $\mathcal{R}$.
Output: An array τ composed of the removal node set, fitness of each individual in the population, and replay buffer $\mathcal{R}$.

1: **for** each individual $pop_i(g)$ in population $pop(g)$ **do**
2: $S, \Phi(S), \mathcal{R} \leftarrow \mathrm{SNS}(pop_i(g), \mathcal{R})$.
3: Put $[S, \Phi(S)]$ into τ.
4: **end for**
5: Sort the population $pop(g)$ in descending order based on their fitness $\Phi(S)$.

4. Experimental results

4.1. Dataset introduction

To verify the effectiveness of the proposed hypergraph cascading failure disassembly algorithm, nine real-world hypergraph datasets were collected. The hypergraph information includes the number of nodes N, the number of hyperedges M, the average hyperdegree size $\langle k^H \rangle$, the average hyperedge size $\langle |e| \rangle$, as well as the average degree $\langle k \rangle$, the clustering coefficient C, and the density of the corresponding ordinary graph (clique expansion). The basic information of the hypergraphs is shown in Table 1.

(i) Algebra and Geometry[28] Sourced from MathOverflow, this dataset models users as nodes. Hyperedges are defined by the co-participation of users, linking those who answered the same thread.

(ii) Bars-Rev[28] The data originates from the review website Yelp. Nodes represent Yelp users and hyperedges are sets of users who reviewed the same bar.

(iii) Congress-bills[29] In this dataset, nodes represent members of the US Congress and hyperedges are sets formed by the sponsors of a specific bill.

(iv) DAWN[29] Based on data from a drug abuse surveillance system, this network maps drugs to nodes. The hyperedges are constructed by grouping all substances associated with a specific patient's visit to the emergency room.

(v) Email-Eu and Email-Enron[29,30] These datasets utilize email exchange records between two institutions. We treat email addresses as nodes, with hyperedges representing the group of users (sender plus recipients) involved in each email transmission.

(vi) Tags-ask-Ubuntu and Tags-math-sx[29] These datasets were collected from Ask Ubuntu and Stack Exchange websites, and they model the relationship between tags. Tags serve as nodes, and a hyperedge is defined as the collection of tags attached to the same question.

Table 1. The statistical information of the datasets.

Dataset	N	M	$\langle k \rangle$	$\langle k^H \rangle$	$\langle \lvert e \rvert \rangle$	M/N	C	Density
Algebra	423	980	78.9	17.53	7.57	2.32	0.8	0.19
Bars–Rev	1 234	1 188	174.3	9.61	9.98	0.96	0.58	0.14
Congress-bills	1 462	3 819	211.23	29.84	11.42	2.61	0.55	0.14
DAWN	849	5 898	23.52	21.95	3.16	6.95	0.55	0.02
Email–Eu	656	2 384	19.75	10.57	2.91	3.63	0.53	0.03
Email–Enron	143	1 457	25.17	31.43	3.09	10.19	0.59	0.18
Geometry	580	888	164.79	19.9	13	1.53	0.82	0.28
Tags-ask-Ubuntu	757	1 710	9.7	6.58	2.91	2.26	0.55	0.01
Tags-math-sx	494	1 801	11.58	10.08	2.77	3.65	0.59	0.02

4.2. Evaluation metrics

To comprehensively evaluate the dismantling performance of the model, we adopt two complementary metrics: the ANC curve to visualize the dynamic degradation process, and the AUC to quantify the overall effectiveness.

Given a removal sequence, the ANC after removing nodes up to step t is defined as

$$\mathrm{ANC}(t) = \frac{\sigma(G_t)}{\sigma(G_0)}, \tag{23}$$

where $\sigma(G_t)$ represents the connectivity of the residual hypergraph at step t (and cascades have stabilized). The curve of $\mathrm{ANC}(n)$ versus n/N provides a visual representation of the speed of the network's collapse.

Based on Eq. (23), the AUC serves as a global metric reflecting the network's robustness. It is calculated by averaging the normalized connectivity over all removal steps:

$$\mathrm{AUC} = \frac{1}{N}\sum_{t=1}^{N} \mathrm{ANC}(t). \tag{24}$$

Note that the AUC metric aligns with our dismantling objective defined in Eq. (9). A lower AUC value indicates a faster

collapse of the giant component, signifying a more effective dismantling strategy.

4.3. Baseline algorithms

To comprehensively verify the performance of HD-EDR, we compare it with multiple baseline algorithms; they include:

Highest degree adaptive (HDA)[31] This strategy operates iteratively, targeting and eliminating the node that possesses the largest degree within the current network topology at every round.

Highest hyper-degree adaptive (HHDA)[32] Functioning as a counterpart to HDA, this method applies the same adaptive logic but prioritizes the removal of the node exhibiting the maximal hyper-degree in each sequence.

CI[33] The CI value of a node is calculated based on the node's degree and the sum of the degrees of its neighbors within a fixed number of hops, which can reflect the reachability between nodes. Therefore, the node with the maximum CI value in the hypergraph is removed at each step.

HITTER[34] HITTER maps hypergraph dismantling into a sequential decision-making problem. It calculates the Q-value of nodes via graph embedding and DQN and removes the node with the highest Q-value in the hypergraph at each step.

FINDER[35] is a state-of-the-art deep reinforcement learning framework designed for ND. Since FINDER was originally proposed for simple graphs, we adapt it by applying it to the clique-expansion of the hypergraphs, allowing the DRL agent to evaluate node importance based on the approximated pairwise topology.

4.4. Parameter settings

Environment and training We adopt an inductive learning paradigm. To ensure the agent learns structural patterns generalizable to unseen networks, we utilize HyperFF to generate a diverse set of synthetic hypergraphs for training. The training set consists of graphs with node counts $N \in [30, 60]$. The burning and expansion probabilities are sampled uniformly from $[0.1, 0.5]$.

Network architecture The HyperSAGE encoder consists of three layers with a hidden dimension of 64. The Q-network is an MLP with two hidden layers (128, 64 units).

Hyperparameters The DRL agent is trained for 3000 episodes with a replay buffer size of 500000. We use the Adam optimizer with a learning rate of 1×10^{-4} and a discount factor of $\gamma = 0.99$. For the evolutionary strategy, the population size is $K = 10$ and the mutation strength is $\sigma = 0.02$.

Cascading model Following Ref. [27], the capacity parameter T is set to 2, and the load parameters α, β are set to 10 and 1, respectively, to simulate a standard load redistribution scenario.

4.5. Main performance comparison

To evaluate HD-EDR against existing works on network cascading failures, we quantitatively compare it with five representative baselines using the AUC of the ANC. Tables 2 and 3 present the results under the hyperdegree-based and residual load-based models, respectively (a lower AUC indicates a faster network collapse).

Overall, HD-EDR significantly outperforms both traditional heuristics and competitive DRL methods. Compared to static heuristic methods (e.g., HDA, HHDA, CI), HD-EDR achieves an average AUC reduction (improvement) ranging from 9.2% to 11.0% across both cascading models. This quantitative gap demonstrates that our evolutionary DRL agent effectively overcomes the myopia of static structural metrics during dynamic load redistributions.

Furthermore, compared to recent DRL frameworks (HITTER, FINDER), HD-EDR still exhibits strong quantitative superiority, securing the best results on the majority of datasets. For instance, on the DAWN dataset under the residual load model (Table 3), HD-EDR (0.0293) outperforms FINDER (0.0393) by 25.4% and HITTER (0.0396) by 26.0%. Similarly, on the Congress-bills dataset (Table 2), it outperforms FINDER by 21.2%. The ANC curves in Fig. 3 visually corroborate these quantitative findings, proving that integrating an inductive HyperSAGE encoder with evolutionary strategies allows HD-EDR to trigger network fragmentation much faster and more efficiently than existing baselines.

Table 2. Hypergraph dismantling AUC values for hyperdegree-based load distribution models.

Dataset	HDA	HHDA	CI	HITTER	FINDER	HD-EDR
Algebra	0.0844	0.0848	0.0844	0.0848	0.0849	**0.0788**
Bars–Rev	**0.2028**	0.2100	0.2105	0.2183	**0.2028**	0.2031
Congress-bills	0.1556	0.1303	0.1556	0.1303	0.1537	**0.1211**
DAWN	0.0399	0.0425	0.0399	0.0431	**0.0353**	0.0357
Email–Eu	0.1388	0.1012	0.1083	0.1112	0.1097	**0.0903**
Email–Enron	**0.1751**	0.1828	0.1833	0.1753	**0.1751**	0.1822
Geometry	0.1036	0.0947	0.1036	0.0947	0.0932	**0.0923**
Tags-ask-Ubuntu	0.0675	0.0604	0.0608	0.0586	**0.0580**	0.0592
Tags-math-sx	0.0644	0.0613	0.0651	0.0604	0.0621	**0.0563**
Average	0.1147	0.1076	0.1124	0.1085	0.1083	**0.1021**

Table 3. Hypergraph dismantling AUC values for load distribution models based on residual load of nodes.

Dataset	HDA	HHDA	CI	HITTER	FINDER	HD-EDR
Algebra	0.0837	0.0838	0.0837	0.0879	0.0835	**0.0827**
Bars-Rev	0.2128	0.2242	0.2088	0.2327	0.2099	**0.2055**
Congress-bills	0.1639	0.1388	0.1639	0.1473	**0.1271**	**0.1271**
DAWN	0.0401	0.0394	0.0401	0.0396	0.0393	**0.0293**
Email-Eu	0.0997	0.1066	**0.0954**	0.1218	0.0977	0.0971
Email-Enron	0.1586	0.1752	0.1807	0.2195	**0.1439**	0.1473
Geometry	**0.0942**	**0.0942**	**0.0942**	0.0943	0.0945	0.0943
Tags-ask-Ubuntu	0.0605	0.0562	0.0677	0.0524	0.0521	**0.0513**
Tags-math-sx	0.0637	0.0607	0.0576	0.0562	0.0566	**0.0521**
Average	0.1086	0.1088	0.1102	0.1169	0.1005	**0.0985**

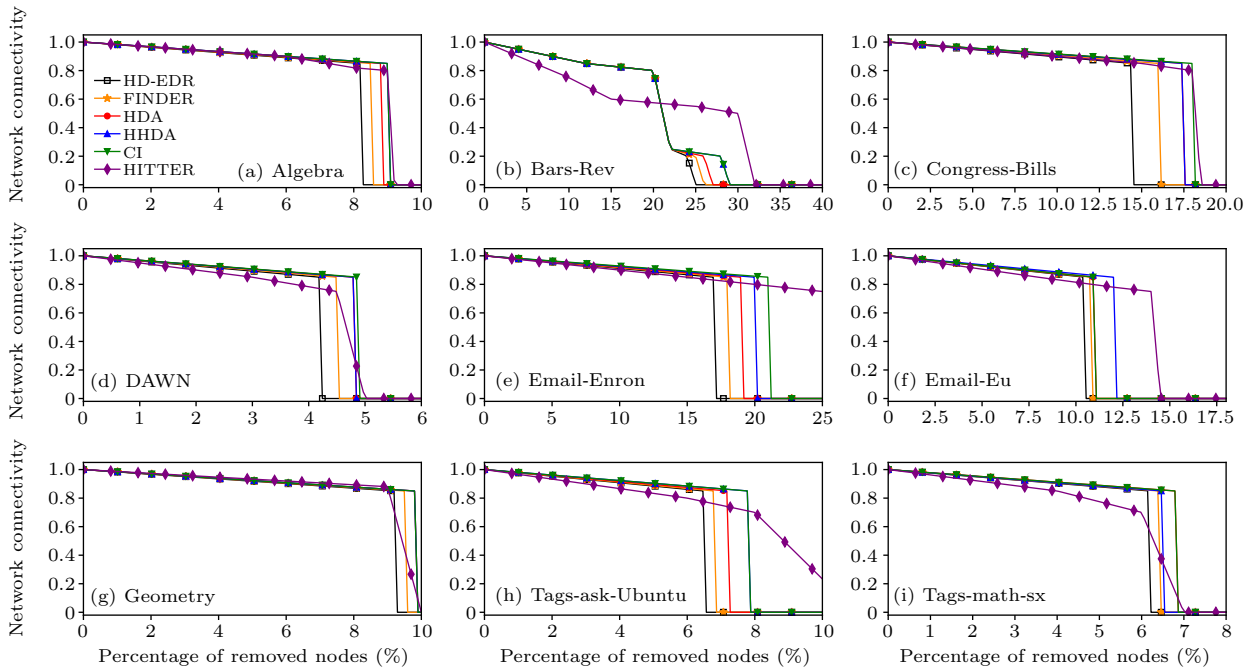


Fig. 3. Network connectivity (ANC) decline under residual capacity-based load distribution models. Compared with baselines (HDA, CI, HHDA, HITTER, FINDER), the proposed HD-EDR (black) consistently induces the fastest network collapse in both scenarios.

4.6. Ablation study

To verify the necessity of each component within HD-EDR, we conducted ablation studies comparing the full framework against three variants: "w/o ES" (gradient learner only), "w/o DRL" (evolutionary strategy only), and "w/o Bi-Sync" (both modules without parameter synchronization). Results (Tables 4 and 5) across nine datasets under both load redistribution models consistently show that the complete HD-EDR achieves the lowest AUC, outperforming all ablated versions.

Table 4. Ablation study for hyperdegree-based load distribution models.

Dataset	HD-EDR	w/o Bi–Sync	w/o ES	w/o DRL
Algebra	**0.0788**	0.0812	0.0835	0.0841
Bars–Rev	**0.2031**	0.2085	0.2140	0.2165
Congress-bills	**0.1211**	0.1265	0.1320	0.1355
DAWN	**0.0357**	0.0382	0.0415	0.0428
Email–Eu	**0.0903**	0.0945	0.1020	0.1055
Email–Enron	**0.1822**	0.1860	0.1915	0.1940
Geometry	**0.0923**	0.0960	0.0998	0.1012
Tags-ask-Ubuntu	**0.0592**	0.0625	0.0660	0.0672
Tags-math-sx	**0.0563**	0.0590	0.0635	0.0648
Average	**0.1021**	**0.1058**	**0.1104**	**0.1124**

The "w/o DRL" variant generally yields the poorest performance (e.g., 0.1355 AUC on Congress-bills), indicating that evolutionary search alone lacks the precise guidance required for efficient convergence. Similarly, the "w/o ES" variant suffers from premature convergence to local optima, evidenced by increased AUC values such as the rise from 0.0357 to 0.0415 on the DAWN dataset. Furthermore, the "w/o Bi-Sync" variant, despite utilizing both algorithms, fails to match the full model's efficacy. Its intermediate performance confirms that simply stacking algorithms is insufficient; the bidirectional synchronization mechanism is critical for enabling the gradient learner to escape local optima via "Elite Reflow" and the evolutionary population to accelerate via "Gradient Injection." Thus, the superior robustness of HD-EDR relies on the synergistic integration of global exploration and local exploitation.

Table 5. Ablation study for load distribution models based on residual load of nodes.

Dataset	HD-EDR	w/o Bi–Sync	w/o ES	w/o DRL
Algebra	**0.0827**	0.0855	0.0882	0.0895
Bars–Rev	**0.2055**	0.2110	0.2185	0.2210
Congress-bills	**0.1271**	0.1330	0.1395	0.1420
DAWN	**0.0293**	0.0325	0.0368	0.0380
Email–Eu	**0.0971**	0.1025	0.1090	0.1125
Email–Enron	**0.1473**	0.1520	0.1595	0.1630
Geometry	**0.0943**	0.0980	0.1025	0.1040
Tags-ask-Ubuntu	**0.0513**	0.0545	0.0590	0.0605
Tags-math-sx	**0.0521**	0.0558	0.0612	0.0625
Average	**0.0985**	**0.1027**	**0.1082**	**0.1103**

4.7. Parameter sensitivity analysis

Figures 4 and 5 present the parameter sensitivity analysis of HD-EDR, evaluated from both external network attack settings (Figs. 4(a)–4(c)) and internal evolutionary hyperparameters (Figs. 5(a)–5(c)).

Impact of network and attack parameters Figures 4(a)–4(c) compare HD-EDR against five baselines, including the newly added FINDER. As the number of removed nodes K increases (Fig. 4(a)), HD-EDR consistently achieves the lowest average network controllability (ANC), progressively widening the performance gap against strong baselines

such as HITTER and FINDER. For the tolerance parameter σ (Fig. 4(b)), our method shows significant dominance under strict capacity constraints ($\sigma < 0.6$). Under varying synchronization strengths I_{sync} (Fig. 4(c)), HD-EDR demonstrates the most stable dismantling effectiveness and the lowest variance, proving its robustness in complex coupled networks.

Impact of evolutionary hyperparameters Figures 5(a)–5(c) evaluate the internal sensitivity of the Evolutionary Strategy (ES) module. Increasing the population size (Fig. 5(a)) enhances global exploration and sharply lowers ANC, though marginal returns diminish beyond a size of 40. Both crossover and mutation rates display distinct U-shaped performance curves. The optimal crossover rate lies between 0.6 and 0.7 (Fig. 5(b)), effectively balancing elite genetic exchange. Conversely, the mutation rate (Fig. 5(c)) is highly sensitive; optimal performance peaks around 0.02 to 0.03. Higher mutation rates (> 0.05) severely disrupt the learned network weights, leading to a rapid degradation in performance.

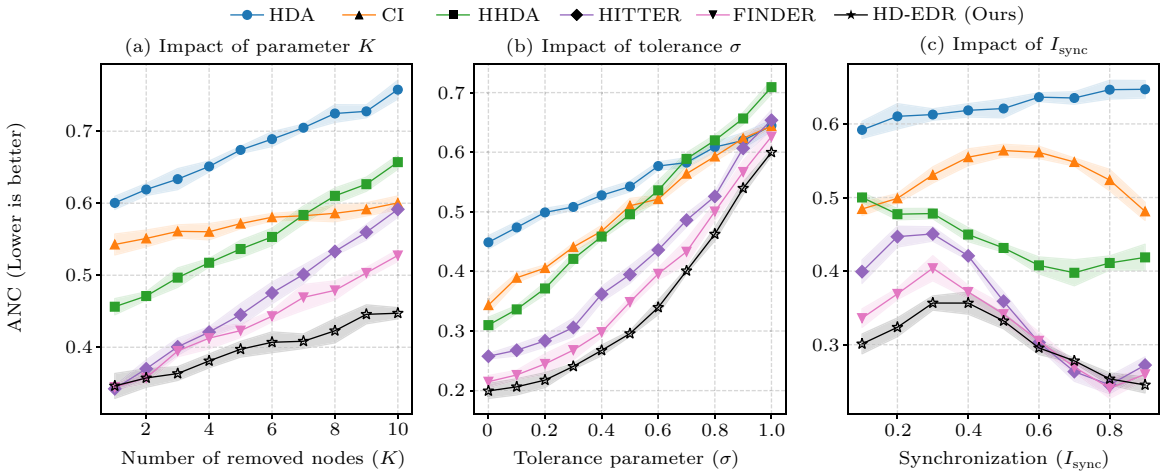


Fig. 4. Parameter sensitivity analysis of average network controllability (ANC). (a) Impact of the number of removed nodes K. (b) Impact of the tolerance parameter σ. (c) Impact of the synchronization strength I_{sync}. The shaded areas represent the standard deviation.

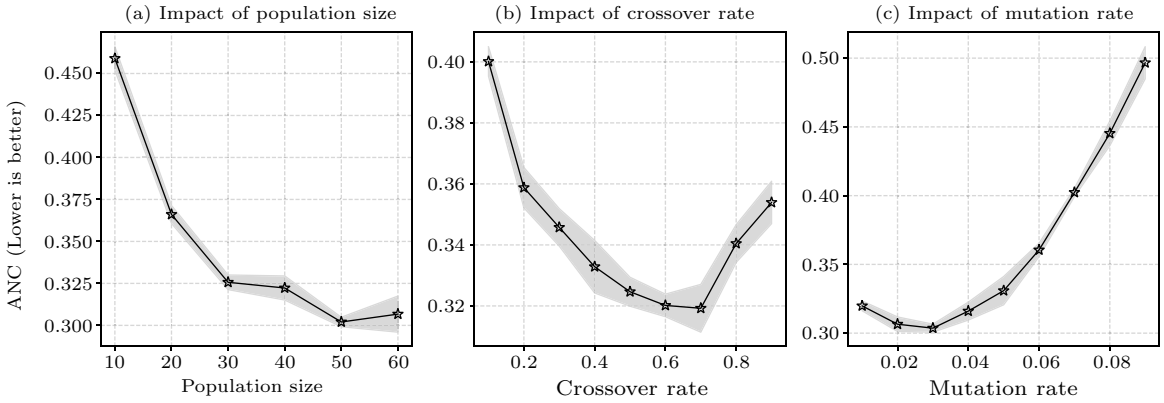


Fig. 5. Parameter sensitivity analysis of average network controllability (ANC). (a) Impact of population size. (b) Impact of crossover rate. (c) Impact of mutation rate. The shaded areas represent the standard deviation.

4.8. Visualization analysis of attacks on real-world hypergraphs

To intuitively verify the efficacy of the HD-EDR algorithm, we conducted a visualization experiment using the classic Les Miserables character co-occurrence network. This dataset comprises 77 nodes representing characters, with hyperedges constructed based on simultaneous appearances in the same scene, reflecting high-order social interactions. The attack budget was set to remove five nodes (approximately 6.5% of the total).

As shown in the left panel of Fig. 6, the HD-EDR algorithm successfully identified five structurally critical nodes: Valjean, Javert, Marius, Gavroche, and Thenardier. These characters serve as vital hubs connecting distinct social communities, demonstrating the algorithm's ability to locate key "structural supports" within the hypergraph.

The right panel illustrates the network state post-attack. Upon removing these five nodes, the giant component size plummeted from 100% to 62.3%. Visually, the dense core of the network was effectively hollowed out, causing peripheral nodes and sub-communities to detach and fragment. This significant structural collapse confirms that HD-EDR can trigger a rapid decline in connectivity with minimal cost, validating its superior performance in identifying vulnerabilities within

real-world hypergraph networks.

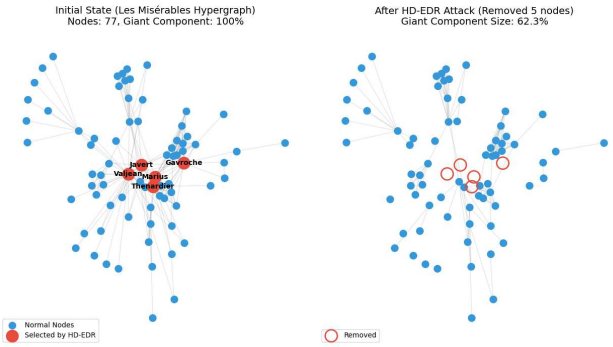


Fig. 6. Impact of the HD-EDR attack on the Les Miserables network. Removing the five structurally critical nodes identified by the algorithm (left, in red) results in significant network fragmentation and a reduction of the giant component to 62.3% (right).

5. Conclusion

In this paper, we propose HD-EDR, a novel evolutionary deep reinforcement learning framework for hypergraph dismantling under cascading failures. Unlike static metrics, our approach explicitly models dynamic load redistribution based on hyperdegree and residual capacity. By synergizing the global exploration of evolutionary strategies with the local exploitation of DRL through a bidirectional parameter synchronization mechanism, HD-EDR effectively adapts to the rugged, evolving optimization landscape. Extensive experiments demonstrate that our framework significantly outperforms state-of-the-art baselines, successfully identifying critical nodes that trigger catastrophic system-wide collapse.

Future work will focus on three key directions: extending the attack space from nodes to hyperedges to uncover diverse vulnerability patterns; incorporating probabilistic failure dynamics to enhance applicability in stochastic environments such as epidemic propagation; and improving scalability through distributed learning techniques to facilitate robustness analysis on billion-scale real-world hypergraphs.

Acknowledgment

Project supported by the National Natural Science Foundation of China (Grant Nos. 72571150 and 62306156).

References

[1] Calderoni F, Catanese S, De Meo P, Ficara A and Fiumara G 2020 *Expert Syst. Appl.* **161** 113666
[2] Ma X Y, Zhou H J and Li Z Y 2021 *Renew. Sustain. Energy Rev.* **152** 111646
[3] Zhang M Y, Huang T, Guo Z X and He Z G 2022 *Physica A* **607** 128063
[4] Ullah A, Shoa J M, Yang Q L, Khan N, Bernard C M and Kumar R 2023 *Expert Syst. Appl.* **228** 120326
[5] Albert R, Jeong H and Barabási A L 2000 *Nature* **406** 378
[6] Freeman L C 1979 *Soc. Netw.* **1** 215
[7] Peng P, Fan T and Lü L 2024 *Entropy* **26** 248
[8] Estrada E and Rodríguez-Velázquez J A 2005 *Phys. Rev. E* **71** 056103
[9] Kapoor K, Sharma D and Srivastava J 2013 *Proceedings of the 2013 IEEE 2nd Network Science Workshop (NSW)* (West Point, NY, USA)
[10] Xie X W, Zhan X X, Zhang Z K and Liu C 2023 *Chaos* **33** 013104
[11] Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young J G and Petri G 2020 *Phys. Rep.* **874** 1
[12] Qu H, Wang X, Song Y R, Ni W, Jiang G P and Sheng Q Z 2025 *Proceedings of the 34th ACM International Conference on Information and Knowledge Management* (Orlando, Florida, USA) p. 2450
[13] Jiang T, Liu K, Xiang B B, Zhang H F and Wang H 2025 *Physica A* **679** 130988
[14] Motter A E and Lai Y C 2002 *Phys. Rev. E* **66** 065102
[15] de Arruda G F, Petri G and Moreno Y 2020 *Phys. Rev. Res.* **2** 023032
[16] Buldyrev S V, Parshani R, Paul G, Stanley H E and Havlin S 2010 *Nature* **464** 1025
[17] Khalil E B, Dai H, Zhang Y, Dilkina B and Song L 2017 *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)* (Long Beach, CA, USA) p. 6348
[18] Manchanda S, Mittal A, Dhawan A, Medya S, Ranu S and Singh A 2020 *Proceedings of the 34th International Conference on Neural Information Processing Systems (Virtual Conference)* p. 20000
[19] Song N, Sheng W, Sun Y, Lin T, Wang Z, Xu Z, Yang F, Zhang Y and Li D 2025 *Neurocomputing* **618** 129117
[20] Newman M E J 2003 *SIAM Rev.* **45** 167
[21] Xu L, Ma L J, Lin Q Z, Li L J, Gong M G and Li J Q 2025 *Inf. Sci.* **698** 121764
[22] Feng Y F, You H X, Zhang Z Z, Ji R R and Gao Y 2019 *Proceedings of the AAAI Conference on Artificial Intelligence* (Honolulu, Hawaii, USA) p. 3558
[23] Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A and Talukdar P 2019 *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Vancouver, Canada) p. 1511
[24] Hamilton W L, Ying R and Leskovec J 2017 *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) p. 1025
[25] Colas C, Sigaud O and Oudeyer P Y 2018 *Proceedings of the 35th International Conference on Machine Learning* (Stockholm, Sweden) p. 1039
[26] Jia C X and Liu R R 2026 *Reliab. Eng. Syst. Saf.* **267** 111850
[27] Zhou B, Ma X J, Ma F X and Gao S J 2023 *AIMS Math.* **8** 4814
[28] Amburg I, Veldt N and Benson A R 2020 arXiv:2006.05863 [math.PR]
[29] Benson A R, Abebe R, Schaub M T and Kleinberg J 2018 *Proc. Natl. Acad. Sci. USA* **115** E11221
[30] Yin H, Benson A R, Leskovec J and Gleich D F 2017 *Proceedings of the 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD '17)* (Halifax, NS, Canada) p. 555
[31] Petri G and Barrat A 2018 *Phys. Rev. Lett.* **121** 228301
[32] Ren X L, Gleinig N, Helbing D and Antulov-Fantulin N 2019 *Proc. Natl. Acad. Sci. USA* **116** 6554
[33] Morone F and Makse H A 2015 *Nature* **524** 65
[34] Yan D, Xie W, Zhang Y, He Q and Yang Y 2022 *IEEE Trans. Netw. Sci. Eng.* **9** 3302
[35] Fan C J, Zeng L, Sun Y Z and Liu Y Y 2020 *Nat. Mach. Intell.* **2** 317